All Classes

AbstractTrigger
AndMatcher
AnnualCalendar
AttributeRestoringConnectionInvocationHandler
BaseCalendar
BroadcastJobListener
BroadcastSchedulerListener
BroadcastTriggerListener
Calendar
CalendarIntervalScheduleBuilder
CalendarIntervalTrigger
CalendarIntervalTriggerImpl
CalendarIntervalTriggerPersistenceDelegate
CascadingClassLoadHelper
CircularLossyQueue
ClassUtils
CloudscapeDelegate
ConnectionProvider
Constants
Constraint
CoreTrigger
Counter
CounterConfig
CounterImpl
CounterManager
CounterManagerImpl
CpuConstraint
CpuConstraint.Operator
CpuEvaluator
CronCalendar
CronExpression
CronScheduleBuilder
CronTrigger
CronTriggerImpl
CronTriggerPersistenceDelegate
CronTriggerSupport
DailyCalendar
DateBuilder
DateBuilder.IntervalUnit
DB2v6Delegate
DB2v7Delegate
DB2v8Delegate
DBConnectionManager
DBSemaphore
DelegatingLocalityJobDetail
DelegatingLocalityTrigger
DirectoryScanJob
DirectoryScanListener
DirectSchedulerFactory
PostgreSQLDelegate
PropertiesParser
PropertySettingJobFactory
QuartzInitializerListener
QuartzInitializerServlet
QuartzScheduler
QuartzSchedulerMBean
QuartzSchedulerMBeanImpl
QuartzSchedulerResources
QuartzSchedulerThread
QuartzServer
QuartzService
QuartzServiceMBean
RAMJobStore
RemotableQuartzScheduler
RemoteMBeanScheduler
RemoteScheduler
SampledCounter
SampledCounterConfig
SampledCounterImpl
SampledRateCounter
SampledRateCounterConfig
SampledRateCounterImpl
SampledStatistics
SampledStatisticsImpl
ScheduleBuilder
Scheduler
SchedulerConfigException
SchedulerContext
SchedulerException
SchedulerFactory
SchedulerListener
SchedulerListenerSupport
SchedulerMetaData
SchedulerPluginWithUserTransactionSupport
SchedulerRepository
SchedulerSignalerImpl
SchedulerStateRecord
Semaphore
SendDestinationMessageJob
SendMailJob
SendMailJob.MailInfo
SendQueueMessageJob
SendTopicMessageJob
ShutdownHookPlugin
SimpleClassLoadHelper
SimpleInstanceIdGenerator
SimpleJobFactory
SimplePropertiesTriggerPersistenceDelegateSupport
SimplePropertiesTriggerProperties
All Classes  

AbstractTrigger  
AndMatcher  
AnnualCalendar  
AttributeRestoringConnectionInvocationHandler  
BaseCalendar  
BroadcastJobListener  
BroadcastSchedulerListener  
BroadcastTriggerListener  
Calendar  
CalendarIntervalScheduleBuilder  
CalendarIntervalTrigger  
CalendarIntervalTriggerImpl  
CalendarIntervalTriggerPersistenceDelegate  
CascadingClassLoadHelper  
CircularLossyQueue  
ClassUtils  
CloudscapeDelegate  
ConnectionProvider  
Constants  
Constraint  
CoreTrigger  
Counter  
CounterConfig  
CounterImpl  
CounterManager  
CounterManagerImpl  
CpuConstraint  
CpuConstraint.Operator  
CpuEvaluator  
CronCalendar  
CronExpression  
CronScheduleBuilder  
CronTrigger  
CronTriggerImpl  
CronTriggerPersistenceDelegate  
CronTriggerSupport  
DailyCalendar  
DateBuilder  
DateBuilder.IntervalUnit  
DB2v6Delegate  
DB2v7Delegate  
DB2v8Delegate  
DBConnectionManager  
DBSemaphore  
DelegatingLocalityJobDetail  
DelegatingLocalityTrigger  
DirectoryScanJob  
DirectoryScanListener  
DirectSchedulerFactory
## Constant Field Values

### Contents

- `org.quartz.*`

### org.quartz.*

<table>
<thead>
<tr>
<th>Class</th>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.Calendar</code></td>
<td><code>month</code></td>
<td>0</td>
</tr>
</tbody>
</table>

### org.quartz.CalendarIntervalTrigger

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MISFIRE_INSTRUCTION_DO NOTHING</code></td>
<td>2</td>
</tr>
<tr>
<td><code>MISFIRE_INSTRUCTION_FIRE_ONCE_NOW</code></td>
<td>1</td>
</tr>
</tbody>
</table>

### org.quartz.CronExpression

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ALL_SPEC_INT</code></td>
<td>99</td>
</tr>
<tr>
<td><code>DAY_OF_MONTH</code></td>
<td>3</td>
</tr>
<tr>
<td><code>DAY_OF_WEEK</code></td>
<td>5</td>
</tr>
<tr>
<td><code>HOUR</code></td>
<td>2</td>
</tr>
<tr>
<td><code>MINUTE</code></td>
<td>1</td>
</tr>
<tr>
<td><code>MONTH</code></td>
<td>4</td>
</tr>
<tr>
<td><code>NO_SPEC_INT</code></td>
<td>98</td>
</tr>
<tr>
<td><code>SECOND</code></td>
<td>0</td>
</tr>
<tr>
<td><code>YEAR</code></td>
<td>6</td>
</tr>
</tbody>
</table>

### org.quartz.CronTrigger

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MISFIRE_INSTRUCTION_DO NOTHING</code></td>
<td>2</td>
</tr>
<tr>
<td><code>MISFIRE_INSTRUCTION_FIRE_ONCE_NOW</code></td>
<td>1</td>
</tr>
<tr>
<td><code>serialVersionUID</code></td>
<td>-86449531464515L</td>
</tr>
</tbody>
</table>

### org.quartz.DateBuilder

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FRIDAY</code></td>
<td>6</td>
</tr>
<tr>
<td><code>MILLISECONDS_IN_DAY</code></td>
<td>86400000L</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>public static final long MILLISECONDS_IN_HOUR</td>
<td>3600000L</td>
</tr>
<tr>
<td>public static final long MILLISECONDS_IN_MINUTE</td>
<td>60000L</td>
</tr>
<tr>
<td>public static final int MONDAY</td>
<td>2</td>
</tr>
<tr>
<td>public static final int SATURDAY</td>
<td>7</td>
</tr>
<tr>
<td>public static final long SECONDS_IN_MOST_DAYS</td>
<td>86400L</td>
</tr>
<tr>
<td>public static final int SUNDAY</td>
<td>1</td>
</tr>
<tr>
<td>public static final int THURSDAY</td>
<td>5</td>
</tr>
<tr>
<td>public static final int TUESDAY</td>
<td>3</td>
</tr>
<tr>
<td>public static final int WEDNESDAY</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.Scheduler</td>
<td>DEFAULT_FAIL_OVER_GROUP</td>
<td>String</td>
</tr>
<tr>
<td>org.quartz.Scheduler</td>
<td>DEFAULT_GROUP</td>
<td>String</td>
</tr>
<tr>
<td>org.quartz.Scheduler</td>
<td>DEFAULT_RECOVERY_GROUP</td>
<td>String</td>
</tr>
<tr>
<td>org.quartz.Scheduler</td>
<td>FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS</td>
<td>String</td>
</tr>
<tr>
<td>org.quartz.Scheduler</td>
<td>FAILED_JOB_ORIGINAL_TRIGGER_GROUP</td>
<td>String</td>
</tr>
<tr>
<td>org.quartz.Scheduler</td>
<td>FAILED_JOB_ORIGINAL_TRIGGER_NAME</td>
<td>String</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.SimpleTrigger</td>
<td>MISFIRE_INSTRUCTION_FIRE_NOW</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.SimpleTrigger</td>
<td>MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.SimpleTrigger</td>
<td>MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.SimpleTrigger</td>
<td>MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.SimpleTrigger</td>
<td>MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.SimpleTrigger</td>
<td>REPEAT_INDEFINITELY</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.SimpleTrigger</td>
<td>serialVersionUID</td>
<td>long</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.Trigger</td>
<td>DEFAULT_PRIORITY</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.Trigger</td>
<td>MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.Trigger</td>
<td>MISFIRE_INSTRUCTION_SMART_POLICY</td>
<td>int</td>
</tr>
<tr>
<td>org.quartz.Trigger</td>
<td>serialVersionUID</td>
<td>long</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core.QuartzSchedulerResources</td>
<td>CREATE_REGISTRY_ALWAYS</td>
<td>String</td>
</tr>
<tr>
<td>org.quartz.core.QuartzSchedulerResources</td>
<td>serialVersionUID</td>
<td>long</td>
</tr>
<tr>
<td>String</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>CREATE_REGISTRY_AS_NEEDED</td>
<td>&quot;as_needed&quot;</td>
<td></td>
</tr>
<tr>
<td>CREATE_REGISTRY_NEVER</td>
<td>&quot;never&quot;</td>
<td></td>
</tr>
</tbody>
</table>

### org.quartz.core.jmx. QuartzSchedulerMBean

<table>
<thead>
<tr>
<th>String</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB_ADDED</td>
<td>&quot;jobAdded&quot;</td>
</tr>
<tr>
<td>JOB_DELETED</td>
<td>&quot;jobDeleted&quot;</td>
</tr>
<tr>
<td>JOB_EXECUTION_VETOED</td>
<td>&quot;jobExecutionVetoed&quot;</td>
</tr>
<tr>
<td>JOB_SCHEDULED</td>
<td>&quot;jobScheduled&quot;</td>
</tr>
<tr>
<td>JOB_TO_BE_EXECUTED</td>
<td>&quot;jobToBeExecuted&quot;</td>
</tr>
<tr>
<td>JOB_UNSCHEDULED</td>
<td>&quot;jobUnscheduled&quot;</td>
</tr>
<tr>
<td>JOB_WAS_EXECUTED</td>
<td>&quot;jobWasExecuted&quot;</td>
</tr>
<tr>
<td>JOBS_PAUSED</td>
<td>&quot;jobsPaused&quot;</td>
</tr>
<tr>
<td>JOBS_RESUMED</td>
<td>&quot;jobsResumed&quot;</td>
</tr>
<tr>
<td>SAMPLED_STATISTICS_ENABLED</td>
<td>&quot;sampledStatisticsEnabled&quot;</td>
</tr>
<tr>
<td>SAMPLED_STATISTICS_RESET</td>
<td>&quot;sampledStatisticsReset&quot;</td>
</tr>
<tr>
<td>SCHEDULER_ERROR</td>
<td>&quot;schedulerError&quot;</td>
</tr>
<tr>
<td>SCHEDULER_PAUSED</td>
<td>&quot;schedulerPaused&quot;</td>
</tr>
<tr>
<td>SCHEDULER_SHUTDOWN</td>
<td>&quot;schedulerShutdown&quot;</td>
</tr>
<tr>
<td>SCHEDULER_STARTED</td>
<td>&quot;schedulerStarted&quot;</td>
</tr>
<tr>
<td>SCHEDULING_DATA_CLEARED</td>
<td>&quot;schedulingDataCleared&quot;</td>
</tr>
<tr>
<td>TRIGGER_FINALIZED</td>
<td>&quot;triggerFinalized&quot;</td>
</tr>
<tr>
<td>TRIGGERS_PAUSED</td>
<td>&quot;triggersPaused&quot;</td>
</tr>
<tr>
<td>TRIGGERS_RESUMED</td>
<td>&quot;triggersResumed&quot;</td>
</tr>
</tbody>
</table>

### org.quartz.ee.jta. UserTransactionHelper

<table>
<thead>
<tr>
<th>String</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_USER_TX_LOCATION</td>
<td>&quot;java:comp/UserTransaction&quot;</td>
</tr>
</tbody>
</table>

### org.quartz.ee.servlet. QuartzInitializerListener

<table>
<thead>
<tr>
<th>String</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUARTZ_FACTORY_KEY</td>
<td>&quot;org.quartz.impl.StdSchedulerFactory&quot;</td>
</tr>
</tbody>
</table>

### org.quartz.ee.servlet. QuartzInitializerServlet

<table>
<thead>
<tr>
<th>String</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUARTZ_FACTORY_KEY</td>
<td>&quot;org.quartz.impl.StdSchedulerFactory&quot;</td>
</tr>
</tbody>
</table>

### org.quartz.impl. DirectSchedulerFactory

<table>
<thead>
<tr>
<th>String</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_INSTANCE_ID</td>
<td>&quot;SIMPLE_NON_CLUSTERED&quot;</td>
</tr>
<tr>
<td>public static final String DEFAULT_SCHEDULER_NAME</td>
<td>&quot;SimpleQuartzScheduler&quot;</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-------------------------</td>
</tr>
</tbody>
</table>

```java
org.quartz.impl.StdSchedulerFactory
```

<table>
<thead>
<tr>
<th>public static final String AUTO_GENERATE_INSTANCE_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String DEFAULT_INSTANCE_ID</td>
</tr>
<tr>
<td>public static final String PROP_CONNECTION_PROVIDER_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_DRIVER</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_JNDI_ALWAYS_LOOKUP</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_JNDI_CREDENTIALS</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_JNDI_INITIAL</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_JNDI_PRINCIPAL</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_JNDI_PROVIDER</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_JNDI_URL</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_MAX_CONNECTIONS</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_PASSWORD</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_URL</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_USER</td>
</tr>
<tr>
<td>public static final String PROP_DATASOURCE_VALIDATION_QUERY</td>
</tr>
<tr>
<td>public static final String PROP_JOB_LISTENER_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_JOB_STORE_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_JOB_STORE_LOCK_HANDLER_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_JOB_STORE_LOCK_HANDLER_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_JOB_STORE_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_JOB_STORE_USE_PROP</td>
</tr>
<tr>
<td>public static final String PROP_LISTENER_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_PLUGIN_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_PLUGIN_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_BATCH_TIME_WINDOW</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_CLASS_LOAD_HELPER_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_CONTEXT_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_DB_FAILURE_RETRY_INTERVAL</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_IDLE_WAIT_TIME</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_INSTANCE_ID</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_INSTANCE_ID_GENERATOR_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_INSTANCE_ID_GENERATOR_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_INSTANCE_NAME</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN_WITH_WAIT</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_JMX_EXPORT</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_JMX_OBJECT_NAME</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_JMX_PROXY</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_JMX_PROXY_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_JOB_FACTORY_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_JOB_FACTORY_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_MAKE_SCHEDULER_THREAD_DAEMON</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_MAX_BATCH_SIZE</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_NAME</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_RMI_BIND_NAME</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_RMI_CREATE_REGISTRY</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_RMI_EXPORT</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_RMI_HOST</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_RMI_PORT</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_RMI_PROXY</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_RMI_SERVER_PORT</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_SCHEDULER_THREADS_INHERIT_CONTEXT_CLASS_LOADER_OF_INITIALIZING_THREAD</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_SKIP_UPDATE_CHECK</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_THREAD_NAME</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_USER_TX_URL</td>
</tr>
<tr>
<td>public static final String PROP_SCHED_WRAP_JOB_IN_USER_TX</td>
</tr>
<tr>
<td>public static final String PROP_TABLE_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_THREAD_POOL_CLASS</td>
</tr>
<tr>
<td>public static final String PROP_THREAD_POOL_PREFIX</td>
</tr>
<tr>
<td>public static final String PROP_TRIGGER_LISTENER_PREFIX</td>
</tr>
<tr>
<td>public static final String PROPERTIES_FILE</td>
</tr>
<tr>
<td>public static final String SYSTEM_PROPERTY_AS_INSTANCE_ID</td>
</tr>
</tbody>
</table>

**org.quartz.impl.jdbcjobstore.Constants**

<p>| public static final String ALIAS_COL_NEXT_FIRE_TIME | &quot;ALIAS_NXT_FR_TM&quot; |
| public static final String ALL_GROUPS_PAUSED | &quot;$<em>ALL_GROUPS_PAUSED$</em>&quot; |
| public static final String COL_BLOB | &quot;BLOB_DATA&quot; |
| public static final String COL_CALENDAR | &quot;CALENDAR&quot; |</p>
<table>
<thead>
<tr>
<th>public static final String</th>
<th>COL(Calendar_Name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String</td>
<td>COL(CHECKIN_INTERVAL)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(CRON_EXPRESSION)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL_DEscription</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(ENTRY_ID)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(ENTRY_STATE)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(ENTRY)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(INSTANCE_NAME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(STATE)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(IS_DURABLE)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(IS_NONCONCURRENT)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(IS_UPDATE_DATA)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(IS_VOLATILE)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(JOB_CLASS)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(JOB_DATA)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(JOB_GROUP)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(JOB_NAME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(LAST_CHECKIN_TIME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(LOCK_NAME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(MISFIRE_INSTRUCTION)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(NEXT_FIRE_TIME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(PREV_FIRE_TIME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(PRIORITY)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(REPEAT_COUNT)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(REPEAT_INTERVAL)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(REQUESTS_RECOVERY)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL_SCHEDULER_NAME</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(SCHED_NAME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(START_TIME)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(TIME_ZONE_ID)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL(TIMES_TRIGGERED)</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL_TRIGGER_GROUP</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL_TRIGGER_NAME</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL_TRIGGER_STATE</td>
</tr>
<tr>
<td>public static final String</td>
<td>COL_TRIGGER_TYPE</td>
</tr>
<tr>
<td>public static final String</td>
<td>DEFAULT_TABLE_PREFIX</td>
</tr>
<tr>
<td>public static final String</td>
<td>STATE_ACQUIRED</td>
</tr>
</tbody>
</table>
### org.quartz.impl.jdbcjobstore.DB2v6Delegate

<table>
<thead>
<tr>
<th>Method Name</th>
<th>SQL Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT_NUM_CALENDARS</td>
<td>&quot;SELECT COUNT(*) FROM {0}CALENDARS WHERE SCHED_NAME = {1}&quot;</td>
<td>Select number of calendars</td>
</tr>
<tr>
<td>SELECT_NUM_JOBS</td>
<td>&quot;SELECT COUNT(*) FROM {0}JOB_DETAILS WHERE SCHED_NAME = {1}&quot;</td>
<td>Select number of jobs</td>
</tr>
<tr>
<td>SELECT_NUM_TRIGGERS</td>
<td>&quot;SELECT COUNT(*) FROM {0}TRIGGERS WHERE SCHED_NAME = {1}&quot;</td>
<td>Select number of triggers</td>
</tr>
</tbody>
</table>
```java
public static final String SELECT_NUM_TRIGGERS_FOR_JOB = "SELECT COUNT(*) FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND JOB_NAME = ? AND JOB_GROUP = ?";

```
protected static final String INSERT_SIMPLE_PROPS_TRIGGER

protected static final String SELECT_SIMPLE_PROPS_TRIGGER
"SELECT {0}SIMPROP_TRIGGERS WHERE SCHED_NAME = {1} AND TRIGGER_NAME = ? AND TRIGGER_GROUP = ?"

protected static final String TABLE_SIMPLE_PROPERTIES_TRIGGERS
"SIMPROP_TRIGGERS"

protected static final String UPDATE_SIMPLE_PROPS_TRIGGER
SET STR_PROP_1 = ?, STR_PROP_2 = ?, STR_PROP_3 = ?, INT_PROP_1 = ?, INT_PROP_2 = ?, LONG_PROP_1 = ?, LONG_PROP_2 = ?, DEC_PROP_1 = ?, DEC_PROP_2 = ?, BOOL_PROP_1 = ?, BOOL_PROP_2 = ?
WHERE SCHED_NAME = {1} AND TRIGGER_NAME = ? AND TRIGGER_GROUP = ?

org.quartz.impl.jdbcjobstore.StdJDBCConstants
public static final String COUNT_MISFIRED_TRIGGERS_IN_STATE
FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND NOT (MISFIRE_INSTR = -1) AND NEXT_FIRE_TIME < ?

public static final String DELETE_ALL_BLOB_TRIGGERS
{0}BLOB_TRIGGERS WHERE

public static final String DELETE_ALL_CALENDARS

public static final String DELETE_ALL_CRON_TRIGGERS
{0}CRON_TRIGGERS WHERE

public static final String DELETE_ALL_JOB_DETAILS
{0}JOB_DETAILS WHERE

public static final String DELETE_ALL_PAUSED_TRIGGER_GRPS
{0}PAUSED_TRIGGER_GRPS WHERE SCHED_NAME = {1}"

public static final String DELETE_ALL_SIMPLE_TRIGGERS
WHERE SCHED_NAME = {1}"

public static final String DELETE_ALL_SIMPROP_TRIGGERS
WHERE SCHED_NAME = {1}"

public static final String DELETE_ALL_TRIGGERS

public static final String DELETE_BLOB_TRIGGER
{0}BLOB_TRIGGERS WHERE

public static final String DELETE_CALENDAR
SCHED_NAME = {1}

public static final String DELETE_CRON_TRIGGER
{0}CRON_TRIGGERS WHERE

public static final String DELETE_SIMPLE_TRIGGER
WHERE SCHED_NAME = {1}"

public static final String DELETE_SIMPROP_TRIGGER
WHERE SCHED_NAME = {1}"

public static final String DELETE_TRIGGER
WHERE SCHED_NAME = {1}"
public static final String DELETE_FIRED_TRIGGER

public static final String DELETE_FIRED_TRIGGERS

public static final String DELETE_INSTANCES_FIRED_TRIGGERS

public static final String DELETE_JOB_DETAIL

public static final String DELETE_NO_RECOVERY_FIRED_TRIGGERS

public static final String DELETE_PAUSED_TRIGGER_GROUP

public static final String DELETE_PAUSED_TRIGGER_GROUPS

public static final String DELETE_SCHEDULER_STATE

public static final String DELETE_SIMPLE_TRIGGER

public static final String DELETE_TRIGGER
public static final String INSERT_BLOB_TRIGGER

public static final String INSERT_CALENDAR

public static final String INSERT_CRON_TRIGGER

public static final String INSERT_FIRED_TRIGGER

public static final String INSERT_JOB_DETAIL
public static final String INSERT_PAUSED_TRIGGER_GROUP

public static final String INSERT_SCHEDULER_STATE

public static final String INSERT_SIMPLE_TRIGGER

public static final String INSERT_TRIGGER

public static final String SCHED_NAME_SUBST
<table>
<thead>
<tr>
<th>public static final String SELECT_BLOB_TRIGGER</th>
<th>{0}</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String SELECT_CALENDAR</td>
<td>{0}</td>
</tr>
<tr>
<td>public static final String SELECT_CALENDAR_EXISTENCE</td>
<td>FROM {0}</td>
</tr>
<tr>
<td>public static final String SELECT_CALENDARS</td>
<td>FROM {0}</td>
</tr>
<tr>
<td>public static final String SELECT_CRON_TRIGGER</td>
<td>{0}</td>
</tr>
<tr>
<td>public static final String SELECT_FIRED_TRIGGER</td>
<td>{0}</td>
</tr>
<tr>
<td>public static final String SELECT_FIRED_TRIGGER_GROUP</td>
<td>{0}</td>
</tr>
<tr>
<td>public static final String SELECT_FIRED_TRIGGER_INSTANCE_NAMES</td>
<td>{0}</td>
</tr>
<tr>
<td>public static final String SELECT_FIRED_TRIGGERS</td>
<td>{0}</td>
</tr>
<tr>
<td>public static final String SELECT_FIRED_TRIGGERS_OF_JOB</td>
<td>{0}</td>
</tr>
<tr>
<td>public static final String SELECT_FIRED_TRIGGERS_OF_JOB_GROUP</td>
<td>{0}</td>
</tr>
<tr>
<td>Method Name</td>
<td>SQL Query</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE</td>
<td>SELECT TRIGGER_NAME, SCHED_NAME = {1} AND NOT (MISFIRE_INSTR = -1) AND NEXT_FIRE_TIME &lt; ? AND TRIGGER_STATE =</td>
</tr>
<tr>
<td>SELECT_INSTANCES_FIRED_TRIGGERS</td>
<td>{0}FIRED_TRIGGERS WHERE SCHED_NAME = {1} AND</td>
</tr>
<tr>
<td>SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS</td>
<td>{0}FIRED_TRIGGERS WHERE SCHED_NAME = {1} AND INSTANCE_NAME = ? AND REQUESTS_RECOVERY = ?</td>
</tr>
<tr>
<td>SELECT_JOB_DETAIL</td>
<td>{0}JOB_DETAILS WHERE SCHED_NAME = {1} AND</td>
</tr>
<tr>
<td>SELECT_JOB_EXECUTION_COUNT</td>
<td>FROM {0}FIRED_TRIGGERS WHERE SCHED_NAME = {1} AND JOB_NAME = ?</td>
</tr>
<tr>
<td>SELECT_JOB_EXISTENCE</td>
<td>&quot;SELECT JOB_NAME FROM {0}JOB_DETAILS WHERE SCHED_NAME = {1} AND &quot;</td>
</tr>
<tr>
<td>SELECT_JOB_FOR_TRIGGER</td>
<td>{0}JOB_DETAILS J WHERE T.SCHED_NAME = {1} AND J.SCHED_NAME = {1} AND T.TRIGGER_NAME = ? AND</td>
</tr>
</tbody>
</table>
public static final String SELECT_JOB_GROUPS

public static final String SELECT_JOB_NONCONCURRENT

public static final String SELECT_JOBS_IN_GROUP

public static final String SELECT_MISFIRED_TRIGGERS

public static final String SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE

public static final String SELECT_MISFIRED_TRIGGERS_IN_STATE
public static final String SELECT_NEXT_FIRE_TIME

SELECT_MIN(NEXT_FIRE_TIME) AS ALIAS_NXT_FR_TM FROM SCHED_NAME = {1} AND TRIGGER_STATE = ? AND NEXT_FIRE_TIME >= 0

public static final String SELECT_NEXT_TRIGGER_TO_ACQUIRE

SELECT TRIGGER_NAME, SCHED_NAME = {1} AND TRIGGER_STATE = ? AND NEXT_FIRE_TIME < ? AND (NEXT_FIRE_TIME >= ?) ORDER BY NEXT_FIRE_TIME

public static final String SELECT_NUM_CALENDARS

COUNT(CALENDAR_NAME) FROM {0}CALENDARS WHERE

public static final String SELECT_NUM_JOBS

SELECT COUNT(JOB_NAME) WHERE SCHED_NAME = {1}

public static final String SELECT_NUM_TRIGGERS

FROM {0}TRIGGERS WHERE

public static final String SELECT_NUM_TRIGGERS_FOR_JOB

FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND

public static final String SELECT_NUM_TRIGGERS_IN_GROUP

FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND TRIGGER_GROUP = ?

public static final String SELECT_PAUSED_TRIGGER_GROUP

"SELECT TRIGGER_GROUP {0}PAUSED_TRIGGER_GRPS WHERE SCHED_NAME = {1} AND TRIGGER_GROUP = ?"
public static final String SELECT_PAUSED_TRIGGER_GROUPS = "SELECT TRIGGER_GROUP {0}PAUSED_TRIGGER_GRPS WHERE SCHED_NAME = {1}";

public static final String SELECT_REFERENCED_CALENDAR = "SELECT CALENDAR_NAME FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND ";

public static final String SELECT_SCHEDULER_STATE = "WHERE SCHED_NAME = {1} AND INSTANCE_NAME = ?";

public static final String SELECT_SCHEDULER_STATES = "WHERE SCHED_NAME = {1}";

public static final String SELECT_SIMPLE_TRIGGER = "WHERE SCHED_NAME = {1} AND TRIGGER_NAME = ? AND TRIGGER_GROUP = ?";

public static final String SELECT_TRIGGER = "SCHED_NAME = {1} AND TRIGGER_NAME = ? AND ";

public static final String SELECT_TRIGGER_DATA = "SELECT JOB_DATA FROM SCHED_NAME = {1} AND TRIGGER_NAME = ? AND ";

public static final String SELECT_TRIGGER_EXISTENCE = "SELECT TRIGGER_NAME FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND TRIGGER_NAME = ? AND ";

public static final String SELECT_TRIGGER_FOR_FIRE_TIME = "SELECT TRIGGER_NAME, SCHED_NAME = {1} AND TRIGGER_STATE = ? AND ";

public static final String SELECT_TRIGGER_GROUPS = "DISTINCT(TRIGGER_GROUP) FROM {0}TRIGGERS WHERE ";
public static final String SELECT_TRIGGER_GROUPS_FILTERED =
"SELECT DISTINCT(TRIGGER_GROUP) FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND TRIGGER_GROUP LIKE ?";

public static final String SELECT_TRIGGER_STATE =
"SELECT TRIGGER_STATE FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND TRIGGER_NAME = ?";

public static final String SELECT_TRIGGER_STATUS =
"SELECT TRIGGER_STATE, SCHED_NAME FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND TRIGGER_NAME = ?";

public static final String SELECT_TRIGGERS_FOR_CALENDAR =
"SELECT TRIGGER_NAME, SCHED_NAME FROM {0}TRIGGERS WHERE SCHED_NAME = {1}";

public static final String SELECT_TRIGGERS_FOR_JOB =
"SELECT TRIGGER_NAME FROM {0}TRIGGERS WHERE SCHED_NAME = {1}";

public static final String SELECT_TRIGGERS_IN_GROUP =
"SELECT TRIGGER_NAME, SCHED_NAME FROM {0}TRIGGERS WHERE SCHED_NAME = {1} AND TRIGGER_GROUP LIKE ?";

public static final String SELECT_TRIGGERS_IN_STATE =
"SELECT TRIGGER_NAME, SCHED_NAME FROM {0}TRIGGERS WHERE SCHED_NAME = {1}";

public static final String TABLE_PREFIX_SUBST =
"{0}BLOB_TRIGGERS SET SCHED_NAME = {1}";

public static final String UPDATE_BLOB_TRIGGER =
"UPDATE {0}BLOB_TRIGGERS SET SCHED_NAME = {1}";
public static final String UPDATE_CALENDAR

"UPDATE {0}CALENDARS
SET CALENDAR = ?
SCHED_NAME = {1} AND
"

public static final String UPDATE_CRON_TRIGGER

"{0}CRON_TRIGGERS
SET CRON_EXPRESSION = ?,
TIME_ZONE_ID = ?
WHERE
SCHED_NAME = {1} AND
"

public static final String UPDATE_FIRED_TRIGGER

{"0}FIRED_TRIGGERS
SET FIRED_TIME = ?,
STATE = IS_NONCONCURRENT = ?,
REQUESTS_RECOVERY = ?
WHERE
SCHED_NAME = {1}
"

public static final String UPDATE_INSTANCES_FIRED_TRIGGER_STATE

{"0}FIRED_TRIGGERS
SET SCHED_NAME = {1}
"

public static final String UPDATE_JOB_DATA

"UPDATE {0}JOB_DETAILS
SET JOB_DATA = ?
SCHED_NAME = {1} AND
"

public static final String UPDATE_JOB_DETAIL

"UPDATE {0}JOB_DETAILS
SET DESCRIPTION = ?,
IS_NONCONCURRENT = ?,
REQUESTS_RECOVERY = ,
SCHED_NAME = {1}
"

public static final String UPDATE_JOB_DETAIL
<table>
<thead>
<tr>
<th>Method Name</th>
<th>SQL Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE_JOB_TRIGGER_STATES</td>
<td>UPDATE {0}TRIGGERS SET TRIGGER_STATE = ? WHERE SCHED_NAME = {1} AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE</td>
<td>UPDATE {0}TRIGGERS SET TRIGGER_STATE = ? WHERE SCHED_NAME = {1} AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE_SCHEDULER_STATE</td>
<td>{0}SCHEDULER_STATE SET LAST_CHECKIN_TIME = ? WHERE SCHED_NAME = {1} AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE_SIMPLE_TRIGGER</td>
<td>{0}SIMPLE_TRIGGERS SET REPEAT_INTERVAL = ?, WHERE SCHED_NAME = {1} AND</td>
</tr>
<tr>
<td></td>
<td>TRIGGER_NAME = ? AND TRIGGER_GROUP = ?</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE_TRIGGER</td>
<td>{0}TRIGGERS SET JOB_NAME = ?, JOB_GROUP = ?, DESCRIPTION = ?, PRIORITY = ?</td>
</tr>
</tbody>
</table>
public static final String UPDATE_TRIGGER_GROUP_STATE_FROM_STATES

public static final String UPDATE_TRIGGER_SKIP_DATA

public static final String UPDATE_TRIGGER_STATE

public static final String UPDATE_TRIGGER_STATE_FROM_STATE

public static final String UPDATE_TRIGGER_STATE_FROM_STATES

public static final String UPDATE_TRIGGER_STATES_FROM_OTHER_STATES
<table>
<thead>
<tr>
<th>Method/Class</th>
<th>Method Type</th>
<th>SQL Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore.StdRowLockSemaphore</td>
<td>INSERT_LOCK</td>
<td>&quot;INSERT INTO {0}LOCKS(SCHED_NAME, LOCK_NAME) VALUES ({1}, ?)&quot;</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore</td>
<td>INSERT_LOCK</td>
<td>&quot;INSERT INTO {0}LOCKS(SCHED_NAME, LOCK_NAME) VALUES ({1}, ?)&quot;</td>
</tr>
<tr>
<td></td>
<td>UPDATE_FOR_LOCK</td>
<td>&quot;UPDATE {0}LOCKS SET LOCK_NAME = LOCK_NAME WHERE SCHED_NAME = {1} AND LOCK_NAME = ? &quot;</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore.oracle.OracleDelegate</td>
<td>INSERT_ORACLE_CALENDAR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INSERT_ORACLE_JOB_DETAIL</td>
<td></td>
</tr>
</tbody>
</table>
public static final String SELECT_ORACLE_CALENDAR_BLOB =
"SELECT CALENDAR FROM {0}CALENDARS WHERE SCHED_NAME = CALENDAR_NAME = ?;"

public static final String SELECT_ORACLE_JOB_DETAIL_BLOB =
"SELECT JOB_DATA WHERE SCHED_NAME = {1} AND JOB_NAME = ? AND JOB_GROUP = ?;"

public static final String SELECT_ORACLE_TRIGGER_JOB_DETAIL_BLOB =
"SELECT JOB_DATA FROM {0}TRIGGERS WHERE SCHED_NAME = TRIGGER_NAME = ? AND TRIGGER_GROUP = ? FOR UPDATE;"

public static final String UPDATE_ORACLE_CALENDAR_BLOB =
{0}CALENDARS SET CALENDAR = ? SCHED_NAME = {1} AND CALENDAR_NAME = ?;

public static final String UPDATE_ORACLE_JOB_DETAIL =
{0}JOB_DETAILS SET DESCRIPTION = ?, JOB_CLASS_NAME = ?, IS_DURABLE = ?, IS_NONCONCURRENT = ?, IS_UPDATE_DATA = ?, REQUESTS_RECOVERY = ?, JOB_DATA = EMPTY_BLOB() SCHED_NAME = {1} AND JOB_NAME = ?;

public static final String UPDATE_ORACLE_JOB_DETAIL_BLOB =
{0}JOB_DETAILS SET JOB_DATA = ? SCHED_NAME = {1} AND JOB_NAME = ?;"
public static final String UPDATE_ORACLE_TRIGGER

{0}TRIGGERS SET DESCRIPTION = ?, NEXT_FIRE_TIME = ?, PREV_FIRE_TIME = ?, TRIGGER_STATE = ?, TRIGGER_TYPE = ?, START_TIME = ?, END_TIME = ?, CALENDAR_NAME = ?, MISFIRE_INSTR = ?, PRIORITY = ? WHERE SCHED_NAME = {1} AND TRIGGER_NAME = TRIGGER_GROUP = ?

public static final String UPDATE_ORACLE_TRIGGER_JOB_DETAIL_BLOB

{0}TRIGGERS SET JOB_DATA = ? WHERE SCHED_NAME = {1} AND TRIGGER_NAME = TRIGGER_GROUP = ?

public static final String UPDATE_ORACLE_TRIGGER_JOB_DETAIL_EMPTY_BLOB

{0}TRIGGERS SET EMPTY_BLOB() WHERE SCHED_NAME = {1} AND TRIGGER_NAME = TRIGGER_GROUP = ?

org.quartz.jobs.DirectoryScanJob

public static final String DIRECTORY_NAME

"DIRECTORY_NAME"

public static final String DIRECTORY_SCAN_LISTENER_NAME

"DIRECTORY_SCAN_LISTENER_NAME"

public static final String MINIMUM_UPDATE_AGE

"MINIMUM_UPDATE_AGE"

org.quartz.jobs.FileScanJob

public static final String FILE_NAME

"FILE_NAME"

public static final String FILE_SCAN_LISTENER_NAME

"FILE_SCAN_LISTENER_NAME"
<table>
<thead>
<tr>
<th>public static final String MINIMUM_UPDATE_AGE</th>
<th>&quot;MINIMUM_UPDATE_AGE&quot;</th>
</tr>
</thead>
</table>
| org.quartz.jobs.

**NativeJob**

<table>
<thead>
<tr>
<th>public static final String PROP_COMMAND</th>
<th>&quot;command&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String PROP_CONSUME_STREAMS</td>
<td>&quot;consumeStreams&quot;</td>
</tr>
<tr>
<td>public static final String PROP_PARAMETERS</td>
<td>&quot;parameters&quot;</td>
</tr>
<tr>
<td>public static final String PROP_WAIT_FOR_PROCESS</td>
<td>&quot;waitForProcess&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final String CREDENTIALS</th>
<th>&quot;java.naming.security.credentials&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String EJB_ARG_TYPES_KEY</td>
<td>&quot;java.naming.security.principal&quot;</td>
</tr>
<tr>
<td>public static final String EJB_ARGS_KEY</td>
<td>&quot;java.naming.provider.url&quot;</td>
</tr>
<tr>
<td>public static final String EJB_JNDI_NAME_KEY</td>
<td>&quot;java.naming.provider.url&quot;</td>
</tr>
<tr>
<td>public static final String EJB_METHOD_KEY</td>
<td>&quot;java.naming.provider.url&quot;</td>
</tr>
<tr>
<td>public static final String INITIAL_CONTEXT_FACTORY</td>
<td>&quot;java.naming.factory.initial&quot;</td>
</tr>
<tr>
<td>public static final String PRINCIPAL</td>
<td>&quot;java.naming.security.principal&quot;</td>
</tr>
<tr>
<td>public static final String PROVIDER_URL</td>
<td>&quot;java.naming.provider.url&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final String CREDENTIALS</th>
<th>&quot;java.naming.security.credentials&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String INITIAL_CONTEXT_FACTORY</td>
<td>&quot;java.naming.factory.initial&quot;</td>
</tr>
<tr>
<td>public static final String JMS_ACK_MODE</td>
<td>&quot;jms.acknowledge&quot;</td>
</tr>
<tr>
<td>public static final String JMS_CONNECTION_FACTORY_JNDI</td>
<td>&quot;jms.connection.factory&quot;</td>
</tr>
<tr>
<td>public static final String JMS_DESTINATION_JNDI</td>
<td>&quot;jms.destination&quot;</td>
</tr>
<tr>
<td>public static final String JMS_MSG_FACTORY_CLASS_NAME</td>
<td>&quot;jms.message.factory.class.name&quot;</td>
</tr>
<tr>
<td>public static final String JMS_PASSWORD</td>
<td>&quot;jms.password&quot;</td>
</tr>
<tr>
<td>public static final String JMS_USE_TXN</td>
<td>&quot;jms.use.transaction&quot;</td>
</tr>
<tr>
<td>public static final String JMS_USER</td>
<td>&quot;jms.user&quot;</td>
</tr>
<tr>
<td>public static final String PRINCIPAL</td>
<td>&quot;java.naming.security.principal&quot;</td>
</tr>
<tr>
<td>public static final String PROVIDER_URL</td>
<td>&quot;java.naming.provider.url&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final String PROP_CC_RECIPIENT</th>
<th>&quot;cc_recipient&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String PROP_CONTENT_TYPE</td>
<td>&quot;content_type&quot;</td>
</tr>
<tr>
<td>public static final String PROP_MESSAGE</td>
<td>&quot;message&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final String CREDENTIALS</th>
<th>&quot;java.naming.security.credentials&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final String INITIAL_CONTEXT_FACTORY</td>
<td>&quot;java.naming.factory.initial&quot;</td>
</tr>
<tr>
<td>public static final String JMS_ACK_MODE</td>
<td>&quot;jms.acknowledge&quot;</td>
</tr>
<tr>
<td>public static final String JMS_CONNECTION_FACTORY_JNDI</td>
<td>&quot;jms.connection.factory&quot;</td>
</tr>
<tr>
<td>public static final String JMS_DESTINATION_JNDI</td>
<td>&quot;jms.destination&quot;</td>
</tr>
<tr>
<td>public static final String JMS_MSG_FACTORY_CLASS_NAME</td>
<td>&quot;jms.message.factory.class.name&quot;</td>
</tr>
<tr>
<td>public static final String JMS_PASSWORD</td>
<td>&quot;jms.password&quot;</td>
</tr>
<tr>
<td>public static final String JMS_USE_TXN</td>
<td>&quot;jms.use.transaction&quot;</td>
</tr>
<tr>
<td>public static final String JMS_USER</td>
<td>&quot;jms.user&quot;</td>
</tr>
<tr>
<td>public static final String PRINCIPAL</td>
<td>&quot;java.naming.security.principal&quot;</td>
</tr>
<tr>
<td>public static final String PROVIDER_URL</td>
<td>&quot;java.naming.provider.url&quot;</td>
</tr>
<tr>
<td>public static final String PROP_RECIPIENT</td>
<td>&quot;recipient&quot;</td>
</tr>
<tr>
<td>public static final String PROP_REPLY_TO</td>
<td>&quot;reply_to&quot;</td>
</tr>
<tr>
<td>public static final String PROP_SENDER</td>
<td>&quot;sender&quot;</td>
</tr>
<tr>
<td>public static final String PROP_SMTP_HOST</td>
<td>&quot;smtp_host&quot;</td>
</tr>
<tr>
<td>public static final String PROP_SUBJECT</td>
<td>&quot;subject&quot;</td>
</tr>
</tbody>
</table>

| org.quartz.simpl.SystemPropertyInstanceIdGenerator |
| public static final String SYSTEM_PROPERTY | "org.quartz.scheduler.instanceId" |

| org.quartz.utils.DBConnectionManager |
| public static final String DB_PROPS_PREFIX | "org.quartz.db." |

| org.quartz.utils.Key<T> |
| public static final String DEFAULT_GROUP | "DEFAULT" |

| org.quartz.utils.PoolingConnectionProvider |
| public static final String DB_DRIVER | |
| public static final String DB_IDLE_VALIDATION_SECONDS | |
| public static final String DB_MAX_CACHED_STATEMENTS_PER_CONNECTION | |
| public static final String DB_MAX_CONNECTIONS | |
| public static final String DB_PASSWORD | |
| public static final String DB_URL | |
| public static final String DB_USER | |
| public static final String DB_VALIDATE_ON_CHECKOUT | |
| public static final String DB_VALIDATION_QUERY | |
| public static final int DEFAULT_DB_MAX_CACHED_STATEMENTS_PER_CONNECTION | |
| public static final int DEFAULT_DB_MAX_CONNECTIONS | |

| org.quartz.xml.XMLSchedulingDataProcessor |
| public static final String QUARTZ_NS | |
| public static final String QUARTZ_SCHEMA_WEB_URL | "scheduler.org/xml/job_scheduling_data_2_0.xsd" |
| public static final String QUARTZ_SYSTEM_ID_JAR_PREFIX | |
| public static final String QUARTZ_XML_DEFAULT_FILE_NAME | |
| public static final String QUARTZ_XSD_PATH_IN_JAR | "org/quartz/xml/job_scheduling_data_2_0.xsd" |
protected static final String XSD_DATE_FORMAT
Deprecated API

Contents

- Deprecated Interfaces
- Deprecated Classes
- Deprecated Fields
- Deprecated Methods
- Deprecated Constructors

### Deprecated Interfaces

`org.quartz.StatefulJob`  
*use DisallowConcurrentExecution and/or PersistJobDataAfterExecution annotations instead.*

### Deprecated Classes

`org.quartz.impl.jdbcjobstore.CloudscapeDelegate`  
*Use the StdJDBCDelegate for latest versions of Derby*

### Deprecated Fields

`org.quartz.impl.jdbcjobstore.Constants.STATE_MISFIRED`  
*Whether a trigger has misfired is no longer a state, but rather now identified dynamically by whether the trigger's next fire time is more than the misfire threshold time in the past.*

### Deprecated Methods

`org.quartz.utils.StringKeyDirtyFlagMap.containsTransientData()`  
*JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.*

`org.quartz.impl.DirectSchedulerFactory.createVolatileScheduler(int)`
see correctly spelled method.

```java
org.quartz.utils.StringKeyDirtyFlagMap.getAllowsTransientData()
    JDBCJobStores no longer prune out transient data. If you include non-
    Serializable values in the Map, you will now get an exception when attempting
    to store it in a database.

org.quartz.SchedulerMetaData.jobStoreSupportsPersistence()
    s

org.quartz.utils.StringKeyDirtyFlagMap.removeTransientData()
    JDBCJobStores no longer prune out transient data. If you include non-
    Serializable values in the Map, you will now get an exception when attempting
    to store it in a database.

org.quartz.impl.jdbcjobstore.StdJDBCDelegate.selectNextFireTime(Connection)
    Does not account for misfires.

org.quartz.impl.jdbcjobstore.DriverDelegate.selectNextFireTime(Connection)
    Does not account for misfires.

org.quartz.utils.StringKeyDirtyFlagMap.setAllowsTransientData(boolean)
    JDBCJobStores no longer prune out transient data. If you include non-
    Serializable values in the Map, you will now get an exception when attempting
    to store it in a database.
```

## Deprecated Constructors

```java
org.quartz.impl.triggers.CronTriggerImpl(String)
    use a TriggerBuilder instead

org.quartz.impl.triggers.CronTriggerImpl(String, String)
    use a TriggerBuilder instead

org.quartz.impl.triggers.CronTriggerImpl(String, String, String)
    use a TriggerBuilder instead

org.quartz.impl.triggers.CronTriggerImpl(String, String, String, String)
    use a TriggerBuilder instead

org.quartz.impl.triggers.CronTriggerImpl(String, String, String, String, Date, Date, String)
    use a TriggerBuilder instead

org.quartz.impl.triggers.CronTriggerImpl(String, String, String, String, Date, Date, String, TimeZone)
```
<table>
<thead>
<tr>
<th>Class</th>
<th>Use instead</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.impl.triggers.CronTriggerImpl(String, String, String, String, String)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.CronTriggerImpl(String, String, String, String, String, TimeZone)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.JobDetailImpl(String, Class)</code></td>
<td><code>JobBuilder</code></td>
</tr>
<tr>
<td><code>org.quartz.impl.JobDetailImpl(String, String, Class)</code></td>
<td><code>JobBuilder</code></td>
</tr>
<tr>
<td><code>org.quartz.impl.JobDetailImpl(String, String, Class, boolean, boolean)</code></td>
<td><code>JobBuilder</code></td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, Date)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, Date, Date, int, long)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, int, long)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, String)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, String, Date)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, String, Date, int, long)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, String, int, long)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, String, String, String, String, Date, Date, int, long)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, String, int, long)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers.SimpleTriggerImpl(String, String, String, String, Date, Date, int, long)</code></td>
<td><code>TriggerBuilder</code> instead</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
How This API Document Is Organized

This API (Application Programming Interface) document has pages corresponding to the items in the navigation bar, described as follows.

Overview

The Overview page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. This page can contain four categories:

- Interfaces (italic)
- Classes
- Enums
- Exceptions
- Errors
- Annotation Types

Class/Interface

Each class, interface, nested class and nested interface has its own separate page. Each of these pages has three sections consisting of a class/interface description, summary tables, and detailed member descriptions:

- Class inheritance diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class/interface declaration
- Class/interface description

- Nested Class Summary
- Field Summary
• Constructor Summary
• Method Summary

• Field Detail
• Constructor Detail
• Method Detail

Each summary entry contains the first sentence from the detailed description for that item. The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

**Annotation Type**

Each annotation type has its own separate page with the following sections:

• Annotation Type declaration
• Annotation Type description
• Required Element Summary
• Optional Element Summary
• Element Detail

**Enum**

Each enum has its own separate page with the following sections:

• Enum declaration
• Enum description
• Enum Constant Summary
• Enum Constant Detail

**Use**

Each documented package, class and interface has its own Use page. This page describes what packages, classes, methods, constructors and fields use any part of the given class or package. Given a class or interface A, its Use page includes subclasses of A, fields declared as A, methods that return A, and methods and constructors with parameters of type A. You can access this page by first going to the package, class or interface, then clicking on
the "Use" link in the navigation bar.

**Tree (Class Hierarchy)**

There is a [Class Hierarchy](#) page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. The classes are organized by inheritance structure starting with `java.lang.Object`. The interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on "Tree" displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking "Tree" displays the hierarchy for only that package.

**Deprecated API**

The [Deprecated API](#) page lists all of the API that have been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

**Index**

The [Index](#) contains an alphabetic list of all classes, interfaces, constructors, methods, and fields.

**Prev/Next**

These links take you to the next or previous class, interface, package, or related page.

**Frames/No Frames**

These links show and hide the HTML frames. All pages are available with or without frames.

**Serialized Form**
Each serializable or externalizable class has a description of its serialization fields and methods. This information is of interest to re-implementors, not to developers using the API. While there is no link in the navigation bar, you can get to this information by going to any serialized class and clicking "Serialized Form" in the "See also" section of the class description.

**Constant Field Values**

The [Constant Field Values](#) page lists the static final fields and their values.

*This help file applies to API documentation generated using the standard doclet.*
AbstractTrigger<T extends Trigger> - Class in org.quartz.impl.triggers
The base abstract class to be extended by all Triggers.

AbstractTrigger() - Constructor for class org.quartz.impl.triggers.AbstractTrigger
Create a Trigger with no specified name, group, or JobDetail.

AbstractTrigger(String) - Constructor for class org.quartz.impl.triggers.AbstractTrigger
Create a Trigger with the given name, and default group.

AbstractTrigger(String, String) - Constructor for class org.quartz.impl.triggers.AbstractTrigger
Create a Trigger with the given name, and group.

AbstractTrigger(String, String, String, String) - Constructor for class org.quartz.impl.triggers.AbstractTrigger
Create a Trigger with the given name, and group.

acquireNextTrigger(Connection, long, int, long) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

acquireNextTriggers(long, int, long) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Get a handle to the next N triggers to be fired, and mark them as 'reserved' by the calling scheduler.

acquireNextTriggers(long, int, long) - Method in class org.quartz.simpl.RAMJobStore
Get a handle to the next trigger to be fired, and mark it as 'reserved' by the calling scheduler.

addCalendar(String, Calendar, boolean, boolean) - Method in class org.quartz.core.QuartzScheduler
Add (register) the given Calendar to the Scheduler.

addCalendar(String, Calendar, boolean, boolean) - Method in interface org.quartz.core.RemotableQuartzScheduler

addCalendar(String, Calendar, boolean, boolean) - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.
**addCalendar(String, Calendar, boolean, boolean)** - Method in class org.quartz.impl.RemoteScheduler

Calls the equivalent method on the 'proxied' quartzScheduler.

**addCalendar(String, Calendar, boolean, boolean)** - Method in class org.quartz.impl.StdScheduler

Calls the equivalent method on the 'proxied' quartzScheduler.

**addCalendar(String, Calendar, boolean, boolean)** - Method in interface org.quartz.Scheduler

Add the given Calendar to the Scheduler.

**addConnectionProvider(String, ConnectionProvider)** - Method in class org.quartz.utils.DBConnectionManager

**addDefaultTriggerPersistenceDelegates()** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**addExcludedDate(Date)** - Method in class org.quartz.impl.calendar.HolidayCalendar

Add the given Date to the list of excluded days.

**addInternalJobListener(JobListener)** - Method in class org.quartz.core.QuartzScheduler

Add the given JobListener to the Scheduler's internal list.

**addInternalSchedulerListener(SchedulerListener)** - Method in class org.quartz.core.QuartzScheduler

Register the given SchedulerListener with the Scheduler's list of internal listeners.

**addInternalTriggerListener(TriggerListener)** - Method in class org.quartz.core.QuartzScheduler

Add the given TriggerListener to the Scheduler's internal list.

**addJob(CompositeData, boolean)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**addJob(Map<String, Object>, boolean)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

Add a durable job described by abstractJobInfo, which must contain the fully-qualified JobDetail class name under the key "jobDetailClass." That JobDetail type must contain a no-arg constructor and have public access.

**addJob(JobDetail, boolean)** - Method in class org.quartz.core.QuartzScheduler

Add the given Job to the Scheduler - with no associated Trigger.

**addJob(CompositeData, boolean)** - Method in class
addJob(Map<String, Object>, boolean) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

addJob(JobDetail, boolean) - Method in interface org.quartz.core.RemotableQuartzScheduler

addJob(JobDetail, boolean) - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

addJob(JobDetail, boolean) - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

addJob(JobDetail, boolean) - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

addJob(JobDetail, boolean) - Method in interface org.quartz.Scheduler
Add the given Job to the Scheduler - with no associated Trigger.

addJobChainLink(JobKey, JobKey) - Method in class org.quartz.listeners.JobChainingJobListener
Add a chain mapping - when the Job identified by the first key completes the job identified by the second key will be triggered.

addJobGroupToNeverDelete(String) - Method in class org.quartz.xml.XMLSchedulingDataProcessor
Add the given group to the list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

addJobListener(JobListener, Matcher<JobKey>...) - Method in class org.quartz.core.ListenerManagerImpl

addJobListener(JobListener, List<Matcher<JobKey>>) - Method in class org.quartz.core.ListenerManagerImpl

addJobListener(JobListener, Matcher<JobKey>...) - Method in interface org.quartz.ListenerManager
Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers.
Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers.

`addJobListenerMatcher(String, Matcher<JobKey>)` - Method in class `org.quartz.core.ListenerManagerImpl`

Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

`addJobToSchedule(JobDetail)` - Method in class `org.quartz.xml.XMLSchedulingDataProcessor`

Register the given SchedulerListener with the Scheduler.

`addSchedulerListener(SchedulerListener)` - Method in class `org.quartz.core.ListenerManagerImpl`

Add the given SchedulerPlugin for the QuartzScheduler to use.

`addSchedulerPlugin(SchedulerPlugin)` - Method in class `org.quartz.core.QuartzSchedulerResources`

Add the given SchedulerPlugin to the Scheduler.

`addNotificationListener(NotificationListener, NotificationFilter, Object)` - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

Add the given SchedulerPlugin for the QuartzScheduler to use.

`addNotificationListener(NotificationListener, NotificationFilter, Object)` - Method in class `org.quartz.core.QuartzSchedulerResources`

Add the given SchedulerPlugin for the QuartzScheduler to use.

`addNoGCObject(Object)` - Method in class `org.quartz.core.QuartzScheduler`

Add the given SchedulerPlugin for the QuartzScheduler to use.

`addSchedulerListener(SchedulerListener)` - Method in interface `org.quartz.ListenerManager`

Add the given SchedulerPlugin for the QuartzScheduler to use.

`addToSet(int, int, int, int)` - Method in class `org.quartz.CronExpression`

Add the given SchedulerPlugin for the QuartzScheduler to use.

`addTriggerGroupToNeverDelete(String)` - Method in class
Add the given group to the list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

`addTriggerListener(TriggerListener, Matcher<TriggerKey>...)` - Method in class `org.quartz.core.ListenerManagerImpl`

`addTriggerListener(TriggerListener, List<Matcher<TriggerKey>>)` - Method in interface `org.quartz.ListenerManager`

Add the given `TriggerListener` to the `Scheduler`, and register it to receive events for Triggers that are matched by ANY of the given Matchers.

`addTriggerListenerMatcher(String, Matcher<TriggerKey>)` - Method in interface `org.quartz.ListenerManager`

Add the given `Matcher` to the set of matchers for which the listener will receive events if ANY of the matchers match.

`addTriggerPersistenceDelegate(TriggerPersistenceDelegate)` - Method in class `org.quartz.impl.jdbcjobstore.StdJDBCDelegate`

`addTriggerToSchedule(Trigger)` - Method in class `org.quartz.xml.XMLSchedulingDataProcessor`

`addValidationException(SAXException)` - Method in class `org.quartz.xml.XMLSchedulingDataProcessor`

Add a detected validation exception.

`ALIAS_COL_NEXT_FIRE_TIME` - Static variable in interface `org.quartz.impl.jdbcjobstore.Constants`
**ALL_GROUPS_PAUSED** - Static variable in interface `org.quartz.impl.jdbcjobstore.Constants`

**ALL_SPEC** - Static variable in class `org.quartz.CronExpression`

**ALL_SPEC_INT** - Static variable in class `org.quartz.CronExpression`

**allJobs()** - Static method in class `org.quartz.impl.matchers.EverythingMatcher`
Create an `EverythingMatcher` that matches all jobs.

**allTriggers()** - Static method in class `org.quartz.impl.matchers.EverythingMatcher`
Create an `EverythingMatcher` that matches all triggers.

**and(Matcher<U>, Matcher<U>)** - Static method in class `org.quartz.impl.matchers.AndMatcher`
Create an `AndMatcher` that depends upon the result of both of the given matchers.

**AndMatcher<T extends Key>** - Class in `org.quartz.impl.matchers`
Matches using an AND operator on two Matcher operands.

**AndMatcher(Matcher<T>, Matcher<T>)** - Constructor for class `org.quartz.impl.matchers.AndMatcher`

**AnnualCalendar** - Class in `org.quartz.impl.calendar`
This implementation of the Calendar excludes a set of days of the year.

**AnnualCalendar()** - Constructor for class `org.quartz.impl.calendar.AnnualCalendar`

**AnnualCalendar(Calendar)** - Constructor for class `org.quartz.impl.calendar.AnnualCalendar`

**AnnualCalendar(TimeZone)** - Constructor for class `org.quartz.impl.calendar.AnnualCalendar`

**AnnualCalendar(Calendar, TimeZone)** - Constructor for class `org.quartz.impl.calendar.AnnualCalendar`

**applyMisfire(TriggerWrapper)** - Method in class `org.quartz.simpl.RAMJobStore`
areAllDaysExcluded() - Method in class org.quartz.impl.calendar.MonthlyCalendar
  Check if all days are excluded.
areAllDaysExcluded() - Method in class org.quartz.impl.calendar.WeeklyCalendar
  Check if all week days are excluded.
atHourMinuteAndSecond(int, int, int) - Method in class org.quartz.DateBuilder

atHourOfDay(int) - Method in class org.quartz.DateBuilder
  Set the hour (0-23) for the Date that will be built by this builder.
atLeastAvailable(int, MemoryConstraint.Unit) - Static method in class org.quartz.locality.constraint.MemoryConstraint
  Creates a MemoryConstraint for at least an amount of memory available
atMinute(int) - Method in class org.quartz.DateBuilder
  Set the minute (0-59) for the Date that will be built by this builder.
atSecond(int) - Method in class org.quartz.DateBuilder
  Set the second (0-59) for the Date that will be built by this builder, and truncate the milliseconds to 000.
AttributeRestoringConnectionInvocationHandler - Class in org.quartz.impl.jdbcjobstore
  Protects a Connection's attributes from being permanently modified.
AttributeRestoringConnectionInvocationHandler(Connection) - Constructor for class org.quartz.impl.jdbcjobstore.AttributeRestoringConnectionInvocationHandler

AUTO_GENERATE_INSTANCE_ID - Static variable in class org.quartz.impl.StdSchedulerFactory
**BaseCalendar** - Class in `org.quartz.impl.calendar`
   This implementation of the Calendar may be used (you don't have to) as a base class for more sophisticated one's.

**BaseCalendar()** - Constructor for class `org.quartz.impl.calendar<BaseCalendar`

**BaseCalendar(Calendar)** - Constructor for class `org.quartz.impl.calendar<BaseCalendar`

**BaseCalendar(TimeZone)** - Constructor for class `org.quartz.impl.calendar<BaseCalendar`

**BaseCalendar(Calendar, TimeZone)** - Constructor for class `org.quartz.impl.calendar<BaseCalendar`

**begin()** - Method in class `org.quartz.core.JobRunShell`

**begin()** - Method in class `org.quartz.ee.jta.JTAJobRunShell`

**bind(Scheduler)** - Method in class `org.quartz.impl.SchedulerRepository`

**blockedJobs** - Variable in class `org.quartz.simpl.RAMJobStore`

**blockForAvailableThreads()** - Method in class `org.quartz.simpl.SimpleThreadPool`

**blockForAvailableThreads()** - Method in class `org.quartz.simpl.ZeroSizeThreadPool`

**BroadcastJobListener** - Class in `org.quartz.listeners`
   Holds a List of references to JobListener instances and broadcasts all events to them (in order).

**BroadcastJobListener(String)** - Constructor for class `org.quartz.listeners.BroadcastJobListener`
   Construct an instance with the given name.

**BroadcastJobListener(String, List<JobListener>)** - Constructor for class
org.quartz.listeners.BroadcastJobListener
Construct an instance with the given name, and List of listeners.

BroadcastSchedulerListener - Class in org.quartz.listeners
Holds a List of references to SchedulerListener instances and broadcasts all events to them (in order).

BroadcastSchedulerListener() - Constructor for class org.quartz.listeners.BroadcastSchedulerListener

BroadcastSchedulerListener(List) - Constructor for class org.quartz.listeners.BroadcastSchedulerListener
Construct an instance with the given List of listeners.

BroadcastTriggerListener - Class in org.quartz.listeners
Holds a List of references to TriggerListener instances and broadcasts all events to them (in order).

BroadcastTriggerListener(String) - Constructor for class org.quartz.listeners.BroadcastTriggerListener
Construct an instance with the given name.

BroadcastTriggerListener(String, List) - Constructor for class org.quartz.listeners.BroadcastTriggerListener
Construct an instance with the given name, and List of listeners.

build() - Method in class org.quartz.CalendarIntervalScheduleBuilder
Build the actual Trigger -- NOT intended to be invoked by end users, but will rather be invoked by a TriggerBuilder which this ScheduleBuilder is given to.

build() - Method in class org.quartz.CronScheduleBuilder
Build the actual Trigger -- NOT intended to be invoked by end users, but will rather be invoked by a TriggerBuilder which this ScheduleBuilder is given to.

build() - Method in class org.quartz.DateBuilder
Build the Date defined by this builder instance.

build() - Method in class org.quartz.JobBuilder
Produce the JobDetail instance defined by this JobBuilder.

build() - Method in class org.quartz.locality.LocalityJobBuilder
Method building an immutable LocalityAware JobDetail, wrapping the actual JobDetail instance If a JobBuilder was used at construction time, its build method is invoked

build() - Method in class org.quartz.locality.LocalityTriggerBuilder
Method building an immutable LocalityAware Trigger, wrapping the actual Trigger instance If a TriggerBuilder was used at construction time, its build
method is invoked

**build()** - Method in class org.quartz.locality.NodeSpecBuilder
Method that creates the NodeSpec represented by this builder

**build()** - Method in class org.quartz.ScheduleBuilder

**build()** - Method in class org.quartz.SimpleScheduleBuilder
Build the actual Trigger -- NOT intended to be invoked by end users, but will rather be invoked by a TriggerBuilder which this ScheduleBuilder is given to.

**build()** - Method in class org.quartz.TriggerBuilder
Produce the Trigger.

**buildExpression(String)** - Method in class org.quartz.CronExpression
calcFailedIfAfter(SchedulerStateRecord) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Calendar - Interface in org.quartz
   An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.
calendarCache - Variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

calendarExists(Connection, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
   Check whether or not a calendar exists.
calendarExists(Connection, String) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

calendarExists(Connection, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   Check whether or not a calendar exists.
calendarIntervalSchedule() - Static method in class org.quartz.CalendarIntervalScheduleBuilder
   Create a CalendarIntervalScheduleBuilder.
CalendarIntervalScheduleBuilder - Class in org.quartz
   CalendarIntervalScheduleBuilder is a ScheduleBuilder that defines calendar time (day, week, month, year) interval-based schedules for Triggers.
CalendarIntervalTrigger - Interface in org.quartz
   A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.
CalendarIntervalTriggerImpl - Class in org.quartz.impl.triggers
   A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.
CalendarIntervalTriggerImpl() - Constructor for class org.quartz.impl.triggers.CalendarIntervalTriggerImpl
   Create a DateIntervalTrigger with no settings.
CalendarIntervalTriggerImpl(String, DateBuilder.IntervalUnit, int) - Constructor for class org.quartz.impl.triggers.CalendarIntervalTriggerImpl
Create a `DateIntervalTrigger` that will occur immediately, and repeat at the given interval.

`CalendarIntervalTriggerImpl(String, String, DateBuilder.IntervalUnit, int)` - Constructor for class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`

Create a `DateIntervalTrigger` that will occur immediately, and repeat at the given interval.

`CalendarIntervalTriggerImpl(String, Date, Date, DateBuilder.IntervalUnit, int)` - Constructor for class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`

Create a `DateIntervalTrigger` that will occur at the given time, and repeat at the given interval until the given end time.

`CalendarIntervalTriggerImpl(String, String, Date, Date, DateBuilder.IntervalUnit, int)` - Constructor for class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`

Create a `DateIntervalTrigger` that will occur at the given time, and repeat at the given interval until the given end time.

`CalendarIntervalTriggerImpl(String, String, String, String, Date, Date, DateBuilder.IntervalUnit, int)` - Constructor for class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`

Create a `DateIntervalTrigger` that will occur at the given time, fire the identified `Job` and repeat at the given interval until the given end time.

`CalendarIntervalTriggerPersistenceDelegate` - Class in `org.quartz.impl.jdbcjobstore`

`CalendarIntervalTriggerPersistenceDelegate()` - Constructor for class `org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate`

`calendarIsReferenced(Connection, String)` - Method in interface `org.quartz.impl.jdbcjobstore.DriverDelegate`

Check whether or not a calendar is referenced by any triggers.

`calendarIsReferenced(Connection, String)` - Method in class `org.quartz.impl.jdbcjobstore.StdJDBCDelegate`

Check whether or not a calendar is referenced by any triggers.

`calendarsByName` - Variable in class `org.quartz.simpl.RAMJobStore`

`cancelMonitoring()` - Method in class `org.quartz.locality.constraint.evaluator.EvaluatorTimerTask`

Permanently cancels the timer backing this instance.

`canHandleTriggerType(OperableTrigger)` - Method in class `org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate`
canHandleTriggerType(OperableTrigger) - Method in class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

canHandleTriggerType(OperableTrigger) - Method in class org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

canHandleTriggerType(OperableTrigger) - Method in interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate

canUseProperties() - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Get whether String-only properties will be handled in JobDataMaps.

canUseProperties() - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

CascadingClassLoadHelper - Class in org.quartz.simpl
  A ClassLoadHelper uses all of the ClassLoadHelper types that are found in this package in its attempts to load a class, when one scheme is found to work, it is promoted to the scheme that will be used first the next time a class is loaded (in order to improve performance).

CascadingClassLoadHelper() - Constructor for class org.quartz.simpl.CascadingClassLoadHelper

checkBlockedState(Connection, JobKey, String) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Determines if a Trigger for the given job should be blocked.

checkExists(JobKey) - Method in class org.quartz.core.QuartzScheduler
  Determine whether a Job with the given identifier already exists within the scheduler.

checkExists(TriggerKey) - Method in class org.quartz.core.QuartzScheduler
  Determine whether a Trigger with the given identifier already exists within the scheduler.

checkExists(JobKey) - Method in interface org.quartz.core.RemotableQuartzScheduler

checkExists(TriggerKey) - Method in interface org.quartz.core.RemotableQuartzScheduler
**checkExists(JobKey)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Determine whether a Job with the given identifier already exists within the scheduler.

**checkExists(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Determine whether a Job with the given identifier already exists within the scheduler.

**checkExists(TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Determine whether a Trigger with the given identifier already exists within the scheduler.

**checkExists(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Determine whether a Trigger with the given identifier already exists within the scheduler.

**checkExists(JobKey)** - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**checkExists(TriggerKey)** - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**checkExists(JobKey)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**checkExists(TriggerKey)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**checkExists(JobKey)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**checkExists(TriggerKey)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**checkExists(JobKey)** - Method in interface org.quartz.Scheduler
Determine whether a Job with the given identifier already exists within the scheduler.

**checkExists(TriggerKey)** - Method in interface org.quartz.Scheduler
Determine whether a Trigger with the given identifier already exists within the scheduler.

**checkExists(JobKey)** - Method in class org.quartz.simpl.RAMJobStore
Determine whether a Job with the given identifier already exists within the scheduler.

**checkExists(TriggerKey)** - Method in class org.quartz.simpl.RAMJobStore
Determine whether a Trigger with the given identifier already exists within the scheduler.
the scheduler.

**checkForUpdate()** - Method in class org.quartz.utils.UpdateChecker

This method ensures that there will be no exception thrown.

**checkNext(int, String, int, int)** - Method in class org.quartz.CronExpression

**CircularLossyQueue<T>** - Class in org.quartz.util

An implementation of a CircularQueue data-structure.

**CircularLossyQueue(int)** - Constructor for class org.quartz.util.CircularLossyQueue

Constructs the circular queue with the specified capacity

**classLoadHelper** - Variable in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**classLoadHelper** - Variable in class org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin

**classLoadHelper** - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

**ClassUtils** - Class in org.quartz.util

**ClassUtils()** - Constructor for class org.quartz.util.ClassUtils

**cleanupConnection(Connection)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Clean up the given database connection.

**clear()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**clear()** - Method in class org.quartz.core.QuartzScheduler

Clears (deletes!) all scheduling data - all Jobs, Triggers, Calendars.

**clear()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**clear()** - Method in interface org.quartz.core.RemotableQuartzScheduler

**clear()** - Method in class org.quartz.impl.RemoteMBeanScheduler

**clear()** - Method in class org.quartz.impl.RemoteScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.

**clear()** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**clear()** - Method in interface org.quartz.Scheduler
Clears (deletes!) all scheduling data - all Jobs, Triggers Calendars.

**clear()** - Method in class org.quartz.utils.DirtyFlagMap

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**clearAllSchedulingData(Connection)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearAllSchedulingData()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**clearFromBusyWorkersList(SimpleThreadPool.WorkerThread)** - Method in class org.quartz.simpl.SimpleThreadPool
Clear the 'dirty' flag (set dirty flag to false).

**clearStatistics()** - Method in class org.quartz.core.SampledStatisticsImpl
Clears the collected statistics.

**clearValidationExceptions()** - Method in class org.quartz.xml.XMLSchedulingDataProcessor
Resets the number of detected validation exceptions.

**clone()** - Method in interface org.quartz.Calendar

**clone()** - Method in class org.quartz.CronExpression
clone() - Method in class org.quartz.impl.calendar.**AnnualCalendar**

clone() - Method in class org.quartz.impl.calendar.**BaseCalendar**

clone() - Method in class org.quartz.impl.calendar.**CronCalendar**

clone() - Method in class org.quartz.impl.calendar.**DailyCalendar**

clone() - Method in class org.quartz.impl.calendar.**HolidayCalendar**

clone() - Method in class org.quartz.impl.calendar.**MonthlyCalendar**

clone() - Method in class org.quartz.impl.calendar.**WeeklyCalendar**

clone() - Method in class org.quartz.impl.**JobDetailImpl**

clone() - Method in class org.quartz.impl.triggers.**AbstractTrigger**

clone() - Method in class org.quartz.impl.triggers.**CronTriggerImpl**

clone() - Method in interface org.quartz.**JobDetail**

clone() - Method in class org.quartz.locality.**DelegatingLocalityJobDetail**

clone() - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

clone() - Method in class org.quartz.utils.**DirtyFlagMap**

close() - Method in class org.quartz.impl.jdbcjobstore.**AttributeRestoringConnectionInvocationHandler**

Attempts to restore the auto commit and transaction isolation connection attributes of the wrapped connection to their original values (if they were overwritten), before finally actually closing the wrapped connection.

closeConnection(Connection) - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

Closes the supplied Connection.

closeResource(Object) - Static method in class org.quartz.jobs.ee.jms.**JmsHelper**

Closes a resource that has a close() method.
**closeResultSet(ResultSet)** - Static method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Cleanup helper method that closes the given ResultSet while ignoring any errors.

**closeResultSet(ResultSet)** - Static method in class org.quartz.impl.jdbcjobstore.Util
Cleanup helper method that closes the given ResultSet while ignoring any errors.

**closeStatement(Statement)** - Static method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Cleanup helper method that closes the given Statement while ignoring any errors.

**closeStatement(Statement)** - Static method in class org.quartz.impl.jdbcjobstore.Util
Cleanup helper method that closes the given Statement while ignoring any errors.

**CloudscapeDelegate** - Class in org.quartz.impl.jdbcjobstore
Deprecated. *Use the StdJDBCDelegate for latest versions of Derby*

**CloudscapeDelegate(Logger, String, String, String, ClassLoadHelper)** - Constructor for class org.quartz.impl.jdbcjobstore.CloudscapeDelegate
Deprecated. Create new CloudscapeDelegate instance.

**CloudscapeDelegate(Logger, String, String, String, ClassLoadHelper, Boolean)** - Constructor for class org.quartz.impl.jdbcjobstore.CloudscapeDelegate
Deprecated. Create new CloudscapeDelegate instance.

**clusterCheckIn(Connection)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**clusterRecover(Connection, List<SchedulerStateRecord>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**COL_BLOB** - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

**COL_BOOL_PROP_1** - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**COL_BOOL_PROP_2** - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport
**COL_CALENDAR** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_CALENDAR_NAME** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_CHECKIN_INTERVAL** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_CRON_EXPRESSION** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_DEC_PROP_1** - Static variable in class org.quartz.impl.jdbcjobstore.**SimplePropertiesTriggerPersistenceDelegateSupport**

**COL_DEC_PROP_2** - Static variable in class org.quartz.impl.jdbcjobstore.**SimplePropertiesTriggerPersistenceDelegateSupport**

**COL_DESCRIPTION** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_END_TIME** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_ENTRY_ID** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COLENTRYSTATE** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL FIRED_TIME** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_INSTANCE_NAME** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_INT_PROP_1** - Static variable in class org.quartz.impl.jdbcjobstore.**SimplePropertiesTriggerPersistenceDelegateSupport**
**COL_INT_PROP_2** - Static variable in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**COL_IS_DURABLE** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_IS_NONCONCURRENT** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_IS_UPDATE_DATA** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_IS_VOLATILE** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_JOB_CLASS** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_JOB_DATAMAP** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_JOB_GROUP** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_JOB_NAME** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_LAST_CHECKIN_TIME** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_LOCK_NAME** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**COL_LONG_PROP_1** - Static variable in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**COL_LONG_PROP_2** - Static variable in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport
**COL_MISFIRE_INSTRUCTION** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_NEXT_FIRE_TIME** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_PREV_FIRE_TIME** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_PRIORITY** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_REPEAT_COUNT** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_REPEAT_INTERVAL** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_REQUESTS_RECOVERY** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_SCHEDULER_NAME** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_START_TIME** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts

**COL_STR_PROP_1** - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**COL_STR_PROP_2** - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**COL_STR_PROP_3** - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**COL_TIME_ZONE_ID** - Static variable in interface org.quartz.impl.jdbcjobstore.Consts
**COL_TIMES_TRIGGERED** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_TRIGGER_GROUP** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_TRIGGER_NAME** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_TRIGGER_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**COL_TRIGGER_TYPE** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**commitConnection(Connection)** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

  Commit the supplied connection

**compare(Trigger, Trigger)** - Method in class org.quartz.**Trigger.TriggerTimeComparator**

**compareTo** - Variable in class org.quartz.impl.matchers.**KeyMatcher**

**compareTo** - Variable in class org.quartz.impl.matchers.**StringMatcher**

**compareTo(Trigger)** - Method in class org.quartz.impl.triggers.**AbstractTrigger**

  Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e.

**compareTo(Trigger)** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

  Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e.

**compareTo(Trigger)** - Method in interface org.quartz.**Trigger**

  Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e.

**compareTo(Key)** - Method in class org.quartz.utils.**Key**

**compareWith** - Variable in class org.quartz.impl.matchers.**StringMatcher**
**complete(boolean)** - Method in class org.quartz.core.JobRunShell

**complete(boolean)** - Method in class org.quartz.ee.jta.JTAJobRunShell

**completeTriggerRetryLoop(OperableTrigger, JobDetail, Trigger.CompletedExecutionInstruction)** - Method in class org.quartz.core.JobRunShell

**computeEndTimeToAllowParticularNumberOfFirings(OperableTrigger, Calendar, int)** - Static method in class org.quartz.TriggerUtils

  Compute the date that is 1 second after the Nth firing of the given Trigger, taking the trigger's associated Calendar into consideration.

**computeFireTimes(OperableTrigger, Calendar, int)** - Static method in class org.quartz.TriggerUtils

  Returns a list of Dates that are the next fire times of a Trigger.

**computeFireTimesBetween(OperableTrigger, Calendar, Date, Date)** - Static method in class org.quartz.TriggerUtils

  Returns a list of Dates that are the next fire times of a Trigger that fall within the given date range.

**computeFirstFireTime(Calendar)** - Method in class org.quartz.impl.triggers.AbstractTrigger

  This method should not be used by the Quartz client.

**computeFirstFireTime(Calendar)** - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl

  Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.

**computeFirstFireTime(Calendar)** - Method in class org.quartz.impl.triggers.CronTriggerImpl

  Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.

**computeFirstFireTime(Calendar)** - Method in class org.quartz.impl.triggers.SimpleTriggerImpl

  Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.

**computeFirstFireTime(Calendar)** - Method in class
org.quartz.locality. DelegatingLocalityTrigger

computeNumTimesFiredBetween(Date, Date) - Method in class org.quartz.impl.triggers. SimpleTriggerImpl

ConnectionProvider - Interface in org.quartz.utils
   Implementations of this interface used by DBConnectionManager to provide connections from various sources.

Constants - Interface in org.quartz.impl.jdbcjobstore
   This interface can be implemented by any DriverDelegate class that needs to use the constants contained herein.

Constraint<OP extends Enum,T> - Interface in org.quartz.locality.constraint
   A constraint about the node on which a job will be executed. A constraint always needs a matching Evaluator type present on the classpath.

containsKey(Object) - Method in class org.quartz.utils. DirtyFlagMap

containsTransientData() - Method in class org.quartz.utils. StringKeyDirtyFlagMap
   Deprecated. JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

containsValue(Object) - Method in class org.quartz.utils. DirtyFlagMap

contextDestroyed(ServletContextEvent) - Method in class org.quartz.ee.servlet. QuartzInitializerListener

contextInitialized(ServletContextEvent) - Method in class org.quartz.ee.servlet. QuartzInitializerListener

convertFromProperty(Properties) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   convert the JobDataMap into a list of properties

convertToProperty(Map<?, ?>) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   convert the JobDataMap into a list of properties

coresAtLeast(int) - Static method in class org.quartz.locality.constraint. CpuConstraint
   Creates a constraint that requires the node to have a least an amount of core

CoreTrigger - Interface in org.quartz.impl.triggers
internal interface preserved for backward compatibility

**COUNT_MISFIRED_TRIGGERS_IN_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**Counter** - Interface in org.quartz.utils.counter
- A simple counter

**CounterConfig** - Class in org.quartz.utils.counter
- Config for a simple Counter

**CounterConfig(long)** - Constructor for class org.quartz.utils.counter.CounterConfig
- Creates a config with the initial value

**CounterImpl** - Class in org.quartz.utils.counter
- A simple counter implementation

**CounterImpl()** - Constructor for class org.quartz.utils.counter.CounterImpl
- Default Constructor

**CounterImpl(long)** - Constructor for class org.quartz.utils.counter.CounterImpl
- Constructor with initial value

**CounterManager** - Interface in org.quartz.utils.counter
- A Counter Manager that accepts a config to create counters.

**CounterManagerImpl** - Class in org.quartz.utils.counter
- An implementation of a CounterManager.

**CounterManagerImpl(Timer)** - Constructor for class org.quartz.utils.counter.CounterManagerImpl
- Constructor that accepts a timer that will be used for scheduling sampled counter if any is created

**countMisfiredTriggersInState(Connection, String, long)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
- Get the number of triggers in the given state that have misfired - according to the given timestamp.

**countMisfiredTriggersInState(Connection, String, long)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
- Get the number of triggers in the given states that have misfired - according to the given timestamp.

**CpuConstraint** - Class in org.quartz.locality.constraint
- Cpu based constraint for best node evaluation

**CpuConstraint(CpuConstraint.Operator, int)** - Constructor for class org.quartz.locality.constraint.CpuConstraint

**CpuConstraint.Operator** - Enum in org.quartz.locality.constraint
**CpuEvaluator** - Class in org.quartz.locality.constraint.evaluator

**CpuEvaluator(Integer)** - Constructor for class org.quartz.locality.constraint.evaluator.CpuEvaluator

**CREATE_REGISTRY_ALWAYS** - Static variable in class org.quartz.core.QuartzSchedulerResources

**CREATE_REGISTRY_AS_NEEDED** - Static variable in class org.quartz.core.QuartzSchedulerResources

**CREATE_REGISTRY_NEVER** - Static variable in class org.quartz.core.QuartzSchedulerResources

**createCounter()** - Method in class org.quartz.utils.counter.CounterConfig
  Creates and returns a Counter based on the initial value

**createCounter(CounterConfig)** - Method in interface org.quartz.utils.counter.CounterManager
  Creates a Counter based on the passed config

**createCounter(CounterConfig)** - Method in class org.quartz.utils.counter.CounterManagerImpl
  Creates a Counter based on the passed config

**createCounter()** - Method in class org.quartz.utils.counter.sampled.SampledCounterConfig
  Creates and returns a Counter based on the initial value

**createCounter()** - Method in class org.quartz.utils.counter.sampled.SampledRateCounterConfig
  Creates and returns a Counter based on the initial value

**createJavaCalendar(long)** - Method in class org.quartz.impl.calendar.BaseCalendar
  Build a Calendar for the given time Stamp.

**createJavaCalendar()** - Method in class org.quartz.impl.calendar.BaseCalendar
  Build a Calendar with the current time.

**createJobRunShell(TriggerFiredBundle)** - Method in interface org.quartz.core.JobRunShellFactory
  Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

**createJobRunShell(TriggerFiredBundle)** - Method in class org.quartz.ee.jta.JTAAnnotationAwareJobRunShellFactory
Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

**createJobRunShell(TriggerFiredBundle)** - Method in class org.quartz.ee.jta.JTAJobRunShellFactory

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

**createJobRunShell(TriggerFiredBundle)** - Method in class org.quartz.impl.StdJobRunShellFactory

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

**createMailInfo()** - Method in class org.quartz.jobs.ee.mail.SendMailJob

**createMessage(JobDataMap, Session)** - Method in interface org.quartz.jobs.ee.jms.JmsMessageFactory

Creates a javax.jms.Message.

**createRemoteScheduler(String, int)** - Method in class org.quartz.impl.DirectSchedulerFactory

Creates a proxy to a remote scheduler.

**createRemoteScheduler(String, String, String, int)** - Method in class org.quartz.impl.DirectSchedulerFactory

Same as DirectSchedulerFactory.createRemoteScheduler(String rmiHost, int rmiPort), with the addition of specifying the scheduler name and instance ID.

**createRemoteScheduler(String, String, String, String, int)** - Method in class org.quartz.impl.DirectSchedulerFactory

Same as DirectSchedulerFactory.createRemoteScheduler(String rmiHost, int rmiPort), with the addition of specifying the scheduler name, instance ID, and rmi bind name.

**createScheduler(ThreadPool, JobStore)** - Method in class org.quartz.impl.DirectSchedulerFactory

Creates a scheduler using the specified thread pool and job store.

**createScheduler(String, String, ThreadPool, JobStore)** - Method in class org.quartz.impl.DirectSchedulerFactory

Same as DirectSchedulerFactory.createScheduler(ThreadPool threadPool, JobStore jobStore), with the addition of specifying the scheduler name and instance ID.

**createScheduler(String, String, ThreadPool, JobStore, String, int, long, long)** - Method in class org.quartz.impl.DirectSchedulerFactory

Creates a scheduler using the specified thread pool and job store and binds it to RMI.

**createScheduler(String, String, ThreadPool, JobStore, Map, String, int, long, long, boolean, String)** - Method in class
org.quartz.impl.**DirectSchedulerFactory**

   Creates a scheduler using the specified thread pool, job store, and plugins, and binds it to RMI.

**createService()** - Method in class org.quartz.ee.jmx.jboss.**QuartzService**

**createUniqueName(String)** - Static method in class org.quartz.utils.**Key**

**createVolatileScheduler(int)** - Method in class org.quartz.impl.**DirectSchedulerFactory**

   **Deprecated. see correctly spelled method.**

**createVolatileScheduler(int)** - Method in class org.quartz.impl.**DirectSchedulerFactory**

   Creates an in memory job store (**RAMJobStore**) The thread priority is set to Thread.NORM_PRIORITY

**createWorkerThreads(int)** - Method in class org.quartz.simpl.**SimpleThreadPool**

**CREDENTIALS** - Static variable in class org.quartz.jobs.ee.ejb.**EJBInvokerJob**

**CREDENTIALS** - Static variable in class org.quartz.jobs.ee.jms.**JmsHelper**

**CronCalendar** - Class in org.quartz.impl.calendar

   This implementation of the Calendar excludes the set of times expressed by a given **CronExpression**.

**CronCalendar(String)** - Constructor for class org.quartz.impl.calendar.**CronCalendar**

   Create a CronCalendar with the given cron expression and no baseCalendar.

**CronCalendar(Calendar, String)** - Constructor for class org.quartz.impl.calendar.**CronCalendar**

   Create a CronCalendar with the given cron expression and baseCalendar.

**CronCalendar(Calendar, String, TimeZone)** - Constructor for class org.quartz.impl.calendar.**CronCalendar**

   Create a CronCalendar with the given cron expression, baseCalendar, and TimeZone.

**CronExpression** - Class in org.quartz

   Provides a parser and evaluator for unix-like cron expressions.

**CronExpression(String)** - Constructor for class org.quartz.**CronExpression**

   Constructs a new CronExpression based on the specified parameter.
cronSchedule(String) - Static method in class org.quartz.CronScheduleBuilder
   Create a CronScheduleBuilder with the given cron-expression.

CronScheduleBuilder - Class in org.quartz
   CronScheduleBuilder is a ScheduleBuilder that defines CronExpression-based schedules for Triggers.

CronTrigger - Interface in org.quartz
   The public interface for inspecting settings specific to a CronTrigger,.

CronTriggerImpl - Class in org.quartz.impl.triggers
   A concrete Trigger that is used to fire a JobDetail at given moments in
time, defined with Unix 'cron-like' definitions.

CronTriggerImpl() - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Create a CronTrigger with no settings.

CronTriggerImpl(String) - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Deprecated. use a TriggerBuilder instead

CronTriggerImpl(String, String) - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Deprecated. use a TriggerBuilder instead

CronTriggerImpl(String, String, String) - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Deprecated. use a TriggerBuilder instead

CronTriggerImpl(String, String, String, String) - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Deprecated. use a TriggerBuilder instead

CronTriggerImpl(String, String, String, String, String, TimeZone) - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Deprecated. use a TriggerBuilder instead

CronTriggerImpl(String, String, String, String, Date, Date, String) - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Deprecated. use a TriggerBuilder instead

CronTriggerImpl(String, String, String, String, Date, Date, String, TimeZone) - Constructor for class
   org.quartz.impl.triggers.CronTriggerImpl
      Deprecated. use a TriggerBuilder instead

CronTriggerPersistenceDelegate - Class in org.quartz.impl.jdbctimer
**CronTriggerPersistenceDelegate()** - Constructor for class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

**CronTriggerSupport** - Class in org.quartz.core.jmx

**CronTriggerSupport()** - Constructor for class org.quartz.core.jmx.CronTriggerSupport

**current()** - Static method in enum org.quartz.locality.constraint.OsConstraint.OS
**dailyAtHourAndMinute(int, int)** - Static method in class org.quartz.CronScheduleBuilder
Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire every day at the given time (hour and minute).

**DailyCalendar** - Class in org.quartz.impl.calendar
This implementation of the Calendar excludes (or includes - see below) a specified time range each day.

**DailyCalendar(String, String)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified strings and no baseCalendar.

**DailyCalendar(Calendar, String, String)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified strings and the specified baseCalendar.

**DailyCalendar(int, int, int, int, int, int, int)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified values and no baseCalendar.

**DailyCalendar(Calendar, int, int, int, int, int, int, int, int)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.

**DailyCalendar(Calendar, Calendar)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified java.util.Calendar and no baseCalendar.

**DailyCalendar(Calendar, Calendar, Calendar)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified java.util.Calendar and the specified baseCalendar.

**DailyCalendar(long, long)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified values and no baseCalendar.
**DailyCalendar(Calendar, long, long)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.

**DailyCalendar(TimeZone, long, long)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified values and no baseCalendar.

**DailyCalendar(Calendar, TimeZone, long, long)** - Constructor for class org.quartz.impl.calendar.DailyCalendar
Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.

**DateBuilder** - Class in org.quartz
DateBuilder is used to conveniently create java.util.Date instances that meet particular criteria.

**DateBuilder.IntervalUnit** - Enum in org.quartz

**dateFormat** - Static variable in class org.quartz.xml.XMLSchedulingDataProcessor

**dateOf(int, int, int)** - Static method in class org.quartz.DateBuilder
Get a Date object that represents the given time, on today's date (equivalent to DateBuilder.todayAt(int, int, int)).

**dateOf(int, int, int, int)** - Static method in class org.quartz.DateBuilder
Get a Date object that represents the given time, on the given date.

**dateOf(int, int, int, int, int)** - Static method in class org.quartz.DateBuilder
Get a Date object that represents the given time, on the given date.

**DAY_OF_MONTH** - Static variable in class org.quartz.CronExpression

**DAY_OF_WEEK** - Static variable in class org.quartz.CronExpression

**dayMap** - Static variable in class org.quartz.CronExpression

**daysOfMonth** - Variable in class org.quartz.CronExpression

**daysOfWeek** - Variable in class org.quartz.CronExpression

**DB2v6Delegate** - Class in org.quartz.impl.jdbcjobstore
Quartz JDBC delegate for DB2 v6 databases.
**DB2v6Delegate***(Logger, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.DB2v6Delegate

**DB2v6Delegate***(Logger, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore.DB2v6Delegate

**DB2v7Delegate** - Class in org.quartz.impl.jdbcjobstore

  Quartz JDBC delegate for DB2 v7 databases.

**DB2v7Delegate***(Logger, String, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.DB2v7Delegate

**DB2v7Delegate***(Logger, String, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore.DB2v7Delegate

**DB2v8Delegate** - Class in org.quartz.impl.jdbcjobstore

  Quartz JDBC delegate for DB2 v8 databases.

**DB2v8Delegate***(Logger, String, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.DB2v8Delegate

**DB2v8Delegate***(Logger, String, String, String, String, Boolean, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.DB2v8Delegate

**DB DRIVER** - Static variable in class org.quartz.utils.PoolingConnectionProvider

  The JDBC database driver.

**DB_IDLE_VALIDATION_SECONDS** - Static variable in class org.quartz.utils.PoolingConnectionProvider

  The number of seconds between tests of idle connections - only enabled if the validation query property is set.

**DB_MAX_CACHED_STATEMENTS_PER_CONNECTION** - Static variable in class org.quartz.utils.PoolingConnectionProvider

  The maximum number of prepared statements that will be cached per connection in the pool.

**DB_MAX_CONNECTIONS** - Static variable in class org.quartz.utils.PoolingConnectionProvider

  The maximum number of database connections to have in the pool.

**DB_PASSWORD** - Static variable in class org.quartz.utils.PoolingConnectionProvider

  The database user password.
**DB_PROPS_PREFIX** - Static variable in class org.quartz.utils.DBConnectionManager

**DB_URL** - Static variable in class org.quartz.utils.PoolingConnectionProvider
The JDBC database URL.

**DB_USER** - Static variable in class org.quartz.utils.PoolingConnectionProvider
The database user name.

**DB_VALIDATE_ON_CHECKOUT** - Static variable in class org.quartz.utils.PoolingConnectionProvider
Whether the database sql query to validate connections should be executed every time a connection is retrieved from the pool to ensure that it is still valid.

**DB_VALIDATION_QUERY** - Static variable in class org.quartz.utils.PoolingConnectionProvider
The database sql query to execute every time a connection is returned to the pool to ensure that it is still valid.

**DBConnectionManager** - Class in org.quartz.utils
Manages a collection of ConnectionProviders, and provides transparent access to their connections.

**DBSemaphore** - Class in org.quartz.impl.jdbcjobstore
Base class for database based lock handlers for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.

**DBSemaphore(String, String, String, String)** - Constructor for class org.quartz.impl.jdbcjobstore.DBSemaphore

**decrement()** - Method in interface org.quartz.utils.counter.Counter
Decrement the counter by 1

**decrement(long)** - Method in interface org.quartz.utils.counter.Counter
Decrement the counter by given amount

**decrement()** - Method in class org.quartz.utils.counter.CounterImpl
Decrement the counter by 1

**decrement(long)** - Method in class org.quartz.utils.counter.CounterImpl
Decrement the counter by given amount

**decrement(long, long)** - Method in interface org.quartz.utils.counter.sampled.SampledRateCounter
Decrements the numerator and denominator by the passed values

**decrement(long, long)** - Method in class org.quartz.utils.counter.sampled.SampledRateCounterImpl
Decrement the numerator and denominator by the passed values

**decrement()** - Method in class
org.quartz.utils.counter.sampled.SampledRateCounterImpl
  throws UnsupportedOperationException

**decrement(long)** - Method in class
org.quartz.utils.counter.sampled.SampledRateCounterImpl
  throws UnsupportedOperationException

**DEFAULT_DB_MAX_CACHED_STATEMENTS_PER_CONNECTION** - Static variable in class org.quartz.utils.PoolingConnectionProvider
  Default maximum number of database connections in the pool.

**DEFAULT_DB_MAX_CONNECTIONS** - Static variable in class org.quartz.utils.PoolingConnectionProvider
  Default maximum number of database connections in the pool.

**DEFAULT_FAIL_OVER_GROUP** - Static variable in interface org.quartz.Scheduler
  A constant Trigger group name used internally by the scheduler - clients should not use the value of this constant ("FAILED_OVER_JOBS") for the name of a Trigger's group.

**DEFAULT_GROUP** - Static variable in interface org.quartz.Scheduler
  A (possibly) useful constant that can be used for specifying the group that Job and Trigger instances belong to.

**DEFAULT_GROUP** - Static variable in class org.quartz.util.Key
  The default group for scheduling entities, with the value "DEFAULT".

**DEFAULT_INSTANCE_ID** - Static variable in class org.quartz.impl.DirectSchedulerFactory

**DEFAULT_INSTANCE_ID** - Static variable in class org.quartz.impl.StdSchedulerFactory

**DEFAULT_PRIORITY** - Static variable in interface org.quartz.Trigger
  The default value for priority.

**DEFAULT_RECOVERY_GROUP** - Static variable in interface org.quartz.Scheduler
  A constant Trigger group name used internally by the scheduler - clients should not use the value of this constant ("RECOVERING_JOBS") for the name of a Trigger's group.

**DEFAULT_SCHEDULER_NAME** - Static variable in class org.quartz.impl.DirectSchedulerFactory
**DEFAULT_TABLE_PREFIX** - Static variable in interface org.quartz.impl.jdbcjobstore.**Constants**

**DEFAULT_TRANSACTION_MANAGER_LOCATION** - Static variable in class org.quartz.impl.jdbcjobstore.**JTANonClusteredSemaphore**

**DEFAULT_USER_TX_LOCATION** - Static variable in class org.quartz.ee.jta.**UserTransactionHelper**

**delegateClass** - Variable in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**delegateClassName** - Variable in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**delegateInitString** - Variable in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**DelegatingLocalityJobDetail** - Class in org.quartz locality. Wrapping a JobDetail instance while adding the **LocalityAware** contract All JobDetail method calls will be delegated to the wrapped JobDetail instance

**DelegatingLocalityJobDetail(JobDetail, NodeSpec)** - Constructor for class org.quartz.locality.**DelegatingLocalityJobDetail** Constructs a **LocalityAware** JobDetail, wrapping an existing JobDetail instance, with additional NodeSpec

**DelegatingLocalityTrigger** - Class in org.quartz locality. Wrapping a Trigger instance while adding the **LocalityAware** contract All Trigger method calls will be delegated to the wrapped Trigger instance

**DelegatingLocalityTrigger(Trigger, NodeSpec)** - Constructor for class org.quartz.locality.**DelegatingLocalityTrigger** Constructs a **LocalityAware** Trigger, wrapping an existing Trigger instance, with additional NodeSpec

**DELETE_ALL_BLOB_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCCConstants**

**DELETE_ALL_CALENDARS** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCCConstants**

**DELETE_ALL_CRON_TRIGGERS** - Static variable in interface
org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_ALL_JOBDETAILS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_ALL_PAUSED_TRIGGER_GRPS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_ALL_SIMPLE_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_ALL_SIMPROP_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_ALL_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_BLOB_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETECALENDAR** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_CRON_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_FIRED_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_FIRED_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_INSTANCES_FIRED_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_JOBDETAIL** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_NO_RECOVERY_FIRED_TRIGGERS** - Static variable in
interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_PAUSED_TRIGGER_GROUP** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_PAUSED_TRIGGER_GROUPS** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_SCHEDULER_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_SIMPLE_PROPS_TRIGGER** - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**DELETE_SIMPLE_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**DELETE_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

deleteAllPausedTriggerGroups(Connection) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

deleteAllPausedTriggerGroups(Connection) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

deleteBlobTrigger(Connection, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  
  Delete the cron trigger data for a trigger.

deleteCalendar(String) - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

deleteCalendar(String) - Method in class org.quartz.core.QuartzScheduler
  
  Delete the identified Calendar from the Scheduler.

deleteCalendar(String) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

deleteCalendar(String) - Method in interface org.quartz.core.Remota bleQuartzScheduler
deleteCalendar(Connection, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
  Delete a calendar.
deleteCalendar(Connection, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  Delete a calendar.
deleteCalendar(String) - Method in class org.quartz.impl.RemoteMBeanScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.
deleteCalendar(String) - Method in class org.quartz.impl.RemoteScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler.
deleteCalendar(String) - Method in class org.quartz.impl.StdScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler.
deleteCalendar(String) - Method in interface org.quartz.Scheduler
  Delete the identified Calendar from the Scheduler.
deleteExtendedTriggerProperties(Connection, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate
deleteExtendedTriggerProperties(Connection, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport
deleteExtendedTriggerProperties(Connection, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate
deleteExtendedTriggerProperties(Connection, TriggerKey) - Method in interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate
deleteFiredTrigger(Connection, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
  Delete a fired trigger.
deleteFiredTrigger(Connection, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  Delete a fired trigger.
deleteFiredTriggers(Connection) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
  Delete all fired triggers.
deleteFiredTriggers(Connection, String) - Method in interface...
Delete all fired triggers of the given instance.

**deleteFiredTriggers(Connection)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Delete all fired triggers.

**deleteFiredTriggers(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**deleteJob(String, String)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean
Delete the identified Job from the Scheduler - and any associated Triggers.

**deleteJob(JobKey)** - Method in class org.quartz.core.QuartzScheduler
Delete the identified Job from the Scheduler - and any associated Triggers.

**deleteJob(JobKey)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**deleteJob(JobKey)** - Method in interface org.quartz.core.RemotableQuartzScheduler
Delete the identified Job from the Scheduler - and any associated Triggers.

**deleteJob(JobKey)** - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' quartzScheduler, passing the SchedulingContext associated with this instance.

**deleteJob(JobKey)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' quartzScheduler.

**deleteJob(JobKey)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' quartzScheduler.

**deleteJob(JobKey)** - Method in interface org.quartz.core.StandaloneQuartzScheduler
Delete the identified Job from the Scheduler - and any associated Triggers.

**deleteJobDetail(Connection, JobKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Delete the job detail record for the given job.

**deleteJobDetail(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Delete the job detail record for the given job.

**deleteJobs(List<JobKey>)** - Method in class org.quartz.core.QuartzScheduler

**deleteJobs(List<JobKey>)** - Method in interface org.quartz.core.RemotableQuartzScheduler
**deleteJobs(List<JobKey>)** - Method in class org.quartz.impl.RemoteMBeanScheduler

**deleteJobs(List<JobKey>)** - Method in class org.quartz.impl.RemoteScheduler

**deleteJobs(List<JobKey>)** - Method in class org.quartz.impl.StdScheduler

**deleteJobs(List<JobKey>)** - Method in interface org.quartz.Scheduler

Delete the identified Jobs from the Scheduler - and any associated Triggers.

**deletePausedTriggerGroup(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

**deletePausedTriggerGroup(Connection, GroupMatcher<TriggerKey>)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

**deletePausedTriggerGroup(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**deletePausedTriggerGroup(Connection, GroupMatcher<TriggerKey>)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**deleteSchedulerState(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Delete a scheduler-instance state record.

**deleteSchedulerState(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**deleteTrigger(Connection, TriggerKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Delete the base trigger data for a trigger.

**deleteTrigger(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Delete the base trigger data for a trigger.

**deleteTriggerExtension(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**depth()** - Method in class org.quartz.utils.CircularLossyQueue

Returns the number of items currently in the queue
**destroy()** - Method in class org.quartz.ee.servlet.**QuartzInitializerServlet**

**destroyService()** - Method in class org.quartz.ee.jmx.jboss.**QuartzService**

**DIRECTORY_NAME** - Static variable in class org.quartz.jobs.**DirectoryScanJob**
   JobDataMap key with which to specify the directory to be monitored - an absolute path is recommended.

**DIRECTORY_SCAN_LISTENER_NAME** - Static variable in class org.quartz.jobs.**DirectoryScanJob**
   JobDataMap key with which to specify the **DirectoryScanListener** to be notified when the directory contents change.

**DirectoryScanJob** - Class in org.quartz.jobs
   Inspects a directory and compares whether any files' "last modified dates" have changed since the last time it was inspected.

**DirectoryScanJob()** - Constructor for class org.quartz.jobs.**DirectoryScanJob**

**DirectoryScanListener** - Interface in org.quartz.jobs
   Interface for objects wishing to receive a 'call-back' from a DirectoryScanJob.

**DirectSchedulerFactory** - Class in org.quartz.impl
   A singleton implementation of SchedulerFactory.

**DirectSchedulerFactory()** - Constructor for class org.quartz.impl.**DirectSchedulerFactory**
   Constructor

**DirtyFlagMap** - Class in org.quartz.util
   An implementation of *Map* that wraps another *Map* and flags itself 'dirty' when it is modified.

**DirtyFlagMap()** - Constructor for class org.quartz.util.**DirtyFlagMap**
   Create a DirtyFlagMap that 'wraps' a *HashMap*.

**DirtyFlagMap(int)** - Constructor for class org.quartz.util.**DirtyFlagMap**
   Create a DirtyFlagMap that 'wraps' a *HashMap* that has the given initial capacity.

**DirtyFlagMap(int, float)** - Constructor for class org.quartz.util.**DirtyFlagMap**
   Create a DirtyFlagMap that 'wraps' a *HashMap* that has the given initial capacity and load factor.

**DisallowConcurrentExecution** - Annotation Type in org.quartz
   An annotation that marks a *Job* class as one that must not have multiple instances executed concurrently (where instance is based-upon a *JobDetail*)
definition - or in other words based upon a \textit{JobKey}).
\textbf{doCheckin()} - Method in class org.quartz.impl.jdbcjobstore.\texttt{JobStoreSupport}

\textbf{doGet(HttpServletRequest, HttpServletResponse)} - Method in class
org.quartz.ee.servlet.\texttt{QuartzInitializerServlet}

\textbf{dontSetNonManagedTXConnectionAutoCommitFalse} - Variable in class
org.quartz.impl.jdbcjobstore.\texttt{JobStoreCMT}

\textbf{doPost(HttpServletRequest, HttpServletResponse)} - Method in class
org.quartz.ee.servlet.\texttt{QuartzInitializerServlet}

\textbf{doRecoverMisfires()} - Method in class
org.quartz.impl.jdbcjobstore.\texttt{JobStoreSupport}

\textbf{DriverDelegate} - Interface in org.quartz.impl.jdbcjobstore
This is the base interface for all driver delegate classes.
\textbf{dsName} - Variable in class org.quartz.impl.jdbcjobstore.\texttt{JobStoreSupport}
E

**EhcacheConstraint** - Class in org.quartz.locality.constraint
Constraints related the locality of Ehcache values

**EhcacheConstraint(EhcacheConstraint.Operator, EhcacheConstraint.Value)** - Constructor for class org.quartz.locality.constraint.EhcacheConstraint
Constructor

The Cache needs to be alive and clustered using Terracotta in Serialization mode

**EhcacheConstraint.Operator** - Enum in org.quartz.locality.constraint

**EhcacheConstraint.Value** - Class in org.quartz.locality.constraint

**EhcacheConstraint.Value(String, String, Collection<?>)** - Constructor for class org.quartz.locality.constraint.EhcacheConstraint.Value

**EhcachEvaluator** - Class in org.terracotta.modules.ehcache.store
Ehcache evaluator, which lets Jobs be executed on certain node, depending of the locality of the data in a Ehcache

**EhcachEvaluator(ConcurrentMap<String, String>)** - Constructor for class org.terracotta.modules.ehcache.store.EhcachEvaluator
Constructor

**EJB_ARG_TYPES_KEY** - Static variable in class org.quartz.jobs.ee.ejb.EJBIInvokerJob

**EJB_ARGS_KEY** - Static variable in class org.quartz.jobs.ee.ejb.EJBIInvokerJob

**EJB_JNDI_NAME_KEY** - Static variable in class org.quartz.jobs.ee.ejb.EJBIInvokerJob

**EJB_METHOD_KEY** - Static variable in class org.quartz.jobs.ee.ejb.EJBIInvokerJob

**EJBIInvokerJob** - Class in org.quartz.jobs.ee.ejb
A Job that invokes a method on an EJB.

**EJBInvokerJob()** - Constructor for class org.quartz.jobs.ee.ejb.EJBInvokerJob

**elements(Cache, Collection)** - Static method in class org.quartz.locality.constraint.EhcacheConstraint

  Creates a EhcacheConstraint for local values to a Set of keys

**emitter** - Variable in class org.quartz.core.QuartzSchedulerMBeanImpl

**endAt(Date)** - Method in class org.quartz.TriggerBuilder

  Set the time at which the Trigger will no longer fire - even if it's schedule has remaining repeats.

**entrySet()** - Method in class org.quartz.utils.DirtyFlagMap

**equals(Object)** - Method in class org.quartz.impl.JobDetailImpl

**equals(Object)** - Method in class org.quartz.impl.matchers.AndMatcher

**equals(Object)** - Method in class org.quartz.impl.matchers.EverythingMatcher

**equals(Object)** - Method in class org.quartz.impl.matchers.KeyMatcher

**equals(Object)** - Method in class org.quartz.impl.matchers.NotMatcher

**equals(Object)** - Method in class org.quartz.impl.matchers.OrMatcher

**equals(Object)** - Method in class org.quartz.impl.triggers.AbstractTrigger

  Trigger equality is based upon the equality of the TriggerKey.

**equals(Object)** - Method in class org.quartz.locality.constraint.CpuConstraint

**equals(Object)** - Method in class org.quartz.localityconstraint.EhcacheConstraint.Value

**equals(Object)** - Method in class org.quartz.locality.constraint.MemoryConstraint

**equals(Object)** - Method in class org.quartz.locality.constraint.NodeGroupConstraint
equals(Object) - Method in class org.quartz.locality.constraint.OsConstraint
equals(Object) - Method in interface org.quartz.Matcher
equals(Object) - Method in interface org.quartz.Trigger
    Trigger equality is based upon the equality of the TriggerKey.
equals(Object) - Method in class org.quartz.utils.DirtyFlagMap
equals(Object) - Method in class org.quartz.utils.Key
equals(Object) - Method in class org.quartz.utils.StringKeyDirtyFlagMap
error(SAXParseException) - Method in class org.quartz.xml.XMLSchedulingDataProcessor
    ErrorHandler interface.
errorTriggerRetryLoop(TriggerFiredBundle) - Method in class org.quartz.core.QuartzSchedulerThread
evaluate(String, String) - Method in enum org.quartz.impl.matchers.StringMatcher.StringOperatorName
evaluate(String, String) - Method in enum org.quartz.locality.constraint.NodeGroupConstraint.Operator
evaluate(OsConstraint.OS, OsConstraint.OS) - Method in enum org.quartz.locality.constraint.OsConstraint.Operator
Evaluator<T extends Constraint> - Interface in org.quartz.locality.constraint.evaluator
    An Evaluator will match and potentially return best suited nodes for a Job to execute on.
EvaluatorTimerTask - Class in org.quartz.locality.constraint.evaluator
    An evaluator that performs monitoring on a regular basis
EvaluatorTimerTask() - Constructor for class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask
EvaluatorTimerTask.Status - Enum in org.quartz.locality.constraint.evaluator
**evenHourDate(Date)** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even hour above the given date.

**evenHourDateAfterNow()** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even hour after the current time.

**evenHourDateBefore(Date)** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the previous even hour below the given date.

**evenMinuteDate(Date)** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even minute above the given date.

**evenMinuteDateAfterNow()** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even minute after the current time.

**evenMinuteDateBefore(Date)** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the previous even minute below the given date.

**evenSecondDate(Date)** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even second above the given date.

**evenSecondDateAfterNow()** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even second after the current time.

**evenSecondDateBefore(Date)** - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the previous even second below the given date.

**EverythingMatcher<T extends Key>** - Class in org.quartz.impl.matchers
Matches on the complete key being equal (both name and group).

**execute(Connection)** - Method in interface org.quartz.impl.jdbcjobstore.JobStoreSupport.TransactionCallback
Called by the Scheduler when a Trigger fires that is associated with the Job.

**execute(JobExecutionContext)** - Method in interface org.quartz.Job
Called by the Scheduler when a Trigger fires that is associated with the Job.

**execute(JobExecutionContext)** - Method in class org.quartz.jobs.DirectoryScanJob
org.quartz.jobs.ee.ejb.\texttt{EJBInvokerJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.ee.jms.\texttt{SendDestinationMessageJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.ee.jms.\texttt{SendQueueMessageJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.ee.jms.\texttt{SendTopicMessageJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.ee.jmx.\texttt{JMXInvokerJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.ee.mail.\texttt{SendMailJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.\texttt{FileScanJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.\texttt{NativeJob}

\texttt{execute(JobExecutionContext)} - Method in class org.quartz.jobs.\texttt{NoOpJob}

Do nothing.

\texttt{ExecuteInJTATransaction} - Annotation Type in \texttt{org.quartz}

An annotation that marks a \texttt{Job} class as one that will have its execution wrapped by a JTA Transaction.

\texttt{executeInLock(String, JobStoreSupport.TransactionCallback)} - Method in class org.quartz.impl.jdbcjobstore.\texttt{JobStoreCMT}

Execute the given callback having optionally acquired the given lock.

\texttt{executeInLock(String, JobStoreSupport.VoidTransactionCallback)} - Method in class org.quartz.impl.jdbcjobstore.\texttt{JobStoreSupport}

Execute the given callback having acquired the given lock.

\texttt{executeInLock(String, JobStoreSupport.TransactionCallback)} - Method in class org.quartz.impl.jdbcjobstore.\texttt{JobStoreSupport}

Execute the given callback having acquired the given lock.

\texttt{executeInLock(String, JobStoreSupport.TransactionCallback)} - Method in class org.quartz.impl.jdbcjobstore.\texttt{JobStoreTX}

Execute the given callback having optionally acquired the given lock.

\texttt{executeInNonManagedTXLock(String,}
  Execute the given callback having optionally acquired the given lock.
executeInNonManagedTXLock(String, JobStoreSupport.TransactionCallback) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Execute the given callback having optionally acquired the given lock.
executePreProcessCommands(Scheduler) - Method in class org.quartz.xml.XMLSchedulingDataProcessor
executeSQL(Connection, String, String, String) - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore
  Execute the SQL that will lock the proper database row.
executeSQL(Connection, String, String, String) - Method in class org.quartz.impl.jdbcjobstore.StdRowLockSemaphore
  Execute the SQL select for update that will lock the proper database row.
executeSQL(Connection, String, String, String) - Method in class org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore
  Execute the SQL select for update that will lock the proper database row.
executeWithoutLock(JobStoreSupport.TransactionCallback) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Execute the given callback in a transaction.
executionComplete(JobExecutionContext, JobExecutionException) - Method in class org.quartz.impl.triggers.AbstractTrigger
  This method should not be used by the Quartz client.
executionComplete(JobExecutionContext, JobExecutionException) - Method in class org.quartz.locality.DelegatingLocalityTrigger
expressionParsed - Variable in class org.quartz.CronExpression
FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS - Static variable in interface org.quartz.Scheduler
   A constant JobDataMap key that can be used to retrieve the scheduled fire time of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

FAILED_JOB_ORIGINAL_TRIGGER_GROUP - Static variable in interface org.quartz.Scheduler
   A constant JobDataMap key that can be used to retrieve the group of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

FAILED_JOB_ORIGINAL_TRIGGER_NAME - Static variable in interface org.quartz.Scheduler
   A constant JobDataMap key that can be used to retrieve the name of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

fatalError(SAXParseException) - Method in class org.quartz.xml.XMLSchedulingDataProcessor
   ErrorHandler interface.

FILE_NAME - Static variable in class org.quartz.jobs.FileScanJob
   JobDataMap key with which to specify the name of the file to monitor.

FILE_SCAN_LISTENER_NAME - Static variable in class org.quartz.jobs.FileScanJob
   JobDataMap key with which to specify the FileScanListener to be notified when the file contents change.

FileScanJob - Class in org.quartz.jobs
   Inspects a file and compares whether it's "last modified date" has changed since the last time it was inspected.

FileScanJob() - Constructor for class org.quartz.jobs.FileScanJob

FileScanListener - Interface in org.quartz.jobs
   Interface for objects wishing to receive a 'call-back' from a FileScanJob.

filesUpdatedOrAdded(File[]) - Method in interface org.quartz.jobs.DirectoryScanListener

fileUpdated(String) - Method in interface org.quartz.jobs.FileScanListener
fileUpdated(String) - Method in class org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin

findFailedInstances(Connection) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Get a list of all scheduler instances in the cluster that may have failed.

findNextWhiteSpace(int, String) - Method in class org.quartz.CronExpression

findTriggerPersistenceDelegate(OperableTrigger) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

findTriggerPersistenceDelegate(String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

firedTriggerBundle - Variable in class org.quartz.core.JobRunShell

FiredTriggerRecord - Class in org.quartz.impl.jdbcjobstore
Conveys the state of a fired-trigger record.
FiredTriggerRecord() - Constructor for class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

firstCheckIn - Variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

forJob(JobKey) - Method in class org.quartz.TriggerBuilder
Set the identity of the Job which should be fired by the produced Trigger.

forJob(String) - Method in class org.quartz.TriggerBuilder
Set the identity of the Job which should be fired by the produced Trigger - a JobKey will be produced with the given name and default group.

forJob(String, String) - Method in class org.quartz.TriggerBuilder
Set the identity of the Job which should be fired by the produced Trigger - a JobKey will be produced with the given name and group.

forJob(JobDetail) - Method in class org.quartz.TriggerBuilder
Set the identity of the Job which should be fired by the produced Trigger, by extracting the JobKey from the given job.

FRIDAY - Static variable in class org.quartz.DateBuilder

futureDate(int, DateBuilder.IntervalUnit) - Static method in class org.quartz.DateBuilder
generateInstanceId() - Method in class
org.quartz.simpl.HostnameInstanceIdGenerator

generateInstanceId() - Method in class
org.quartz.simpl.SimpleInstanceIdGenerator

generateInstanceId() - Method in class
org.quartz.simpl.SystemPropertyInstanceIdGenerator

- Returns the cluster wide value for this scheduler instance's id, based on a system property

generateJMXObjectName(String, String) - Static method in class
org.quartz.core.QuartzSchedulerResources

- Create the name under which this scheduler should be registered in JMX.

get(Object) - Method in class org.quartz.impl.JobExecutionContextImpl

- Get the value with the given key from the context's data map.

get(Object) - Method in class org.quartz.utils.DirtyFlagMap

getAllJobDetails() - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

getAllJobDetails() - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

getAllowsTransientData() - Method in class org.quartz.utils.StringKeyDirtyFlagMap

- Deprecated. JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

getAllSampleValues() - Method in interface
org.quartz.utils.counter.sampled.SampledCounter

- Returns all samples in history

getAllSampleValues() - Method in class
org.quartz.utils.counter.sampled.SampledCounterImpl
Returns all samples in history

**getAllSchedulers()** - Method in class org.quartz.impl.**DirectSchedulerFactory**
Returns a handle to all known Schedulers (made by any StdSchedulerFactory instance.).

**getAllSchedulers()** - Method in class org.quartz.impl.**StdSchedulerFactory**
Returns a handle to all known Schedulers (made by any StdSchedulerFactory instance.).

**getAllSchedulers()** - Method in interface org.quartz.**SchedulerFactory**
Returns handles to all known Schedulers (made by any SchedulerFactory within this jvm.).

**getAllTriggers()** - Method in interface org.quartz.core.jmx.**QuartzSchedulerMBean**

**getAllTriggers()** - Method in class org.quartz.core.**QuartzSchedulerMBeanImpl**

**getAndReset()** - Method in interface org.quartz.utils.counter.sampled.**SampledCounter**
Returns the current value of the counter and resets it to 0

**getAndReset()** - Method in class org.quartz.utils.counter.sampled.**SampledCounterImpl**
Returns the current value of the counter and resets it to 0

**getAndReset()** - Method in class org.quartz.utils.counter.sampled.**SampledRateCounterImpl**
Returns the current value of the counter and resets it to 0

**getAndSet(long)** - Method in interface org.quartz.utils.counter.**Counter**
Returns the value of the counter and sets it to the new value

**getAndSet(long)** - Method in class org.quartz.utils.counter.**CounterImpl**
Returns the value of the counter and sets it to the new value

**getAndSet(long)** - Method in class org.quartz.utils.counter.**SampledRateCounterImpl**
throws **UnsupportedOperationException**

**getAttribute(String)** - Method in class org.quartz.ee.jmx.jboss.**JBoss4RMIRemoteMBeanScheduler**

**getAttribute(String)** - Method in class org.quartz.impl.**RemoteMBeanScheduler**
Get the given attribute of the remote Scheduler MBean.

**getAttributeRestoringConnection(Connection)** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**
Wrap the given connection in a Proxy such that attributes that might be set
will be restored before the connection is closed (and potentially restored to a pool).

**getAttributes(String[])** - Method in class org.quartz.ee.jmx.jboss.JBoss4RMIRemoteMBeanScheduler

**getAttributes(String[])** - Method in class org.quartz.impl.RemoteMBeanScheduler

  Get the given attributes of the remote Scheduler MBean.

**getBaseCalendar()** - Method in interface org.quartz.Calendar

  Get the base calendar.

**getBaseCalendar()** - Method in class org.quartz.impl.calendar.BaseCalendar

  Get the base calendar.

**getBatchTimeWindow()** - Method in class org.quartz.core.QuartzSchedulerResources

**getBoolean(ResultSet, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

  Retrieves the value of the designated column in the current row as a boolean.

**getBoolean(ResultSet, int)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

  Retrieves the value of the designated column index in the current row as a boolean.

**getBoolean(String)** - Method in class org.quartz.util.StringKeyDirtyFlagMap

  Retrieve the identified boolean value from the StringKeyDirtyFlagMap.

**getBoolean(XPath, String, Document)** - Method in class org.quartz.xml.XMLSchedulingDataProcessor

**getBooleanFromString(String)** - Method in class org.quartz.JobDataMap

  Retrieve the identified Boolean value from the JobDataMap.

**getBooleanProperty(String)** - Method in class org.quartz.util.PropertiesParser

  Retrieve the identified boolean value from the JobDataMap.

**getBooleanProperty(String, boolean)** - Method in class org.quartz.util.PropertiesParser

**getBooleanValue(String)** - Method in class org.quartz.JobDataMap

  Retrieve the identified boolean value from the JobDataMap.

**getBooleanValueFromString(String)** - Method in class org.quartz.JobDataMap

  Retrieve the identified boolean value from the JobDataMap.
**getByteProperty(String)** - Method in class org.quartz.utils.PropertiesParser

**getByteProperty(String, byte)** - Method in class org.quartz.utils.PropertiesParser

**getCacheManagerName()** - Method in class org.quartz.locality.constraint.EhcacheConstraint
- Gets the CacheManager name for the Cache

**getCacheManagerName()** - Method in class org.quartz.locality.constraint.EhcacheConstraint.Value

**getCacheName()** - Method in class org.quartz.locality.constraint.EhcacheConstraint
- Gets the Cache name

**getCacheName()** - Method in class org.quartz.locality.constraint.EhcacheConstraint.Value

**getCalendar(String)** - Method in class org.quartz.core.QuartzScheduler
- Get the Calendar instance with the given name.

**getCalendar(String)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getCalendar()** - Method in class org.quartz.impl.JobExecutionContextImpl

**getCalendar(String)** - Method in class org.quartz.impl.RemoteMBeanScheduler
- Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**getCalendar(String)** - Method in class org.quartz.impl.RemoteScheduler
- Calls the equivalent method on the 'proxied' QuartzScheduler.

**getCalendar(String)** - Method in class org.quartz.impl.StdScheduler
- Calls the equivalent method on the 'proxied' QuartzScheduler.

**getCalendar()** - Method in interface org.quartz.JobExecutionContext
- Get a handle to the Calendar referenced by the Trigger instance that fired the Job.

**getCalendar(String)** - Method in interface org.quartz.Scheduler
- Get the Calendar instance with the given name.

**getCalendarName()** - Method in class org.quartz.impl.triggers.AbstractTrigger
- Get the name of the Calendar associated with this Trigger.

**getCalendarName()** - Method in class
org.quartz.locality.\texttt{DelegatingLocalityTrigger}
Get the name of the calendar associated with this Trigger.

\texttt{getCalendarName()} - Method in interface \texttt{org.quartz.Trigger}
Get the name of the calendar associated with this Trigger.

\texttt{getCalendarNames()} - Method in interface \texttt{org.quartz.core.jmx.QuartzSchedulerMBean}

\texttt{getCalendarNames()} - Method in class \texttt{org.quartz.core.QuartzScheduler}
Get the names of all registered calendars.

\texttt{getCalendarNames()} - Method in class
\texttt{org.quartz.core.QuartzSchedulerMBeanImpl}

\texttt{getCalendarNames()} - Method in interface \texttt{org.quartz.core.RemotableQuartzScheduler}

\texttt{getCalendarNames()} - Method in class
\texttt{org.quartz.impl.jdbcjobstore.JobStoreSupport}
Get the names of all of the calendars in the JobStore.

\texttt{getCalendarNames(Connection)} - Method in class
\texttt{org.quartz.impl.jdbcjobstore.JobStoreSupport}

\texttt{getCalendarNames()} - Method in class
\texttt{org.quartz.impl.RemoteMBeanScheduler}
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

\texttt{getCalendarNames()} - Method in class \texttt{org.quartz.impl.RemoteScheduler}
Calls the equivalent method on the 'proxied' QuartzScheduler.

\texttt{getCalendarNames()} - Method in class \texttt{org.quartz.impl.StdScheduler}
Calls the equivalent method on the 'proxied' QuartzScheduler.

\texttt{getCalendarNames()} - Method in interface \texttt{org.quartz.Scheduler}
Get the names of all registered calendars.

\texttt{getCalendarNames()} - Method in class \texttt{org.quartz.impl.jdbcjobstore.RAMJobStore}
Get the names of all of the calendars in the JobStore.

\texttt{getCc()} - Method in class \texttt{org.quartz.jobs.ee.mail.SendMailJob.MailInfo}
Retrieve the identified \texttt{char} value from the \texttt{StringKeyDirtyFlagMap}.

\texttt{getChar(String)} - Method in class \texttt{org.quartz.utils.StringKeyDirtyFlagMap}
Retrieve the identified \texttt{Character} value from the \texttt{JobDataMap}.

\texttt{getCharacterFromString(String)} - Method in class \texttt{org.quartz.JobDataMap}
Retrieve the identified \texttt{Character} value from the \texttt{JobDataMap}.
**getCharFromString(String)** - Method in class org.quartz.JobDataMap
Retrieve the identified char value from the JobDataMap.

**getCharProperty(String)** - Method in class org.quartz.utils.PropertiesParser

**getCharProperty(String, char)** - Method in class org.quartz.utils.PropertiesParser

**getCheckinInterval()** - Method in class org.quartz.impl.jdbcjobstore.SchedulerStateRecord

**getCheckinTimestamp()** - Method in class org.quartz.impl.jdbcjobstore.SchedulerStateRecord

**getClassLoader()** - Method in class org.quartz.simpl.CascadingClassLoadHelper
Enable sharing of the "best" class-loader with 3rd party.

**getClassLoader()** - Method in class org.quartz.simpl.InitThreadContextClassLoadHelper
Enable sharing of the class-loader with 3rd party.

**getClassLoader()** - Method in class org.quartz.simpl.LoadingLoaderClassLoadHelper
Enable sharing of the class-loader with 3rd party.

**getClassLoader()** - Method in class org.quartz.simpl.SimpleClassLoadHelper
Enable sharing of the class-loader with 3rd party.

**getClassLoader()** - Method in class org.quartz.simpl.ThreadContextClassLoadHelper
Enable sharing of the class-loader with 3rd party.

**getClassLoadHelper()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**getClusterCheckinInterval()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Get the frequency (in milliseconds) at which this instance "checks-in" with the other instances of the cluster.

**getCompareToValue()** - Method in class org.quartz.impl.matchers.KeyMatcher

**getCompareToValue()** - Method in class org.quartz.impl.matchers.StringMatcher
**getCompareWithOperator()** - Method in class `org.quartz.impl.matchers.StringMatcher`

**getConnection()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**getConnection()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**getConnection()** - Method in interface `org.quartz.utils.ConnectionProvider`

**getConnection(String)** - Method in class `org.quartz.utils.DBConnectionManager`

Get a database connection from the DataSource with the given name.

**getConnection()** - Method in class `org.quartz.utils.JNDIConnectionProvider`

**getConnection()** - Method in class `org.quartz.utils.PoolingConnectionProvider`

**getConnection()** - Method in class `org.quartz.utils.weblogic.WeblogicConnectionProvider`

**getConstraint()** - Method in exception `org.quartz.locality.LocalityException`

Constraint being the issue

**getConstraints()** - Method in interface `org.quartz.locality.NodeSpec`

Returns a List of `Constraint` to be used to identify the targeted node

**getContentType()** - Method in class `org.quartz.jobs.ee.mail.SendMailJob.MailInfo`

**getContext()** - Method in class `org.quartz.impl.RemoteMBeanScheduler`

Returns the SchedulerContext of the Scheduler.

**getContext()** - Method in class `org.quartz.impl.RemoteScheduler`

Returns the SchedulerContext of the Scheduler.

**getContext()** - Method in class `org.quartz.impl.StdScheduler`

Returns the SchedulerContext of the Scheduler.

**getContext()** - Method in interface `org.quartz.Scheduler`

Returns the SchedulerContext of the Scheduler.

**getContextProperties()** - Method in class `org.quartz.ee.jmx.jboss.JBoss4RMIRemoteMBeanScheduler`

Get the properties to use when creating a JNDI InitialContext.

**getCounterValue()** - Method in class `org.quartz.utils.counter.sampled.TimeStampedCounterValue`

Get the counter value

**getCronExpression()** - Method in class `org.quartz.CronExpression`
**getCronExpression()** - Method in interface org.quartz.CronTrigger

**getCronExpression()** - Method in class org.quartz.impl.calendar.CronCalendar
   Returns the object representation of the cron expression that defines the
dates and times this calendar excludes.

**getCronExpression()** - Method in class
org.quartz.impl.triggers.CronTriggerImpl

**getCurrentlyExecutingJobs()** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean

**getCurrentlyExecutingJobs()** - Method in class
org.quartz.core.QuartzScheduler
   Return a list of JobExecutionContext objects that represent all currently
   executing Jobs in this Scheduler instance.

**getCurrentlyExecutingJobs()** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**getCurrentlyExecutingJobs()** - Method in interface
org.quartz.core.RemotableQuartzScheduler

**getCurrentlyExecutingJobs()** - Method in class
org.quartz.impl.RemoteMBeanScheduler
   Calls the equivalent method on the 'proxied' QuartzScheduler.

**getCurrentlyExecutingJobs()** - Method in class
org.quartz.impl.RemoteScheduler
   Calls the equivalent method on the 'proxied' QuartzScheduler.

**getCurrentlyExecutingJobs()** - Method in class org.quartz.impl.StdScheduler
   Calls the equivalent method on the 'proxied' QuartzScheduler.

**getCurrentlyExecutingJobs()** - Method in interface org.quartz.Scheduler
   Return a list of JobExecutionContext objects that represent all currently
   executing Jobs in this Scheduler instance.

**getCurrentTime()** - Method in class org.quartz.simpl.SimpleTimeBroker
   Get the current time, simply using new Date().

**getDataSource()** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
   Get the name of the DataSource that should be used for performing
database functions.
**getDataSource()** - Method in class org.quartz.utils.PoolingConnectionProvider
Get the C3PO ComboPooledDataSource created during initialization.

**getDayOfWeekNumber(String)** - Method in class org.quartz.CronExpression

**getDaysExcluded()** - Method in class org.quartz.impl.calendar.AnnualCalendar
Get the array which defines the exclude-value of each day of month

**getDaysExcluded()** - Method in class org.quartz.impl.calendar.MonthlyCalendar
Get the array which defines the exclude-value of each day of month.

**getDaysExcluded()** - Method in class org.quartz.impl.calendar.WeeklyCalendar
Get the array with the week days

**getDbRetryInterval()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**getDecimal1()** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**getDecimal2()** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**getDefaultScheduler()** - Static method in class org.quartz.impl.StdSchedulerFactory
Returns a handle to the default Scheduler, creating it if it does not yet exist.

**getDelegate()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Get the driver delegate for DB operations.

**getDescription()** - Method in interface org.quartz.Calendar
Return the description given to the Calendar instance by its creator (if any).

**getDescription()** - Method in class org.quartz.impl.calendar.BaseCalendar
Return the description given to the Calendar instance by its creator (if any).

**getDescription()** - Method in class org.quartz.impl.JobDetailImpl

**getDescription()** - Method in class org.quartz.impl.triggers(AbstractTrigger
Return the description given to the Trigger instance by its creator (if any).

**getDescription()** - Method in interface org.quartz.JobDetail
Return the description given to the Job instance by its creator (if any).

**getDescription()** - Method in class org.quartz.locality.DelegatingLocalityJobDetail
Return the description given to the Job instance by its creator (if any).

**getDescription()** - Method in class org.quartz.locality.DelegatingLocalityTrigger
Return the description given to the Trigger instance by its creator (if any).
**getDescription()** - Method in interface `org.quartz.Trigger`
Return the description given to the Trigger instance by its creator (if any).

**getDouble(String)** - Method in class `org.quartz.utils.StringKeyDirtyFlagMap`
Retrieve the identified double value from the StringKeyDirtyFlagMap.

**getDoubleCheckLockMisfireHandler()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
Get whether to check to see if there are Triggers that have misfired before actually acquiring the lock to recover them.

**getDoubleFromString(String)** - Method in class `org.quartz.JobDataMap`
Retrieve the identified Double value from the JobDataMap.

**getDoubleProperty(String)** - Method in class `org.quartz.utils.PropertiesParser`

**getDoubleProperty(String, double)** - Method in class `org.quartz.utils.PropertiesParser`

**getDoubleValue(String)** - Method in class `org.quartz.JobDataMap`
Retrieve the identified double value from the JobDataMap.

**getDoubleValueFromString(String)** - Method in class `org.quartz.JobDataMap`
Retrieve the identified double value from the JobDataMap.

**getDriverDelegateClass()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
Get the JDBC driver delegate class name.

**getDriverDelegateInitString()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
Get the JDBC driver delegate's initialization string.


**getEndOfDayJavaCalendar(long)** - Method in class `org.quartz.impl.calendar.BaseCalendar`
Returns the end of the given day Calendar.

**getEndTime()** - Method in class `org.quartz.impl.triggers.AbstractTrigger`
Get the time at which the Trigger should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).

**getEndTime()** - Method in class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`
Get the time at which the DateIntervalTrigger should quit repeating.

**getEndTime()** - Method in class `org.quartz.impl.triggers.CronTriggerImpl`
Get the time at which the CronTrigger should quit repeating - even if
repeastCount isn't yet satisfied.

**getEndTime**() - Method in class org.quartz.impl.triggers.*SimpleTriggerImpl*
Get the time at which the *SimpleTrigger* should quit repeating - even if repeastCount isn't yet satisfied.

**getEndTime**() - Method in class org.quartz.locality.*DelegatingLocalityTrigger*
Get the time at which the *Trigger* should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).

**getEndTime**() - Method in interface org.quartz.*Trigger*
Get the time at which the *Trigger* should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).

**getEstimatedTimeToReleaseAndAcquireTrigger**() - Method in class org.quartz.impl.jdbcjobstore.*JobStoreSupport*

**getEstimatedTimeToReleaseAndAcquireTrigger**() - Method in class org.quartz.impl.*RAMJobStore*

**getExcludedDates**() - Method in class org.quartz.impl.calendar.*HolidayCalendar*
Returns a *SortedSet* of Dates representing the excluded days.

**getExpressionSetSummary**(*Set*) - Method in class org.quartz.*CronExpression*

**getExpressionSetSummary**(*ArrayList*) - Method in class org.quartz.*CronExpression*

**getExpressionSummary**() - Method in class org.quartz.*CronExpression*

**getExpressionSummary**() - Method in interface org.quartz.*CronTrigger*

**getExpressionSummary**() - Method in class org.quartz.impl.triggers.*CronTriggerImpl*

**getFileNames**() - Method in class org.quartz.plugins.xml.*XMLSchedulingDataProcessorPlugin*
Comma separated list of file names (with paths) to the XML files that should be read.

**getFinalFireTime**() - Method in class org.quartz.*CronExpression*
NOT YET IMPLEMENTED: Returns the final time that the CronExpression will match.

**getFinalFireTime**() - Method in class org.quartz.impl.triggers.*AbstractTrigger*
Returns the last time at which the Trigger will fire, if the Trigger will repeat indefinitely, null will be returned.

**getFinalFireTime()** - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl

Returns the final time at which the DateIntervalTrigger will fire, if there is no end time set, null will be returned.

**getFinalFireTime()** - Method in class org.quartz.impl.triggers.CronTriggerImpl

NOT YET IMPLEMENTED: Returns the final time at which the CronTrigger will fire.

**getFinalFireTime()** - Method in class org.quartz.impl.triggers.SimpleTriggerImpl

Returns the final time at which the SimpleTrigger will fire, if repeatCount is REPEAT_INFINITY, null will be returned.

**getFinalFireTime()** - Method in class org.quartz.locality.DelegatingLocalityTrigger

Returns the last time at which the Trigger will fire, if the Trigger will repeat indefinitely, null will be returned.

**getFinalFireTime()** - Method in interface org.quartz.Trigger

Returns the last time at which the Trigger will fire, if the Trigger will repeat indefinitely, null will be returned.

**getFiredTriggerRecordId()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**getFiredTriggerRecordId()** - Method in class org.quartz.simpl.RAMJobStore

**getFireInstanceId()** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

**getFireInstanceId()** - Method in class org.quartz.impl.triggers.AbstractTrigger

This method should not be used by the Quartz client.

**getFireInstanceId()** - Method in class org.quartz.locality.DelegatingLocalityTrigger

**getFireInstanceState()** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

**getFireTime()** - Method in class org.quartz.impl.JobExecutionContextImpl

**getFireTime()** - Method in interface org.quartz.JobExecutionContext

The actual time the trigger fired.
**getFireTimeAfter(Date)** - Method in class org.quartz.impl.triggers.**AbstractTrigger**

Returns the next time at which the Trigger will fire, after the given time.

**getFireTimeAfter(Date)** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**

Returns the next time at which the DateIntervalTrigger will fire, after the given time.

**getFireTimeAfter(Date, boolean)** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**

**getFireTimeAfter(Date)** - Method in class org.quartz.impl.triggers.**CronTriggerImpl**

Returns the next time at which the CronTrigger will fire, after the given time.

**getFireTimeAfter(Date)** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**

Returns the next time at which the SimpleTrigger will fire, after the given time.

**getFireTimeAfter(Date)** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

Returns the next time at which the Trigger will fire, after the given time.

**getFireTimeAfter(Date)** - Method in interface org.quartz.**Trigger**

Returns the next time at which the Trigger will fire, after the given time.

**getFireTimeBefore(Date)** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**

Returns the last time at which the SimpleTrigger will fire, before the given time.

**getFireTimestamp()** - Method in class org.quartz.impl.jdbcjobstore.**FiredTriggerRecord**

**getFloat(String)** - Method in class org.quartz.utils.**StringKeyDirtyFlagMap**

Retrieve the identified float value from the StringKeyDirtyFlagMap.

**getFloatFromString(String)** - Method in class org.quartz.**JobDataMap**

Retrieve the identified float value from the JobDataMap.

**getFloatProperty(String)** - Method in class org.quartz.utils.**PropertiesParser**

**getFloatProperty(String, float)** - Method in class org.quartz.utils.**PropertiesParser**
**getFloatValue(String)** - Method in class org.quartz.JobDataMap
Retrieve the identified float value from the JobDataMap.

**getFloatValueFromFromString(String)** - Method in class org.quartz.JobDataMap
Retrieve the identified float value from the JobDataMap.

**getFrom()** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**getFullJobName()** - Method in class org.quartz.impl.triggers.AbstractTrigger
Returns the 'full name' of the Job that the Trigger points to, in the format "group.name".

**getFullName()** - Method in class org.quartz.impl.JobDetailImpl
Returns the 'full name' of the JobDetail in the format "group.name".

**getFullName()** - Method in class org.quartz.impl.triggers.AbstractTrigger
Returns the 'full name' of the Trigger in the format "group.name".

**getGroup()** - Method in class org.quartz.impl.JobDetailImpl
Get the group of this Job.

**getGroup()** - Method in class org.quartz.impl.triggers.AbstractTrigger
Get the group of this Trigger.

**getGroup()** - Method in class org.quartz.utils.Key
Get the group portion of the key.

**getHandledTriggerTypeDiscriminator()** - Method in class org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate

**getHandledTriggerTypeDiscriminator()** - Method in class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

**getHandledTriggerTypeDiscriminator()** - Method in class org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

**getHandledTriggerTypeDiscriminator()** - Method in interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate

**getHistorySize()** - Method in class org.quartz.utils.counter.sampled.SampledCounterConfig
Returns the history size

**getIndexNames()** - Method in class org.quartz.core.jmx.TriggerSupport

**getInitialContext(JobDataMap)** - Static method in class org.quartz.jobs.ee.jms.JmsHelper
getInitialValue() - Method in class org.quartz.utils.counter.CounterConfig
  Gets the initial value
getInputStream(String) - Method in class
  org.quartz.xml.XMLSchedulingDataProcessor
  Returns an InputStream from the fileName as a resource.
ggetInstance() - Static method in class org.quartz.impl.DirectSchedulerFactory
  Get the class instance.
ggetInstance() - Static method in class org.quartz.impl.SchedulerRepository
  Get the class instance.
ggetInstance() - Static method in class org.quartz.core.QuartzSchedulerResources
  Get the instance Id for the QuartzScheduler.
ggetInstanceId() - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Get the instance Id of the Scheduler (must be unique within a cluster).
ggetInstanceName() - Method in class
  org.quartz.impl.jdbcjobstore.JobStoreSupport
  Get the instance name of the Scheduler (must be unique within this server instance).
gGetInt(String) - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Retrieve the identified int value from the StringKeyDirtyFlagMap.
gGetInt1() - Method in class
  org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

getInternalJobListener(String) - Method in class
  org.quartz.core.QuartzScheduler
  Get the internal JobListener that has the given name.
getInternalJobListeners() - Method in class org.quartz.core.QuartzScheduler
Get a List containing all of the JobListeners in the Scheduler's *internal* list.

`getInternalSchedulerListeners()` - Method in class org.quartz.core.QuartzScheduler

Get a List containing all of the *internal* SchedulerListeners registered with the Scheduler.

`getInternalTriggerListener(String)` - Method in class org.quartz.core.QuartzScheduler

Get the *internal* TriggerListener that has the given name.

`getInternalTriggerListeners()` - Method in class org.quartz.core.QuartzScheduler

Get a list containing all of the TriggerListeners in the Scheduler's *internal* list.

`getInterval()` - Static method in class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask

`getIntervalMillis()` - Method in class org.quartz.utils.counter.sampled.SampledCounterImpl

Returns the sampling thread interval in millis

`getIntervalSecs()` - Method in class org.quartz.utils.counter.sampled.SampledCounterConfig

Returns the interval time (seconds)

`getIntFromString(String)` - Method in class org.quartz.JobDataMap

Retrieve the identified int value from the JobDataMap.

`getIntProperty(String)` - Method in class org.quartz.utils.PropertiesParser

`getIntProperty(String, int)` - Method in class org.quartz.utils.PropertiesParser

`getIntValue(String)` - Method in class org.quartz.JobDataMap

Retrieve the identified int value from the JobDataMap.

`getInvertTimeRange()` - Method in class org.quartz.impl.calendar.DailyCalendar

Indicates whether the time range represents an inverted time range (see class description).

`getItemDescriptions()` - Static method in class org.quartz.core.jmx.CronTriggerSupport

`getItemDescriptions()` - Static method in class org.quartz.core.jmx.SimpleTriggerSupport
**getItemDescriptions()** - Static method in class org.quartz.core.jmx.TriggerSupport

**getItemNames()** - Static method in class org.quartz.core.jmx.CronTriggerSupport

**getItemNames()** - Static method in class org.quartz.core.jmx.SimpleTriggerSupport

**getItemNames()** - Static method in class org.quartz.core.jmx.TriggerSupport

**getItemTypes()** - Static method in class org.quartz.core.jmx.CronTriggerSupport

**getItemTypes()** - Static method in class org.quartz.core.jmx.SimpleTriggerSupport

**getItemTypes()** - Static method in class org.quartz.core.jmx.TriggerSupport

**getJMXExport()** - Method in class org.quartz.core.QuartzSchedulerResources

Get whether the QuartzScheduler should be registered with the local MBeanServer.

**getJMXObjectName()** - Method in class org.quartz.core.QuartzSchedulerResources

Get the name under which the QuartzScheduler should be registered with the local MBeanServer.

**getJndiName()** - Method in class org.quartz.ee.jmx.jboss.QuartzService

**getJndiName()** - Method in interface org.quartz.ee.jmx.jboss.QuartzServiceMBean

**getJobBuilder()** - Method in class org.quartz.impl.JobDetailImpl

Get a JobBuilder that is configured to produce a JobDetail identical to this one.

**getJobBuilder()** - Method in interface org.quartz.JobDetail

**getJobBuilder()** - Method in class org.quartz.locality.DelegatingLocalityJobDetail

Get a JobBuilder that is configured to produce a JobDetail identical to
getJobClass() - Method in class org.quartz.impl.JobDetailImpl
getJobClass() - Method in interface org.quartz.JobDetail
   Get the instance of Job that will be executed.
getJobClass() - Method in class org.quartz.locality.DelegatingLocalityJobDetail
   Get the instance of Job that will be executed.
getJobDataFromBlob(ResultSet, String) - Method in class org.quartz.impl.jdbcjobstore.HSQLDBDelegate
getJobDataFromBlob(ResultSet, String) - Method in class org.quartz.impl.jdbcjobstore.MSSQLDelegate
getJobDataFromBlob(ResultSet, String) - Method in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
   This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.
getJobDataFromBlob(ResultSet, String) - Method in class org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
getJobDataFromBlob(ResultSet, String) - Method in class org.quartz.impl.jdbcjobstore.StdBDBCDelegate
   This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.
getJobDataFromBlob(ResultSet, String) - Method in class org.quartz.impl.jdbcjobstore.SybaseDelegate
getJobDataFromBlob(ResultSet, String) - Method in class org.quartz.impl.jdbcjobstore.WebLogicDelegate
getJobDataMap() - Method in class org.quartz.impl.JobDetailImpl
getJobDataMap() - Method in class org.quartz.impl.triggers.AbstractTrigger
   Get the JobDataMap that is associated with the Trigger.
getJobDataMap() - Method in interface org.quartz.JobDetail
   Get the JobDataMap that is associated with the Job.
**getJobDataMap()** - Method in class org.quartz.locality.DelegatingLocalityJobDetail
Get the JobDataMap that is associated with the Job.

**getJobDataMap()** - Method in class org.quartz.locality.DelegatingLocalityTrigger
Get the JobDataMap that is associated with the Trigger.

**getJobDataMap()** - Method in interface org.quartz.Trigger
Get the JobDataMap that is associated with the Trigger.

**getJobDetail(String, String)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**getJobDetail(JobKey)** - Method in class org.quartz.core.QuartzScheduler
Get the JobDetail for the Job instance with the given name and group.

**getJobDetail(String, String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**getJobDetail(JobKey)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getJobDetail()** - Method in class org.quartz.impl.JobExecutionContextImpl

**getJobDetail(JobKey)** - Method in interface org.quartz.Scheduler
Get the JobDetail for the Job instance with the given key.

**getJobFailedMessage()** - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin
Get the message that is logged when a Job fails its execution.

**getJobGroup()** - Method in class org.quartz.impl.triggers.AbstractTrigger
Get the name of the associated JobDetail's group.

**getJobGroupNames()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**getJobGroupNames()** - Method in class org.quartz.core.QuartzScheduler

Get the names of all known Job groups.

**getJobGroupNames()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

Get the names of all known JobDetail groups.

**getJobGroupNames()** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getJobGroupNames()** - Method in class org.quartz.core.QuartzScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**getJobGroupNames()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Get the names of all the Job groups.

**getJobGroupNames()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Calls the equivalent method on the 'proxied' QuartzScheduler.

**getJobGroupNames()** - Method in class org.quartz.impl.RemoteMBeanScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.

**getJobGroupNames()** - Method in class org.quartz.impl.StdScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.

**getJobGroupNames()** - Method in interface org.quartz.Scheduler

Get the names of all known JobDetail groups.

**getJobGroupsToNeverDelete()** - Method in class org.quartz.xml.XMLSchedulingDataProcessor

Get the (unmodifiable) list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

**getJobInstance()** - Method in class org.quartz.impl.JobExecutionContextImpl

Get the instance of the Job that was created for this execution.

**getJobInstance()** - Method in interface org.quartz.JobExecutionContext

Get the instance of the Job that was created for this execution.
**getJobKey()** - Method in class org.quartz.impl.jdbcjobstore.*FiredTriggerRecord*

**getJobKey()** - Method in class org.quartz.impl.jdbcjobstore.*TriggerStatus*

**getJobKey()** - Method in class org.quartz.impl.triggers.*AbstractTrigger*

**getJobKey()** - Method in class org.quartz.locality.*DelegatingLocalityTrigger*

**getJobKey()** - Method in interface org.quartz.*Trigger*

**getJobKeys(GroupMatcher<JobKey>)** - Method in class org.quartz.core.*QuartzScheduler*

Get the names of all the Jobs in the matching groups.

**getJobKeys(GroupMatcher<JobKey>)** - Method in interface org.quartz.core.*RemotableQuartzScheduler*

**getJobKeys(GroupMatcher<JobKey>)** - Method in class org.quartz.impl.jdbcjobstore.*JobStoreSupport*

Get the names of all of the Jobs that matcher the given groupMatcher.

**getJobKeys(GroupMatcher<JobKey>)** - Method in class org.quartz.impl.*RemoteMBeanScheduler*

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**getJobKeys(GroupMatcher<JobKey>)** - Method in class org.quartz.impl.*RemoteScheduler*

Calls the equivalent method on the 'proxied' QuartzScheduler.

**getJobKeys(GroupMatcher<JobKey>)** - Method in class org.quartz.impl.*StdScheduler*

Calls the equivalent method on the 'proxied' QuartzScheduler.

**getJobKeys(GroupMatcher<JobKey>)** - Method in interface org.quartz.*Scheduler*

Get the keys of all the JobDetails in the matching groups.

**getJobKeys(GroupMatcher<JobKey>)** - Method in class org.quartz.simpl.*RAMJobStore*

Get the names of all of the Jobs that match the given groupMatcher.

**getJobListener(String)** - Method in class org.quartz.core.*ListenerManagerImpl*

Get the JobListener that has the given name.

**getJobListenerMatchers(String)** - Method in class org.quartz.core.*ListenerManagerImpl*
**getJobListenerMatchers(String)** - Method in interface org.quartz.core.ListenerManager

Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

**getJobListeners()** - Method in class org.quartz.core.ListenerManagerImpl

Get a List containing all of the JobListeners in the Scheduler.

**getJobName()** - Method in class org.quartz.impl.triggers.AbstractTrigger

Get the name of the associated JobDetail.

**getJobNames(String)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**getJobNames(String)** - Method in class org.quartz.core.jmx.QuartzSchedulerMBeanImpl

**getJobNames(Connection, GroupMatcher<JobKey>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Get the JobRunShellFactory for the QuartzScheduler to use.

**getJobRunTime()** - Method in class org.quartz.impl.JobExecutionContextImpl

The amount of time the job ran for (in milliseconds).

**getJobsCompletedMostRecentSample()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**getJobsCompletedMostRecentSample()** - Method in class org.quartz.core.NullSampledStatisticsImpl

**getJobsCompletedMostRecentSample()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**getJobsCompletedMostRecentSample()** - Method in interface org.quartz.core.SampledStatistics
**getJobsCompletedMostRecentSample()** - Method in class `org.quartz.core.SampledStatisticsImpl`

**getJobsExecutedMostRecentSample()** - Method in interface `org.quartz.core.jmx.QuartzSchedulerMBean`

**getJobsExecutedMostRecentSample()** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**getJobsExecutingMostRecentSample()** - Method in class `org.quartz.core.NullSampledStatisticsImpl`

**getJobsExecutingMostRecentSample()** - Method in interface `org.quartz.core.SampledStatistics`

**getJobsExecutingMostRecentSample()** - Method in class `org.quartz.core.SampledStatisticsImpl`

**getJobsScheduledMostRecentSample()** - Method in interface `org.quartz.core.jmx.QuartzSchedulerMBean`

**getJobsScheduledMostRecentSample()** - Method in class `org.quartz.core.NullSampledStatisticsImpl`

**getJobsScheduledMostRecentSample()** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**getJobsScheduledMostRecentSample()** - Method in interface `org.quartz.core.SampledStatistics`

**getJobsScheduledMostRecentSample()** - Method in class `org.quartz.core.SampledStatisticsImpl`

**getJobStore()** - Method in class `org.quartz.core.QuartzSchedulerResources`

Get the JobStore for the QuartzScheduler to use.

**getJobStoreClass()** - Method in class `org.quartz.core.QuartzScheduler`

**getJobStoreClass()** - Method in interface
org.quartz.core.**RemotableQuartzScheduler**

**getJobStoreClass()** - Method in class org.quartz.**SchedulerMetaData**

Returns the class-name of the JobStore instance that is being used by the Scheduler.

**getJobStoreClassName()** - Method in interface org.quartz.core.jmx.**QuartzSchedulerMBean**

**getJobStoreClassName()** - Method in class org.quartz.core.**QuartzSchedulerMBeanImpl**

**getJobSuccessMessage()** - Method in class org.quartz.plugins.history.**LoggingJobHistoryPlugin**

Get the message that is logged when a Job successfully completes its execution.

**getJobToBeFiredMessage()** - Method in class org.quartz.plugins.history.**LoggingJobHistoryPlugin**

Get the message that is logged when a Job is about to execute.

**getJobWasVetoedMessage()** - Method in class org.quartz.plugins.history.**LoggingJobHistoryPlugin**

Get the message that is logged when a Job execution is vetoed by a trigger listener.

**getKey()** - Method in class org.quartz.impl.jdbcjobstore.**TriggerStatus**

**getKey()** - Method in class org.quartz.impl.**JobDetailImpl**

**getKey()** - Method in class org.quartz.impl.triggers.**AbstractTrigger**

**getKey()** - Method in interface org.quartz.**JobDetail**

**getKey()** - Method in class org.quartz.locality.**DelegatingLocalityJobDetail**

**getKey()** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

**getKey()** - Method in interface org.quartz.**Trigger**

**getKeyOfNonSerializableValue(Map<?, ?>)** - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**

Find the key of the first non-serializable value in the given Map.

**getKeys()** - Method in class org.quartz.locality.constraint.**EhcacheConstraint**

Accessor
**getKeys()** - Method in class org.quartz.utils.**StringKeyDirtyFlagMap**
Get a copy of the Map's String keys in an array of Strings.

**getKeySet()** - Method in class
org.quartz.locality.constraint.**EhcacheConstraint.Value**

**getLastDayOfMonth(int, int)** - Method in class org.quartz.**CronExpression**

**getLastModifiedDate(String)** - Method in class org.quartz.jobs.**FileScanJob**

**getLeftOperand()** - Method in class org.quartz.impl.matchers.**AndMatcher**

**getLeftOperand()** - Method in class org.quartz.impl.matchers.**OrMatcher**

**getListenerManager()** - Method in class org.quartz.core.**QuartzScheduler**

Calls the equivalent method on the 'proxied' QuartzScheduler.

**getListenerManager()** - Method in class org.quartz.impl.**RemoteScheduler**

**getListenerManager()** - Method in class org.quartz.impl.**RemoteMBeanScheduler**

Get a reference to the scheduler's ListenerManager, through which listeners may be registered.

**getListeners()** - Method in class org.quartz.listeners.**BroadcastJobListener**

**getListeners()** - Method in class org.quartz.listeners.**BroadcastSchedulerListener**

**getListeners()** - Method in class org.quartz.listeners.**BroadcastTriggerListener**

**getLoadedJobs()** - Method in class
org.quartz.xml.**XMLSchedulingDataProcessor**
Returns a List of jobs loaded from the xml file.

**getLoadedTriggers()** - Method in class
org.quartz.xml.**XMLSchedulingDataProcessor**
Returns a List of triggers loaded from the xml file.

**getLockHandler()** - Method in class
org.quartz.impl.jdbcjobstore.**JobStoreSupport**
getLog() - Method in class org.quartz.core.JobRunShell

getLog() - Method in class org.quartz.core.QuartzScheduler

getLog() - Method in class org.quartz.core.QuartzSchedulerThread

getLog() - Method in class org.quartz.impl.DirectSchedulerFactory

getLog() - Method in class org.quartz.impl.jdbcjobstore.AttributeRestoringConnectionInvocationHandler

getLog() - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore

getLog() - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

getLog() - Method in class org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore

getLog() - Method in class org.quartz.impl.jdbcjobstore.SimpleSemaphore

getLog() - Method in class org.quartz.impl.StdSchedulerFactory

getLog() - Method in class org.quartz.jobs.ee.jmx.JMXInvokerJob

getLog() - Method in class org.quartz.jobs.ee.mail.SendMailJob

getLog() - Method in class org.quartz.listeners.JobListenerSupport
  Get the Logger for this class's category.

getLog() - Method in class org.quartz.listeners.SchedulerListenerSupport
  Get the Logger for this class's category.

getLog() - Method in class org.quartz.listeners.TriggerListenerSupport
  Get the Logger for this class's category.

getLog() - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin

getLog() - Method in class
  org.quartz.plugins.history.LoggingTriggerHistoryPlugin
**getLog()** - Method in class `org.quartz.plugins.management.ShutdownHookPlugin`

**getLog()** - Method in class `org.quartz.plugins.SchedulerPluginWithUserTransactionSupport`
   Get the commons Logger for this class.

**getLog()** - Method in class `org.quartz.simpl.RAMJobStore`

**getLog()** - Method in class `org.quartz.simpl.SimpleJobFactory`

**getLog()** - Method in class `org.quartz.simpl.SimpleThreadPool`

**getLog()** - Method in class `org.quartz.simpl.ZeroSizeThreadPool`

**getLog()** - Method in class `org.quartz.utils.JNDIConnectionProvider`

**getLong(String)** - Method in class `org.quartz.utils.StringKeyDirtyFlagMap`
   Retrieve the identified long value from the StringKeyDirtyFlagMap.

**getLong1()** - Method in class `org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties`

**getLong2()** - Method in class `org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties`

**getLongFromString(String)** - Method in class `org.quartz.JobDataMap`
   Retrieve the identified Long value from the JobDataMap.

**getLongProperty(String)** - Method in class `org.quartz.utils.PropertiesParser`

**getLongProperty(String, long)** - Method in class `org.quartz.utils.PropertiesParser`

**getLongValue(String)** - Method in class `org.quartz.JobDataMap`
   Retrieve the identified long value from the JobDataMap.

**getLongValueFromString(String)** - Method in class `org.quartz.JobDataMap`
   Retrieve the identified long value from the JobDataMap.

**getMailSession(SendMailJob.MailInfo)** - Method in class `org.quartz.jobs.ee.mail.SendMailJob`
**getMakeSchedulerThreadDaemon()** - Method in class org.quartz.core.QuartzSchedulerResources
Get whether to mark the Quartz scheduling thread as daemon.

**getMakeThreadsDaemons()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Get whether the threads spawned by this JobStore should be marked as daemon.

**getMaxBatchSize()** - Method in class org.quartz.core.QuartzSchedulerResources

**getMaxMisfiresToHandleAtATime()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Get the maximum number of misfired triggers that the misfire handling thread will try to recover at one time (within one transaction).

**getMaxValue()** - Method in class org.quartz.utils.counter.sampled.SampledRateCounterImpl
Throws UnsupportedOperationException

**getMergedJobDataMap()** - Method in class org.quartz.impl.JobExecutionContextImpl
**getMergedJobDataMap()** - Method in interface org.quartz.JobExecutionContext
Get the convenience JobDataMap of this execution context.

**getMessage()** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo
Returns the detail message string.

**getMessageFactory(String)** - Static method in class org.quartz.jobs.ee.jms.JmsHelper
Creates the JmsMessageFactory

**getMetaData()** - Method in class org.quartz.impl.RemoteMBeanScheduler
**getMetaData()** - Method in class org.quartz.impl.RemoteScheduler
**getMetaData()** - Method in class org.quartz.impl.StdScheduler
**getMetaData()** - Method in interface org.quartz.Scheduler
Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.
**getMinValue()** - Method in class
org.quartz.utils.counter.sampled.**SampledRateCounterImpl**
   throws UnsupportedOperationException

**getMisfireInstruction()** - Method in class
org.quartz.impl.triggers.**AbstractTrigger**
   Get the instruction the Scheduler should be given for handling misfire situations for this Trigger - the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be passed to this method.

**getMisfireInstruction()** - Method in class
org.quartz.locality.**DelegatingLocalityTrigger**
   Get the instruction the Scheduler should be given for handling misfire situations for this Trigger - the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be set as this property's value.

**getMisfireInstruction()** - Method in interface org.quartz.**Trigger**
   Get the instruction the Scheduler should be given for handling misfire situations for this Trigger - the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be set as this property's value.

**getMisfireThreshold()** - Method in class
org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**getMisfireThreshold()** - Method in class org.quartz.simpl.**RAMJobStore**

**getMisfireTime()** - Method in class
org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**getMonthNumber(String)** - Method in class org.quartz.**CronExpression**

**getMostRecentSample()** - Method in interface
org.quartz.utils.counter.sampled.**SampledCounter**
   Returns the most recent sampled value

**getMostRecentSample()** - Method in class
org.quartz.utils.counter.sampled.**SampledCounterImpl**
   Returns the most recent sampled value

**getName()** - Method in class org.quartz.core.**QuartzSchedulerMBeanImpl**

**getName()** - Method in class org.quartz.core.**QuartzSchedulerResources**
Get the name for the QuartzScheduler.

**getName()** - Method in class org.quartz.core. `SampledStatisticsImpl`

**getName()** - Method in class org.quartz.ee.jmx.jboss. `QuartzService`

**getName()** - Method in class org.quartz.impl. `JobDetailImpl`
   
   Get the name of this Job.

**getName()** - Method in class org.quartz.impl.triggers. `AbstractTrigger`
   
   Get the name of this Trigger.

**getName()** - Method in interface org.quartz. `JobListener`
   
   Get the name of the JobListener.

**getName()** - Method in class org.quartz.listeners. `BroadcastJobListener`

**getName()** - Method in class org.quartz.listeners. `BroadcastTriggerListener`

**getName()** - Method in class org.quartz.listeners. `JobChainingJobListener`

**getName()** - Method in class org.quartz.plugins.history. `LoggingJobHistoryPlugin`

**getName()** - Method in class org.quartz.plugins.history. `LoggingTriggerHistoryPlugin`

**getName()** - Method in class org.quartz.plugins. `SchedulerPluginWithUserTransactionSupport`
   
   Get the name of this plugin.

**getName()** - Method in interface org.quartz. `TriggerListener`
   
   Get the name of the TriggerListener.

**getName()** - Method in class org.quartz.utils. `Key`
   
   Get the name portion of the key.

**getNextFireTime()** - Method in class org.quartz.impl.jdbcjobstore. `TriggerStatus`
   
   Get the group portion of the key.

**getNextFireTime()** - Method in class org.quartz.impl. `JobExecutionContextImpl`

**getNextFireTime()** - Method in class org.quartz.impl.triggers. `AbstractTrigger`
   
   Returns the next time at which the Trigger is scheduled to fire.

**getNextFireTime()** - Method in class org.quartz.impl.triggers. `CalendarIntervalTriggerImpl`
   
   Returns the next time at which the Trigger is scheduled to fire.
**getNextFireTime()** - Method in class `org.quartz.impl.triggers.CronTriggerImpl`  
Returns the next time at which the Trigger is scheduled to fire.

**getNextFireTime()** - Method in class `org.quartz.impl.triggers.SimpleTriggerImpl`  
Returns the next time at which the Trigger is scheduled to fire.

**getNextFireTime()** - Method in interface `org.quartz.JobExecutionContext`  

**getNextFireTime()** - Method in class `org.quartz.locality.DelegatingLocalityTrigger`  
Returns the next time at which the Trigger is scheduled to fire.

**getNextFireTime()** - Method in interface `org.quartz.Trigger`  
Returns the next time at which the Trigger is scheduled to fire.

**getNextIncludedTime(long)** - Method in interface `org.quartz.Calendar`  
Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.

**getNextIncludedTime(long)** - Method in class `org.quartz.impl.calendar.AnnualCalendar`  
Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.

**getNextIncludedTime(long)** - Method in class `org.quartz.impl.calendar.BaseCalendar`  
Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.

**getNextIncludedTime(long)** - Method in class `org.quartz.impl.calendar.CronCalendar`  
Determines the next time included by the CronCalendar after the specified time.

**getNextIncludedTime(long)** - Method in class `org.quartz.impl.calendar.DailyCalendar`  
Determines the next time included by the DailyCalendar after the specified time.

**getNextIncludedTime(long)** - Method in class `org.quartz.impl.calendar.HolidayCalendar`  
Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.

**getNextIncludedTime(long)** - Method in class `org.quartz.impl.calendar.MonthlyCalendar`  
Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.
**getNextIncludedTime(long)** - Method in class org.quartz.impl.calendar.**WeeklyCalendar**

Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.

**getNextInvalidTimeAfter(Date)** - Method in class org.quartz.**CronExpression**

Returns the next date/time after the given date/time which does not satisfy the expression.

**getNextValidTimeAfter(Date)** - Method in class org.quartz.**CronExpression**

Returns the next date/time after the given date/time which satisfies the cron expression.

**getNodeGroupForJobGroup(String)** - Method in class org.quartz.locality.constraint.evaluator.**NodeGroupEvaluator**

Retrieves the node group for a particular trigger group.

**getNodeGroupForTriggerGroup(String)** - Method in class org.quartz.locality.constraint.evaluator.**NodeGroupEvaluator**

Retrieves the node group for a particular trigger group.

**getNodeSpec()** - Method in class org.quartz.locality.**DelegatingLocalityJobDetail**

Accessor to the **NodeSpec** instance to evaluate constraints.

**getNodeSpec()** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

Accessor to the **NodeSpec** instance to evaluate constraints.

**getNodeSpec()** - Method in interface org.quartz.locality.**LocalityAware**

Accessor to the **NodeSpec** instance to evaluate constraints.

**getNonManagedTXConnection()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreCMT**

**getNonManagedTXConnection()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**getNonManagedTXConnection()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreTX**

For JobStoreTX, the non-managed TX connection is just the normal connection because it is not CMT.

**getNonManagedTXDataSource()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreCMT**

Get the name of the **DataSource** that should be used for performing database functions.

**getNotificationInfo()** - Method in class org.quartz.core.**QuartzSchedulerMBeanImpl**
**getNumberOfCalendars()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`  
Get the number of Calendar s that are stored in the JobsStore.

**getNumberOfCalendars(Connection)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`  
Get the number of Calendar s that are stored in the JobsStore.

**getNumberOfJobs()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`  
Get the number of Job s that are stored in the JobStore.

**getNumberOfJobs(Connection)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`  
Get the number of Job s that are stored in the JobStore.

**getNumberOfTriggers()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`  
Get the number of Trigger s that are stored in the JobStore.

**getNumberOfTriggers(Connection)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`  
Get the number of Trigger s that are stored in the JobStore.

**getNumericValue(String, int)** - Method in class `org.quartz.CronExpression`  

**getObjectFromBlob(ResultSet, String)** - Method in class `org.quartz.impl.jdbcjobstore.CloudscapeDelegate`  
*Deprecated.* This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**getObjectFromBlob(ResultSet, String)** - Method in class `org.quartz.impl.jdbcjobstore.HSQLDBDelegate`  
This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**getObjectFromBlob(ResultSet, String)** - Method in class `org.quartz.impl.jdbcjobstore.MSSQLDelegate`  

**getNumberOfJobsExecuted()** - Method in class `org.quartz.SchedulerMetaData`  
Returns the number of jobs executed since the Scheduler started..
This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**getObjectFromBlob(ResultSet, String)** - Method in class org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**

**getObjectFromBlob(ResultSet, String)** - Method in class org.quartz.impl.jdbcjobstore.**PointbaseDelegate**

This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**getObjectFromBlob(ResultSet, String)** - Method in class org.quartz.impl.jdbcjobstore.**PostgreSQLDelegate**

This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**getObjectFromBlob(ResultSet, String)** - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**

This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**getObjectFromBlob(ResultSet, String)** - Method in class org.quartz.impl.jdbcjobstore.**SybaseDelegate**

This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**getOp()** - Method in class org.quartz.locality.constraint.**CpuConstraint**

**getOp()** - Method in class org.quartz.locality.constraint.**MemoryConstraint**

Getter to operator

**getOperand()** - Method in class org.quartz.impl.matchers.**NotMatcher**

**getOptionalParm(JobDataMap, String)** - Method in class org.quartz.jobs.ee.mail.**SendMailJob**

**getPausedTriggerGroups()** - Method in interface org.quartz.core.jmx.**QuartzSchedulerMBean**

**getPausedTriggerGroups()** - Method in class org.quartz.core.**QuartzScheduler**
**getPausedTriggerGroups()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**getPausedTriggerGroups()** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getPausedTriggerGroups()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**getPausedTriggerGroups()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

  Pause all of the Triggers in the given group.

**getPausedTriggerGroups()** - Method in class org.quartz.impl.RemoteMBeanScheduler

**getPausedTriggerGroups()** - Method in class org.quartz.impl.RemoteScheduler

**getPausedTriggerGroups()** - Method in class org.quartz.impl.StdScheduler

**getPausedTriggerGroups()** - Method in interface org.quartz.Scheduler

  Get the names of all Trigger groups that are paused.

**getPausedTriggerGroups()** - Method in class org.quartz.simpl.RAMJobStore

**getPerformanceMetrics()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**getPerformanceMetrics()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**getPoolSize()** - Method in class org.quartz.simpl.SimpleThreadPool

**getPoolSize()** - Method in class org.quartz.simpl.ZeroSizeThreadPool

**getPreviousFireTime()** - Method in class org.quartz.impl.JobExecutionContextImpl

**getPreviousFireTime()** - Method in class org.quartz.impl.triggers.AbstractTrigger

  Returns the previous time at which the Trigger fired.
**getPreviousFireTime()** - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl
Returns the previous time at which the DateIntervalTrigger fired.

**getPreviousFireTime()** - Method in class org.quartz.impl.triggers.CronTriggerImpl
Returns the previous time at which the CronTrigger fired.

**getPreviousFireTime()** - Method in class org.quartz.impl.triggers.SimpleTriggerImpl
Returns the previous time at which the SimpleTrigger fired.

**getPreviousFireTime()** - Method in interface org.quartz.JobExecutionContext

**getPreviousFireTime()** - Method in class org.quartz.locality.DelegatingLocalityTrigger
Returns the previous time at which the Trigger fired.

**getPreviousFireTime()** - Method in interface org.quartz.Trigger
Returns the previous time at which the Trigger fired.

**getPriority()** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord
The priority of a Trigger acts as a tiebreaker such that if two Triggers have the same scheduled fire time, then the one with the higher priority will get first access to a worker thread.

**getPriority()** - Method in class org.quartz.locality.DelegatingLocalityTrigger
The priority of a Trigger acts as a tiebreaker such that if two Triggers have the same scheduled fire time, then the one with the higher priority will get first access to a worker thread.

**getPriority()** - Method in interface org.quartz.Trigger
The priority of a Trigger acts as a tiebreaker such that if two Triggers have the same scheduled fire time, then the one with the higher priority will get first access to a worker thread.


**getProperties()** - Method in class org.quartz.ee.jmx.jboss.QuartzService

**getPropertiesFile()** - Method in class org.quartz.ee.jmx.jboss.QuartzService

**getPropertyGroup(String)** - Method in class org.quartz.utils.PropertiesParser
getPropertyGroup(String, boolean) - Method in class org.quartz.utils.PropertiesParser

g.getPropertyGroup(String, boolean, String[]) - Method in class org.quartz.utils.PropertiesParser
   Get all properties that start with the given prefix.

getPropertyGroups(String) - Method in class org.quartz.utils.PropertiesParser

getRefireCount() - Method in class org.quartz.impl.JobExecutionContextImpl

getRefireCount() - Method in interface org.quartz.JobExecutionContext


getRepeatCount() - Method in class org.quartz.impl.triggers.SimpleTriggerImpl
   Get the number of times the SimpleTrigger should repeat, after which it will be automatically deleted.

getRepeatInterval() - Method in interface org.quartz.CalendarIntervalTrigger
   Get the time interval that will be added to the DateIntervalTrigger's fire time (in the set repeat interval unit) in order to calculate the time of the next trigger repeat.

getRepeatInterval() - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl

getRepeatInterval() - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl

getRepeatInterval() - Method in interface org.quartz.SimpleTrigger
   Get the time interval (in milliseconds) at which the SimpleTrigger should repeat.

getRepeatIntervalUnit() - Method in interface org.quartz.CalendarIntervalTrigger
   Get the interval unit - the time unit on with the interval applies.

getRepeatIntervalUnit() - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl

getReplyTo() - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo
**getRequiredParm(JobDataMap, String, String)** - Method in class org.quartz.jobs.ee.mail.SendMailJob

**getResource(String)** - Method in class org.quartz.simpl.CascadingClassLoadHelper
Finds a resource with a given name.

**getResource(String)** - Method in class org.quartz.simpl.InitThreadContextClassLoadHelper
Finds a resource with a given name.

**getResource(String)** - Method in class org.quartz.simpl.LoadingLoaderClassLoadHelper
Finds a resource with a given name.

**getResource(String)** - Method in class org.quartz.simpl.SimpleClassLoadHelper
Finds a resource with a given name.

**getResourceAsStream(String)** - Method in class org.quartz.simpl.CascadingClassLoadHelper
Finds a resource with a given name.

**getResourceAsStream(String)** - Method in class org.quartz.simpl.InitThreadContextClassLoadHelper
Finds a resource with a given name.

**getResourceAsStream(String)** - Method in class org.quartz.simpl.LoadingLoaderClassLoadHelper
Finds a resource with a given name.

**getResourceAsStream(String)** - Method in class org.quartz.simpl.SimpleClassLoadHelper
Finds a resource with a given name.

**getResourceAsStream(String)** - Method in class org.quartz.simpl.ThreadContextClassLoadHelper
Finds a resource with a given name.

**getResult()** - Method in class org.quartz.impl.JobExecutionContextImpl
Returns the result (if any) that the Job set before its execution completed (the type of object set as the result is entirely up to the particular job).

**getResult()** - Method in interface org.quartz.JobExecutionContext

**getRightOperand()** - Method in class org.quartz.impl.matchers.AndMatcher
**getRightOperand()** - Method in class `org.quartz.impl.matchers.OrMatcher`

**getRMIBindName()** - Method in class `org.quartz.core.QuartzSchedulerResources`
  Get the name under which to bind the QuartzScheduler in RMI.

**getRMICreateRegistryStrategy()** - Method in class `org.quartz.core.QuartzSchedulerResources`
  Get the setting of whether or not Quartz should create an RMI Registry, and if so, how.

**getRMIRegistryHost()** - Method in class `org.quartz.core.QuartzSchedulerResources`
  Get the host name of the RMI Registry that the scheduler should export itself to.

**getRMIRegistryPort()** - Method in class `org.quartz.core.QuartzSchedulerResources`
  Get the port number of the RMI Registry that the scheduler should export itself to.

**getRMIServerPort()** - Method in class `org.quartz.core.QuartzSchedulerResources`
  Get the port number the scheduler server will be bound to.

**getRunningSince()** - Method in class `org.quartz.SchedulerMetaData`
  Returns the date at which the Scheduler started running.

**getScanInterval()** - Method in class `org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin`
  The interval (in seconds) at which to scan for changes to the file.

**getSchedName()** - Method in class `org.quartz.impl.jdbcjobstore.DBSemaphore`

**getScheduleBuilder()** - Method in class `org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPropertyBundle`

**getScheduleBuilder()** - Method in class `org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPropertyBundle`

**getScheduleBuilder()** - Method in class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`
  Get a `ScheduleBuilder` that is configured to produce a schedule identical to this trigger's schedule.

**getScheduleBuilder()** - Method in class `org.quartz.impl.triggers.CronTriggerImpl`
Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

**getScheduleBuilder()** - Method in class org.quartz.impl.triggers. SimpleTriggerImpl
Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

**getScheduleBuilder()** - Method in class org.quartz.locality. DelegatingLocalityTrigger
Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

**getScheduleBuilder()** - Method in interface org.quartz. Trigger
Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

**getScheduledFireTime()** - Method in class org.quartz.impl. JobExecutionContextImpl

The scheduled time the trigger fired for.

**getScheduler()** - Method in class org.quartz.impl. DirectSchedulerFactory
Returns a handle to the Scheduler produced by this factory.

**getScheduler(String)** - Method in class org.quartz.impl. DirectSchedulerFactory
Returns a handle to the Scheduler with the given name, if it exists.

**getScheduler()** - Method in class org.quartz.impl. JobExecutionContextImpl

Get a handle to the Scheduler instance that fired the Job.

**getScheduler()** - Method in class org.quartz.plugins. SchedulerPluginWithUserTransactionSupport
Get this plugin's Scheduler.

**getScheduler()** - Method in interface org.quartz. SchedulerFactory
Returns a client usable handle to a Scheduler.

**getScheduler(String)** - Method in interface org.quartz. SchedulerFactory
Returns a handle to the Scheduler with the given name, if it exists.

**getSchedulerClass()** - Method in class org.quartz. SchedulerMetaData
Returns the class-name of the Scheduler instance.

**getSchedulerContext()** - Method in class org.quartz.core.QuartzScheduler

Returns the SchedulerContext of the Scheduler.

**getSchedulerContext()** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getSchedulerInstanceId()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**getSchedulerInstanceId()** - Method in class org.quartz.core.QuartzScheduler

Returns the instance Id of the QuartzScheduler.

**getSchedulerInstanceId()** - Method in class org.quartz.core.RemotableQuartzScheduler

**getSchedulerInstanceId()** - Method in class org.quartz.core.jmx.FiredTriggerRecord

**getSchedulerInstanceId()** - Method in class org.quartz.core.jdbcjobstore.SchedulerStateRecord

**getSchedulerInstanceId()** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

**getSchedulerInstanceId()** - Method in class org.quartz.impl.RemoteMBeanScheduler

Returns the instance Id of the Scheduler.

**getSchedulerInstanceId()** - Method in class org.quartz.impl.RemoteScheduler

Returns the instance Id of the Scheduler.

**getSchedulerInstanceId()** - Method in class org.quartz.impl.StdScheduler

Returns the instance Id of the Scheduler.

**getSchedulerInstanceId()** - Method in interface org.quartz.Scheduler

Returns the instance Id of the Scheduler.

**getSchedulerInstanceId()** - Method in class org.quartz.SchedulerMetaData

Returns the instance Id of the Scheduler.

**getSchedulerListeners()** - Method in class org.quartz.core.ListenerManagerImpl

Get a List containing all of the SchedulerListeners registered with the
getSchedulerName() - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

Returns the name of the QuartzScheduler.

getSchedulerName() - Method in class org.quartz.core.QuartzScheduler

Returns the name of the Scheduler.

getSchedulerName() - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

getSchedulerName() - Method in interface org.quartz.core.RemotableQuartzScheduler

getSchedulerName() - Method in class org.quartz.impl.RemoteMBeanScheduler

Returns the name of the Scheduler.

getSchedulerName() - Method in class org.quartz.impl.RemoteScheduler

Returns the name of the Scheduler.

getSchedulerName() - Method in class org.quartz.impl.StdScheduler

Returns the name of the Scheduler.

getSchedulerName() - Method in interface org.quartz.Scheduler

Returns the name of the Scheduler.

getSchedulerNameLiteral() - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore

Returns the name of the Scheduler.

getSchedulerNameLiteral() - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

getSchedulerObjectName() - Method in class org.quartz.impl.RemoteMBeanScheduler

Get the name under which the Scheduler MBean is registered on the remote MBean server.

getSchedulerPlugins() - Method in class org.quartz.core.QuartzSchedulerResources

Get the List of all SchedulerPlugins for the QuartzScheduler to use.

getSchedulerSignaler() - Method in class org.quartz.core.QuartzScheduler

getSchedulerThreadGroup() - Method in class
org.quartz.core.QuartzScheduler
  Returns the name of the thread group for Quartz's main threads.

getSelectWithLockSQL() - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

getSelectWithLockSQL() - Method in class org.quartz.impl.jdbcjobstore.StdRowLockSemaphore

getSet(int) - Method in class org.quartz.CronExpression

getShortProperty(String) - Method in class org.quartz.utils.PropertiesParser

getShortProperty(String, short) - Method in class org.quartz.utils.PropertiesParser

getSignaledNextFireTime() - Method in class org.quartz.core.QuartzSchedulerThread

getSmtpHost() - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

getSQL() - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore

getStartOfDayJavaCalendar(long) - Method in class org.quartz.impl.calendar.BaseCalendar
  Returns the start of the given day as a Calendar.

getStartScheduler() - Method in class org.quartz.ee.jmx.jboss.QuartzService

getStartTime() - Method in class org.quartz.impl.triggers.AbstractTrigger
  Get the time at which the Trigger should occur.

getStartTime() - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl
  Get the time at which the TimeIntervalTrigger should occur.

getStartTime() - Method in class org.quartz.impl.triggers.CronTriggerImpl
  Get the time at which the CronTrigger should occur.

getStartTime() - Method in class org.quartz.impl.triggers.SimpleTriggerImpl
  Get the time at which the SimpleTrigger should occur.

getStartTime() - Method in class org.quartz.locality.DelegatingLocalityTrigger
  Get the time at which the Trigger should occur.

getStartTime() - Method in interface org.quartz.Trigger
Get the time at which the Trigger should occur.

**getStateChangedPropertyNames()** - Method in class
org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPropertyBundle

**getStateChangedPropertyValues()** - Method in class
org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPropertyBundle

**getStatus()** - Method in class org.quartz.impl.jdbcjobstore.TriggerStatus
  Get the status of the key.

**getStateChangedPropertyValues()** - Method in class
org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPropertyBundle

**getString(String)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Retrieve the identified string value from the StringKeyDirtyFlagMap.

**getString1()** - Method in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**getString2()** - Method in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**getString3()** - Method in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**getStringArrayProperty(String)** - Method in class
org.quartz.utils.PropertiesParser

**getStringArrayProperty(String, String[])** - Method in class
org.quartz.utils.PropertiesParser

**getStringProperty(String)** - Method in class org.quartz.utils.PropertiesParser
  Get the trimmed String value of the property with the given name.

**getStringProperty(String, String)** - Method in class
org.quartz.utils.PropertiesParser
  Get the trimmed String value of the property with the given name or the
given default value if the value is null or empty after trimming.

**getSubject()** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**getSummary()** - Method in class org.quartz.SchedulerMetaData
  Returns a formatted (human readable) String describing all the Scheduler's
meta-data values.

**getSystemIdForFileName(String)** - Method in class org.quartz.xml.XMLSchedulingDataProcessor

For the given **fileName**, attempt to expand it to its full path for use as a system id.

**getTablePrefix()** - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore

**getTablePrefix()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Get the prefix that should be pre-pended to all table names.

**getTargetedOs()** - Method in class org.quartz.locality.constraint.OsConstraint

Getter to targeted OS

**getTargetNodeGroup()** - Method in class org.quartz.locality.constraint.NodeGroupConstraint

Getter to the target node group

**getThreadCount()** - Method in class org.quartz.simpl.SimpleThreadPool

Get the number of worker threads in the pool.

**getThreadName()** - Method in class org.quartz.core.QuartzSchedulerResources

Get the name for the QuartzSchedulerThread.

**getThreadNamePrefix()** - Method in class org.quartz.simpl.SimpleThreadPool

**getThreadPool()** - Method in class org.quartz.core.QuartzSchedulerResources

Get the ThreadPool for the QuartzScheduler to use.

**getThreadPoolClass()** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getThreadPoolClass()** - Method in class org.quartz.SchedulerMetaData

Returns the class-name of the ThreadPool instance that is being used by the Scheduler.

**getThreadPoolClassName()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**getThreadPoolClassName()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**getThreadPoolSize()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean
**getThreadPoolSize()** - Method in class `org.quartz.core.QuartzScheduler`

**getThreadPoolSize()** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**getThreadPoolSize()** - Method in interface `org.quartz.core.RemotableQuartzScheduler`

**getThreadPoolSize()** - Method in class `org.quartz.SchedulerMetaData`

  Returns the number of threads currently in the Scheduler's ThreadPool.

**getThreadPriority()** - Method in class `org.quartz.simpl.SimpleThreadPool`

  Get the thread priority of worker threads in the pool.

**getTimeAfter(Date)** - Method in class `org.quartz.CronExpression`

**getTimeBefore(Date)** - Method in class `org.quartz.impl.triggers.CronTriggerImpl`

  NOT YET IMPLEMENTED: Returns the time before the given time that this CronTrigger will fire.

**getTimeRangeEndingTimeInMillis(long)** - Method in class `org.quartz.impl.calendar.DailyCalendar`

  Returns the end time of the time range (in milliseconds) of the day specified in `timeInMillis`

**getTimeRangeStartingTimeInMillis(long)** - Method in class `org.quartz.impl.calendar.DailyCalendar`

  Returns the start time of the time range (in milliseconds) of the day specified in `timeInMillis`

**getTimerTask()** - Method in class `org.quartz.utils.counter.sampled.SampledCounterImpl`

  Returns the timer task for this sampled counter

**getTimestamp()** - Method in class `org.quartz.utils.counter.sampled.TimeStampedCounterValue`

  Get value of the timestamp

**getTimesTriggered()** - Method in interface `org.quartz.CalendarIntervalTrigger`
Get the number of times the `DateIntervalTrigger` has already fired.

**getTimesTriggered()** - Method in class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`

**getTimesTriggered()** - Method in class `org.quartz.impl.triggers.SimpleTriggerImpl`

Get the number of times the `SimpleTrigger` has already fired.

**getTimesTriggered()** - Method in interface `org.quartz.SimpleTrigger`

Get the number of times the `SimpleTrigger` has already fired.

**getTimeZone()** - Method in class `org.quartz.CronExpression`

Returns the time zone for which this `CronExpression` will be resolved.

**getTimeZone()** - Method in interface `org.quartz.CronTrigger`

Returns the time zone for which the `cronExpression` of this `CronTrigger` will be resolved.

**getTimeZone()** - Method in class `org.quartz.impl.calendar.BaseCalendar`

Returns the time zone for which this `Calendar` will be resolved.

**getTimeZone()** - Method in class `org.quartz.impl.calendar.CronCalendar`

Returns the time zone for which the `CronExpression` of this `CronCalendar` will be resolved.

**getTimeZone()** - Method in class `org.quartz.impl.triggers.CronTriggerImpl`

**getTo()** - Method in class `org.quartz.jobs.ee.mail.SendMailJob.MailInfo`

**getTransaction()** - Method in class `org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore`

Helper method to get the current `Transaction` from the `TransactionManager` in JNDI.

**getTrigger(String, String)** - Method in interface `org.quartz.core.jmx.QuartzSchedulerMBean`

**getTrigger(TriggerKey)** - Method in class `org.quartz.core.QuartzScheduler`

Get the `Trigger` instance with the given name and group.

**getTrigger(TriggerKey)** - Method in class `org.quartz.core.QuartzSchedulerImpl`

**getTrigger(TriggerKey)** - Method in interface `org.quartz.core.RemotableQuartzScheduler`

**getTrigger()** - Method in class `org.quartz.impl.JobExecutionContextImpl`
getTrigger(TriggerKey) - Method in class org.quartz.impl.RemoteMBeanScheduler
    Calls the equivalent method on the 'proxied' QuartzScheduler.
getTrigger(TriggerKey) - Method in class org.quartz.impl.RemoteScheduler
    Calls the equivalent method on the 'proxied' QuartzScheduler.
getTrigger(TriggerKey) - Method in class org.quartz.impl.StdScheduler
    Calls the equivalent method on the 'proxied' QuartzScheduler.
getTrigger() - Method in interface org.quartz.JobExecutionContext
    Get a handle to the Trigger instance that fired the Job.
getTrigger(TriggerKey) - Method in interface org.quartz.Scheduler
    Get the Trigger instance with the given key.
getTriggerBuilder() - Method in interface org.quartz.CalendarIntervalTrigger
getTriggerBuilder() - Method in interface org.quartz.CronTrigger
getTriggerBuilder() - Method in interface org.quartz.SimpleTrigger
getTriggerBuilder() - Method in class org.quartz.impl.triggers.AbstractTrigger
    Get a TriggerBuilder that is configured to produce a Trigger identical to this one.
getTriggerBuilder() - Method in interface org.quartz.Scheduler
    Get a TriggerBuilder that is configured to produce a Trigger identical to this one.
getTriggerCompleteMessage() - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin
    Get the message that is printed upon the completion of a trigger's firing.
getTriggerFiredMessage() - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin
    Get the message that is printed upon a trigger's firing.
getTriggerGroupNames() - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean
    Get the names of all known Trigger groups.
getTriggerGroupNames() - Method in class org.quartz.core.QuartzScheduler

org.quartz.core.QuartzSchedulerMBeanImpl

**getTriggerGroupNames()** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getTriggerGroupNames()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
   Get the names of all of the Trigger groups.

**getTriggerGroupNames(Connection)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**getTriggerGroupNames()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
   Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**getTriggerGroupNames()** - Method in class org.quartz.impl.RemoteMBeanScheduler
   Calls the equivalent method on the 'proxied' QuartzScheduler.

**getTriggerGroupNames()** - Method in class org.quartz.impl.StdScheduler
   Get the names of all known Trigger groups.

**getTriggerGroupNames()** - Method in class org.quartz.simpl.RAMJobStore
   Get the names of all of the Trigger groups.

**getTriggerGroupsToNeverDelete()** - Method in class org.quartz.xml.XMLSchedulingDataProcessor
   Get the (unmodifiable) list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

**getTriggerKey()** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

**getTriggerKeys(GroupMatcher<TriggerKey>)** - Method in class org.quartz.core.QuartzScheduler
   Get the names of all the Triggers in the matching groups.

**getTriggerKeys(GroupMatcher<TriggerKey>)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**getTriggerKeys(GroupMatcher<TriggerKey>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Get the names of all of the Triggers that match the given group Matcher.

`getTriggerKeys(GroupMatcher<TriggerKey>)` - Method in class org.quartz.impl.RemoteMBeanScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

`getTriggerKeys(GroupMatcher<TriggerKey>)` - Method in class org.quartz.impl.RemoteScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.

`getTriggerKeys(GroupMatcher<TriggerKey>)` - Method in class org.quartz.impl.StdScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.

`getTriggerKeys(GroupMatcher<TriggerKey>)` - Method in interface org.quartz.Scheduler

Get the names of all the Triggers in the given group.

`getTriggerKeys(GroupMatcher<TriggerKey>)` - Method in class org.quartz.simpl.RAMJobStore

Get the names of all of the Triggers that match the given groupMatcher.

`getTriggerListener(String)` - Method in class org.quartz.core.ListenerManagerImpl

Get the TriggerListener that has the given name.

`getTriggerListenerMatchers(String)` - Method in interface org.quartz.ListenerManager

Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

`getTriggerListeners()` - Method in class org.quartz.core.ListenerManagerImpl

Get a List containing all of the TriggerListeners in the Scheduler.

`getTriggerMisfiredMessage()` - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin

Get the message that is printed upon a trigger's mis-firing.

`getTriggerNames(String)` - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean
**getTriggerNames(String)** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**getTriggerNames(Connection, GroupMatcher<TriggerKey>)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**getTriggerProperties(OperableTrigger)** - Method in class `org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate`

**getTriggerProperties(OperableTrigger)** - Method in class `org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport`

**getTriggerPropertyBundle(SimplePropertiesTriggerProperties)** - Method in class `org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate`

**getTriggerPropertyBundle(SimplePropertiesTriggerProperties)** - Method in class `org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport`

**getTriggersForJob(JobKey)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

Get all of the Triggers that are associated to the given Job.

**getTriggersForJob(Connection, JobKey)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**getTriggersForJob(JobKey)** - Method in class `org.quartz.simpl.RAMJobStore`

Get all of the Triggers that are associated to the given Job.

**getTriggersOfJob(String, String)** - Method in interface `org.quartz.core.jmx.QuartzSchedulerMBean`

**getTriggersOfJob(JobKey)** - Method in class `org.quartz.core.QuartzScheduler`

Get all Triggers that are associated with the identified JobDetail.

**getTriggersOfJob(String, String)** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**getTriggersOfJob(JobKey)** - Method in interface `org.quartz.core.RemotableQuartzScheduler`

**getTriggersOfJob(JobKey)** - Method in class
org.quartz.impl.RemoteMBeanScheduler
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

getTriggersOfJob(JobKey) - Method in class org.quartz.impl.RemoteScheduler
    Calls the equivalent method on the 'proxied' QuartzScheduler.

getTriggersOfJob(JobKey) - Method in class org.quartz.impl.StdScheduler
    Calls the equivalent method on the 'proxied' QuartzScheduler.

getTriggersOfJob(JobKey) - Method in interface org.quartz.Scheduler
    Get all Trigger s that are associated with the identified JobDetail.

getTriggersOfJob(JobKey) - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

getTriggerState(TriggerKey) - Method in class org.quartz.core.QuartzScheduler
    Get the current state of the identified Trigger.

getTriggerState(TriggerKey) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

getTriggerState(TriggerKey) - Method in interface org.quartz.core.RemotableQuartzScheduler

getTriggerState(TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
    Get the current state of the identified Trigger.

getTriggerState(TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

getTriggerState(TriggerKey) - Method in interface org.quartz.Scheduler
    Get the current state of the identified Trigger.

getTriggerState(TriggerKey) - Method in class org.quartz.simpl.RAMJobStore
Get the current state of the identified Trigger.

**getTriggerWrappersForCalendar(String)** - Method in class org.quartz.simpl.RAMJobStore

**getTriggerWrappersForJob(JobKey)** - Method in class org.quartz.simpl.RAMJobStore

**getTrimmedToNullString(XPath, String, Node)** - Method in class org.quartz.xml.XMLSchedulingDataProcessor

**getUnderlyingException()** - Method in exception org.quartz.SchedulerException
  Return the exception that is the underlying cause of this exception.

**getUnderlyingProperties()** - Method in class org.quartz.util.PropertiesParser

**getUniqueIdentifier(String, String)** - Static method in class org.quartz.core.QuartzSchedulerResources

**getUniqueIdentifier()** - Method in class org.quartz.core.QuartzSchedulerResources

**getUpdatedOrNewFiles(String, long, long)** - Method in class org.quartz.jobs.DirectoryScanJob

**getUpdateLockRowSQL()** - Method in class org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore

**getURL(String)** - Method in class org.quartz.xml.XMLSchedulingDataProcessor
  Returns an URL from the fileName as a resource.

**getUseDBLocks()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Get whether this instance should use database-based thread synchronization.

**getUserTxLocation()** - Static method in class org.quartz.ee.jta.UserTransactionHelper

**getValidationExceptions()** - Method in exception org.quartz.xml.ValidationException
Returns collection of errors.

`getValue(int, String, int)` - Method in class `org.quartz.CronExpression`

`getValue(T)` - Method in class `org.quartz.impl.matchers.GroupMatcher`

`getValue(T)` - Method in class `org.quartz.impl.matchers.NameMatcher`

`getValue(T)` - Method in class `org.quartz.impl.matchers.StringMatcher`

`getValue()` - Method in class `org.quartz.locality.constraint.CpuConstraint`

`getValue()` - Method in interface `org.quartz.utils.counter.Counter`

`getValue()` - Method in class `org.quartz.utils.counter.CounterImpl`

`getValue()` - Method in class `org.quartz.utils.counter.sampled.SampledRateCounterImpl`

`getVersion()` - Method in interface `org.quartz.core.jmx.QuartzSchedulerMBean`

`getVersion()` - Method in class `org.quartz.core.QuartzScheduler`

`getVersion()` - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

`getVersion()` - Method in interface `org.quartz.core.RemotableQuartzScheduler`

`getVersion()` - Method in class `org.quartz.SchedulerMetaData`

Returns the version of Quartz that is running.

`getVersionIteration()` - Static method in class `org.quartz.core.QuartzScheduler`

`getVersionMajor()` - Static method in class `org.quartz.core.QuartzScheduler`

`getVersionMinor()` - Static method in class `org.quartz.core.QuartzScheduler`

`getWrapInUserTransaction()` - Method in class

`getWrappedConnection()` - Method in class

Wrap the start() and shutdown() methods in a UserTransaction.

`AttributeValueRestoringConnectionInvocationHandler`
Gets the underlying connection to which all operations ultimately defer. 

**getWrappedMap()** - Method in class org.quartz.utils.DirtyFlagMap
Get a direct handle to the underlying Map.

**groupContains(String)** - Static method in class org.quartz.impl.matchers.GroupMatcher
Create a GroupMatcher that matches groups containing the given string.

**groupEndsWith(String)** - Static method in class org.quartz.impl.matchers.GroupMatcher
Create a GroupMatcher that matches groups ending with the given string.

**groupEquals(String)** - Static method in class org.quartz.impl.matchers.GroupMatcher
Create a GroupMatcher that matches groups equaling the given string.

**GroupMatcher<T extends Key>** - Class in org.quartz.impl.matchers
Matches on group (ignores name) property of Keys.

**GroupMatcher(String, StringMatcher.StringOperatorName)** - Constructor for class org.quartz.impl.matchers.GroupMatcher

**groupStartsWith(String)** - Static method in class org.quartz.impl.matchers.GroupMatcher
Create a GroupMatcher that matches groups starting with the given string.
has(Constraint) - Method in class org.quartz.locality. NodeSpecBuilder
DSL method to add a constraint

hasAdditionalProperties() - Method in class org.quartz.impl.triggers. CalendarIntervalTriggerImpl

hasAdditionalProperties() - Method in interface org.quartz.impl.triggers. CoreTrigger

hasAdditionalProperties() - Method in class org.quartz.impl.triggers. CronTriggerImpl
Used by extensions of CronTrigger to imply that there are additional properties, specifically so that extensions can choose whether to be stored as a serialized blob, or as a flattened CronTrigger table.

hasAdditionalProperties() - Method in class org.quartz.impl.triggers. SimpleTriggerImpl
Used by extensions of SimpleTrigger to imply that there are additional properties, specifically so that extensions can choose whether to be stored as a serialized blob, or as a flattened SimpleTrigger table.

hasAvailableThread(String) - Method in class org.quartz.locality.constraint.evaluator. CpuEvaluator

hasAvailableThread() - Method in class org.quartz.locality.constraint.evaluator. CpuEvaluator

hashCode() - Method in class org.quartz.impl. JobDetailImpl

hashCode() - Method in class org.quartz.impl.matchers. AndMatcher

hashCode() - Method in class org.quartz.impl.matchers. EverythingMatcher

hashCode() - Method in class org.quartz.impl.matchers. KeyMatcher

hashCode() - Method in class org.quartz.impl.matchers. NotMatcher

hashCode() - Method in class org.quartz.impl.matchers. OrMatcher
hashCode() - Method in class org.quartz.impl.matchers.StringMatcher

hashCode() - Method in class org.quartz.impl.triggers.AbstractTrigger

hashCode() - Method in class org.quartz.locality.constraint.CpuConstraint

hashCode() - Method in class org.quartz.locality.constraint.EhcacheConstraint.Value

hashCode() - Method in class org.quartz.locality.constraint.MemoryConstraint

hashCode() - Method in class org.quartz.locality.constraint.NodeGroupConstraint

hashCode() - Method in class org.quartz.locality.constraint.OsConstraint

hashCode() - Method in interface org.quartz.Matcher

hashCode() - Method in class org.quartz.utils.DirtyFlagMap

hashCode() - Method in class org.quartz.utils.Key

hashCode() - Method in class org.quartz.utils.StringKeyDirtyFlagMap

hasMisfiredTriggersInState(Connection, String, long, int, List<TriggerKey>) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
  Get the names of all of the triggers in the given states that have misfired - according to the given timestamp.

hasMisfiredTriggersInState(Connection, String, long, int, List<TriggerKey>) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  Get the names of all of the triggers in the given state that have misfired - according to the given timestamp.


history - Variable in class org.quartz.utils.counter.sampled.SampledCounterImpl
The history of this counter

**HolidayCalendar** - Class in `org.quartz.impl.calendar`  
This implementation of the Calendar stores a list of holidays (full days that are excluded from scheduling).

**HolidayCalendar()** - Constructor for class  
`org.quartz.impl.calendar.HolidayCalendar`

**HolidayCalendar(Calendar)** - Constructor for class  
`org.quartz.impl.calendar.HolidayCalendar`

**HolidayCalendar(TimeZone)** - Constructor for class  
`org.quartz.impl.calendar.HolidayCalendar`

**HolidayCalendar(Calendar, TimeZone)** - Constructor for class  
`org.quartz.impl.calendar.HolidayCalendar`

**HostnameInstanceIdGenerator** - Class in `org.quartz.simpl`  
InstanceIdGenerator that names the scheduler instance using just the machine hostname.

**HostnameInstanceIdGenerator()** - Constructor for class  
`org.quartz.simpl.HostnameInstanceIdGenerator`

**HOUR** - Static variable in class `org.quartz.CronExpression`

**hours** - Variable in class `org.quartz.CronExpression`

**HSQLDBDelegate** - Class in `org.quartz.impl.jdbcjobstore`  
This is a driver delegate for the HSQLDB database.

**HSQLDBDelegate(Logger, String, String, String, ClassLoadHelper)** - Constructor for class  
`org.quartz.impl.jdbcjobstore.HSQLDBDelegate`
Create new HSQLDBDelegate instance.

**HSQLDBDelegate(Logger, String, String, String, ClassLoadHelper, Boolean)** - Constructor for class  
`org.quartz.impl.jdbcjobstore.HSQLDBDelegate`
Create new MSSQLDelegate instance.
**increment()** - Method in interface org.quartz.utils.counter.Counter
Increment the counter by 1

**increment(long)** - Method in interface org.quartz.utils.counter.Counter
Increment the counter by given amount

**increment()** - Method in class org.quartz.utils.counter.CounterImp
Increment the counter by 1

**increment(long)** - Method in class org.quartz.utils.counter.CounterImp
Increment the counter by given amount

**increment(long, long)** - Method in interface org.quartz.utils.counter.sampled.SampledRateCounter
Increments the numerator and denominator by the passed values

**increment(long, long)** - Method in class org.quartz.utils.counter.sampled.SampledRateCounterImp
Increments the numerator and denominator by the passed values

**increment()** - Method in class org.quartz.utils.counter.sampled.SampledRateCounterImp
throws UnsupportedOperationException

**increment(long)** - Method in class org.quartz.utils.counter.sampled.SampledRateCounterImp
throws UnsupportedOperationException

**incrementRefireCount()** - Method in class org.quartz.impl.JobExecutionContextImp

**init(ServletConfig)** - Method in class org.quartz.ee.servlet.QuartzInitializerServlet

**initDocumentParser()** - Method in class org.quartz.xml.XMLSchedulingDataProcessor
Initializes the XML parser.

**INITIAL_CONTEXT_FACTORY** - Static variable in class org.quartz.jobs.ee.ejb.EJBInvokerJob

**INITIAL_CONTEXT_FACTORY** - Static variable in class org.quartz.jobs.ee.jms.JmsHelper
**initialize(QuartzScheduler)** - Method in class org.quartz.core.JobRunShell

**initialize(Scheduler)** - Method in interface org.quartz.core.JobRunShellFactory

  Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it.

**initialize()** - Method in class org.quartz.core.QuartzScheduler

**initialize(Scheduler)** - Method in class org.quartz.ee.jmx.jboss.JBoss4RMIRemoteMBeanScheduler

  Initialize this remote MBean scheduler, getting the JBoss RMIAdaptor for communication.

**initialize(Scheduler)** - Method in class org.quartz.ee.jta.JTAAnnotationAwareJobRunShellFactory

  Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.

**initialize(Scheduler)** - Method in class org.quartz.ee.jta.JTAJobRunShellFactory

  Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.

**initialize(String, String)** - Method in class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

**initialize(String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

**initialize(ClassLoadHelper, SchedulerSignaler)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreCMT

**initialize(ClassLoadHelper, SchedulerSignaler)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

  Called by the QuartzScheduler before the JobStore is used, in order to give it a chance to initialize.

**initialize(ClassLoadHelper, SchedulerSignaler)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreTX
**initialize(String, String)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**initialize(String, String)** - Method in class org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

**initialize(String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

  InitStrings are of the format:
  settingName=settingValue|otherSettingName=otherSettingValue|...

**initialize(String, String)** - Method in interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate

**initialize()** - Method in class org.quartz.impl.RemoteMBeanScheduler

  Initialize this RemoteMBeanScheduler instance, connecting to the remote MBean server.

**initialize(Scheduler)** - Method in class org.quartz.impl.StdJobRunShellFactory

  Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it.

**initialize()** - Method in class org.quartz.impl.StdSchedulerFactory

  Initialize the SchedulerFactory with the contents of a Properties file and overriding System properties.

**initialize(String)** - Method in class org.quartz.impl.StdSchedulerFactory

  Initialize the SchedulerFactory with the contents of the Properties file with the given name.

**initialize(InputStream)** - Method in class org.quartz.impl.StdSchedulerFactory

  Initialize the SchedulerFactory with the contents of the Properties file opened with the given InputStream.

**initialize(Properties)** - Method in class org.quartz.impl.StdSchedulerFactory

  Initialize the SchedulerFactory with the contents of the given Properties object.

**initialize(String, ConcurrentMap<String, ConcurrentMap<CpuConstraint.Operator, Integer>>>)** - Method in class org.quartz.locality.constraint.evaluator.CpuEvaluator

**initialize(String, ConcurrentMap<String, Long>)** - Method in class org.quartz.locality.constraint.evaluator.MemoryEvaluator
Will start the monitoring, to store available memory on the local node in the shared map.

`initialize(String, ConcurrentMap<String, Set<String>>)` - Method in class org.quartz.locality.constraint.evaluator.OsEvaluator

During its bootstrapping, the JobStore will call this method once on each node, passing a Clustered ConcurrentMap instance and the local instanceId of the Scheduler

`initialize(String, ConcurrentMap<String, V>)` - Method in interface org.quartz.locality.constraint.evaluator.PersistentEvaluator

During its bootstrapping, the JobStore will call this method once on each node, passing a Clustered ConcurrentMap instance and the local instanceId of the Scheduler

`initialize(String, Scheduler)` - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

`initialize(String, Scheduler)` - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

`initialize(String, Scheduler)` - Method in class org.quartz.plugins.management.ShutdownHookPlugin

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

`initialize(String, Scheduler)` - Method in class org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

`initialize()` - Method in class org.quartz.simpl.CascadingClassLoadHelper

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

`initialize()` - Method in class org.quartz.simpl.InitThreadContextClassLoadHelper

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is
the thread that is initializing Quartz.

**initialize()** - Method in class org.quartz.simpl.**LoadingLoaderClassLoadHelper**

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

**initialize(ClassLoadHelper, SchedulerSignaler)** - Method in class org.quartz.simpl.**RAMJobStore**

Called by the QuartzScheduler before the JobStore is used, in order to give the it a chance to initialize.

**initialize()** - Method in class org.quartz.simpl.**SimpleClassLoadHelper**

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

**initialize()** - Method in class org.quartz.simpl.**SimpleThreadPool**

**initialize()** - Method in class org.quartz.simpl.**SimpleTimeBroker**

**initialize()** - Method in class org.quartz.simpl.**ThreadContextClassLoadHelper**

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

**initialize()** - Method in class org.quartz.simpl.**ZeroSizeThreadPool**

**initializeTrigger(MutableTrigger, CompositeData)** - Static method in class org.quartz.core.jmx.**TriggerSupport**

**initializeTrigger(MutableTrigger, Map<String, Object>)** - Static method in class org.quartz.core.jmx.**TriggerSupport**

**InitThreadContextClassLoadHelper** - Class in org.quartz.simpl

A ClassLoadHelper that uses either the context class loader of the thread that initialized Quartz.

**InitThreadContextClassLoadHelper()** - Constructor for class org.quartz.simpl.**InitThreadContextClassLoadHelper**

**inLocale(Locale)** - Method in class org.quartz.**DateBuilder**

Set the Locale for the Date that will be built by this builder (if "null", system default will be used)

**inMonth(int)** - Method in class org.quartz.**DateBuilder**
Set the month (1-12) for the Date that will be built by this builder.

**inMonthOnDay(int, int)** - Method in class org.quartz.**DateBuilder**

**INSERT_BLOB_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCConstants**

**INSERT_CALENDAR** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCConstants**

**INSERT_CRON_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCConstants**

**INSERT_FIRED_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCConstants**

**INSERT_JOB_DETAIL** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCConstants**

**INSERT_LOCK** - Static variable in class org.quartz.impl.jdbcjobstore.**StdRowLockSemaphore**

**INSERT_LOCK** - Static variable in class org.quartz.impl.jdbcjobstore.**UpdateLockRowSemaphore**

**INSERT_ORACLE_CALENDAR** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**

**INSERT_ORACLE_JOB_DETAIL** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**

**INSERT_PAUSED_TRIGGER_GROUP** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCConstants**

**INSERT_SCHEDULER_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCConstants**

**INSERT_SIMPLE_PROPS_TRIGGER** - Static variable in class org.quartz.impl.jdbcjobstore.**SimplePropertiesTriggerPersistenceDelegateSupport**
**INSERT_SIMPLE_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**INSERT_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**insertBlobTrigger(Connection, OperableTrigger)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

  Insert the blob trigger data.

**insertCalendar(Connection, String, Calendar)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

  Insert a new calendar.

**insertCalendar(Connection, String, Calendar)** - Method in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

  Insert a new calendar.

**insertExtendedTriggerProperties(Connection, OperableTrigger, String, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

**insertExtendedTriggerProperties(Connection, OperableTrigger, String, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**insertExtendedTriggerProperties(Connection, OperableTrigger, String, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegateSupport

**insertExtendedTriggerProperties(Connection, OperableTrigger, String, JobDetail)** - Method in interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate

**insertFiredTrigger(Connection, OperableTrigger, String, JobDetail)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Insert a fired trigger.

**insertFiredTrigger**<br>Method in class org.quartz.impl.jbdcjobstore.**StdJDBCDelegate**

Insert a fired trigger.

**insertJobDetail** - Method in interface org.quartz.impl.jbdcjobstore.**DriverDelegate**

Insert the job detail record.

**insertJobDetail** - Method in class org.quartz.impl.jbdcjobstore.oracle.**OracleDelegate**

**insertJobDetail** - Method in class org.quartz.impl.jbdcjobstore.**PointbaseDelegate**

Insert the job detail record.

**insertJobDetail** - Method in class org.quartz.impl.jbdcjobstore.**StdJDBCDelegate**

**insertPausedTriggerGroup** - Method in interface org.quartz.impl.jbdcjobstore.**DriverDelegate**

**insertPausedTriggerGroup** - Method in class org.quartz.impl.jbdcjobstore.**OracleDelegate**

**insertSchedulerState** - Method in interface org.quartz.impl.jbdcjobstore.**DriverDelegate**

Insert a scheduler-instance state record.

**insertSchedulerState** - Method in class org.quartz.impl.jbdcjobstore.**StdJDBCDelegate**

**insertTrigger**<br>Method in interface org.quartz.impl.jbdcjobstore.**DriverDelegate**

Insert the base trigger data.

**insertTrigger**<br>Method in class org.quartz.impl.jbdcjobstore.oracle.**OracleDelegate**

**insertTrigger**<br>Method in class org.quartz.impl.jbdcjobstore.**PointbaseDelegate**

**insertTrigger**<br>Method in class org.quartz.impl.jbdcjobstore.**StdJDBCDelegate**
Insert the base trigger data.

**instanceId** - Variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**instanceId** - Variable in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**instanceName** - Variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**instantiate(QuartzSchedulerResources, QuartzScheduler)** - Method in class org.quartz.impl.StdSchedulerFactory

**interrupt(JobKey)** - Method in class org.quartz.core.QuartzScheduler
    Interrupt all instances of the identified InterruptableJob executing in this Scheduler instance.

**interrupt(JobKey)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**interrupt(JobKey)** - Method in class org.quartz.impl.RemoteMBeanScheduler

**interrupt(JobKey)** - Method in class org.quartz.impl.RemoteScheduler

**interrupt(JobKey)** - Method in class org.quartz.impl.StdScheduler

**interrupt()** - Method in interface org.quartz.InterruptableJob
    Called by the Scheduler when a user interrupts the Job.

**interrupt(JobKey)** - Method in interface org.quartz.Scheduler
    Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which must be an implementor of the InterruptableJob interface.

**InterruptableJob** - Interface in org.quartz
    The interface to be implemented by Jobs that provide a mechanism for having their execution interrupted.

**interruptJob(String, String)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**interruptJob(String, String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**inTimeZone(TimeZone)** - Method in class org.quartz.CronScheduleBuilder
    The TimeZone in which to base the schedule.
**inTimeZone(TimeZone)** - Method in class org.quartz.**DateBuilder**
Set the TimeZone for the Date that will be built by this builder (if "null", system default will be used)

**invalidateHandleCreateException(String, Exception)** - Method in class org.quartz.impl.**RemoteScheduler**

**InvalidConfigurationException** - Exception in org.quartz.impl.jdbcjobstore
Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.

**InvalidConfigurationException(String)** - Constructor for exception org.quartz.impl.jdbcjobstore.**InvalidConfigurationException**

**InvalidConfigurationException()** - Constructor for exception org.quartz.impl.jdbcjobstore.**InvalidConfigurationException**

**invoke(String, Object[], String[])** - Method in class org.quartz.ee.jmx.jboss.**JBoss4RMIRemoteMBeanScheduler**

**invoke(Object, Method, Object[])** - Method in class org.quartz.impl.jdbcjobstore.**AttributeRestoringConnectionInvocationHandler**

**invoke(String, Object[], String[])** - Method in class org.quartz.impl.**RemoteMBeanScheduler**
Invoke the given operation on the remote Scheduler MBean.

**inYear(int)** - Method in class org.quartz.**DateBuilder**
Set the year for the Date that will be built by this builder.

**is(Constraint)** - Method in class org.quartz.locality.**NodeSpecBuilder**
DSL method to add a constraint

**isAcquireTriggersWithinLock()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**
Whether or not the query and update to acquire a Trigger for firing should be performed after obtaining an explicit DB lock (to avoid possible race conditions on the trigger's db row).

**isAlwaysLookup()** - Method in class org.quartz.utils.**JNDIConnectionProvider**

**isAnnotationPresent(Class<?>, Class<? extends Annotation>)** - Static method in class org.quartz.utils.**ClassUtils**

**isBoolean1()** - Method in class
isBoolean2() - Method in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

isCleanShutdown() - Method in class
org.quartz.plugins.management.ShutdownHookPlugin
  Determine whether or not the plug-in is configured to cause a clean
  shutdown of the scheduler.

isClustered() - Method in class org.quartz.core.QuartzScheduler

isClustered() - Method in interface org.quartz.core.RemotableQuartzScheduler

isClustered() - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Get whether this instance is part of a cluster.

isClustered() - Method in class org.quartz.simple.RAMJobStore

isConcurrentExectionDisallowed() - Method in class
org.quartz.impl.JobDetailImpl

isConcurrentExectionDisallowed() - Method in interface org.quartz.JobDetail

isConcurrentExectionDisallowed() - Method in class
org.quartz.locality.DelegatingLocalityJobDetail

isCurrent() - Method in enum org.quartz.locality.constraint.OsConstraint.OS

isDayExcluded(Calendar) - Method in class
org.quartz.impl.calendar.AnnualCalendar
  Return true, if day is defined to be exluded.

isDayExcluded(int) - Method in class
org.quartz.impl.calendar.MonthlyCalendar
  Return true, if day is defined to be excluded.

isDayExcluded(int) - Method in class org.quartz.impl.calendar.WeeklyCalendar
  Return true, if wday (see Calendar.get()) is defined to be exluded.

isDestinationSecure(JobDataMap) - Static method in class
org.quartz.jobs.ee.jms.JmsHelper

isDirty() - Method in class org.quartz.utilsDirtyFlagMap
  Determine whether the Map is flagged dirty.
**isDontSetAutoCommitFalse()** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport

**isDontSetNonManagedTXConnectionAutoCommitFalse()** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreCMT

**isDurable()** - Method in class org.quartz.impl.JobDetailImpl
Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

**isDurable()** - Method in interface org.quartz.JobDetail
Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

**isEmpty()** - Method in class org.quartz.utils.DirtyFlagMap

**isEmtpy()** - Method in class org.quartz.utils.CircularLossyQueue
Returns true if the queue is empty, otherwise false

**isExistingTriggerGroup(Connection, String)** - Method in interface
org.quartz.impl.jdbcjobstore.DriverDelegate

**isExistingTriggerGroup(Connection, String)** - Method in class
org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**isFailOnFileNotFound()** - Method in class
org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin
Whether or not initialization of the plugin should fail (throw an exception) if the file cannot be found.

**isIgnoreDuplicates()** - Method in class
org.quartz.xml.XMLSchedulingDataProcessor
If true (and OverWriteExistingData is false) then any job/triggers encountered in this file that have names that already exist in the scheduler will be ignored, and no error will be produced.

**isInStandbyMode()** - Method in class org.quartz.core.QuartzScheduler
Reports whether the Scheduler is paused.

**isInStandbyMode()** - Method in interface
org.quartz.core.RemotableQuartzScheduler

**isInStandbyMode()** - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**isInStandbyMode()** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**isInStandbyMode()** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**isInStandbyMode()** - Method in interface org.quartz.Scheduler
Reports whether the Scheduler is in stand-by mode.

**isInStandbyMode()** - Method in class org.quartz.SchedulerMetaData
Reports whether the Scheduler is in standby mode.

**isInterruptJobsOnShutdown()** - Method in class org.quartz.core.QuartzSchedulerResources

**isInterruptJobsOnShutdownWithWait()** - Method in class org.quartz.core.QuartzSchedulerResources

**isJobDisallows ConcurrentExecution()** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

**isJobNonConcurrent(Connection, JobKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Check whether or not the given job disallows concurrent execution.

**isJobNonConcurrent(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Check whether or not the given job is stateful.

**isJobRequestsRecovery()** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

**isJobStoreClustered()** - Method in class org.quartz.SchedulerMetaData
Returns whether or not the Scheduler's JobStore is clustered.

**isJobStoreSupportsPersistence()** - Method in class org.quartz.SchedulerMetaData
Returns whether or not the Scheduler's JobStore instance supports persistence.

**isLeapYear(int)** - Method in class org.quartz.CronExpression

**isLockOnInsert()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**isLockOwner(Connection, String)** - Method in class
org.quartz.impl.jdbcjobstore.**DBSemaphore**
   Determine whether the calling thread owns a lock on the identified resource.

**isLockOwner(Connection, String)** - Method in class
org.quartz.impl.jdbcjobstore.**JTANonClusteredSemaphore**
   Determine whether the calling thread owns a lock on the identified resource.

**isLockOwner(Connection, String)** - Method in interface
org.quartz.impl.jdbcjobstore.**Semaphore**
   Determine whether the calling thread owns a lock on the identified resource.

**isLockOwner(Connection, String)** - Method in class
org.quartz.impl.jdbcjobstore.**SimpleSemaphore**
   Determine whether the calling thread owns a lock on the identified resource.

**isMakeThreadsDaemons()** - Method in class
org.quartz.simpl.**SimpleThreadPool**

**isMatch(T)** - Method in class org.quartz.impl.matchers.**AndMatcher**

**isMatch(T)** - Method in class org.quartz.impl.matchers.**EverythingMatcher**

**isMatch(T)** - Method in class org.quartz.impl.matchers.**KeyMatcher**

**isMatch(T)** - Method in class org.quartz.impl.matchers.**NotMatcher**

**isMatch(T)** - Method in class org.quartz.impl.matchers.**OrMatcher**

**isMatch(T)** - Method in interface org.quartz.**Matcher**

**isNodeInGroup(String, String)** - Method in class
org.quartz.locality.constraint.evaluator.**NodeGroupEvaluator**
   Checks whether a node is part of a group

**isOverWriteExistingData()** - Method in class
org.quartz.xml.**XMLSchedulingDataProcessor**
   Whether the existing scheduling data (with same identifiers) will be overwritten.
isPersistJobDataAfterExecution() - Method in class org.quartz.impl.JobDetailImpl

isPersistJobDataAfterExecution() - Method in interface org.quartz.JobDetail

isPersistJobDataAfterExecution() - Method in class org.quartz.locality.DelegatingLocalityJobDetail

isRecovering() - Method in class org.quartz.impl.JobExecutionContextImpl

isRecovering() - Method in interface org.quartz.JobExecutionContext

If the Job is being re-executed because of a 'recovery' situation, this method will return true.

isResetOnSample() - Method in class org.quartz.utils.counter.sampled.SampledCounterConfig

Returns true if counters created from this config will reset on each sample

isRunUpdateCheck() - Method in class org.quartz.core.QuartzSchedulerResources

isSampledStatisticsEnabled() - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

isSampledStatisticsEnabled() - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

isSatisfiedBy(Date) - Method in class org.quartz.CronExpression

Indicates whether the given date satisfies the cron expression.

isScheduleChanged() - Method in class org.quartz.core.QuartzSchedulerThread

isSchedulerRemote() - Method in class org.quartz.SchedulerMetaData

Returns whether the Scheduler is being used remotely (via RMI).

isShutdown() - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

isShutdown() - Method in class org.quartz.core.QuartzScheduler

Reports whether the Scheduler has been shutdown.

isShutdown() - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

isShutdown() - Method in class org.quartz.core.RemotableQuartzScheduler

isShutdown() - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**isShutdown()** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**isShutdown()** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**isShutdown()** - Method in interface org.quartz.Scheduler
Reports whether the Scheduler has been shutdown.

**isShutdown()** - Method in class org.quartz.SchedulerMetaData
Reports whether the Scheduler has been shutdown.

**isShuttingDown()** - Method in class org.quartz.core.QuartzScheduler

**isSignalOnSchedulingChange()** - Method in class org.quartz.core.QuartzScheduler

**isStandbyMode()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**isStandbyMode()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**isStarted()** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**isStarted()** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**isStarted()** - Method in class org.quartz.impl.RemoteMBeanScheduler
Whether the scheduler has been started.

**isStarted()** - Method in class org.quartz.impl.RemoteScheduler
Whether the scheduler has been started.

**isStarted()** - Method in class org.quartz.impl.StdScheduler
Whether the scheduler has been started.

**isStarted()** - Method in interface org.quartz.Scheduler
Whether the scheduler has been started.

**isStarted()** - Method in class org.quartz.SchedulerMetaData
Returns whether the scheduler has been started.

**isThreadsInheritContextClassLoaderOfInitializingThread()** - Method in class org.quartz.simpl.SimpleThreadPool
isThreadsInheritGroupOfInitializingThread() - Method in class org.quartz.simpl.SimpleThreadPool

isThreadsInheritInitializersClassLoadContext() - Method in class org.quartz.core.QuartzSchedulerResources
   Get whether to set the class load context of spawned threads to that of the initializing thread.

isThreadsInheritInitializersClassLoadContext() - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
   Get whether to set the class load context of spawned threads to that of the initializing thread.

isThrowIfPropertyNotFound() - Method in class org.quartz.simpl.PropertySettingJobFactory
   Whether the JobInstantiation should fail and throw an exception if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

isTimeIncluded(long) - Method in interface org.quartz.Calendar
   Determine whether the given time (in milliseconds) is 'included' by the Calendar.

isTimeIncluded(long) - Method in class org.quartz.impl.calendar.AnualCalendar
   Determine whether the given time (in milliseconds) is 'included' by the Calendar.

isTimeIncluded(long) - Method in class org.quartz.impl.calendar.BaseCalendar
   Check if date/time represented by timeStamp is included.

isTimeIncluded(long) - Method in class org.quartz.impl.calendar.CronCalendar
   Determines whether the given time (in milliseconds) is 'included' by the BaseCalendar

isTimeIncluded(long) - Method in class org.quartz.impl.calendar.DailyCalendar
   Determines whether the given time (in milliseconds) is 'included' by the BaseCalendar

isTimeIncluded(long) - Method in class org.quartz.impl.calendar.HolidayCalendar
   Determine whether the given time (in milliseconds) is 'included' by the Calendar.

isTimeIncluded(long) - Method in class org.quartz.impl.calendar.MonthlyCalendar
   Determine whether the given time (in milliseconds) is 'included' by the Calendar.
**isTimeIncluded(long)** - Method in class org.quartz.impl.calendar.**WeeklyCalendar**
Determine whether the given time (in milliseconds) is 'included' by the Calendar.

**isTriggerGroupPaused(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**

**isTriggerGroupPaused(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**

**isTxIsolationLevelReadCommitted()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreCMT**

**isTxIsolationLevelSerializable()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**isValidExpression(String)** - Static method in class org.quartz.**CronExpression**
Indicates whether the specified cron expression can be parsed into a valid cron expression

**isWarnIfPropertyNotFound()** - Method in class org.quartz.simpl.**PropertySettingJobFactory**
Whether a warning should be logged if a key (name) and value (type) found in the JobDataMap does not correspond to a proptery setter on the Job class.
**JBoss4RMIRemoteMBeanScheduler** - Class in `org.quartz.ee.jmx.jboss`  
An implementation of the `Scheduler` interface that remotely proxies all method calls to the equivalent call on a given `QuartzScheduler` instance, via JBoss's JMX RMIAdaptor.

**JBoss4RMIRemoteMBeanScheduler()** - Constructor for class `org.quartz.ee.jmx.jboss.JBoss4RMIRemoteMBeanScheduler`

**jec** - Variable in class `org.quartz.core.JobRunShell`

**JMS_ACK_MODE** - Static variable in class `org.quartz.jobs.ee.jms.JmsHelper`

**JMS_CONNECTION_FACTORY_JNDI** - Static variable in class `org.quartz.jobs.ee.jms.JmsHelper`

**JMS_DESTINATION_JNDI** - Static variable in class `org.quartz.jobs.ee.jms.JmsHelper`

**JMS_MSG_FACTORY_CLASS_NAME** - Static variable in class `org.quartz.jobs.ee.jms.JmsHelper`

**JMS_PASSWORD** - Static variable in class `org.quartz.jobs.ee.jms.JmsHelper`

**JMS_USE_TXN** - Static variable in class `org.quartz.jobs.ee.jms.JmsHelper`

**JMS_USER** - Static variable in class `org.quartz.jobs.ee.jms.JmsHelper`

**JmsHelper** - Class in `org.quartz.jobs.ee.jms`  
Utility class that aids in the processing of JMS based jobs and sending of `javax.jms.Message`

**JmsJobException** - Exception in `org.quartz.jobs.ee.jms`  
The JmsJobException is used to indicate an error during sending of a `javax.jms.Message`.

**JmsJobException()** - Constructor for exception `org.quartz.jobs.ee.jms.JmsJobException`
**JmsJobException(String)** - Constructor for exception
org.quartz.jobs.ee.jms.JmsJobException

**JmsJobException(Throwable)** - Constructor for exception
org.quartz.jobs.ee.jms.JmsJobException

**JmsJobException(String, Throwable)** - Constructor for exception
org.quartz.jobs.ee.jms.JmsJobException

**JmsMessageFactory** - Interface in org.quartz.jobs.ee.jms
The JmsMessageFactory interface allows for the creation of javax.jms.Message.

**JMXInvokerJob** - Class in org.quartz.jobs.ee.jmx
Generic JMX invoker Job.

**JMXInvokerJob()** - Constructor for class
org.quartz.jobs.ee.jmx.JMXInvokerJob

**JNDIConnectionProvider** - Class in org.quartz.utils
A ConnectionProvider that provides connections from a DataSource that is managed by an application server, and made available via JNDI.

**JNDIConnectionProvider(String, boolean)** - Constructor for class
org.quartz.utils.JNDIConnectionProvider
Constructor

**JNDIConnectionProvider(String, Properties, boolean)** - Constructor for class
org.quartz.utils.JNDIConnectionProvider
Constructor

**Job** - Interface in org.quartz
The interface to be implemented by classes which represent a ‘job’ to be performed.

**JOB_ADDED** - Static variable in interface
org.quartz.core.jmx.QuartzSchedulerMBean

**JOB_DELETED** - Static variable in interface
org.quartz.core.jmx.QuartzSchedulerMBean

**JOB_EXECUTION_VETOED** - Static variable in interface
org.quartz.core.jmx.QuartzSchedulerMBean

**JOB_SCHEDULED** - Static variable in interface
**jobAdded(JobDetail)** - Method in class
org.quartz.core.**QuartzSchedulerMBeanImpl**

**jobAdded(JobDetail)** - Method in class org.quartz.core.**SampledStatisticsImpl**

**jobAdded(JobDetail)** - Method in class org.quartz.listeners.**BroadcastSchedulerListener**

**jobAdded(JobDetail)** - Method in class org.quartz.listeners.**SchedulerListenerSupport**

**jobAdded(JobDetail)** - Method in interface org.quartz.**SchedulerListener**

  Called by the Scheduler when a JobDetail has been added.

**JobBuilder** - Class in org.quartz

  JobBuilder is used to instantiate JobDetails.

**JobChainingJobListener** - Class in org.quartz.listeners

  Keeps a collection of mappings of which Job to trigger after the completion of a given job.

**JobChainingJobListener(String)** - Constructor for class org.quartz.listeners.**JobChainingJobListener**

  Construct an instance with the given name.

**JobDataMap** - Class in org.quartz

  Holds state information for Job instances.

**JobDataMap()** - Constructor for class org.quartz.**JobDataMap**

  Create an empty JobDataMap.

**JobDataMap(Map)** - Constructor for class org.quartz.**JobDataMap**

  Create a JobDataMap with the given data.

**JobDataMapSupport** - Class in org.quartz.core.jmx
**JobDataMapSupport()** - Constructor for class `org.quartz.core.jmx.JobDataMapSupport`

**jobDeleted(JobKey)** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**jobDeleted(String, String)** - Method in class `org.quartz.core.SampledStatisticsImpl`

**jobDeleted(JobKey)** - Method in class `org.quartz.listeners.BroadcastSchedulerListener`

**jobDeleted(JobKey)** - Method in class `org.quartz.listeners.SchedulerListenerSupport`

**jobDeleted(JobKey)** - Method in interface `org.quartz.SchedulerListener`

Called by the Scheduler when a JobDetail has been deleted.

**JobDetail** - Interface in `org.quartz`
Conveys the detail properties of a given Job instance.

**JobDetailImpl** - Class in `org.quartz.impl`
Conveys the detail properties of a given Job instance.

**JobDetailImpl()** - Constructor for class `org.quartz.impl.JobDetailImpl`
Create a JobDetail with no specified name or group, and the default settings of all the other properties.

**JobDetailImpl(String, Class<? extends Job>)** - Constructor for class `org.quartz.impl.JobDetailImpl`

Deprecated. use `JobBuilder`

**JobDetailImpl(String, String, Class<? extends Job>)** - Constructor for class `org.quartz.impl.JobDetailImpl`

Deprecated. use `JobBuilder`

**JobDetailImpl(String, String, Class<? extends Job>, boolean, boolean)** - Constructor for class `org.quartz.impl.JobDetailImpl`

Deprecated. use `JobBuilder`

**JobDetailSupport** - Class in `org.quartz.core.jmx`

**JobDetailSupport()** - Constructor for class `org.quartz.core.jmx.JobDetailSupport`
**JobExecutionContext** - Interface in org.quartz

A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.

**JobExecutionContextImpl** - Class in org.quartz.impl

**JobExecutionContextImpl**(Scheduler, TriggerFiredBundle, Job) - Constructor for class org.quartz.impl.JobExecutionContextImpl

Create a JobExecutionContext with the given context data.

**JobExecutionContextSupport** - Class in org.quartz.core.jmx

**JobExecutionContextSupport()** - Constructor for class org.quartz.core.jmx.JobExecutionContextSupport

**JobExecutionContext** - Exception in org.quartz

An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.

**JobExecutionException()** - Constructor for exception org.quartz.JobExecutionException

Create a JobExecutionException, with the 're-fire immediately' flag set to false.

**JobExecutionException(Throwable)** - Constructor for exception org.quartz.JobExecutionException

Create a JobExecutionException, with the given cause.

**JobExecutionException(String)** - Constructor for exception org.quartz.JobExecutionException

Create a JobExecutionException, with the given message.

**JobExecutionException(boolean)** - Constructor for exception org.quartz.JobExecutionException

Create a JobExecutionException with the 're-fire immediately' flag set to the given value.

**JobExecutionException(Throwable, boolean)** - Constructor for exception org.quartz.JobExecutionException

Create a JobExecutionException with the given underlying exception, and the 're-fire immediately' flag set to the given value.

**JobExecutionException(String, Throwable)** - Constructor for exception org.quartz.JobExecutionException
Create a JobExecutionException with the given message, and underlying exception.

**JobExecutionException(String, Throwable, boolean)** - Constructor for exception org.quartz.JobExecutionException

Create a JobExecutionException with the given message, and underlying exception, and the 're-fire immediately' flag set to the given value.

**JobExecutionException(String, boolean)** - Constructor for exception org.quartz.JobExecutionException

Create a JobExecutionException with the given message and the 're-fire immediately' flag set to the given value.

**jobExecutionVetoed(JobExecutionContext)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

**jobExecutionVetoed(JobExecutionContext)** - Method in class org.quartz.core.SampledStatisticsImpl

**jobExecutionVetoed(JobExecutionContext)** - Method in class org.quartz.core.JobListener

**jobExecutionVetoed(JobExecutionContext)** - Method in class org.quartz.core.BroadcastJobListener

**jobExecutionVetoed(JobExecutionContext)** - Method in class org.quartz.core.JobListenerSupport

**jobExecutionVetoed(JobExecutionContext)** - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin

**jobExists(Connection, JobKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Check whether or not the given job exists.

**jobExists(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Check existence of a given job.

**jobExists(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Check whether or not the given job exists.
**jobGroupsToDelete** - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

**jobGroupsToNeverDelete** - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

**JobKey** - Class in org.quartz

  Uniquely identifies a JobDetail.

**JobKey(String)** - Constructor for class org.quartz.JobKey

**JobKey(String, String)** - Constructor for class org.quartz.JobKey

**jobKey(String)** - Static method in class org.quartz.JobKey

**jobKey(String, String)** - Static method in class org.quartz.JobKey

**JobListener** - Interface in org.quartz

  The interface to be implemented by classes that want to be informed when a JobDetail executes.

**JobListenerSupport** - Class in org.quartz.listeners

  A helpful abstract base class for implementors of JobListener.

**JobListenerSupport()** - Constructor for class org.quartz.listeners.JobListenerSupport

**jobPaused(JobKey)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**jobPaused(JobKey)** - Method in class org.quartz.listeners.BroadcastSchedulerListener

**jobPaused(JobKey)** - Method in class org.quartz.listeners.SchedulerListenerSupport

**jobPaused(JobKey)** - Method in interface org.quartz.SchedulerListener

  Called by the Scheduler when a JobDetail has been paused.

**JobPersistenceException** - Exception in org.quartz

  An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.

**JobPersistenceException(String)** - Constructor for exception
org.quartz.\textit{JobPersistenceException}  
Create a JobPersistenceException with the given message.  
\textit{JobPersistenceException(String, Throwable)} - Constructor for exception  
org.quartz.\textit{JobPersistenceException}  
Create a JobPersistenceException with the given message and cause.  
\textit{jobResumed(JobKey)} - Method in class  
org.quartz.core.\textit{QuartzSchedulerMBeanImpl}  
\textit{jobResumed(JobKey)} - Method in class  
org.quartz.listeners.\textit{BroadcastSchedulerListener}  
\textit{jobResumed(JobKey)} - Method in class  
org.quartz.listeners.\textit{SchedulerListenerSupport}  
\textit{jobResumed(JobKey)} - Method in interface org.quartz.\textit{SchedulerListener}  
Called by the Scheduler when a JobDetail has been un-paused.  
\textit{JobRunShell} - Class in org.quartz.core  
JobRunShell instances are responsible for providing the 'safe' environment for Job s to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the \textit{Trigger} with the Job's completion code, etc.  
\textit{JobRunShell(Scheduler, TriggerFiredBundle)} - Constructor for class  
org.quartz.core.\textit{JobRunShell}  
Create a JobRunShell instance with the given settings.  
\textit{JobRunShellFactory} - Interface in org.quartz.core  
Responsible for creating the instances of JobRunShell to be used within the \textit{QuartzScheduler} instance.  
\textit{JOBS_PAUSED} - Static variable in interface  
org.quartz.core.jmx.\textit{QuartzSchedulerMBean}  
\textit{JOBS_RESUMED} - Static variable in interface  
org.quartz.core.jmx.\textit{QuartzSchedulerMBean}  
\textit{jobsByGroup} - Variable in class org.quartz.simpl.\textit{RAMJobStore}  
\textit{jobsByKey} - Variable in class org.quartz.simpl.\textit{RAMJobStore}  
\textit{jobScheduled(Trigger)} - Method in class  
org.quartz.core.\textit{QuartzSchedulerMBeanImpl}
**jobScheduled(Trigger)** - Method in class `org.quartz.core.SampledStatisticsImpl`  
Called by the Scheduler when a JobDetail is scheduled.

**jobScheduled(Trigger)** - Method in class `org.quartz.listeners.BroadcastSchedulerListener`  

**jobScheduled(Trigger)** - Method in class `org.quartz.listeners.SchedulerListenerSupport`  

**jobsPaused(String)** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`  
Called by the Scheduler when a group of JobDetails has been paused.

**jobsPaused(String)** - Method in class `org.quartz.listeners.BroadcastSchedulerListener`  

**jobsPaused(String)** - Method in class `org.quartz.listeners.SchedulerListenerSupport`  

**jobsPaused(String)** - Method in class `org.quartz.listeners.BroadcastSchedulerListener`  

**jobsResumed(String)** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`  
Called by the Scheduler when a group of JobDetails has been un-paused.

**jobsResumed(String)** - Method in class `org.quartz.listeners.BroadcastSchedulerListener`  

**jobsResumed(String)** - Method in class `org.quartz.listeners.SchedulerListenerSupport`  

**jobsResumed(String)** - Method in class `org.quartz.listeners.BroadcastSchedulerListener`  

**jobsToDelete** - Variable in class `org.quartz.xml.XMLSchedulingDataProcessor`  

**JobStoreCMT** - Class in `org.quartz.impl.jdbcjobstore`  
JobStoreCMT is meant to be used in an application-server environment that provides container-managed-transactions.

**JobStoreCMT()** - Constructor for class
**JobStoreSupport** - Class in `org.quartz.impl.jdbcjobstore`
Contains base functionality for JDBC-based JobStore implementations.

**JobStoreSupport()** - Constructor for class
`org.quartz.impl.jdbcjobstore.JobStoreSupport`

**JobStoreSupport.RecoverMisfiredJobsResult** - Class in `org.quartz.impl.jdbcjobstore`
Helper class for returning the composite result of trying to recover misfired jobs.

**JobStoreSupport.RecoverMisfiredJobsResult(boolean, int, long)** - Constructor for class

**JobStoreSupport.TransactionCallback** - Interface in `org.quartz.impl.jdbcjobstore`
Implement this interface to provide the code to execute within the a transaction template.

**JobStoreSupport.VoidTransactionCallback** - Interface in `org.quartz.impl.jdbcjobstore`
Implement this interface to provide the code to execute within the a transaction template that has no return value.

**jobStoreSupportsPersistence()** - Method in class `org.quartz.SchedulerMetaData`
Deprecated.

**JobStoreTX** - Class in `org.quartz.impl.jdbcjobstore`
JobStoreTX is meant to be used in a standalone environment.

**JobStoreTX()** - Constructor for class `org.quartz.impl.jdbcjobstore.JobStoreTX`

**jobToBeExecuted(JobExecutionContext)** - Method in class `org.quartz.core.QuartzSchedulerMBEanImpl`

**jobToBeExecuted(JobExecutionContext)** - Method in class `org.quartz.core.SampledStatisticsImpl`

**jobToBeExecuted(JobExecutionContext)** - Method in interface `org.quartz.JobListener`
Called by the Scheduler when a JobDetail is about to be executed (an
associated Trigger has occurred).

**jobToBeExecuted(JobExecutionContext)** - Method in class org.quartz.listeners.BroadcastJobListener

**jobToBeExecuted(JobExecutionContext)** - Method in class org.quartz.listeners.JobListenerSupport

**jobToBeExecuted(JobExecutionContext)** - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin

**jobUnscheduled(TriggerKey)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**jobUnscheduled(TriggerKey)** - Method in class org.quartz.listeners.BroadcastSchedulerListener

**jobUnscheduled(TriggerKey)** - Method in class org.quartz.listeners.SchedulerListenerSupport

**jobUnscheduled(TriggerKey)** - Method in interface org.quartz.SchedulerListener

  Called by the Scheduler when a JobDetail is unscheduled.

**jobWasExecuted(JobExecutionContext, JobExecutionException)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**jobWasExecuted(JobExecutionContext, JobExecutionException)** - Method in class org.quartz.core.SampledStatisticsImpl

**jobWasExecuted(JobExecutionContext, JobExecutionException)** - Method in interface org.quartz.JobListener

  Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

**jobWasExecuted(JobExecutionContext, JobExecutionException)** - Method in class org.quartz.listeners.BroadcastJobListener

**jobWasExecuted(JobExecutionContext, JobExecutionException)** - Method in class org.quartz.listeners.JobChainingJobListener

**jobWasExecuted(JobExecutionContext, JobExecutionException)** - Method
in class org.quartz.listeners.JobListenerSupport

**jobWasExecuted(JobExecutionContext, JobExecutionException)** - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin

**JTAAnnotationAwareJobRunShellFactory** - Class in org.quartz.ee.jta

- Responsible for creating the instances of a JobRunShell to be used within the QuartzScheduler instance.

**JTAAnnotationAwareJobRunShellFactory()** - Constructor for class org.quartz.ee.jta.JTAAnnotationAwareJobRunShellFactory

**JTAJobRunShell** - Class in org.quartz.ee.jta

- An extension of JobRunShell that begins an XA transaction before executing the Job, and commits (or rolls-back) the transaction after execution completes.

**JTAJobRunShell(Scheduler, TriggerFiredBundle)** - Constructor for class org.quartz.ee.jta.JTAJobRunShell

- Create a JTAJobRunShell instance with the given settings.

**JTAJobRunShellFactory** - Class in org.quartz.ee.jta

- Responsible for creating the instances of JTAJobRunShell11 to be used within the QuartzScheduler instance.

**JTAJobRunShellFactory()** - Constructor for class org.quartz.ee.jta.JTAJobRunShellFactory

**JTANonClusteredSemaphore** - Class in org.quartz.impl.jdbcjobstore

- Provides in memory thread/resource locking that is JTA Transaction aware.

**JTANonClusteredSemaphore()** - Constructor for class org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore
K

**Key**<T> - Class in [org.quartz.utils](http://example.com)
Object representing a job or trigger key.

**Key(String, String)** - Constructor for class org.quartz.utils.Key
Construct a new key with the given name and group.

**keyEquals(U)** - Static method in class org.quartz.impl.matchers.KeyMatcher
Create a KeyMatcher that matches Keys that equal the given key.

**KeyMatcher**<T extends Key> - Class in [org.quartz.impl.matchers](http://example.com)
Matches on the complete key being equal (both name and group).

**KeyMatcher(T)** - Constructor for class org.quartz.impl.matchers.KeyMatcher

**keySet()** - Method in class org.quartz.utils.DirtyFlagMap
lastCheckin - Variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

lastdayOffset - Variable in class org.quartz.CronExpression

lastdayOfMonth - Variable in class org.quartz.CronExpression

lastdayOfWeek - Variable in class org.quartz.CronExpression

leftOperand - Variable in class org.quartz.impl.matchers.AndMatcher

leftOperand - Variable in class org.quartz.impl.matchers.OrMatcher

LINUX - Static variable in class org.quartz.locality.constraint.OsConstraint

ListenerManager - Interface in org.quartz

Client programs may be interested in the 'listener' interfaces that are available from Quartz.

ListenerManagerImpl - Class in org.quartz.core

ListenerManagerImpl() - Constructor for class org.quartz.core.ListenerManagerImpl

loadAtMost(double) - Static method in class org.quartz.locality.constraint.CpuConstraint

Creates a constraint that requires the node to have at most a certain load

loadClass(String) - Method in class org.quartz.simpl.CascadingClassLoadHelper

Return the class with the given name.

loadClass(String) - Method in class org.quartz.simpl.InitThreadContextClassLoadHelper

Return the class with the given name.

loadClass(String) - Method in class org.quartz.simpl>LoadingLoaderClassLoadHelper

Return the class with the given name.

loadClass(String) - Method in class org.quartz.simpl.SimpleClassLoadHelper
Return the class with the given name.

**loadClass(String)** - Method in class org.quartz.simpl.ThreadContextClassLoadHelper

Return the class with the given name.

**loadedJobs** - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

**loadedTriggers** - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

**loadExtendedTriggerProperties(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

**loadExtendedTriggerProperties(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**loadExtendedTriggerProperties(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

**loadExtendedTriggerProperties(Connection, TriggerKey)** - Method in interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate

**LoadingLoaderClassLoadHelper** - Class in org.quartz.simpl.LoadingLoaderClassLoadHelper

A ClassLoadHelper that uses either the loader of it's own class (this.getClass().getClassLoader().loadClass(. ..

**LoadingLoaderClassLoadHelper()** - Constructor for class org.quartz.simpl.LoadingLoaderClassLoadHelper

**LocalityAware** - Interface in org.quartz.locality

LocalityAware types can provide node specifications to potentially dispatch the instance to a particular node in the cluster.

**LocalityException** - Exception in org.quartz.locality

Exception thrown when some Constraint cannot be handled

**LocalityException(String, Constraint)** - Constructor for exception org.quartz.locality.LocalityException

**LocalityJobBuilder** - Class in org.quartz.locality

Builder for LocalityAware JobDetail instances.

**LocalityJobBuilder(JobBuilder)** - Constructor for class org.quartz.locality.LocalityJobBuilder
**LocalityJobDetail** - Interface in org.quartz.locality
A specialized JobDetail that contains Quartz Where information

**LocalityTrigger** - Interface in org.quartz.locality
A specialized Trigger that contains Quartz Where information

**LocalityTriggerBuilder** - Class in org.quartz.locality
Builder for LocalityAware Triggers, which decorate the Trigger and adds the Locality feature to it, by adding a reference to a NodeSpec.

**LocalityTriggerBuilder(TriggerBuilder)** - Constructor for class org.quartz.locality.LocalityTriggerBuilder

**LocalityTriggerBuilder(Trigger)** - Constructor for class org.quartz.locality.LocalityTriggerBuilder

**localJob(JobDetail)** - Static method in class org.quartz.locality.LocalityJobBuilder
Factory method to create builder to a wrapping LocalityAware JobDetail

**localJob(JobBuilder)** - Static method in class org.quartz.locality.LocalityJobBuilder
Factory method to create builder to a wrapping LocalityAware JobDetailBuilder

**localTrigger(TriggerBuilder)** - Static method in class org.quartz.locality.LocalityTriggerBuilder
Creates a delegating LocalityTriggerBuilder based on the TriggerBuilder

**localTrigger(Trigger)** - Static method in class org.quartz.locality.LocalityTriggerBuilder
Creates a delegating LocalityTriggerBuilder based on the Trigger

**lock** - Variable in class org.quartz.simpl.RAMJobStore

**LOCK_CALENDAR_ACCESS** - Static variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**LOCK_JOB_ACCESS** - Static variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**LOCK_MISFIRE_ACCESS** - Static variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**LOCK_STATE_ACCESS** - Static variable in class
org.quartz.impl.jdbcjobstore.JobStoreSupport

**LOCK_TRIGGER_ACCESS** - Static variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**LockException** - Exception in org.quartz.impl.jdbcjobstore
Exception class for when there is a failure obtaining or releasing a resource lock.

**LockException(String)** - Constructor for exception
org.quartz.impl.jdbcjobstore.LockException

**LockException(String, Throwable)** - Constructor for exception
org.quartz.impl.jdbcjobstore.LockException

**logger** - Variable in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**LoggingJobHistoryPlugin** - Class in org.quartz.plugins.history
Logs a history of all job executions (and execution vetos) via the Jakarta Commons-Logging framework.

**LoggingJobHistoryPlugin()** - Constructor for class
org.quartz.plugins.history.LoggingJobHistoryPlugin

**LoggingTriggerHistoryPlugin** - Class in org.quartz.plugins.history
Logs a history of all trigger firings via the Jakarta Commons-Logging framework.

**LoggingTriggerHistoryPlugin()** - Constructor for class
org.quartz.plugins.history.LoggingTriggerHistoryPlugin

**logWarnIfNonZero(int, String)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport

**lookup(String)** - Method in class org.quartz.impl.SchedulerRepository

**lookupAll()** - Method in class org.quartz.impl.SchedulerRepository

**lookupUserTransaction()** - Static method in class
org.quartz.ee.jta.UserTransactionHelper
Create/Lookup a UserTransaction in the InitialContext via the name set in setUserTxLocation().
**main(String[])** - Static method in class org.quartz.helpers.**VersionPrinter**

**main(String[])** - Static method in class org.quartz.impl.**QuartzServer**

**main(String[])** - Static method in class org.quartz.utils.**UpdateChecker**

**makeAvailable(SimpleThreadPool.WorkerThread)** - Method in class org.quartz.simpl.**SimpleThreadPool**

**Matcher<T> extends Key** - Interface in org.quartz

Matcher can be used in various **Scheduler** API methods to select the entities that should be operated upon.

**matches(Integer)** - Method in class org.quartz.locality.constraint.**CpuConstraint**

**matches(Integer, Integer)** - Method in enum org.quartz.locality.constraint.**CpuConstraint.Operator**

**matches(CpuConstraint)** - Method in class org.quartz.locality.constraint.evaluator.**CpuEvaluator**

**matches(T)** - Method in interface org.quartz.locality.constraint.evaluator.**Evaluator**

Evaluates the constraint against local node.

**matches(MemoryConstraint)** - Method in class org.quartz.locality.constraint.evaluator.**MemoryEvaluator**

Evaluates the constraint against local node.

**matches(MemoryConstraint, String)** - Method in class org.quartz.locality.constraint.evaluator.**MemoryEvaluator**

Verifies whether the constraint passed is validating on the node

**matches(NodeGroupConstraint)** - Method in class org.quartz.locality.constraint.evaluator.**NodeGroupEvaluator**

Evaluates the constraint against local node.

**matches(OsConstraint)** - Method in class org.quartz.locality.constraint.evaluator.**OsEvaluator**

Evaluates the constraint against local node.
**matches(Long)** - Method in class org.quartz.locality.constraint.**MemoryConstraint**

**matches(String)** - Method in class org.quartz.locality.constraint.**NodeGroupConstraint**

**matches()** - Method in class org.quartz.locality.constraint.**OsConstraint**
    Evaluate against current node

**matches(OsConstraint.OS)** - Method in class org.quartz.locality.constraint.**OsConstraint**

**matches(EhcachaeConstraint)** - Method in class org.terracotta.modules.ehcache.store.**EhcachaeEvaluator**
    Evaluates the constraint against local node.

**MAX_YEAR** - Static variable in class org.quartz.**CronExpression**

**maxToRecoverAtATime** - Variable in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**maybeThrowValidationException()** - Method in class org.quartz.xml.**XMLSchedulingDataProcessor**
    Throws a ValidationException if the number of validationExceptions detected is greater than zero.

**mayFireAgain()** - Method in class org.quartz.impl.triggers.**AbstractTrigger**
    Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

**mayFireAgain()** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**
    Determines whether or not the DateIntervalTrigger will occur again.

**mayFireAgain()** - Method in class org.quartz.impl.triggers.**CronTriggerImpl**
    Determines whether or not the CronTrigger will occur again.

**mayFireAgain()** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**
    Determines whether or not the SimpleTrigger will occur again.

**mayFireAgain()** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**
    Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

**mayFireAgain()** - Method in interface org.quartz.**Trigger**
    Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

**MemoryConstraint** - Class in org.quartz.locality.constraint
    Constraint on memory characteristics of the node to execute the **Job** on
**MemoryConstraint(MemoryConstraint.Operator, long)** - Constructor for class org.quartz.locality.constraint.MemoryConstraint
MemoryConstraint constructor

**MemoryConstraint.Operator** - Enum in org.quartz.locality.constraint

**MemoryConstraint.Unit** - Enum in org.quartz.locality.constraint
Memory units

**MemoryEvaluator** - Class in org.quartz.locality.constraint.evaluator
Evaluates memory constraints

**MemoryEvaluator()** - Constructor for class org.quartz.locality.constraint.evaluator.MemoryEvaluator

**MILLISECONDS_IN_DAY** - Static variable in class org.quartz.DateBuilder

**MILLISECONDS_IN_HOUR** - Static variable in class org.quartz.DateBuilder

**MILLISECONDS_IN_MINUTE** - Static variable in class org.quartz.DateBuilder

**MINIMUM_UPDATE_AGE** - Static variable in class org.quartz.jobs.DirectoryScanJob
JobDataMap key with which to specify a long value that represents the minimum number of milliseconds that must have past since the file's last modified time in order to consider the file new/altered.

**MINIMUM_UPDATE_AGE** - Static variable in class org.quartz.jobs.FileScanJob
JobDataMap key with which to specify a long value that represents the minimum number of milliseconds that must have past since the file's last modified time in order to consider the file new/altered.

**MINUTE** - Static variable in class org.quartz.CronExpression

**minutes** - Variable in class org.quartz.CronExpression

**MISFIRE_INSTRUCTION_DO NOTHING** - Static variable in interface org.quartz.CalendarIntervalTrigger
Instructs the Scheduler that upon a mis-fire situation, the CalendarIntervalTrigger wants to have it's next-fire-time updated to the next time in the schedule after the current time (taking into account any associated calendar, but it does not want to be fired now.
**MISFIRE_INSTRUCTION_DO NOTHING** - Static variable in interface `org.quartz.CronTrigger`  
Instructs the Scheduler that upon a mis-fire situation, the CronTrigger wants to have it's next-fire-time updated to the next time in the schedule after the current time (taking into account any associated Calendar, but it does not want to be fired now.

**MISFIRE_INSTRUCTION_FIRE_NOW** - Static variable in interface `org.quartz.SimpleTrigger`  
Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be fired now by Scheduler.

**MISFIRE_INSTRUCTION_FIRE_ONCE_NOW** - Static variable in interface `org.quartz.CalendarIntervalTrigger`  
Instructs the Scheduler that upon a mis-fire situation, the CalendarIntervalTrigger wants to be fired now by Scheduler.

**MISFIRE_INSTRUCTION_FIRE_ONCE_NOW** - Static variable in interface `org.quartz.CronTrigger`  
Instructs the Scheduler that upon a mis-fire situation, the CronTrigger wants to be fired now by Scheduler.

**MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY** - Static variable in interface `org.quartz.Trigger`  
Instructs the Scheduler that the Trigger will never be evaluated for a misfire situation, and that the scheduler will simply try to fire it as soon as it can, and then update the Trigger as if it had fired at the proper time.

**MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT** - Static variable in interface `org.quartz.SimpleTrigger`  
Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to the next scheduled time after 'now' - taking into account any associated Calendar, and with the repeat count left unchanged.

**MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT** - Static variable in interface `org.quartz.SimpleTrigger`  
Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to the next scheduled time after 'now' - taking into account any associated Calendar, and with the repeat count set to what it would be, if it had not missed any firings.

**MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT** - Static variable in interface `org.quartz.SimpleTrigger`  
Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to 'now' (even if the associated Calendar excludes 'now') with the repeat count left as-is.
**MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING** - Static variable in interface org.quartz.SimpleTrigger
Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to 'now' (even if the associated Calendar excludes 'now') with the repeat count set to what it would be, if it had not missed any firings.

**MISFIRE_INSTRUCTION_SMART_POLICY** - Static variable in interface org.quartz.Trigger
Instructs the Scheduler that upon a mis-fire situation, the updateAfterMisfire() method will be called on the Trigger to determine the mis-fire instruction, which logic will be trigger-implementation-dependent.

**misfireThreshold** - Variable in class org.quartz.simpl.RAMJobStore

**modifiedByCalendar(String)** - Method in class org.quartz.TriggerBuilder
Set the name of the Calendar that should be applied to this Trigger's schedule.

**MONDAY** - Static variable in class org.quartz.DateBuilder

**monitor()** - Method in class org.quartz.locality.constraint.evaluator.CpuEvaluator

**monitor()** - Method in class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask

**monitor()** - Method in class org.quartz.locality.constraint.evaluator.MemoryEvaluator
Performs the monitoring of the free memory on the local node

**MONTH** - Static variable in interface org.quartz.Calendar

**MONTH** - Static variable in class org.quartz.CronExpression

**MonthlyCalendar** - Class in org.quartz.impl.calendar
This implementation of the Calendar excludes a set of days of the month.

**MonthlyCalendar()** - Constructor for class org.quartz.impl.calendar.MonthlyCalendar

**MonthlyCalendar(Calendar)** - Constructor for class org.quartz.impl.calendar.MonthlyCalendar
**MonthlyCalendar(TimeZone)** - Constructor for class org.quartz.impl.calendar.MonthlyCalendar

**MonthlyCalendar(Calendar, TimeZone)** - Constructor for class org.quartz.impl.calendar.MonthlyCalendar

**monthlyOnDayAndHourAndMinute(int, int, int)** - Static method in class org.quartz.CronScheduleBuilder

Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire one per month on the given day of month at the given time (hour and minute).

**monthMap** - Static variable in class org.quartz.CronExpression

**months** - Variable in class org.quartz.CronExpression

**MSSQLDelegate** - Class in org.quartz.impl.jdbcjobstore

This is a driver delegate for the MSSQL JDBC driver.

**MSSQLDelegate(Logger, String, String, String, ClassLoadHelper)** - Constructor for class org.quartz.impl.jdbcjobstore.MSSQLDelegate

Create new MSSQLDelegate instance.

**MSSQLDelegate(Logger, String, String, String, String, ClassLoadHelper, Boolean)** - Constructor for class org.quartz.impl.jdbcjobstore.MSSQLDelegate

**mxBeanAvgLoadMonitoring** - Static variable in class org.quartz.locality.constraint.evaluator.CpuEvaluator
nameContains(String) - Static method in class org.quartz.impl.matchers.NameMatcher
  Create a NameMatcher that matches names containing the given string.
nameEndsWith(String) - Static method in class org.quartz.impl.matchers.NameMatcher
  Create a NameMatcher that matches names ending with the given string.
nameEquals(String) - Static method in class org.quartz.impl.matchers.NameMatcher
  Create a NameMatcher that matches names equaling the given string.
NameMatcher<T> extends Key - Class in org.quartz.impl.matchers
  Matches on name (ignores group) property of Keys.
NameMatcher(String, StringMatcher.StringOperatorName) - Constructor for class org.quartz.impl.matchers.NameMatcher

nameStartsWith(String) - Static method in class org.quartz.impl.matchers.NameMatcher
  Create a NameMatcher that matches names starting with the given string.
NativeJob - Class in org.quartz.jobs
  Built in job for executing native executables in a separate process.
NativeJob() - Constructor for class org.quartz.jobs.NativeJob

nearestWeekday - Variable in class org.quartz.CronExpression

newDate() - Static method in class org.quartz.DateBuilder
  Create a DateBuilder, with initial settings for the current date and time in the system default timezone.
newDateInLocale(Locale) - Static method in class org.quartz.DateBuilder
  Create a DateBuilder, with initial settings for the current date and time in the given locale.
newDateInTimeZone(TimeZone) - Static method in class org.quartz.DateBuilder
  Create a DateBuilder, with initial settings for the current date and time in the given timezone.
newDateInTimeZoneAndLocale(TimeZone, Locale) - Static method in class org.quartz.DateBuilder
Create a DateBuilder, with initial settings for the current date and time in the given timezone and locale.

**newJob()** - Static method in class org.quartz.JobBuilder
Create a JobBuilder with which to define a JobDetail.

**newJob(Class<? extends Job>)** - Static method in class org.quartz.JobBuilder
Create a JobBuilder with which to define a JobDetail, and set the class name of the Job to be executed.

**newJob(TriggerFiredBundle, Scheduler)** - Method in class org.quartz.simpl.PropertySettingJobFactory

**newJob(TriggerFiredBundle, Scheduler)** - Method in class org.quartz.simpl.SimpleJobFactory

**newJobDataMap(TabularData)** - Static method in class org.quartz.core.jmx.JobDataMapSupport

**newJobDataMap(Map<String, Object>)** - Static method in class org.quartz.core.jmx.JobDataMapSupport

**newJobDetail(CompositeData)** - Static method in class org.quartz.core.jmx.JobDetailSupport

**newJobDetail(Map<String, Object>)** - Static method in class org.quartz.core.jmx.JobDetailSupport

**newTrigger(CompositeData)** - Static method in class org.quartz.core.jmx.CronTriggerSupport

**newTrigger(Map<String, Object>)** - Static method in class org.quartz.core.jmx.CronTriggerSupport

**newTrigger(CompositeData)** - Static method in class org.quartz.core.jmx.SimpleTriggerSupport

**newTrigger(Map<String, Object>)** - Static method in class org.quartz.core.jmx.SimpleTriggerSupport

**newTrigger(CompositeData)** - Static method in class org.quartz.core.jmx.TriggerSupport
newTrigger(Map<String, Object>) - Static method in class org.quartz.core.jmx.TriggerSupport

newTrigger() - Static method in class org.quartz.TriggerBuilder
Create a new TriggerBuilder with which to define a specification for a Trigger.

nextGivenMinuteDate(Date, int) - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even multiple of the given minute.

nextGivenSecondDate(Date, int) - Static method in class org.quartz.DateBuilder
Returns a date that is rounded to the next even multiple of the given minute.

NO_OP - Static variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport.RecoverMisfiredJobsResult

NO_SPEC - Static variable in class org.quartz.CronExpression

NO_SPEC_INT - Static variable in class org.quartz.CronExpression
	node() - Static method in class org.quartz.locality.NodeSpecBuilder
Factory method

NodeGroupConstraint - Class in org.quartz.locality.constraint
Constraint to have a Job execute on a specific node group.

NodeGroupConstraint(String) - Constructor for class org.quartz.locality.constraint.NodeGroupConstraint
Constructor

NodeGroupConstraint(NodeGroupConstraint.Operator, String) - Constructor for class org.quartz.locality.constraint.NodeGroupConstraint

NodeGroupConstraint.Operator - Enum in org.quartz.locality.constraint

NodeGroupEvaluator - Class in org.quartz.locality.constraint.evaluator
Evaluator that matches job execution to particular nodes based on the Trigger and JobDetail groups they are in.

NodeGroupEvaluator() - Constructor for class org.quartz.locality.constraint.evaluator.NodeGroupEvaluator
Constructor initializing all the local data structures

NodeSpec - Interface in org.quartz.locality
Encapsulates all metadata about Locality.

**NodeSpecBuilder** - Class in `org.quartz.locality
Builder for **NodeSpec**, specifically required for the DSL builder-style API:
**NodeSpecBuilder()** - Constructor for class org.quartz.locality.**NodeSpecBuilder**

**nonManagedTxDsName** - Variable in class
org.quartz.impl.jdbcjobstore.**JobStoreCMT**

**NoOpJob** - Class in `org.quartz.jobs
An implementation of Job, that does absolutely nothing - useful for system
which only wish to use **TriggerListener**s and **JobListener**s, rather than
writing Jobs that perform work.
**NoOpJob()** - Constructor for class org.quartz.jobs.**NoOpJob**

**NoSuchDelegateException** - Exception in org.quartz.impl.jdbcjobstore
Exception class for when a driver delegate cannot be found for a given
configuration, or lack thereof.
**NoSuchDelegateException(String)** - Constructor for exception
org.quartz.impl.jdbcjobstore.**NoSuchDelegateException**

**NoSuchDelegateException(String, Throwable)** - Constructor for exception
org.quartz.impl.jdbcjobstore.**NoSuchDelegateException**

**not(Matcher<U>)** - Static method in class org.quartz.impl.matchers.**NotMatcher**
Create a NotMatcher that reverses the result of the given matcher.

**notifyJobListenersToBeExecuted(JobExecutionContext)** - Method in class
org.quartz.core.**QuartzScheduler**

**notifyJobListenersWasExecuted(JobExecutionContext, JobExecutionException)** - Method in class
org.quartz.core.**QuartzScheduler**

**notifyJobListenersWasVetoed(JobExecutionContext)** - Method in class
org.quartz.core.**QuartzScheduler**

**notifyJobStoreJobComplete(OperableTrigger, JobDetail, Trigger.CompletedExecutionInstruction)** - Method in class
org.quartz.core.**QuartzScheduler**

**notifyJobStoreJobVetoed(OperableTrigger, JobDetail,**
**Trigger.CompletedExecutionInstruction** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersError(String, SchedulerException)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersFinalized(Trigger)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersFinalized(Trigger)** - Method in class org.quartz.core.SchedulerSignalerImpl

**notifySchedulerListenersInStandbyMode()** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersJobAdded(JobDetail)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersJobDeleted(JobKey)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersJobDeleted(JobKey)** - Method in class org.quartz.core.SchedulerSignalerImpl

**notifySchedulerListenersPausedJob(JobKey)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersPausedJobs(String)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersPausedTrigger(TriggerKey)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersPausedTriggers(String)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersResumedJob(JobKey)** - Method in class org.quartz.core.QuartzScheduler
**notifySchedulerListenersResumedJobs(String)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersResumedTrigger(TriggerKey)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersResumedTriggers(String)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersScheduled(Trigger)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersShutdown()** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersShuttingdown()** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersStarted()** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerListenersUnscheduled(TriggerKey)** - Method in class org.quartz.core.QuartzScheduler

**notifySchedulerThread(long)** - Method in class org.quartz.core.QuartzScheduler

**notifyTriggerListenersComplete(JobExecutionContext, Trigger.CompletedExecutionInstruction)** - Method in class org.quartz.core.QuartzScheduler

**notifyTriggerListenersFired(JobExecutionContext)** - Method in class org.quartz.core.QuartzScheduler

**notifyTriggerListenersMisfired(Trigger)** - Method in class org.quartz.core.SchedulerSignalerImpl

**notifyTriggerListenersMisfired(Trigger)** - Method in class org.quartz.core.SchedulerSignalerImpl
**NotMatcher**<sup>T</sup> extends **Key** - Class in `org.quartz.impl.matchers`  
Matches using an NOT operator on another Matcher.

**NotMatcher(Matcher<T>)** - Constructor for class `org.quartz.impl.matchers.NotMatcher`

**nthdayOfWeek** - Variable in class `org.quartz.CronExpression`

**NullSampledStatisticsImpl** - Class in `org.quartz.core`

**NullSampledStatisticsImpl()** - Constructor for class `org.quartz.core.NullSampledStatisticsImpl`

**numJobsExecuted()** - Method in class `org.quartz.core.QuartzScheduler`

**numJobsExecuted()** - Method in interface `org.quartz.core.RemotableQuartzScheduler`
**ObjectAlreadyExistsException** - Exception in `org.quartz`

An exception that is thrown to indicate that an attempt to store a new object (i.e.

**ObjectAlreadyExistsException(String)** - Constructor for exception

`org.quartz.ObjectAlreadyExistsException`

Create a `ObjectAlreadyExistsException` with the given message.

**ObjectAlreadyExistsException(JobDetail)** - Constructor for exception

`org.quartz.ObjectAlreadyExistsException`

Create a `ObjectAlreadyExistsException` and auto-generate a message using the name/group from the given JobDetail.

**ObjectAlreadyExistsException(Trigger)** - Constructor for exception

`org.quartz.ObjectAlreadyExistsException`

Create a `ObjectAlreadyExistsException` and auto-generate a message using the name/group from the given Trigger.

**obtainLock(Connection, String)** - Method in class

`org.quartz.impl.jdbcjobstore.DBSemaphore`

Grants a lock on the identified resource to the calling thread (blocking until it is available).

**obtainLock(Connection, String)** - Method in class

`org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore`

Grants a lock on the identified resource to the calling thread (blocking until it is available).

**obtainLock(Connection, String)** - Method in interface

`org.quartz.impl.jdbcjobstore.Semaphore`

Grants a lock on the identified resource to the calling thread (blocking until it is available).

**obtainLock(Connection, String)** - Method in class

`org.quartz.impl.jdbcjobstore.SimpleSemaphore`

Grants a lock on the identified resource to the calling thread (blocking until it is available).

**ofType(Class<? extends Job>)** - Method in class `org.quartz.JobBuilder`

Set the class which will be instantiated and executed when a Trigger fires that is associated with this JobDetail.

**onDay(int)** - Method in class `org.quartz.DateBuilder`

Set the day of month (1-31) for the Date that will be built by this builder.
operand - Variable in class org.quartz.impl.matchers.NotMatcher

or(Matcher<?>, Matcher<?>) - Static method in class org.quartz.impl.matchers.OrMatcher
Create an OrMatcher that depends upon the result of at least one of the given matchers.

OracleDelegate - Class in org.quartz.impl.jdbcjobstore.oracle
This is a driver delegate for the Oracle 10 and 11 database.

OracleDelegate(Logger, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
Create new OrcaleDelegate instance.

OracleDelegate(Logger, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
Create new OrcaleDelegate instance.

orderDescending() - Static method in class org.quartz.locality.constraint.MemoryConstraint
Returns a MemoryConstraint to sort the nodes by memory available

org.quartz - package org.quartz
The main package of Quartz, containing the client-side interfaces.

org.quartz.core - package org.quartz.core
Contains the core classes and interfaces for the Quartz job scheduler.

org.quartz.core.jmx - package org.quartz.core.jmx

org.quartz.ee.jmx.jboss - package org.quartz.ee.jmx.jboss

org.quartz.ee.jta - package org.quartz.ee.jta

org.quartz.ee.servlet - package org.quartz.ee.servlet

org.quartz.helpers - package org.quartz.helpers
Contains helper classes to make working with Quartz easier.

org.quartz.impl - package org.quartz.impl
Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.

org.quartz.impl.calendar - package org.quartz.impl.calendar

org.quartz.impl.jdbcjobstore - package org.quartz.impl.jdbcjobstore

org.quartz.impl.jdbcjobstore.oracle - package
This package contains Trigger implementations that ship with Quartz.

Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the
org.quartz.core.QuartzScheduler.

**org.quartz.utils** - package org.quartz.utils

**org.quartz.utils.counter** - package org.quartz.utils.counter

**org.quartz.utils.counter.sampled** - package org.quartz.utils.counter.sampled

**org.quartz.utils.weblogic** - package org.quartz.utils.weblogic

**org.quartz.xml** - package org.quartz.xml

**org.terracotta.modules.ehcache.store** - package org.terracotta.modules.ehcache.store

**OrMatcher**<T extends Key> - Class in org.quartz.impl.matchers

Matches using an OR operator on two Matcher operands.

**OrMatcher(Matcher<T>, Matcher<T>)** - Constructor for class org.quartz.impl.matchers.OrMatcher

**OsConstraint** - Class in org.quartz.locality.constraint

Constraint to a specific operating system

**OsConstraint(OsConstraint.Operator, OsConstraint.OS)** - Constructor for class org.quartz.locality.constraint.OsConstraint

**OsConstraint.Operator** - Enum in org.quartz.locality.constraint

**OsConstraint.OS** - Enum in org.quartz.locality.constraint

**OsEvaluator** - Class in org.quartz.locality.constraint.evaluator

Evaluator that persists all nodes’ Operation Systems

**OsEvaluator()** - Constructor for class org.quartz.locality.constraint.evaluator.OsEvaluator

**OSX** - Static variable in class org.quartz.locality.constraint.OsConstraint
partOfNodeGroup(String) - Static method in class org.quartz.locality.constraint.NodeGroupConstraint
  Creates a NodeGroupConstraint that specifies the node group on which the Job has to be executed

passivate() - Method in class org.quartz.core.JobRunShell
  Override passivate() to ensure we always cleanup the UserTransaction.

pauseAll() - Method in class org.quartz.core.QuartzScheduler
  Pause all triggers - equivalent of calling pauseTriggers(GroupMatcher) with a matcher matching all known groups.

pauseAll() - Method in interface org.quartz.core.RemotableQuartzScheduler

pauseAll() - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Pause all triggers - equivalent of calling pauseTriggerGroup(group) on every group.

pauseAll(Connection) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Pause all triggers - equivalent of calling pauseTriggerGroup(group) on every group.

pauseAll() - Method in class org.quartz.impl.RemoteMBeanScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

pauseAll() - Method in class org.quartz.impl.RemoteScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler.

pauseAll() - Method in class org.quartz.impl.StdScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler.

pauseAll() - Method in interface org.quartz.Scheduler
  Pause all triggers - similar to calling pauseTriggerGroup(group) on every group, however, after using this method resumeAll() must be called to clear the scheduler's state of 'remembering' that all new triggers will be paused as they are added.

pauseAll() - Method in class org.quartz.simpl.RAMJobStore
  Pause all triggers - equivalent of calling pauseTriggerGroup(group) on every group.
**pauseAllTriggers()** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean

**pauseAllTriggers()** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**pausedJobGroups** - Variable in class org.quartz.simpl.RAMJobStore

**pausedTriggerGroups** - Variable in class org.quartz.simpl.RAMJobStore

**pauseJob(String, String)** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean

**pauseJob(String, String)** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**pauseJob(JobKey)** - Method in interface
org.quartz.core.RemotableQuartzScheduler

**pauseJob(JobKey)** - Method in class
org.quartz.simpl.RAMJobStore

**pauseJob(JobKey)** - Method in class
org.quartz.core.QuartzScheduler

**pauseJob(JobKey)** - Method in class
org.quartz.core.RemotableQuartzScheduler

**pauseJob(JobKey)** - Method in class
org.quartz.core.RemoteMBeanScheduler

**pauseJob(JobKey)** - Method in class
org.quartz.core.StdScheduler

**pauseJob(JobKey)** - Method in interface
org.quartz.Scheduler

**pauseJob(JobKey)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport

**pauseJob(JobKey)** - Method in class
org.quartz.impl.jdbcjobstore.RemoteMBeanScheduler

**pauseJob(JobKey)** - Method in class
org.quartz.impl.jdbcjobstore.RemoteScheduler

**pauseJob(JobKey)** - Method in class
org.quartz.impl.jdbcjobstore.StdScheduler

**pauseJob(JobKey)** - Method in interface
org.quartz.simpl.RAMJobStore

**pauseJobGroup(String)** - Method in interface
pauseJobGroup(String) - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

pauseJobs(GroupMatcher<JobKey>) - Method in class
org.quartz.core.QuartzScheduler
  Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.

pauseJobs(GroupMatcher) - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

pauseJobs(GroupMatcher<JobKey>) - Method in interface
org.quartz.core.RemotableQuartzScheduler

pauseJobs(GroupMatcher<JobKey>) - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
  Pause all of the Jobs matching the given groupMatcher - by pausing all of their Triggers.

pauseJobs(GroupMatcher<JobKey>) - Method in class
org.quartz.impl.RemoteMBeanScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

pauseJobs(GroupMatcher<JobKey>) - Method in class
org.quartz.impl.RemoteScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler.

pauseJobs(GroupMatcher<JobKey>) - Method in class
org.quartz.impl.StdScheduler
  Calls the equivalent method on the 'proxied' QuartzScheduler.

pauseJobs(GroupMatcher<JobKey>) - Method in interface
org.quartz.Scheduler
  Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.

pauseJobs(GroupMatcher<JobKey>) - Method in class
org.quartz.simpl.RAMJobStore
  Pause all of the JobDetails in the given group - by pausing all of their Triggers.

pauseJobsContaining(String) - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean
Pause all jobs whose group contains jobGroupToken
**pauseJobsContaining(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

Pause all jobs whose group ends with jobGroupSuffix
**pauseJobsEndingWith(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

Pause all jobs whose group starts with jobGroupPrefix
**pauseJobsStartingWith(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**pauseMonitoring()** - Method in class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask
Pauses the monitoring

**pauseTrigger(String, String)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**pauseTrigger(TriggerKey)** - Method in class org.quartz.core.QuartzScheduler
Pause the Trigger with the given name.

**pauseTrigger(TriggerKey)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**pauseTrigger(TriggerKey)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**pauseTrigger(TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Pause the Trigger with the given name.

**pauseTrigger(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Pause the Trigger with the given name.

**pauseTrigger(TriggerKey)** - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

**pauseTrigger(TriggerKey)** - Method in class org.quartz.impl.RemoteScheduler
- Calls the equivalent method on the 'proxied' QuartzScheduler.

**pauseTrigger(TriggerKey)** - Method in class org.quartz.impl.StdScheduler
- Calls the equivalent method on the 'proxied' QuartzScheduler.

**pauseTrigger(TriggerKey)** - Method in interface org.quartz.Scheduler
- Pause the Trigger with the given key.

**pauseTrigger(TriggerKey)** - Method in class org.quartz.simpl.RAMJobStore
- Pause the Trigger with the given name.

**pauseTriggerGroup(String)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

**pauseTriggerGroup(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in class org.quartz.core.jdbcjobstore.JobStoreSupport
- Pause all of the Triggers matching the given groupMatcher.

**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in class org.quartz.core.QuartzScheduler
- Pause all of the Triggers in the matching groups.

**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
- Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in class org.quartz.impl.RemoteMBeanScheduler
- Calls the equivalent method on the 'proxied' QuartzScheduler.

**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in class org.quartz.impl.StdScheduler
- Calls the equivalent method on the 'proxied' QuartzScheduler.

**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in interface org.quartz.Scheduler
Pause all of the Triggers in the groups matching. 
**pauseTriggers(GroupMatcher<TriggerKey>)** - Method in class org.quartz.simpl.RAMJobStore

Pause all of the known Triggers matching. 
**pauseTriggersContaining(String)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

Pause all triggers whose group contains triggerGroupToken 
**pauseTriggersContaining(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

Pause all triggers whose group ends with triggerGroupSuffix 
**pauseTriggersEndingWith(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

Pause all triggers whose group starts with triggerGroupPrefix 
**pauseTriggersStartingWith(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**peek()** - Method in class org.quartz.utils.CircularLossyQueue

Returns value at the tail of the queue 
**peekTriggers()** - Method in class org.quartz.simpl.RAMJobStore

**PersistentEvaluator<T extends Constraint,V>** - Interface in org.quartz.locality.constraint.evaluator

An particular Evaluator that will be passed a clustered ConcurrentHashMap instance to store shared data. 

**PersistJobDataAfterExecution** - Annotation Type in org.quartz

An annotation that marks a Job class as one that makes updates to its JobDataMap during execution, and wishes the scheduler to re-store the JobDataMap when execution completes.

**PointbaseDelegate** - Class in org.quartz.impl.jdbcjobstore

This is a driver delegate for the Pointbase JDBC driver. 
**PointbaseDelegate(Logger, String, String, String, ClassLoadHelper)** - Constructor for class org.quartz.impl.jdbcjobstore.PointbaseDelegate

Create new PointbaseJDBCDelegate instance.
**PointbaseDelegate**(Logger, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore.PointbaseDelegate

Create new PointbaseJDBCDelegate instance.

**PoolingConnectionProvider** - Class in org.quartz.utils

A ConnectionProvider implementation that creates its own pool of connections.

**PoolingConnectionProvider**(String, String, String, String, int, String) - Constructor for class org.quartz.utils.PoolingConnectionProvider

Create a new PoolingConnectionProvider instance.

**PoolingConnectionProvider**(Properties) - Constructor for class org.quartz.utils.PoolingConnectionProvider

Create a connection pool using the given properties.

**populateMailInfo**(JobDataMap, SendMailJob.MailInfo) - Method in class org.quartz.jobs.ee.mail.SendMailJob

**PostgreSQLDelegate** - Class in org.quartz.impl.jdbcjobstore

This is a driver delegate for the PostgreSQL JDBC driver.

**PostgreSQLDelegate**(Logger, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.PostgreSQLDelegate

Create new PostgreSQLDelegate instance.

**PostgreSQLDelegate**(Logger, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore.PostgreSQLDelegate

Create new PostgreSQLDelegate instance.

**prepareMimeMessage**(SendMailJob.MailInfo) - Method in class org.quartz.jobs.ee.mail.SendMailJob

**prepForProcessing()** - Method in class org.quartz.xml.XMLSchedulingDataProcessor

**PRINCIPAL** - Static variable in class org.quartz.jobs.ee.ejb.EJBInvokerJob

**PRINCIPAL** - Static variable in class org.quartz.jobs.ee.jms.JmsHelper

**process**(InputSource) - Method in class org.quartz.xml.XMLSchedulingDataProcessor

**processFile**(String) - Method in class org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin
**processFile()** - Method in class `org.quartz.xml.XMLSchedulingDataProcessor`
Process the xml file in the default location (a file named "quartz_jobs.xml" in the current working directory).

**processFile(String)** - Method in class
`org.quartz.xml.XMLSchedulingDataProcessor`
Process the xml file named `fileName`.

**processFile(String, String)** - Method in class
`org.quartz.xml.XMLSchedulingDataProcessor`
Process the xml file named `fileName` with the given system ID.

**processFileAndScheduleJobs(Scheduler, boolean)** - Method in class
`org.quartz.xml.XMLSchedulingDataProcessor`
Process the xml file in the default location, and schedule all of the jobs defined within it.

**processFileAndScheduleJobs(String, Scheduler)** - Method in class
`org.quartz.xml.XMLSchedulingDataProcessor`
Process the xml file in the given location, and schedule all of the jobs defined within it.

**processFileAndScheduleJobs(String, String, Scheduler)** - Method in class
`org.quartz.xml.XMLSchedulingDataProcessor`
Process the xml file in the given location, and schedule all of the jobs defined within it.

**processStreamAndScheduleJobs(InputStream, String, Scheduler)** - Method in class
`org.quartz.xml.XMLSchedulingDataProcessor`
Process the xml file named `fileName` with the given system ID.

**PROP_CC_RECIPIENT** - Static variable in class
`org.quartz.jobs.ee.mail.SendMailJob`
The e-mail address to cc the mail to.

**PROP_COMMAND** - Static variable in class `org.quartz.jobs.NativeJob`
Required parameter that specifies the name of the command (executable) to be ran.

**PROP_CONNECTION_PROVIDER_CLASS** - Static variable in class
`org.quartz.impl.StdSchedulerFactory`

**PROP_CONSUME_STREAMS** - Static variable in class `org.quartz.jobs.NativeJob`
Optional parameter (value should be 'true' or 'false') that specifies whether the spawned process's stdout and stderr streams should be consumed.

**PROP_CONTENT_TYPE** - Static variable in class
The message content type.

**PROP_DATASOURCE_DRIVER** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_JNDI_ALWAYS_LOOKUP** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_JNDI_CREDENTIALS** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_JNDI_INITIAL** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_JNDI_PRINCIPAL** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_JNDI_PROVDER** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_JNDI_URL** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_MAX_CONNECTIONS** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_PASSWORD** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_PREFIX** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_URL** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_USER** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_DATASOURCE_VALIDATION_QUERY** - Static variable in class
PROP_JOB_LISTENER PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_JOB_STORE CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_JOB_STORE_LOCK_HANDLER CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_JOB_STORE_LOCK_HANDLER PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_JOB_STORE_PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_JOBSTORE_USE_PROP - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_LISTENER_CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_MESSAGE - Static variable in class org.quartz.jobs.ee.mail.SendMailJob
  The e-mail message body.
PROP_PARAMETERS - Static variable in class org.quartz.jobs.NativeJob
  Optional parameter that specifies the parameters to be passed to the executed command.
PROP_PLUGIN_CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_PLUGIN_PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory
PROP_RECIPIENT - Static variable in class org.quartz.jobs.ee.mail.SendMailJob
  The e-mail address to send the mail to.
PROP_REPLY_TO - Static variable in class
org.quartz.jobs.ee.mail.SendMailJob

- The e-mail address the message should say to reply to.

PROP_SCHED_BATCH_TIME_WINDOW - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_CLASS_LOAD_HELPER_CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_CONTEXT_PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_DB_FAILURE_RETRY_INTERVAL - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_IDLE_WAIT_TIME - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_INSTANCE_ID - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_INSTANCE_ID_GENERATOR_CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_INSTANCE_ID_GENERATOR_PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_INSTANCE_NAME - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN_WITH_WAIT - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_JMX_EXPORT - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_JMX_OBJECT_NAME - Static variable in class
org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_JMX_PROXY** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_JMX_PROXY_CLASS** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_JOB_FACTORY_CLASS** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_JOB_FACTORY_PREFIX** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_MAKE_SCHEDULER_THREAD_DAEMON** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_MAX_BATCH_SIZE** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_NAME** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_RMI_BIND_NAME** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_RMI_CREATE_REGISTRY** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_RMI_EXPORT** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_RMI_HOST** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_RMI_PORT** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PROP_SCHED_RMI_PROXY** - Static variable in class
org.quartz.impl.StdSchedulerFactory

PROP_SCHED_RMI_SERVER_PORT - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_SCHEDULER_THREADS_INHERIT_CONTEXT_CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_SKIP_UPDATE_CHECK - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_THREAD_NAME - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_USER_TX_URL - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SCHED_WRAP_JOB_IN_USER_TX - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_SENDER - Static variable in class org.quartz.jobs.ee.mail.SendMailJob
  The e-mail address to claim the mail is from.
PROP_SMTP_HOST - Static variable in class org.quartz.jobs.ee.mail.SendMailJob
  The host name of the smtp server.
PROP_SUBJECT - Static variable in class org.quartz.jobs.ee.mail.SendMailJob
  The subject to place on the e-mail.
PROP_TABLE_PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_THREAD_POOL_CLASS - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_THREAD_POOL_PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory

PROP_TRIGGER_LISTENER_PREFIX - Static variable in class org.quartz.impl.StdSchedulerFactory
**PROP_WAIT_FOR_PROCESS** - Static variable in class org.quartz.jobs.NativeJob
Optional parameter (value should be 'true' or 'false') that specifies whether the job should wait for the execution of the native process to complete before it completes.

**PROPERTIES_FILE** - Static variable in class org.quartz.impl.StdSchedulerFactory

**PropertiesParser** - Class in org.quartz.utils
This is an utility class used to parse the properties.

**PropertiesParser(Properties)** - Constructor for class org.quartz.utils.PropertiesParser

**PropertySettingJobFactory** - Class in org.quartz.simpl
A JobFactory that instantiates the Job instance (using the default no-arg constructor, or more specifically: class.newInstance()), and then attempts to set all values in the JobExecutionContext's JobDataMap onto bean properties of the Job.

**PropertySettingJobFactory()** - Constructor for class org.quartz.simpl.PropertySettingJobFactory

**PROVIDER_URL** - Static variable in class org.quartz.jobs.ee.ejb.EJBInvokerJob

**PROVIDER_URL** - Static variable in class org.quartz.jobs.ee.jms.JmsHelper

**push(T)** - Method in class org.quartz.utils.CircularLossyQueue
Adds a new item

**put(Object, Object)** - Method in class org.quartz.impl.JobExecutionContextImpl
Put the specified value into the context's data map with the given key.

**put(Object, Object)** - Method in interface org.quartz.JobExecutionContext
Put the specified value into the context's data map with the given key.

**put(String, int)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
Adds the given int value to the StringKeyDirtyFlagMap.

**put(String, long)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
Adds the given long value to the StringKeyDirtyFlagMap.
**put(String, float)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Adds the given float value to the StringKeyDirtyFlagMap.

**put(String, double)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Adds the given double value to the StringKeyDirtyFlagMap.

**put(String, boolean)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Adds the given boolean value to the StringKeyDirtyFlagMap.

**put(String, char)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Adds the given char value to the StringKeyDirtyFlagMap.

**put(String, String)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Adds the given String value to the StringKeyDirtyFlagMap.

**put(Object, Object)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Adds the given Object value to the StringKeyDirtyFlagMap.

**putAll(Map)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
  Adds the name-value pairs in the given Map to the StringKeyDirtyFlagMap.

**putAsString(String, boolean)** - Method in class org.quartz.JobDataMap
  Adds the given boolean value as a string version to the Job's data map.

**putAsString(String, Boolean)** - Method in class org.quartz.JobDataMap
  Adds the given Boolean value as a string version to the Job's data map.

**putAsString(String, char)** - Method in class org.quartz.JobDataMap
  Adds the given char value as a string version to the Job's data map.

**putAsString(String, Character)** - Method in class org.quartz.JobDataMap
  Adds the given Character value as a string version to the Job's data map.

**putAsString(String, double)** - Method in class org.quartz.JobDataMap
  Adds the given double value as a string version to the Job's data map.

**putAsString(String, Double)** - Method in class org.quartz.JobDataMap
  Adds the given Double value as a string version to the Job's data map.

**putAsString(String, float)** - Method in class org.quartz.JobDataMap
  Adds the given float value as a string version to the Job's data map.

**putAsString(String, Float)** - Method in class org.quartz.JobDataMap
  Adds the given Float value as a string version to the Job's data map.

**putAsString(String, int)** - Method in class org.quartz.JobDataMap
  Adds the given int value as a string version to the Job's data map.

**putAsString(String, Integer)** - Method in class org.quartz.JobDataMap
  Adds the given Integer value as a string version to the Job's data map.

**putAsString(String, long)** - Method in class org.quartz.JobDataMap
  Adds the given long value as a string version to the Job's data map.

**putAsString(String, Long)** - Method in class org.quartz.JobDataMap
Adds the given Long value as a string version to the Job's data map.
**qs** - Variable in class org.quartz.core.JobRunShell

**QUARTZ_FACTORY_KEY** - Static variable in class org.quartz.ee.servlet.QuartzInitializerListener

**QUARTZ_FACTORY_KEY** - Static variable in class org.quartz.ee.servlet.QuartzInitializerServlet

**QUARTZ_NS** - Static variable in class org.quartz.xml.XMLSchedulingDataProcessor

**QUARTZ_SCHEMA_WEB_URL** - Static variable in class org.quartz.xml.XMLSchedulingDataProcessor

**QUARTZ_SYSTEM_ID_JAR_PREFIX** - Static variable in class org.quartz.xml.XMLSchedulingDataProcessor

**QUARTZ_XML_DEFAULT_FILE_NAME** - Static variable in class org.quartz.xml.XMLSchedulingDataProcessor

**QUARTZ_XSD_PATH_IN_JAR** - Static variable in class org.quartz.xml.XMLSchedulingDataProcessor

**QuartzInitializerListener** - Class in org.quartz.ee.servlet

  A ServletContextListener that can be used to initialize Quartz.

**QuartzInitializerListener()** - Constructor for class org.quartz.ee.servlet.QuartzInitializerListener

**QuartzInitializerServlet** - Class in org.quartz.ee.servlet

  A Servlet that can be used to initialize Quartz, if configured as a load-on-startup servlet in a web application.

**QuartzInitializerServlet()** - Constructor for class org.quartz.ee.servlet.QuartzInitializerServlet

**QuartzScheduler** - Class in org.quartz.core
This is the heart of Quartz, an indirect implementation of the Scheduler interface, containing methods to schedule Jobs, register JobListener instances, etc.

**QuartzScheduler(QuartzSchedulerResources, long, long)** - Constructor for class org.quartz.core.QuartzScheduler

Create a QuartzScheduler with the given configuration properties.

**QuartzSchedulerMBean** - Interface in org.quartz.core.jmx

**QuartzSchedulerMBeanImpl** - Class in org.quartz.core

**QuartzSchedulerMBeanImpl(QuartzScheduler)** - Constructor for class org.quartz.core.QuartzSchedulerMBeanImpl

QuartzSchedulerMBeanImpl

**QuartzSchedulerResources** - Class in org.quartz.core

Contains all of the resources (JobStore,ThreadPool, etc.) necessary to create a QuartzScheduler instance.

**QuartzSchedulerResources()** - Constructor for class org.quartz.core.QuartzSchedulerResources

Create an instance with no properties initialized.

**QuartzSchedulerThread** - Class in org.quartz.core

The thread responsible for performing the work of firing Trigger s that are registered with the QuartzScheduler.

**QuartzServer** - Class in org.quartz.impl

Instantiates an instance of Quartz Scheduler as a stand-alone program, if the scheduler is configured for RMI it will be made available.

**QuartzService** - Class in org.quartz.ee.jmx.jboss

JBoss specific MBean implementation for configuring, starting, and binding to JNDI a Quartz Scheduler instance.

**QuartzService()** - Constructor for class org.quartz.ee.jmx.jboss.QuartzService

**QuartzServiceMBean** - Interface in org.quartz.ee.jmx.jboss

Interface exposed via JMX for MBean for configuring, starting, and binding to JNDI a Quartz Scheduler instance.
**RAMJobStore** - Class in `org.quartz.simpl`
   This class implements a JobStore that utilizes RAM as its storage device.

**RAMJobStore()** - Constructor for class `org.quartz.simpl.RAMJobStore`
   Create a new RAMJobStore.

**recoverJobs()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
   Recover any failed or misfired jobs and clean up the data store as appropriate.

**recoverJobs(Connection)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
   Will recover any failed or misfired jobs and clean up the data store as appropriate.

**recoverMisfiredJobs(Connection, boolean)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**refireImmediately()** - Method in exception `org.quartz.JobExecutionException`

**releaseAcquiredTrigger(OperableTrigger)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
   Inform the JobStore that the scheduler no longer plans to fire the given Trigger, that it had previously acquired (reserved).

**releaseAcquiredTrigger(Connection, OperableTrigger)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**releaseAcquiredTrigger(OperableTrigger)** - Method in class `org.quartz.simpl.RAMJobStore`
   Inform the JobStore that the scheduler no longer plans to fire the given Trigger, that it had previously acquired (reserved).

**releaseLock(Connection, String)** - Method in class `org.quartz.impl.jdbcjobstore.DBSemaphore`
   Release the lock on the identified resource if it is held by the calling thread.

**releaseLock(Connection, String, boolean)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**releaseLock(Connection, String)** - Method in class `org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore`
Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.

**releaseLock(String, boolean)** - Method in class org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore
Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.

**releaseLock(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.Semaphore
Release the lock on the identified resource if it is held by the calling thread.

**releaseLock(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.SimpleSemaphore
Release the lock on the identified resource if it is held by the calling thread.

**releaseTriggerRetryLoop(OperableTrigger)** - Method in class org.quartz.core.QuartzSchedulerThread

**RemotableQuartzScheduler** - Interface in org.quartz.core

**RemoteMBeanScheduler** - Class in org.quartz.impl
An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JMX.

**RemoteMBeanScheduler()** - Constructor for class org.quartz.impl.RemoteMBeanScheduler

**RemoteScheduler** - Class in org.quartz.impl
An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via RMI.

**RemoteScheduler(String, String, int)** - Constructor for class org.quartz.impl.RemoteScheduler
Construct a RemoteScheduler instance to proxy the given RemoteableQuartzScheduler instance, and with the given SchedulingContext.

**remove(String)** - Method in class org.quartz.impl.SchedulerRepository

**remove(Object)** - Method in class org.quartz.utils.DirtyFlagMap

**removeCalendar(String)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Remove (delete) the calendar with the given name.

**removeCalendar**([`Connection`, `String`]) - Method in class
org.quartz.impl.jdbcjobstore.[`JobStoreSupport`](https://www.javalobby.org/topics/)

**removeCalendar**(`String`) - Method in class org.quartz.simpl.[`RAMJobStore`](https://www.javalobby.org/topics/)
Remove (delete) the calendar with the given name.

**removeExcludedDate**([`Date`]) - Method in class
org.quartz.impl.calendar.[`HolidayCalendar`](https://www.javalobby.org/topics/)

**removeExcludedDay**([`Calendar`]) - Method in class
org.quartz.impl.calendar.[`AnnualCalendar`](https://www.javalobby.org/topics/)
Remove the given day from the list of excluded days.

**removeInternalJobListener**(`String`) - Method in class
org.quartz.core.[`QuartzScheduler`](https://www.javalobby.org/topics/)
Remove the identified JobListener from the Scheduler's list of `internal` listeners.

**removeInternalSchedulerListener**([`SchedulerListener`]) - Method in class
org.quartz.core.[`QuartzScheduler`](https://www.javalobby.org/topics/)
Remove the given SchedulerListener from the Scheduler's list of internal listeners.

**removeInternalTriggerListener**(`String`) - Method in class
org.quartz.core.[`QuartzScheduler`](https://www.javalobby.org/topics/)
Remove the identified TriggerListener from the Scheduler's list of `internal` listeners.

**removeJob**([`JobKey`]) - Method in class
org.quartz.impl.jdbcjobstore.[`JobStoreSupport`](https://www.javalobby.org/topics/)
Remove (delete) the Job with the given name, and any Trigger s that reference it.

**removeJob**([`Connection`, `JobKey`, `boolean`]) - Method in class
org.quartz.impl.jdbcjobstore.[`JobStoreSupport`](https://www.javalobby.org/topics/)
Remove (delete) the Job with the given name, and any Trigger s that reference it.

**removeJob**([`JobKey`]) - Method in class org.quartz.simpl.[`RAMJobStore`](https://www.javalobby.org/topics/)
Remove (delete) the Job with the given name, and any Trigger s that reference it.

**removeJobGroupToNeverDelete**(`String`) - Method in class
org.quartz.xml.[`XMLSchedulingDataProcessor`](https://www.javalobby.org/topics/)
Remove the given group to the list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.
**removeJobListener(String)** - Method in class org.quartz.core.ListenerManagerImpl

**removeJobListener(String)** - Method in interface org.quartz.ListenerManager

Remove the identified JobListener from the Scheduler.

**removeJobListenerMatcher(String, Matcher<JobKey>)** - Method in class org.quartz.core.ListenerManagerImpl

**removeJobListenerMatcher(String, Matcher<JobKey>)** - Method in interface org.quartz.ListenerManager

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

**removeJobs(List<JobKey>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**removeJobs(List<JobKey>)** - Method in class org.quartz.simpl.RAMJobStore

**removeListener(JobListener)** - Method in class org.quartz.listeners.BroadcastJobListener

**removeListener(String)** - Method in class org.quartz.listeners.BroadcastJobListener

**removeListener(SchedulerListener)** - Method in class org.quartz.listeners.BroadcastSchedulerListener

**removeListener(TriggerListener)** - Method in class org.quartz.listeners.BroadcastTriggerListener

**removeListener(String)** - Method in class org.quartz.listeners.BroadcastTriggerListener

**removeNoGCObject(Object)** - Method in class org.quartz.core.QuartzScheduler

**removeNotificationListener(NotificationListener)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**removeNotificationListener(NotificationListener, NotificationFilter, Object)**
- Method in class org.quartz.core.QuartzSchedulerMBeanImpl

removeSchedulerListener(SchedulerListener) - Method in class org.quartz.core.ListenerManagerImpl

removeSchedulerListener(SchedulerListener) - Method in interface org.quartz.ListenerManager

Remove the given SchedulerListener from the Scheduler.

removeTransientData() - Method in class org.quartz.utils.StringKeyDirtyFlagMap

Deprecated. JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

removeTrigger(TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Remove (delete) the Trigger with the given name.

removeTrigger(Connection, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

removeTrigger(TriggerKey) - Method in class org.quartz.simpl.RAMJobStore

Remove (delete) the Trigger with the given name.

removeTriggerGroupToNeverDelete(String) - Method in class org.quartz.xml.XMLSchedulingDataProcessor

Remove the given group to the list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

removeTriggerListener(String) - Method in class org.quartz.core.ListenerManagerImpl

removeTriggerListener(String) - Method in interface org.quartz.ListenerManager

Remove the identified TriggerListener from the Scheduler.

removeTriggerListenerMatcher(String, Matcher<TriggerKey>) - Method in class org.quartz.core.ListenerManagerImpl

removeTriggerListenerMatcher(String, Matcher<TriggerKey>) - Method in interface org.quartz.ListenerManager

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.
**removeTriggers(List<TriggerKey>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**removeTriggers(List<TriggerKey>)** - Method in class org.quartz.simpl.RAMJobStore

**REPEAT_INDEFINITELY** - Static variable in interface org.quartz.SimpleTrigger

  Used to indicate the 'repeat count' of the trigger is indefinite.

**repeatForever()** - Method in class org.quartz.SimpleScheduleBuilder

  Specify that the trigger will repeat indefinitely.

**repeatHourlyForever()** - Static method in class org.quartz.SimpleScheduleBuilder

  Create a SimpleScheduleBuilder set to repeat forever with a 1 hour interval.

**repeatHourlyForever(int)** - Static method in class org.quartz.SimpleScheduleBuilder

  Create a SimpleScheduleBuilder set to repeat forever with an interval of the given number of hours.

**repeatHourlyForTotalCount(int)** - Static method in class org.quartz.SimpleScheduleBuilder

  Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 hour interval.

**repeatHourlyForTotalCount(int, int)** - Static method in class org.quartz.SimpleScheduleBuilder

  Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of hours.

**repeatMinutelyForever()** - Static method in class org.quartz.SimpleScheduleBuilder

  Create a SimpleScheduleBuilder set to repeat forever with a 1 minute interval.

**repeatMinutelyForever(int)** - Static method in class org.quartz.SimpleScheduleBuilder

  Create a SimpleScheduleBuilder set to repeat forever with an interval of the given number of minutes.

**repeatMinutelyForTotalCount(int)** - Static method in class org.quartz.SimpleScheduleBuilder

  Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 minute interval.

**repeatMinutelyForTotalCount(int, int)** - Static method in class
org.quartz.SimpleScheduleBuilder

Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of minutes.

repeatSecondlyForever() - Static method in class org.quartz.SimpleScheduleBuilder

Create a SimpleScheduleBuilder set to repeat forever with a 1 second interval.

repeatSecondlyForever(int) - Static method in class org.quartz.SimpleScheduleBuilder

Create a SimpleScheduleBuilder set to repeat forever with an interval of the given number of seconds.

repeatSecondlyForTotalCount(int) - Static method in class org.quartz.SimpleScheduleBuilder

Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 second interval.

repeatSecondlyForTotalCount(int, int) - Static method in class org.quartz.SimpleScheduleBuilder

Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of seconds.

replaceTrigger(TriggerKey, OperableTrigger) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

replaceTrigger(Connection, TriggerKey, OperableTrigger) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

replaceTrigger(TriggerKey, OperableTrigger) - Method in class org.quartz.simpl.RAMJobStore

requestRecovery() - Method in class org.quartz.JobBuilder

Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.

requestRecovery(boolean) - Method in class org.quartz.JobBuilder

Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.

requestShutdown() - Method in class org.quartz.core.JobRunShell

requestsRecovery() - Method in class org.quartz.impl.JobDetailImpl

requestsRecovery() - Method in interface org.quartz.JobDetail
Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.

requestsRecovery() - Method in class org.quartz.locality.DelegatingLocalityJobDetail
Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.

requiresConnection() - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore
This Semaphore implementation does use the database.

requiresConnection() - Method in class org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore
This Semaphore implementation does not use the database.

requiresConnection() - Method in interface org.quartz.impl.jdbcjobstoreSemaphore
Whether this Semaphore implementation requires a database connection for its lock management operations.

requiresConnection() - Method in class org.quartz.impl.jdbcjobstore.SimpleSemaphore
This Semaphore implementation does not use the database.

rescheduleJob(TriggerKey, Trigger) - Method in class org.quartz.core.QuartzScheduler
Remove (delete) the Trigger with the given name, and store the new given one - which must be associated with the same job.

rescheduleJob(TriggerKey, Trigger) - Method in interface org.quartz.core.RemotableQuartzScheduler

rescheduleJob(TriggerKey, Trigger) - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

rescheduleJob(TriggerKey, Trigger) - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

rescheduleJob(TriggerKey, Trigger) - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

rescheduleJob(TriggerKey, Trigger) - Method in interface org.quartz.Scheduler
Remove (delete) the Trigger with the given key, and store the new given
one - which must be associated with the same job (the new trigger must have the job name & group specified) - however, the new trigger need not have the same name as the old trigger.

**resetOnSample** - Variable in class
org.quartz.utils.counter.sampled.[SampledCounterImpl](#)
Should the counter reset on each sample?

**resolveSchemaSource()** - Method in class
org.quartz.xml.[XMLSchedulingDataProcessor](#)

**restoreOriginalAttributes()** - Method in class
org.quartz.impl.jdbcjobstore.[AttributeRestoringConnectionInvocationHandler](#)
Attempts to restore the auto commit and transaction isolation connection attributes of the wrapped connection to their original values (if they were overwritten).

**resumeAll()** - Method in class org.quartz.core.[QuartzScheduler](#)
Resume (un-pause) all triggers - equivalent of calling resumeTriggerGroup(group) on every group.

**resumeAll()** - Method in interface org.quartz.core.[RemotableQuartzScheduler](#)

**resumeAll()** - Method in class org.quartz.impl.jdbcjobstore.[JobStoreSupport](#)
Resume (un-pause) all triggers - equivalent of calling resumeTriggerGroup(group) on every group.

**resumeAll()** - Method in class org.quartz.impl.jdbcjobstore.[JobStoreSupport](#)
protected

**resumeAll()** - Method in class org.quartz.impl.[RemoteMBeanScheduler](#)
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**resumeAll()** - Method in class org.quartz.impl.[RemoteScheduler](#)
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeAll()** - Method in class org.quartz.impl.[StdScheduler](#)
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeAll()** - Method in interface org.quartz.[Scheduler](#)
Resume (un-pause) all triggers - similar to calling resumeTriggerGroup(group) on every group.

**resumeAll()** - Method in class org.quartz.simpl.[RAMJobStore](#)
Resume (un-pause) all triggers - equivalent of calling resumeTriggerGroup(group) on every group.

**resumeAllTriggers()** - Method in interface
resumeAllTriggers() - Method in class org.quartz.core.QuartzSchedulerMBean

resumeJob(String, String) - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

resumeJob(JobKey) - Method in class org.quartz.core.QuartzScheduler
Resume (un-pause) the Job with the given name.

resumeJob(String, String) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

resumeJob(JobKey) - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

resumeJob(JobKey) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

resumeJob(JobKey) - Method in interface org.quartz.core.RemotableQuartzScheduler

resumeJob(JobKey) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl
Resume (un-pause) the Job with the given name.

resumeJob(JobKey) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

resumeJob(JobKey) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Resume (un-pause) the Job with the given name.

resumeJob(JobKey) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Calls the equivalent method on the 'proxied' QuartzScheduler.

resumeJob(JobKey) - Method in class org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

resumeJob(JobKey) - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

resumeJob(JobKey) - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

resumeJob(JobKey) - Method in class org.quartz.impl.RAMJobStore
Resume (un-pause) the Job with the given key.

resumeJob(JobKey) - Method in class org.quartz.impl.RAMJobStore
Resume (un-pause) the Job with the given name.

resumeJobGroup(String) - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

resumeJobGroup(String) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

resumeJobs(GroupMatcher<JobKey>) - Method in class org.quartz.core.QuartzScheduler
Resume (un-pause) all of the JobDetails in the matching groups.
**resumeJobs(GroupMatcher)** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**resumeJobs(GroupMatcher<JobKey>)** - Method in interface
org.quartz.core.RemotableQuartzScheduler

**resumeJobs(GroupMatcher<JobKey>)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
Resume (un-pause) all of the Jobs in the given group.

**resumeJobs(GroupMatcher<JobKey>)** - Method in class
org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**resumeJobs(GroupMatcher<JobKey>)** - Method in class
org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeJobs(GroupMatcher<JobKey>)** - Method in class
org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeJobs(GroupMatcher<JobKey>)** - Method in interface
org.quartz.Scheduler
Resume (un-pause) all of the JobDetails in matching groups.

**resumeJobs(GroupMatcher<JobKey>)** - Method in class
org.quartz.simpl.RAMJobStore
Resume (un-pause) all of the JobDetails in the given group.

**resumeJobsContaining(String)** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean
Resume all jobs whose group contains jobGroupToken

**resumeJobsContaining(String)** - Method in class
org.quartz.core.jmx.QuartzSchedulerMBeanImpl

**resumeJobsEndingWith(String)** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean
Resume all jobs whose group ends with jobGroupSuffix

**resumeJobsEndingWith(String)** - Method in class
org.quartz.core.jmx.QuartzSchedulerMBeanImpl

**resumeJobsStartingWith(String)** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean
Resume all jobs whose group starts with jobGroupPrefix

**resumeJobsStartingWith(String)** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**resumeMonitoring()** - Method in class
org.quartz.locality.constraint.evaluator.EvaluatorTimerTask
resumes the monitoring

**resumeTrigger(String, String)** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean

**resumeTrigger(TriggerKey)** - Method in class org.quartz.core.QuartzScheduler
Resume (un-pause) the Trigger with the given name.

**resumeTrigger(String, String)** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**resumeTrigger(TriggerKey)** - Method in interface
org.quartz.core.RemotableQuartzScheduler

**resumeTrigger(TriggerKey)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
Resume (un-pause) the Trigger with the given name.

**resumeTrigger(Connection, TriggerKey)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
Resume (un-pause) the Trigger with the given name.

**resumeTrigger(TriggerKey)** - Method in class
org.quartz.impl.RemoteMBeanScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

**resumeTrigger(TriggerKey)** - Method in class
org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeTrigger(TriggerKey)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeTrigger(TriggerKey)** - Method in interface org.quartz.Scheduler
Resume (un-pause) the Trigger with the given key.

**resumeTrigger(TriggerKey)** - Method in class org.quartz.simpl.RAMJobStore
Resume (un-pause) the Trigger with the given key.

**resumeTriggerGroup(String)** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean
**resumeTriggerGroup(String)** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**resumeTriggerGroup(Connection, GroupMatcher<TriggerKey>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Resume (un-pause) all of the Triggers matching the given groupMatcher.

**resumeTriggers(GroupMatcher<TriggerKey>)** - Method in class
org.quartz.core.QuartzScheduler
Resume (un-pause) all of the Triggers in the matching groups.

**resumeTriggers(GroupMatcher<TriggerKey>)** - Method in interface
org.quartz.core.RemotableQuartzScheduler
Resume (un-pause) all of the Triggers matching the given groupMatcher.

**resumeTriggers(GroupMatcher<TriggerKey>)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**resumeTriggers(GroupMatcher<TriggerKey>)** - Method in class
org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeTriggers(GroupMatcher<TriggerKey>)** - Method in class
org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**resumeTriggers(GroupMatcher<TriggerKey>)** - Method in interface
org.quartz. Scheduler
Resume (un-pause) all of the Triggers in matching groups.

**resumeTriggers(GroupMatcher<TriggerKey>)** - Method in class
org.quartz.simpl.RAMJobStore
Resume (un-pause) all of the Triggers in the given group.

**resumeTriggersContaining(String)** - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean
Resume all triggers whose group contains triggerGroupToken

**resumeTriggersContaining(String)** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

**resumeTriggersEndingWith(String)** - Method in interface
Resume all triggers whose group ends with triggerGroupSuffix

**resumeTriggersEndingWith(String)** - Method in class
org.quartz.core.QuartzSchedulerMBean

Resume all triggers whose group starts with triggerGroupPrefix

**resumeTriggersStartingWith(String)** - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

Retrieve the given Trigger.

**retrieveCalendar(String)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport

Retrieve the given JobDetail for the given Job.

**retrieveJob(JobKey)** - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport

Return a UserTransaction that was retrieved via getUserTransaction().

**rightOperand** - Variable in class org.quartz.impl.matchers.AndMatcher
**rightOperand** - Variable in class org.quartz.impl.matchers.OrMatcher

**rollbackConnection(Connection)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
   Rollback the supplied connection.

**rtp(String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   Replace the table prefix in a query by replacing any occurrences of "{0}" with the table prefix.

**rtp(String, String, String)** - Static method in class org.quartz.impl.jdbcjobstore.Util
   Replace the table prefix in a query by replacing any occurrences of "{0}" with the table prefix.

**run()** - Method in class org.quartz.core.JobRunShell

**run()** - Method in class org.quartz.core.QuartzSchedulerThread
   The main processing loop of the QuartzSchedulerThread.

**run()** - Method in class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask

**run()** - Method in class org.quartz.utils.UpdateChecker
   Run the update check

**runInThread(Runnable)** - Method in class org.quartz.simpl.SimpleThreadPool
   Run the given Runnable object in the next available Thread.

**runInThread(Runnable)** - Method in class org.quartz.simpl.ZeroSizeThreadPool

**runningSince()** - Method in class org.quartz.core.QuartzScheduler

**runningSince()** - Method in interface org.quartz.core.RemotableQuartzScheduler
**SAMPLED_STATISTICS_ENABLED** - Static variable in interface org.quartz.core.jmx.QuartzSchedulerMBean

**SAMPLED_STATISTICS_RESET** - Static variable in interface org.quartz.core.jmx.QuartzSchedulerMBean

**SampledCounter** - Interface in org.quartz.utils.counter.sampled

Interface of a sampled counter -- a counter that keeps sampled values

**SampledCounterConfig** - Class in org.quartz.utils.counter.sampled

Config for a SampledCounter

**SampledCounterConfig(int, int, boolean, long)** - Constructor for class org.quartz.utils.counter.sampled.SampledCounterConfig

Make a new timed counter config (duh)

**SampledCounterImpl** - Class in org.quartz.utils.counter.sampled

An implementation of SampledCounter

**SampledCounterImpl(SampledCounterConfig)** - Constructor for class org.quartz.utils.counter.sampled.SampledCounterImpl

todo GL how many threads is this creating? Constructor accepting a SampledCounterConfig

**SampledRateCounter** - Interface in org.quartz.utils.counter.sampled

Interface of a sampled rate counter -- a counter that keeps sampled values of rates

**SampledRateCounterConfig** - Class in org.quartz.utils.counter.sampled

An implementation of SampledCounterConfig

**SampledRateCounterConfig(int, int, boolean)** - Constructor for class org.quartz.utils.counter.sampled.SampledRateCounterConfig

Constructor accepting the interval time in seconds, history-size and whether counters should reset on each sample or not.

**SampledRateCounterConfig(int, int, boolean, long, long)** - Constructor for class org.quartz.utils.counter.sampled.SampledRateCounterConfig

Constructor accepting the interval time in seconds, history-size and whether counters should reset on each sample or not.

**SampledRateCounterImpl** - Class in org.quartz.utils.counter.sampled

An implementation of SampledRateCounter

**SampledRateCounterImpl(SampledRateCounterConfig)** - Constructor for
class org.quartz.utils.counter.sampled.\texttt{SampledRateCounterImpl}
Constructor accepting the config \texttt{SampledStatistics} - Interface in \texttt{org.quartz.core}

\texttt{SampledStatisticsImpl} - Class in \texttt{org.quartz.core}

\texttt{SATURDAY} - Static variable in class \texttt{org.quartz.DateBuilder}

\texttt{sched} - Variable in class \texttt{org.quartz.core.SchedulerSignalerImpl}

\texttt{SCHED\_NAME\_SUBST} - Static variable in interface \texttt{org.quartz.impl.jdbcjobstore.StdJDBCConstants}

\texttt{schedName} - Variable in class \texttt{org.quartz.impl.jdbcjobstore.StdJDBCDelegate}

\texttt{schedNameLiteral} - Variable in class \texttt{org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate}

\texttt{schedNameLiteral} - Variable in class \texttt{org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport}

\texttt{schedNameLiteral} - Variable in class \texttt{org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate}

\texttt{schedThread} - Variable in class \texttt{org.quartz.core.SchedulerSignalerImpl}

\texttt{scheduleBasicJob(Map\texttt{<String, Object>>, Map\texttt{<String, Object>>})} - Method in interface \texttt{org.quartz.core.jmx.QuartzSchedulerMBean}

Schedules a job using the given Cron/Simple triggerInfo.

\texttt{scheduleBasicJob(Map\texttt{<String, Object>>, Map\texttt{<String, Object>>})} - Method in class \texttt{org.quartz.core.QuartzSchedulerMBeanImpl}

\texttt{ScheduleBuilder\texttt{<T}} extends \texttt{Trigger} - Class in \texttt{org.quartz}

\texttt{ScheduleBuilder()} - Constructor for class \texttt{org.quartz.ScheduleBuilder}

\texttt{scheduleJob(String, String, String, String)} - Method in interface \texttt{org.quartz.core.jmx.QuartzSchedulerMBean}

Schedule an existing job with an existing trigger.
**scheduleJob(Map<String, Object>, Map<String, Object>)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

Schedules an arbitrary job described by abstractJobInfo using a trigger specified by abstractTriggerInfo.

**scheduleJob(String, String, Map<String, Object>)** - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

Schedules the specified job using a trigger described by abstractTriggerInfo, which must contain the fully-qualified trigger class name under the key "triggerClass." That trigger type must contain a no-arg constructor and have public access.

**scheduleJob(JobDetail, Trigger)** - Method in class org.quartz.core.QuartzScheduler

Add the Job identified by the given JobDetail to the Scheduler, and associate the given Trigger with it.

**scheduleJob(Trigger)** - Method in class org.quartz.core.QuartzScheduler

Schedule the given Trigger with the Job identified by the Trigger's settings.

**scheduleJob(Map<String, Object>, Map<String, Object>)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**scheduleJob(String, String, Map<String, Object>)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**scheduleJob(String, String, String, String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**scheduleJob(JobDetail, Trigger)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**scheduleJob(Trigger)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**scheduleJob(JobDetail, Trigger)** - Method in class org.quartz.impl.RemoteMBeanScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**scheduleJob(Trigger)** - Method in class org.quartz.impl.RemoteMBeanScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

**scheduleJob(JobDetail, Trigger)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**scheduleJob(Trigger)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**scheduleJob(JobDetail, Trigger)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**scheduleJob(Trigger)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**scheduleJob(JobDetail, Trigger)** - Method in interface org.quartz.Scheduler
Add the given JobDetail to the Scheduler, and associate the given Trigger with it.

**scheduleJob(Trigger)** - Method in interface org.quartz.Scheduler
Schedule the given Trigger with the Job identified by the Trigger's settings.

**scheduleJobs(Map<JobDetail, List<Trigger>>, boolean)** - Method in class org.quartz.core.QuartzScheduler
Schedule all of the given jobs with the related set of triggers.

**scheduleJobs(Map<JobDetail, List<Trigger>>, boolean)** - Method in class org.quartz.core.RemotalbeQuartzScheduler

**scheduleJobs(Map<JobDetail, List<Trigger>>, boolean)** - Method in class org.quartz.impl.RemoteMBeanScheduler

**scheduleJobs(Map<JobDetail, List<Trigger>>, boolean)** - Method in class org.quartz.impl.RemoteScheduler

**scheduleJobs(Map<JobDetail, List<Trigger>>, boolean)** - Method in class org.quartz.impl.StdScheduler

**scheduleJobs(Map<JobDetail, List<Trigger>>, boolean)** - Method in interface org.quartz.Scheduler
Schedule the given sets of jobs and triggers.

**scheduler** - Variable in class org.quartz.core.JobRunShell
**Scheduler** - Interface in `org.quartz`
This is the main interface of a Quartz Scheduler.

**SCHEDULER_ERROR** - Static variable in interface org.quartz.core.jmx. `QuartzSchedulerMBean`

**SCHEDULER_PAUSED** - Static variable in interface org.quartz.core.jmx. `QuartzSchedulerMBean`

**SCHEDULER_SHUTDOWN** - Static variable in interface org.quartz.core.jmx. `QuartzSchedulerMBean`

**SCHEDULER_STARTED** - Static variable in interface org.quartz.core.jmx. `QuartzSchedulerMBean`

**SchedulerConfigException** - Exception in `org.quartz`
An exception that is thrown to indicate that there is a misconfiguration of the SchedulerFactory- or one of the components it configures.

**SchedulerConfigException(String)** - Constructor for exception org.quartz. `SchedulerConfigException`
Create a JobPersistenceException with the given message.

**SchedulerConfigException(String, Throwable)** - Constructor for exception org.quartz. `SchedulerConfigException`
Create a JobPersistenceException with the given message and cause.

**SchedulerContext** - Class in `org.quartz`
Holds context/environment data that can be made available to Jobs as they are executed.

**SchedulerContext()** - Constructor for class org.quartz. `SchedulerContext`
Create an empty SchedulerContext.

**SchedulerContext(Map)** - Constructor for class org.quartz. `SchedulerContext`
Create a SchedulerContext with the given data.

**schedulerError(String, SchedulerException)** - Method in class org.quartz.core. `QuartzSchedulerMBeanImpl`

**schedulerError(String, SchedulerException)** - Method in class org.quartz.impl. `QuartzServer`
Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.
schedulerError(String, SchedulerException) - Method in class org.quartz.listeners.BroadcastSchedulerListener

schedulerError(String, SchedulerException) - Method in class org.quartz.listenersSchedulerListenerSupport

schedulerError(String, SchedulerException) - Method in interface org.quartz.SchedulerListener
Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

SchedulerException - Exception in org.quartz
Base class for exceptions thrown by the Quartz Scheduler.

SchedulerException() - Constructor for exception org.quartz.SchedulerException

SchedulerException(String) - Constructor for exception org.quartz.SchedulerException

SchedulerException( Throwable) - Constructor for exception org.quartz.SchedulerException

SchedulerException(String, Throwable) - Constructor for exception org.quartz.SchedulerException

SchedulerFactory - Interface in org.quartz
Provides a mechanism for obtaining client-usable handles to Scheduler instances.

schedulerInStandbyMode() - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

schedulerInStandbyMode() - Method in class org.quartz.listeners.BroadcastSchedulerListener

schedulerInStandbyMode() - Method in class org.quartz.listenersSchedulerListenerSupport

schedulerInStandbyMode() - Method in interface org.quartz.SchedulerListener
Called by the Scheduler to inform the listener that it has move to standby
mode.

**SchedulerListener** - Interface in [org.quartz](https://api.quartz-scheduler.org/javadoc/org/quartz/SchedulerListener.html)

The interface to be implemented by classes that want to be informed of major Scheduler events.

**SchedulerListenerSupport** - Class in [org.quartz.listeners](https://api.quartz-scheduler.org/javadoc/org/quartz/listeners/SchedulerListenerSupport.html)

A helpful abstract base class for implementors of `SchedulerListener`.

**SchedulerListenerSupport()** - Constructor for class `org.quartz.listeners.SchedulerListenerSupport`

**SchedulerMetaData** - Class in [org.quartz](https://api.quartz-scheduler.org/javadoc/org/quartz/SchedulerMetaData.html)

Describes the settings and capabilities of a given Scheduler instance.

**SchedulerMetaData(String, String, Class, boolean, boolean, boolean, boolean, Date, int, Class, boolean, boolean, Class, int, String)** - Constructor for class `org.quartz.SchedulerMetaData`

**SchedulerPluginWithUserTransactionSupport** - Class in [org.quartz.plugins](https://api.quartz-scheduler.org/javadoc/org/quartz/plugins/SchedulerPluginWithUserTransactionSupport.html)

Base class for plugins that wish to support having their start and shutdown methods run within a `UserTransaction`.

**SchedulerPluginWithUserTransactionSupport()** - Constructor for class `org.quartz.plugins.SchedulerPluginWithUserTransactionSupport`

**SchedulerRepository** - Class in [org.quartz.impl](https://api.quartz-scheduler.org/javadoc/org/quartz/impl/SchedulerRepository.html)

Holds references to Scheduler instances - ensuring uniqueness, and preventing garbage collection, and allowing 'global' lookups - all within a ClassLoader space.

**schedulerShutdown()** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**schedulerShutdown()** - Method in class `org.quartz.impl.QuartzServer`

Called by the Scheduler to inform the listener that it has shutdown.

**schedulerShutdown()** - Method in class `org.quartz.listeners.BroadcastSchedulerListener`

**schedulerShutdown()** - Method in class `org.quartz.listeners.SchedulerListenerSupport`

**schedulerShutdown()** - Method in interface `org.quartz.SchedulerListener`

Called by the Scheduler to inform the listener that it has shutdown.

**schedulerShuttingdown()** - Method in class `org.quartz.core.JobRunShell`
**schedulerShuttingdown()** - Method in class org.quartz.core.**QuartzSchedulerMBeanImpl**

**schedulerShuttingdown()** - Method in class org.quartz.listeners.**BroadcastSchedulerListener**

**schedulerShuttingdown()** - Method in class org.quartz.listeners.**SchedulerListenerSupport**

**schedulerShuttingdown()** - Method in interface org.quartz.**SchedulerListener**
   Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

**SchedulerSignalerImpl** - Class in org.quartz.core
   An interface to be used by JobStore instances in order to communicate signals back to the QuartzScheduler.

**SchedulerSignalerImpl(QuartzScheduler, QuartzSchedulerThread)** - Constructor for class org.quartz.core.**SchedulerSignalerImpl**

**schedulerStarted()** - Method in class org.quartz.core.**QuartzSchedulerMBeanImpl**

**schedulerStarted()** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

**schedulerStarted()** - Method in class org.quartz.listeners.**BroadcastSchedulerListener**

**schedulerStarted()** - Method in class org.quartz.listeners.**SchedulerListenerSupport**

**schedulerStarted()** - Method in interface org.quartz.**SchedulerListener**
   Called by the Scheduler to inform the listener that it has started.

**schedulerStarted()** - Method in class org.quartz.simpl.**RAMJobStore**

**SchedulerStateRecord** - Class in org.quartz.impl.jdbcjobstore
   Conveys a scheduler-instance state record.

**SchedulerStateRecord()** - Constructor for class org.quartz.impl.jdbcjobstore.**SchedulerStateRecord**
SCHEDULING_DATA_CLEARED - Static variable in interface org.quartz.core.jmx.QuartzSchedulerMBean

schedulingDataCleared() - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

schedulingDataCleared() - Method in class org.quartz.listeners.BroadcastSchedulerListener

schedulingDataCleared() - Method in class org.quartz.listeners.SchedulerListenerSupport

schedulingDataCleared() - Method in interface org.quartz.SchedulerListener
    Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.

SECOND - Static variable in class org.quartz.CronExpression

seconds - Variable in class org.quartz.CronExpression

SECONDS_IN_MOST_DAYS - Static variable in class org.quartz.DateBuilder

SELECT_BLOB_TRIGGER - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECTCALENDAR - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECTCALENDAR_EXISTENCE - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECTCALENDARS - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECTCRONTTRIGGER - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECTFIRED_TRIGGER - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
**SELECT_FIRED_TRIGGER_GROUP** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_FIRED_TRIGGER_INSTANCE_NAMES** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_FIRED_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_FIRED_TRIGGERS_OF_JOB** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_FIRED_TRIGGERS_OF_JOB_GROUP** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_FOR_LOCK** - Static variable in class org.quartz.impl.jdbcjobstore.\texttt{StdRowLockSemaphore}

**SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_INSTANCES_FIRED_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_JOB_DETAIL** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_JOB_EXECUTION_COUNT** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_JOB_EXISTENCE** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}

**SELECT_JOB_FOR_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{StdJDBCConstants}
SELECT_JOB_GROUPS - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_JOB_NONCONCURRENT - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_JOBS_IN_GROUP - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_MISFIRED_TRIGGERS - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_MISFIRED_TRIGGERS_IN_STATE - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_NEXT_FIRE_TIME - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_NEXT_TRIGGER_TO_ACQUIRE - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_NUMCALENDARS - Static variable in class org.quartz.impl.jdbcjobstore.DB2v6Delegate

SELECT_NUMCALENDARS - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_NUM_JOBS - Static variable in class org.quartz.impl.jdbcjobstore.DB2v6Delegate

SELECT_NUM_JOBS - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

SELECT_NUM_TRIGGERS - Static variable in class org.quartz.impl.jdbcjobstore.DB2v6Delegate
**SELECT_NUM_TRIGGERS** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_NUM_TRIGGERS_FOR_JOB** - Static variable in class org.quartz.impl.jdbcjobstore.**DB2v6Delegate**

**SELECT_NUM_TRIGGERS_FOR_JOB** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_NUM_TRIGGERS_IN_GROUP** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_ORACLECALENDAR_BLOB** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**

**SELECT_ORACLEJOBDETAIL_BLOB** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**

**SELECT_ORACLE_TRIGGER_JOB_DETAIL_BLOB** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**

**SELECT_PAUSED_TRIGGER_GROUP** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_PAUSED_TRIGGERGROUPS** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_REFERENCED_CALENDAR** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_SCHEDULER_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_SCHEDULERSTATES** - Static variable in interface org.quartz.impl.jdbcjobstore.**StdJDBCClConstants**

**SELECT_SIMPLE_PROPS_TRIGGER** - Static variable in class org.quartz.impl.jdbcjobstore.**SimplePropertiesTriggerPersistenceDelegateSupport**
**SELECT_SIMPLE_TRIGGER** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER_DATA** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER_EXISTENCE** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER_FOR_FIRE_TIME** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER_GROUPS** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER_GROUPS_FILTERED** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER_STATE** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGER_STATUS** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGERS_FOR_CALENDAR** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGERS_FOR_JOB** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGERS_IN_GROUP** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants

**SELECT_TRIGGERS_IN_STATE** - Static variable in interface org.quartz_impl.jdbcjobstore.StdJDBCClConstants
**selectCalendar(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Select a calendar.

**selectCalendar(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Select a calendar.

**selectCalendars(Connection)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Select all of the stored calendars.

**selectCalendars(Connection)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Select all of the stored calendars.

**selectFiredTriggerInstanceNames(Connection)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Select the distinct instance names of all fired-trigger records.

**selectFiredTriggerInstanceNames(Connection)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Select the distinct instance names of all fired-trigger records.

**selectFiredTriggerRecords(Connection, String, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Select the states of all fired-trigger records for a given trigger, or trigger group if trigger name is null.

**selectFiredTriggerRecords(Connection, String, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Select the states of all fired-trigger records for a given trigger, or trigger group if trigger name is null.

**selectFiredTriggerRecordsByJob(Connection, String, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Select the states of all fired-trigger records for a given job, or job group if job name is null.

**selectFiredTriggerRecordsByJob(Connection, String, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Select the states of all fired-trigger records for a given job, or job group if job name is null.

**selectInstancesFiredTriggerRecords(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Select the states of all fired-trigger records for a given scheduler instance.
org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**selectJobDetail(Connection, JobKey, ClassLoadHelper)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select the JobDetail object for a given job name / group name.

**selectJobDetail(Connection, JobKey, ClassLoadHelper)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select the JobDetail object for a given job name / group name.

**selectJobExecutionCount(Connection, JobKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Get the number instances of the identified job currently executing.

**selectJobExecutionCount(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**selectJobForTrigger(Connection, ClassLoadHelper, TriggerKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select the job to which the trigger is associated.

**selectJobForTrigger(Connection, ClassLoadHelper, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select the job to which the trigger is associated.

**selectJobGroups(Connection)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select all of the job group names that are stored.

**selectJobGroups(Connection)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select all of the job group names that are stored.

**selectJobsInGroup(Connection, GroupMatcher&lt;JobKey&gt;)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select all of the jobs contained in a given group.

**selectJobsInGroup(Connection, GroupMatcher&lt;JobKey&gt;)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select all of the jobs contained in a given group.

**selectMisfiredTriggers(Connection, long)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Get the names of all of the triggers that have misfired - according to the given timestamp.

**selectMisfiredTriggers(Connection, long)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Get the names of all of the triggers that have misfired.
**selectMisfiredTriggersInGroupInState**(*Connection, String, String, long*) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Get the names of all of the triggers in the given group and state that have misfired - according to the given timestamp.

**selectMisfiredTriggersInGroupInState**(*Connection, String, String, long*) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Get the names of all of the triggers in the given group and state that have misfired.

**selectMisfiredTriggersInState**(*Connection, String, long*) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Get the names of all of the triggers in the given state that have misfired - according to the given timestamp.

**selectMisfiredTriggersInState**(*Connection, String, long*) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**selectNextFireTime**(*Connection*) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
**Deprecated.** Does not account for misfires.

**selectNextFireTime**(*Connection*) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
**Deprecated.** Does not account for misfires.

**selectNumCalendars**(*Connection*) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select the total number of calendars stored.

**selectNumCalendars**(*Connection*) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select the total number of calendars stored.

**selectNumJobs**(*Connection*) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select the total number of jobs stored.

**selectNumJobs**(*Connection*) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select the total number of jobs stored.
**selectNumTriggers(Connection)** - Method in class org.quartz.impl.jdbcjobstore.**DB2v6Delegate**

**selectNumTriggers(Connection)** - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select the total number of triggers stored.

**selectNumTriggers(Connection)** - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select the total number of triggers stored.

**selectNumTriggersForJob(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.**DB2v6Delegate**

**selectNumTriggersForJob(Connection, JobKey)** - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select the number of triggers associated with a given job.

**selectNumTriggersForJob(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select the number of triggers associated with a given job.

**selectPausedTriggerGroups(Connection)** - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**

**selectSchedulerStateRecords(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
A List of all current SchedulerStateRecords.

**selectSchedulerStateRecords(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**

**selectTrigger(Connection, TriggerKey)** - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select a trigger.

**selectTrigger(Connection, TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select a trigger.

**selectTriggerForFireTime(Connection, long)** - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select the trigger that will be fired at the given fire time.
**selectTriggerForFireTime(Connection, long)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select the trigger that will be fired at the given fire time.

**selectTriggerGroups(Connection)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select all of the trigger group names that are stored.

**selectTriggerGroups(Connection, GroupMatcher<TriggerKey>)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select all of the trigger group names that are stored.

**selectTriggerJobDataMap(Connection, String, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select a trigger's JobDataMap.

**selectTriggerJobDataMap(Connection, String, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select a trigger's JobDataMap.

**selectTriggerKeysForJob(Connection, JobKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Get the names of all of the triggers associated with the given job.

**selectTriggerKeysForJob(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Get the names of all of the triggers associated with the given job.

**selectTriggersForCalendar(Connection, String)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select the triggers for a calendar

**selectTriggersForCalendar(Connection, String)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select the triggers for a calendar

**selectTriggersForJob(Connection, JobKey)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Select the triggers for a job

**selectTriggersForJob(Connection, JobKey)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Select the triggers for a job
**selectTriggersForRecoveringJobs**(*Connection*) - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select all of the triggers for jobs that are requesting recovery.

**selectTriggersForRecoveringJobs**(*Connection*) - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select all of the triggers for jobs that are requesting recovery.

**selectTriggersInGroup**(*Connection*, GroupMatcher<TriggerKey>) - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select all of the triggers contained in a given group.

**selectTriggersInGroup**(*Connection*, GroupMatcher<TriggerKey>) - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select all of the triggers contained in a given group.

**selectTriggersInState**(*Connection*, String) - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select all of the triggers in a given state.

**selectTriggersInState**(*Connection*, String) - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select all of the triggers in a given state.

**selectTriggerState**(*Connection*, TriggerKey) - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select a trigger's state value.

**selectTriggerState**(*Connection*, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select a trigger's state value.

**selectTriggerStatus**(*Connection*, TriggerKey) - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select a trigger's status (state & next fire time).

**selectTriggerStatus**(*Connection*, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select a trigger's status (state & next fire time).

**selectTriggerToAcquire**(*Connection*, long, long) - Method in interface org.quartz.impl.jdbcjobstore.**DriverDelegate**
Select the next trigger which will fire to fire between the two given timestamps in ascending order of fire time, and then descending by priority.

**selectTriggerToAcquire**(*Connection*, long, long) - Method in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**
Select the next trigger which will fire to fire between the two given timestamps in ascending order of fire time, and then descending by priority.

**Semaphore** - Interface in org.quartz.impl.jdbcjobstore
An interface for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.

**SendDestinationMessageJob** - Class in org.quartz.jobs.ee.jms

A Job that sends a javax.jms.Message to a javax.jms.Destination.

**SendDestinationMessageJob()** - Constructor for class org.quartz.jobs.ee.jms.SendDestinationMessageJob

**SendMailJob** - Class in org.quartz.jobs.ee.mail

A Job which sends an e-mail with the configured content to the configured recipient.

**SendMailJob()** - Constructor for class org.quartz.jobs.ee.mail.SendMailJob

**SendMailJob.MailInfo** - Class in org.quartz.jobs.ee.mail

**SendMailJob.MailInfo()** - Constructor for class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**sendNotification(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**sendNotification(String, Object)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**sendNotification(String, Object, String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**SendQueueMessageJob** - Class in org.quartz.jobs.ee.jms

A Job that sends a javax.jms.Message to a javax.jms.Queue

**SendQueueMessageJob()** - Constructor for class org.quartz.jobs.ee.jms.SendQueueMessageJob

**SendTopicMessageJob** - Class in org.quartz.jobs.ee.jms

A Job that sends a javax.jms.Message to a javax.jms.Topic.

**SendTopicMessageJob()** - Constructor for class org.quartz.jobs.ee.jms.SendTopicMessageJob

**sequenceNumber** - Variable in class org.quartz.core.QuartzSchedulerMBeanImpl

sequenceNumber
**serializeJobData(JobDataMap)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Remove the transient data from and then create a serialized java.util.ByteArrayOutputStream version of a JobDataMap.

**serializeObject(Object)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Create a serialized java.util.ByteArrayOutputStream version of an Object.

**serialVersionUID** - Static variable in interface org.quartz.CronTrigger

**serialVersionUID** - Static variable in interface org.quartz.SimpleTrigger

**serialVersionUID** - Static variable in interface org.quartz.Trigger

**serve(SchedulerFactory, boolean)** - Method in class org.quartz.impl.QuartzServer

**set(Constraint...)** - Method in class org.quartz.locality.NodeSpecBuilder

**setAcquireTriggersWithinLock(boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Whether or not the query and update to acquire a Trigger for firing should be performed after obtaining an explicit DB lock.

**setAllowsTransientData(boolean)** - Method in class org.quartz.utils.StringKeyDirtyFlagMap
**Deprecated. JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.**

**setAllTriggersOfJobToState(JobKey, int)** - Method in class org.quartz.simpl.RAMJobStore

**setAlwaysLookup(boolean)** - Method in class org.quartz.utils.JNDIConnectionProvider

**setAutoCommit(boolean)** - Method in class org.quartz.impl.jdbcjobstore.AttributeRestoringConnectionInvocationHandler
Sets this connection's auto-commit mode to the given state, saving the original mode.

**setBaseCalendar(Calendar)** - Method in interface org.quartz.Calendar
Set a new base calendar or remove the existing one.

`setBaseCalendar(Calendar)` - Method in class org.quartz.impl.calendar.BaseCalendar

Set a new base calendar or remove the existing one

`setBatchTimeWindow(long)` - Method in class org.quartz.core.QuartzSchedulerResources

`setBeanProps(Object, String[], Object[])` - Static method in class org.quartz.impl.jdbcjobstore.Util

`setBeanProps(Object, JobDataMap)` - Method in class org.quartz.simpl.PropertySettingJobFactory

`setBoolean(PreparedStatement, int, boolean)` - Method in class org.quartz.impl.jdbcjobstore.DB2v7Delegate

Sets the designated parameter to the given Java boolean value.

`setBoolean(PreparedStatement, int, boolean)` - Method in class org.quartz.impl.jdbcjobstore.DB2v8Delegate

Sets the designated parameter to the given Java boolean value.

`setBoolean(PreparedStatement, int, boolean)` - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Sets the designated parameter to the given Java boolean value.

`setBoolean(boolean)` - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

`setBoolean2(boolean)` - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

`setBytes(PreparedStatement, int, ByteArrayOutputStream)` - Method in class org.quartz.impl.jdbcjobstore.DB2v7Delegate

Sets the designated parameter to the byte array of the given ByteArrayInputStream.

`setBytes(PreparedStatement, int, ByteArrayOutputStream)` - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Sets the designated parameter to the byte array of the given ByteArrayInputStream.

`setBytes(PreparedStatement, int, ByteArrayOutputStream)` - Method in class org.quartz.impl.jdbcjobstore.SybaseDelegate

Sets the designated parameter to the byte array of the given ByteArrayInputStream.
ByteArrayOutputStream.

**setCalendarHour(Calendar, int)** - Method in class org.quartz.CronExpression
Advance the calendar to the particular hour paying particular attention to daylight saving problems.

**setCalendarName(String)** - Method in class org.quartz.impl.triggers.AbstractTrigger
Associate the calendar with the given name with this Trigger.

**setCalendarName(String)** - Method in class org.quartz.locality.DelegatingLocalityTrigger

**setCc(String)** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**setCheckinInterval(long)** - Method in class org.quartz.impl.jdbcjobstore.SchedulerStateRecord

**setCheckinTimestamp(long)** - Method in class org.quartz.impl.jdbcjobstore.SchedulerStateRecord

**setCleanShutdown(boolean)** - Method in class org.quartz.plugins.management.ShutdownHookPlugin
Set whether or not the plug-in is configured to cause a clean shutdown of the scheduler.

**setClusterCheckinInterval(long)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Set the frequency (in milliseconds) at which this instance "checks-in" with the other instances of the cluster.

**setContentType(String)** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**setCronExpression(String)** - Method in class org.quartz.impl.calendar.CronCalendar
Sets the cron expression for the calendar to a new value

**setCronExpression(CronExpression)** - Method in class org.quartz.impl.calendar.CronCalendar
Sets the cron expression for the calendar to a new value

**setCronExpression(String)** - Method in class org.quartz.impl.triggers.CronTriggerImpl

**setCronExpression(CronExpression)** - Method in class org.quartz.impl.triggers.CronTriggerImpl
Set the CronExpression to the given one.

**setDataSource(String)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Set the name of the dataSource that should be used for performing database functions.

**setDayExcluded(Calendar, boolean)** - Method in class org.quartz.impl.calendar.AnnualCalendar

Redefine a certain day to be excluded (true) or included (false).

**setDayExcluded(int, boolean)** - Method in class org.quartz.impl.calendar.MonthlyCalendar

Redefine a certain day of the month to be excluded (true) or included (false).

**setDayExcluded(int, boolean)** - Method in class org.quartz.impl.calendar.WeeklyCalendar

Redefine a certain day of the week to be excluded (true) or included (false).

**setDaysExcluded(ArrayList)** - Method in class org.quartz.impl.calendar.AnnualCalendar

Redefine the list of days excluded.

**setDaysExcluded(boolean[])** - Method in class org.quartz.impl.calendar.MonthlyCalendar

Redefine the array of days excluded.

**setDaysExcluded(boolean[])** - Method in class org.quartz.impl.calendar.WeeklyCalendar

Redefine the array of days excluded.

**setDbFailureRetryInterval(long)** - Method in class org.quartz.core.QuartzSchedulerThread

**setDbRetryInterval(long)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**setDecimal1(BigDecimal)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setDecimal2(BigDecimal)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setDenominatorValue(long)** - Method in interface org.quartz.utils.counter.sampled.SampledRateCounter

Sets the value of the denominator to the passed value
**setDenominatorValue(long)** - Method in class org.quartz.utils.counter.sampled.SampledRateCounterImpl
Sets the value of the denominator to the passed value

**setDescription(String)** - Method in interface org.quartz.Calendar
Set a description for the Calendar instance - may be useful for remembering/displaying the purpose of the calendar, though the description has no meaning to Quartz.

**setDescription(String)** - Method in class org.quartz.impl.calendar.BaseCalendar
Set a description for the Calendar instance - may be useful for remembering/displaying the purpose of the calendar, though the description has no meaning to Quartz.

**setDescription(String)** - Method in class org.quartz.impl.JobDetailImpl
Set a description for the Job instance - may be useful for remembering/displaying the purpose of the job, though the description has no meaning to Quartz.

**setDescription(String)** - Method in class org.quartz.impl.triggers.AbstractTrigger
Set a description for the Trigger instance - may be useful for remembering/displaying the purpose of the trigger, though the description has no meaning to Quartz.

**setDontSetAutoCommitFalse(boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Don't call set autocommit(false) on connections obtained from the DataSource.

**setDontSetNonManagedTXConnectionAutoCommitFalse(boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreCMT
Don't call set autocommit(false) on connections obtained from the DataSource.

**setDoubleCheckLockMisfireHandler(boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Set whether to check to see if there are Triggers that have misfired before actually acquiring the lock to recover them.

**setDriverDelegateClass(String)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Set the JDBC driver delegate class.

**setDriverDelegateInitString(String)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Set the JDBC driver delegate's initialization string.

**setDurability(boolean)** - Method in class `org.quartz.impl.JobDetailImpl`
Set whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

**setEndTime(Date)** - Method in class `org.quartz.impl.triggers.AbstractTrigger`
Set the time at which the Trigger should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).

**setEndTime(Date)** - Method in class `org.quartz.impl.triggers.CalendarIntervalTriggerImpl`
Set the time at which the `CalendarIntervalTrigger` should quit repeating (and be automatically deleted).

**setEndTime(Date)** - Method in class `org.quartz.impl.triggers.CronTriggerImpl`
Set the time at which the `CronTrigger` should quit repeating (and be automatically deleted).

**setEndTime(Date)** - Method in class `org.quartz.impl.triggers.SimpleTriggerImpl`
Set the time at which the `SimpleTrigger` should quit repeating (and be automatically deleted).

**setEndTime(Date)** - Method in class `org.quartz.locale.DelegatingLocalityTrigger`

**setFailOnFileNotFound(boolean)** - Method in class `org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin`
Whether or not initialization of the plugin should fail (throw an exception) if the file cannot be found.

**setFileNames(String)** - Method in class `org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin`
The file name (and path) to the XML file that should be read.

**setFireInstanceId(String)** - Method in class `org.quartz.impl.jdbcjobstore.FiredTriggerRecord`
This method should not be used by the Quartz client.

**setFireInstanceId(String)** - Method in class `org.quartz.locale.DelegatingLocalityTrigger`

**setFireInstanceState(String)** - Method in class `org.quartz.impl.jdbcjobstore.FiredTriggerRecord`

**setFireTimestamp(long)** - Method in class `org.quartz.impl.jdbcjobstore.FiredTriggerRecord`
**setFrom(String)** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**setGroup(String)** - Method in class org.quartz.impl.JobDetailImpl
  Set the group of this Job.

**setGroup(String)** - Method in class org.quartz.impl.triggers.AbstractTrigger
  Set the name of this Trigger.

**setIgnoreDuplicates(boolean)** - Method in class org.quartz.xml.XMLSchedulingDataProcessor
  If true (and OverWriteExistingData is false) then any job/triggers encountered in this file that have names that already exist in the scheduler will be ignored, and no error will be produced.

**setInsertSQL(String)** - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore

**setInstanceId(String)** - Method in class org.quartz.core.QuartzSchedulerResources
  Set the name for the QuartzScheduler.

**setInstanceId(String)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Set the instance Id of the Scheduler (must be unique within a cluster).

**setInstanceId(String)** - Method in class org.quartz.simpl.RAMJobStore

**setInstanceId(String)** - Method in class org.quartz.simpl.SimpleThreadPool

**setInstanceId(String)** - Method in class org.quartz.simpl.ZeroSizeThreadPool

**setInstanceName(String)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
  Set the instance name of the Scheduler (must be unique within this server instance).

**setInstanceName(String)** - Method in class org.quartz.simpl.RAMJobStore

**setInstanceName(String)** - Method in class org.quartz.simpl.SimpleThreadPool

**setInstanceName(String)** - Method in class org.quartz.simpl.ZeroSizeThreadPool
**setInt1(int)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setInt2(int)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setInterruptJobsOnShutdown(boolean)** - Method in class org.quartz.core.QuartzSchedulerResources

**setInterruptJobsOnShutdownWithWait(boolean)** - Method in class org.quartz.core.QuartzSchedulerResources

**setInvertTimeRange(boolean)** - Method in class org.quartz.impl.calendar.DailyCalendar

Indicates whether the time range represents an inverted time range (see class description).

**setIsClustered(boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Set whether this instance is part of a cluster.

**setJMXExport(boolean)** - Method in class org.quartz.core.QuartzSchedulerResources

Set whether the QuartzScheduler should be registered with the local MBeanServer.

**setJMXObjectName(String)** - Method in class org.quartz.core.QuartzSchedulerResources

Set the name under which the QuartzScheduler should be registered with the local MBeanServer.

**setJndiName(String)** - Method in class org.quartz.ee.jmx.jboss.QuartzService

**setJndiName(String)** - Method in interface org.quartz.ee.jmx.jboss.QuartzServiceMBean

**setJobClass(Class<? extends Job>)** - Method in class org.quartz.impl.JobDetailImpl

Set the instance of Job that will be executed.

**setJobDataMap(JobDataMap)** - Method in class org.quartz.impl.JobDetailImpl

Set the JobDataMap to be associated with the Job.

**setJobDataMap(JobDataMap)** - Method in class org.quartz.impl.JobDetailImpl
Set the JobDataMap to be associated with the Trigger.

**setJobDataMap(JobDataMap)** - Method in class `org.quartz.impl.triggers.AbstractTrigger`

Set the JobFactory that will be responsible for producing instances of Job classes.

**setJobFailedMessage(String)** - Method in class `org.quartz.plugins.history.LoggingJobHistoryPlugin`

Set the name of the associated JobDetail's group.

**setJobGroup(String)** - Method in class `org.quartz.impl.triggers.AbstractTrigger`

Set the JobFactory to be associated with the Trigger.

**setJobDisallowsConcurrentExecution(boolean)** - Method in class `org.quartz.impl.jdbcjobstore.FiredTriggerRecord`

Set the JobFactory that will be responsible for producing instances of Job classes.

**setJobFactory(JobFactory)** - Method in class `org.quartz.locality.DelegatingLocalityTrigger`

**setJobFactory(JobFactory)** - Method in class `org.quartz.impl.jdbcjobstore.FiredTriggerRecord`

**setJobFactory(JobFactory)** - Method in class `org.quartz.impl.RemoteMBeanScheduler`

**setJobFactory(JobFactory)** - Method in class `org.quartz.impl.RemoteScheduler`

**setJobFactory(JobFactory)** - Method in class `org.quartz.impl.StdScheduler`

**setJobFactory(JobFactory)** - Method in interface `org.quartz.Scheduler`

Set the message that is logged when a Job fails its execution.

**setJobFailedMessage(String)** - Method in class `org.quartz.impl.triggers.AbstractTrigger`

Set the JobFactory to be associated with the Trigger.

**setJobGroup(String)** - Method in class `org.quartz.impl.jdbcjobstore.FiredTriggerRecord`

**setJobGroup(String)** - Method in class `org.quartz.impl.jdbcjobstore.TriggerStatus`

**setJobGroup(String)** - Method in class `org.quartz.impl.triggers.AbstractTrigger`

**setJobGroup(String)** - Method in class `org.quartz.locality.DelegatingLocalityTrigger`

**setJobListenerMatchers(String, List<Matcher<JobKey>>>)** - Method in class `org.quartz.core.ListenerManagerImpl`

**setJobListenerMatchers(String, List<Matcher<JobKey>>>)** - Method in class `org.quartz.impl.triggers.AbstractTrigger`

**setJobListenerMatchers(String, List<Matcher<JobKey>>>)** - Method in class `org.quartz.locality.DelegatingLocalityTrigger`

**setJobListenerMatchers(String, List<Matcher<JobKey>>>)** - Method in class `org.quartz.impl.jdbcjobstore.FiredTriggerRecord`
interface org.quartz.ListenerManager
    Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

setJobName(String) - Method in class org.quartz.impl.triggers.AbstractTrigger
    Set the name of the associated JobDetail.

setJobRequestsRecovery(boolean) - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord
    Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

setJobRunShellFactory(JobRunShellFactory) - Method in class org.quartz.core.QuartzSchedulerResources
    Set the JobRunShellFactory for the QuartzScheduler to use.

setJobRunTime(long) - Method in class org.quartz.impl.JobExecutionContextImpl
    Set the time at which the Job should be run.

setJobStore(JobStore) - Method in class org.quartz.core.QuartzSchedulerResources
    Set the JobStore for the QuartzScheduler to use.

setJobSuccessMessage(String) - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin
    Set the message that is logged when a Job successfully completes its execution.

setJobToBeFiredMessage(String) - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin
    Set the message that is logged when a Job is about to execute.

setJobWasVetoedMessage(String) - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin
    Set the message that is logged when a Job execution is vetoed by a trigger listener.

setKey(TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.TriggerStatus

setKey(JobKey) - Method in class org.quartz.impl.JobDetailImpl

setKey(TriggerKey) - Method in class org.quartz.impl.triggers.AbstractTrigger

setKey(TriggerKey) - Method in class org.quartz.locality.DelegatingLocalityTrigger

setLockHandler(Semaphore) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
**setLockOnInsert(boolean)** - Method in class org.quartz.impl.jdbcjobstore.[JobStoreSupport](#)
   Whether or not to obtain locks when inserting new jobs/triggers.

**setLong1(long)** - Method in class org.quartz.impl.jdbcjobstore.[SimplePropertiesTriggerProperties](#)

**setLong2(long)** - Method in class org.quartz.impl.jdbcjobstore.[SimplePropertiesTriggerProperties](#)

**setMakeSchedulerThreadDaemon(boolean)** - Method in class org.quartz.core.[QuartzSchedulerResources](#)
   Set whether to mark the Quartz scheduling thread as daemon.

**setMakeThreadsDaemons(boolean)** - Method in class org.quartz.impl.jdbcjobstore.[JobStoreSupport](#)
   Set whether the threads spawned by this JobStore should be marked as daemon.

**setMakeThreadsDaemons(boolean)** - Method in class org.quartz.simpl.[SimpleThreadPool](#)

**setMaxBatchSize(int)** - Method in class org.quartz.core.[QuartzSchedulerResources](#)

**setMaxMisfiresToHandleAtATime(int)** - Method in class org.quartz.impl.jdbcjobstore.[JobStoreSupport](#)
   Set the maximum number of misfired triggers that the misfire handling thread will try to recover at one time (within one transaction).

**setMessage(String)** - Method in class org.quartz.jobs.ee.mail.[SendMailJob.MailInfo](#)

**setMimeMessageContent(MimeMessage, SendMailJob.MailInfo)** - Method in class org.quartz.jobs.ee.mail.[SendMailJob](#)

**setMisfireInstruction(int)** - Method in class org.quartz.impl.triggers.[AbstractTrigger](#)
   Set the instruction the Scheduler should be given for handling misfire situations for this Trigger- the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be passed to this method.
setMisfireInstruction(int) - Method in class
org.quartz.locality.DelegatingLocalityTrigger
setMisfireThreshold(long) - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
  The number of milliseconds by which a trigger must have missed its
  next-fire-time, in order for it to be considered "misfired" and thus have its
  misfire instruction applied.
setMisfireThreshold(long) - Method in class org.quartz.simpl.RAMJobStore
  The number of milliseconds by which a trigger must have missed its next-
  fire-time, in order for it to be considered "misfired" and thus have its
  misfire instruction applied.
setName(String) - Method in class org.quartz.core.QuartzSchedulerResources
  Set the name for the QuartzScheduler.
setName(String) - Method in class org.quartz.impl.JobDetailImpl
  Set the name of this Job.
setName(String) - Method in class org.quartz.impl.triggers.AbstractTrigger
  Set the name of this Trigger.
setNextFireTime(Date) - Method in class
org.quartz.impl.triggers.CalendarIntervalTriggerImpl
  Set the next time at which the DateIntervalTrigger should fire.
setNextFireTime(Date) - Method in class
org.quartz.impl.triggers.CronTriggerImpl
  Sets the next time at which the CronTrigger will fire.
setNextFireTime(Date) - Method in class
org.quartz.impl.triggers.SimpleTriggerImpl
  Set the next time at which the SimpleTrigger should fire.
setNextFireTime(Date) - Method in class
org.quartz.locality.DelegatingLocalityTrigger
setNonManagedTXDataSource(String) - Method in class
org.quartz.impl.jdbcjobstore.JobStoreCMT
  Set the name of the DataSource that should be used for performing
database functions.
setNumeratorValue(long) - Method in interface
org.quartz.utils.counter.sampled.SampledRateCounter
  Sets the value of the numerator to the passed value
setNumeratorValue(long) - Method in class
org.quartz.utils.counter.sampled.SampledRateCounterImpl
  Sets the value of the numerator to the passed value
setOverWriteExistingData(boolean) - Method in class
Whether the existing scheduling data (with same identifiers) will be overwritten.

**setPreviousFireTime(Date)** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**

Set the previous time at which the DateIntervalTrigger fired.

**setPreviousFireTime(Date)** - Method in class org.quartz.impl.triggers.**CronTriggerImpl**

Set the previous time at which the CronTrigger fired.

**setPreviousFireTime(Date)** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**

Set the previous time at which the SimpleTrigger fired.

**setPreviousFireTime(Date)** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

**setPriority(int)** - Method in class org.quartz.impl.jdbcjobstore.**FiredTriggerRecord**

The priority of a Trigger acts as a tie breaker such that if two Triggers have the same scheduled fire time, then Quartz will do its best to give the one with the higher priority first access to a worker thread.

**setPriority(int)** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

**setProperties(String)** - Method in class org.quartz.ee.jmx.jboss.**QuartzService**

**setProperties(String)** - Method in interface org.quartz.ee.jmx.jboss.**QuartzServiceMBean**

**setPropertiesFile(String)** - Method in class org.quartz.ee.jmx.jboss.**QuartzService**

**setPropertiesFile(String)** - Method in interface org.quartz.ee.jmx.jboss.**QuartzServiceMBean**

**setProviderURL(String)** - Method in class org.quartz.ee.jmx.jboss.**JBoss4RMIRemoteMBeanScheduler**

Set the remote MBean server URL.

**setRefireImmediately(boolean)** - Method in exception org.quartz.**JobExecutionException**
**setRepeatCount(int)** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**
Set the number of times the **SimpleTrigger** should repeat, after which it will be automatically deleted.

**setRepeatInterval(int)** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**
Set the time interval that will be added to the **DateIntervalTrigger**'s fire time (in the set repeat interval unit) in order to calculate the time of the next trigger repeat.

**setRepeatInterval(long)** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**
Set the time interval (in milliseconds) at which the **SimpleTrigger** should repeat.

**setRepeatIntervalUnit(DateBuilder.IntervalUnit)** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**
Set the interval unit - the time unit on with the interval applies.

**setReplyTo(String)** - Method in class org.quartz.jobs.ee.mail.**SendMailJob.MailInfo**

**setRequestsRecovery(boolean)** - Method in class org.quartz.impl.**JobDetailImpl**
Set whether or not the **Scheduler** should re-execute the **Job** if a 'recovery' or 'fail-over' situation is encountered.

**setResult(Object)** - Method in class org.quartz.impl.**JobExecutionContextImpl**
Set the result (if any) of the **Job**'s execution (the type of object set as the result is entirely up to the particular job).

**setRMI callable methods**

**setRMIBindName(String)** - Method in class org.quartz.core.**QuartzSchedulerResources**
Set the name under which to bind the QuartzScheduler in RMI.

**setRMICreateRegistryStrategy(String)** - Method in class org.quartz.core.**QuartzSchedulerResources**
Set whether or not Quartz should create an RMI Registry, and if so, how.

**setRMIRegistryHost(String)** - Method in class org.quartz.core.**QuartzSchedulerResources**
Set the host name of the RMI Registry that the scheduler should export itself to.

**setRMIRegistryPort(int)** - Method in class
org.quartz.core.**QuartzSchedulerResources**

Set the port number of the RMI Registry that the scheduler should export itself to.

**setRMIServerPort(int)** - Method in class

org.quartz.core.**QuartzSchedulerResources**

Set the port number the scheduler server will be bound to.

**setRunUpdateCheck(boolean)** - Method in class

org.quartz.core.**QuartzSchedulerResources**

**setSampledStatisticsEnabled(boolean)** - Method in interface

org.quartz.core.jmx.**QuartzSchedulerMBean**

**setSampledStatisticsEnabled(boolean)** - Method in class

org.quartz.core.**QuartzSchedulerMBeanImpl**

**setScanInterval(long)** - Method in class

org.quartz.plugins.xml.**XMLSchedulingDataProcessorPlugin**

The interval (in seconds) at which to scan for changes to the file.

**setSchedName(String)** - Method in class

org.quartz.impl.jdbcjobstore.**DBSemaphore**

**setSchedName(String)** - Method in interface

org.quartz.impl.jdbcjobstore.**TablePrefixAware**

**setSchedulerInstanceId(String)** - Method in class

org.quartz.impl.jdbcjobstore.**FiredTriggerRecord**

**setSchedulerInstanceId(String)** - Method in class

org.quartz.impl.jdbcjobstore.**SchedulerStateRecord**

**setSchedulerObjectName(String)** - Method in class

org.quartz.impl.**RemoteMBeanScheduler**

Set the name under which the Scheduler MBean is registered on the remote MBean server.

**setSchedulerObjectName(ObjectName)** - Method in class

org.quartz.impl.**RemoteMBeanScheduler**

Set the name under which the Scheduler MBean is registered on the remote MBean server.

**setSelectWithLockSQL(String)** - Method in class
set the SQL statement to use to select and lock a row in the "locks" table.

**setSelectWithLockSQL(String)** - Method in class org.quartz.impl.jdbcjobstore.StdRowLockSemaphore

**setSignalOnSchedulingChange(boolean)** - Method in class org.quartz.core.QuartzScheduler

**setSmtpHost(String)** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**setSQL(String)** - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore

**setStartScheduler(boolean)** - Method in class org.quartz.ee.jmx.jboss.QuartzService

**setStartScheduler(boolean)** - Method in interface org.quartz.ee.jmx.jboss.QuartzServiceMBean

**setStartTime(Date)** - Method in class org.quartz.impl.triggers.AbstractTrigger

The time at which the trigger's scheduling should start.

**setStartTime(Date)** - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl

Set the time at which the DateIntervalTrigger should occur.

**setStartTime(Date)** - Method in class org.quartz.impl.triggers.CronTriggerImpl

**setStartTime(Date)** - Method in class org.quartz.locality.DelegatingLocalityTrigger

**setString1(String)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setString2(String)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setString3(String)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setString2(String)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

**setString3(String)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties
**setSubject(String)** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**setTablePrefix(String)** - Method in class org.quartz.impl.jdbcjobstore.DBSemaphore

**setTablePrefix(String)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

  Set the prefix that should be pre-pended to all table names.

**setTablePrefix(String)** - Method in interface org.quartz.impl.jdbcjobstore.TablePrefixAware

**setThreadCount(int)** - Method in class org.quartz.simpl.SimpleThreadPool

  Set the number of worker threads in the pool - has no effect after initialize() has been called.

**setThreadName(String)** - Method in class org.quartz.core.QuartzSchedulerResources

  Set the name for the QuartzSchedulerThread.

**setThreadNamePrefix(String)** - Method in class org.quartz.simpl.SimpleThreadPool

**setThreadPool(ThreadPool)** - Method in class org.quartz.core.QuartzSchedulerResources

  Set the ThreadPool for the QuartzScheduler to use.

**setThreadPoolSize(int)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**setThreadPoolSize(int)** - Method in class org.quartz.simpl.RAMJobStore

**setThreadPriority(int)** - Method in class org.quartz.simpl.SimpleThreadPool

  Set the thread priority of worker threads in the pool - has no effect after initialize() has been called.

**setThreadsInheritContextClassLoaderOfInitializingThread(boolean)** - Method in class org.quartz.simpl.SimpleThreadPool

**setThreadsInheritGroupOfInitializingThread(boolean)** - Method in class org.quartz.simpl.SimpleThreadPool
setThreadsInheritInitializersClassLoadContext(boolean) - Method in class org.quartz.core.QuartzSchedulerResources
   Set whether to set the class load context of spawned threads to that of the initializing thread.

setThreadsInheritInitializersClassLoadContext(boolean) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
   Set whether to set the class load context of spawned threads to that of the initializing thread.

setThrowIfPropertyNotFound(boolean) - Method in class org.quartz.simpl.PropertySettingJobFactory
   Whether the JobInstantiation should fail and throw an exception if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

setTimeRange(String, String) - Method in class org.quartz.impl.calendar.DailyCalendar
   Sets the time range for the DailyCalendar to the times represented in the specified Strings.

setTimeRange(int, int, int, int, int, int, int, int) - Method in class org.quartz.impl.calendar.DailyCalendar
   Sets the time range for the DailyCalendar to the times represented in the specified values.

setTimeRange(Calendar, Calendar) - Method in class org.quartz.impl.calendar.DailyCalendar
   Sets the time range for the DailyCalendar to the times represented in the specified java.util.Calendar objects.

setTimesTriggered(int) - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl
   Set the number of times the DateIntervalTrigger has already fired.

setTimesTriggered(int) - Method in class org.quartz.impl.triggers.SimpleTriggerImpl
   Set the number of times the SimpleTrigger has already fired.

setTimeZone(TimeZone) - Method in class org.quartz.CronExpression
   Sets the timezone for which this CronExpression will be resolved.

setTimeZone(TimeZone) - Method in class org.quartz.impl.calendar.BaseCalendar
Sets the time zone for which this Calendar will be resolved.

**setTimeZone(TimeZone)** - Method in class org.quartz.impl.calendar.CronCalendar

Sets the time zone for which the CronExpression of this CronCalendar will be resolved.

**setTimeZone(TimeZone)** - Method in class org.quartz.impl.triggers.CronTriggerImpl

Sets the time zone for which the cronExpression of this CronTrigger will be resolved.

**setTo(String)** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**setTransactionIsolation(int)** - Method in class org.quartz.impl.jdbcjobstore.AttributeRestoringConnectionInvocationHandler

Attempts to change the transaction isolation level to the given level, saving the original level.

**setTransactionManagerJNDIName(String)** - Method in class org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore

**setTriggerCompleteMessage(String)** - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin

Set the message that is printed upon the completion of a trigger's firing.

**setTriggerFiredMessage(String)** - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin

Set the message that is printed upon a trigger's firing.

**setTriggerKey(TriggerKey)** - Method in class org.quartz.impl.jdbcjobstore.FiredTriggerRecord

**setTriggerListenerMatchers(String, List<Matcher<TriggerKey>>)** - Method in interface org.quartz.ListenerManager

Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

**setTriggerMisfiredMessage(String)** - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin

Set the message that is printed upon a trigger's firing.

**setTxIsolationLevelReadCommitted** - Variable in class org.quartz.impl.jdbcjobstore.JobStoreCMT
setTxIsolationLevelReadCommitted(boolean) - Method in class
org.quartz.impl.jdbcjobstore.JobStoreCMT
   Set the transaction isolation level of DB connections to sequential.
setTxIsolationLevelSerializable(boolean) - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
   Set the transaction isolation level of DB connections to sequential.
setUnscheduleAllTriggers(boolean) - Method in exception
org.quartz.JobExecutionException

setUnscheduleFiringTrigger(boolean) - Method in exception
org.quartz.JobExecutionException

setUpdateLockRowSQL(String) - Method in class
org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore

setUseDBLocks(boolean) - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
   Set whether this instance should use database-based thread synchronization.
setUseProperties(String) - Method in class
org.quartz.impl.jdbcjobstore.JobStoreSupport
   Set whether String-only properties will be handled in JobDataMaps.
setUserTxLocation(String) - Static method in class
org.quartz.ee.jta.UserTransactionHelper
   Set the JNDI URL at which the Application Server's UserTransaction can be found.
setValue(long) - Method in interface org.quartz.utils.counter.Counter
   Sets the value of the counter to the supplied value
setValue(long) - Method in class org.quartz.utils.counter.CounterImpl
   Sets the value of the counter to the supplied value
setValue(long, long) - Method in interface
org.quartz.utils.counter.sampled.SampledRateCounter
   Sets the values of the numerator and denominator to the passed values
setValue(long, long) - Method in class
org.quartz.utils.counter.sampled.SampledRateCounterImpl
   Sets the values of the numerator and denominator to the passed values
setValue(long) - Method in class
org.quartz.utils.counter.sampled.SampledRateCounterImpl
   throws UnsupportedOperationException
**setWarnIfPropertyNotFound(boolean)** - Method in class `org.quartz.simpl.PropertySettingJobFactory`
Whether a warning should be logged if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

**setWrapInUserTransaction(boolean)** - Method in class `org.quartz.plugins.SchedulerPluginWithUserTransactionSupport`
Wrap the start() and shutdown() methods in a UserTransaction.

**shutdown()** - Method in interface `org.quartz.core.jmx.QuartzSchedulerMBean`

**shutdown()** - Method in class `org.quartz.core.NullSampledStatisticsImpl`
Halts the QuartzScheduler's firing of Triggers, and cleans up all resources associated with the QuartzScheduler.

**shutdown(boolean)** - Method in class `org.quartz.core.QuartzScheduler`
Halts the QuartzScheduler's firing of Triggers, and cleans up all resources associated with the QuartzScheduler.

**shutdown()** - Method in class `org.quartz.core.QuartzSchedulerMBeanImpl`

**shutdown()** - Method in interface `org.quartz.core.RemotableQuartzScheduler`

**shutdown(boolean)** - Method in interface `org.quartz.core.RemotableQuartzScheduler`

**shutdown()** - Method in interface `org.quartz.core.SampledStatistics`

**shutdown()** - Method in class `org.quartz.core.SampledStatisticsImpl`

**shutdown()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreCMT`
Called by the QuartzScheduler to inform the JobStore that it should free up all of it's resources because the scheduler is shutting down.

**shutdown()** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
Calls the equivalent method on the 'proxied' QuartzScheduler.

**shutdown(boolean)** - Method in class `org.quartz.impl.jdbcjobstore.RemoteMBeanScheduler`
Calls the equivalent method on the 'proxied' QuartzScheduler.

**shutdown()** - Method in class `org.quartz.impl.jdbcjobstore.RemoteMBeanScheduler`
Calls the equivalent method on the 'proxied' QuartzScheduler.
**shutdown(boolean)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**shutdown()** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**shutdown(boolean)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**shutdown()** - Method in class org.quartz.plugins.history.LoggingJobHistoryPlugin
Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.

**shutdown()** - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin
Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.

**shutdown()** - Method in class org.quartz.plugins.management.ShutdownHookPlugin
Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.

**shutdown(UserTransaction)** - Method in class org.quartz.plugins.SchedulerPluginWithUserTransactionSupport
Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.

**shutdown()** - Method in class org.quartz.plugins.SchedulerPluginWithUserTransactionSupport
Based on the value of wrapInUserTransaction, wraps the call to shutdown(UserTransaction) in a UserTransaction.

**shutdown()** - Method in class org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin
Overriden to ignore wrapInUserTransaction because shutdown() does not interact with the Scheduler.

**shutdown()** - Method in interface org.quartz.Scheduler
Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler.

**shutdown(boolean)** - Method in interface org.quartz.Scheduler
Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler.

**shutdown()** - Method in class org.quartz.simpl.RAMJobStore
Called by the QuartzScheduler to inform the JobStore that it should free up all of its resources because the scheduler is shutting down.
**shutdown()** - Method in class org.quartz.simpl.*SimpleThreadPool*
Terminate any worker threads in this thread group.

**shutdown(boolean)** - Method in class org.quartz.simpl.*SimpleThreadPool*
Terminate any worker threads in this thread group.

**shutdown()** - Method in class org.quartz.simpl.*SimpleTimeBroker*

**shutdown()** - Method in class org.quartz.simpl.*ZeroSizeThreadPool*

**shutdown(boolean)** - Method in class org.quartz.simpl.*ZeroSizeThreadPool*

**shutdown()** - Method in interface org.quartz.utils.*ConnectionProvider*

**shutdown(boolean)** - Method in interface org.quartz.utils.counter.*CounterManager*
Shuts down this counter manager

**shutdown(boolean)** - Method in class org.quartz.utils.counter.*CounterManagerImpl*
Shuts down this counter manager

**shutdown()** - Method in interface org.quartz.utils.counter.sampled.*SampledCounter*
Shutdown this counter

**shutdown()** - Method in class org.quartz.utils.counter.sampled.*SampledCounterImpl*
Shutdown this counter

**shutdown(String)** - Method in class org.quartz.utils.*DBConnectionManager*
Shuts down database connections from the DataSource with the given name, if applicable for the underlying provider.

**shutdown()** - Method in class org.quartz.utils.*JNDIConnectionProvider*

**shutdown()** - Method in class org.quartz.utils.*PoolingConnectionProvider*

**shutdown()** - Method in class org.quartz.utils.weblogic.*WeblogicConnectionProvider*

**shutdownCounter(Counter)** - Method in interface org.quartz.utils.counter.*CounterManager*
Shuts down the counter

**shutdownCounter(Counter)** - Method in class org.quartz.utils.counter.*CounterManagerImpl*
Shuts down the counter

**ShutdownHookPlugin** - Class in `org.quartz.plugins.management`

This plugin catches the event of the JVM terminating (such as upon a CRTL-C) and tells the scheduler to shutdown.

**ShutdownHookPlugin()** - Constructor for class

`org.quartz.plugins.management.ShutdownHookPlugin`

**shutdownRequested** - Variable in class `org.quartz.core.JobRunShell`

**sigChangeForTxCompletion** - Variable in class

`org.quartz.impl.jdbcjobstore.JobStoreSupport`

**signaler** - Variable in class `org.quartz.simpl.RAMJobStore`

**signalSchedulingChange(long)** - Method in class

`org.quartz.core.QuartzSchedulerThread`

Signals the main processing loop that a change in scheduling has been made - in order to interrupt any sleeping that may be occurring while waiting for the fire time to arrive.

**signalSchedulingChange(long)** - Method in class

`org.quartz.core.SchedulerSignalerImpl`

**signalSchedulingChangeImmediately(long)** - Method in class

`org.quartz.impl.jdbcjobstore.JobStoreSupport`

**signalSchedulingChangeOnTxCompletion(long)** - Method in class

`org.quartz.impl.jdbcjobstore.JobStoreSupport`

**SimpleClassLoadHelper** - Class in `org.quartz.simpl`

A ClassLoadHelper that simply calls `Class.forName()`.

**SimpleClassLoadHelper()** - Constructor for class

`org.quartz.simpl.SimpleClassLoadHelper`

**SimpleInstanceIdGenerator** - Class in `org.quartz.simpl`

The default InstanceIdGenerator used by Quartz when instance id is to be automatically generated.

**SimpleInstanceIdGenerator()** - Constructor for class

`org.quartz.simpl.SimpleInstanceIdGenerator`
**SimpleJobFactory** - Class in `org.quartz.simpl`  
The default JobFactory used by Quartz - simply calls `newInstance()` on the job class.

**SimpleJobFactory()** - Constructor for class `org.quartz.simpl.SimpleJobFactory`

**SimplePropertiesTriggerPersistenceDelegateSupport** - Class in `org.quartz.impl.jdbcjobstore`  
A base implementation of `TriggerPersistenceDelegate` that persists trigger fields in the "QRTZ_SIMPROP_TRIGGERS" table.

**SimplePropertiesTriggerPersistenceDelegateSupport()** - Constructor for class `org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport`

**SimplePropertiesTriggerProperties** - Class in `org.quartz.impl.jdbcjobstore`  

**SimplePropertiesTriggerProperties()** - Constructor for class `org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties`

**simpleSchedule()** - Static method in class `org.quartz.SimpleScheduleBuilder`  
Create a `SimpleScheduleBuilder`.

**SimpleScheduleBuilder** - Class in `org.quartz`  
`SimpleScheduleBuilder` is a `ScheduleBuilder` that defines strict/literal interval-based schedules for Triggers.

**SimpleSemaphore** - Class in `org.quartz.impl.jdbcjobstore`  
Internal in-memory lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.

**SimpleSemaphore()** - Constructor for class `org.quartz.impl.jdbcjobstore.SimpleSemaphore`

**SimpleThreadPool** - Class in `org.quartz.simpl`  
This is class is a simple implementation of a thread pool, based on the `ThreadPool` interface.

**SimpleThreadPool()** - Constructor for class `org.quartz.simpl.SimpleThreadPool`  
Create a new (unconfigured) `SimpleThreadPool`.

**SimpleThreadPool(int, int)** - Constructor for class `org.quartz.simpl.SimpleThreadPool`  
Create a new `SimpleThreadPool` with the specified number of `Threads` that have the given priority.

**SimpleTimeBroker** - Class in `org.quartz.simpl`
The interface to be implemented by classes that want to provide a mechanism by which the QuartzScheduler can reliably determine the current time.

**SimpleTimeBroker()** - Constructor for class org.quartz.simpl. SimpleTimeBroker

**SimpleTrigger** - Interface in org.quartz
A Trigger that is used to fire a Job at a given moment in time, and optionally repeated at a specified interval.

**SimpleTriggerImpl** - Class in org.quartz.impl.triggers
A concrete Trigger that is used to fire a JobDetail at a given moment in time, and optionally repeated at a specified interval.

**SimpleTriggerImpl()** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Create a SimpleTrigger with no settings.

**SimpleTriggerImpl(String)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead

**SimpleTriggerImpl(String, String)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead

**SimpleTriggerImpl(String, int, long)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead

**SimpleTriggerImpl(String, String, String)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead

**SimpleTriggerImpl(String, String, int, long)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead

**SimpleTriggerImpl(String, String, Date)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead

**SimpleTriggerImpl(String, String, Date, int, long)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead

**SimpleTriggerImpl(String, String, Date, Date, int, long)** - Constructor for class org.quartz.impl.triggers. SimpleTriggerImpl
Deprecated. use a TriggerBuilder instead
**SimpleTriggerImpl(String, String, String, String, Date, Date, int, long)** - Constructor for class org.quartz.impl.triggers.**SimpleTriggerImpl**

*Deprecated.* use a TriggerBuilder instead

**SimpleTriggerPersistenceDelegate** - Class in org.quartz.impl.jdbcjobstore.**SimpleTriggerPersistenceDelegate**

**SimpleTriggerSupport** - Class in org.quartz.core.jmx.**SimpleTriggerSupport**

**size()** - Method in class org.quartz.utils.**DirtyFlagMap**

**skipWhiteSpace(int, String)** - Method in class org.quartz.**CronExpression**

**SOLARIS** - Static variable in class org.quartz.locality.constraint.**OsConstraint**

**standby()** - Method in interface org.quartz.core.jmx.**QuartzSchedulerMBean**

**standby()** - Method in class org.quartz.core.**QuartzScheduler**

Temporarily halts the QuartzScheduler's firing of Triggers.

**standby()** - Method in class org.quartz.core.**QuartzSchedulerImpl**

**standby()** - Method in interface org.quartz.core.**RemotableQuartzScheduler**

**standby()** - Method in class org.quartz.impl.**RemoteMBeanScheduler**

Calls the equivalent method on the 'proxied' QuartzScheduler.

**standby()** - Method in class org.quartz.impl.**RemoteScheduler**

Calls the equivalent method on the 'proxied' QuartzScheduler.

**standby()** - Method in class org.quartz.impl.**StdScheduler**

Calls the equivalent method on the 'proxied' QuartzScheduler.

**standby()** - Method in interface org.quartz.**Scheduler**

Temporarily halts the Scheduler's firing of Triggers.

**start()** - Method in interface org.quartz.core.jmx.**QuartzSchedulerMBean**

**start()** - Method in class org.quartz.core.**QuartzScheduler**

Starts the QuartzScheduler's threads that fire Triggers.
**start**() - Method in class org.quartz.core.**QuartzSchedulerMBeanImpl**

**start**() - Method in interface org.quartz.core.**RemotableQuartzScheduler**

**start**() - Method in class org.quartz.impl.**RemoteMBeanScheduler**
   Calls the equivalent method on the 'proxied' QuartzScheduler.

**start**() - Method in class org.quartz.impl.**RemoteScheduler**
   Calls the equivalent method on the 'proxied' QuartzScheduler.

**start**() - Method in class org.quartz.impl.**StdScheduler**
   Calls the equivalent method on the 'proxied' QuartzScheduler.

**start**() - Method in class org.quartz.plugins.history.**LoggingJobHistoryPlugin**

**start**() - Method in class org.quartz.plugins.history.**LoggingTriggerHistoryPlugin**

**start**() - Method in class org.quartz.plugins.management.**ShutdownHookPlugin**

**start(UserTransaction)** - Method in class org.quartz.plugins.**SchedulerPluginWithUserTransactionSupport**
   Called when the associated Scheduler is started, in order to let the plug-in know it can now make calls into the scheduler if it needs to.

**start**() - Method in class org.quartz.plugins.**SchedulerPluginWithUserTransactionSupport**
   Based on the value of **wrapInUserTransaction**, wraps the call to **start(UserTransaction)** in a UserTransaction.

**start(UserTransaction)** - Method in class org.quartz.plugins.xml.**XMLSchedulingDataProcessorPlugin**

**start**() - Method in interface org.quartz.**Scheduler**
   Starts the Scheduler's threads that fire Triggers.

**startAt(Date)** - Method in class org.quartz.**TriggerBuilder**
   Set the time the Trigger should start at - the trigger may or may not fire at this time - depending upon the schedule configured for the Trigger.

**startDelayed(int)** - Method in class org.quartz.core.**QuartzScheduler**

**startDelayed(int)** - Method in interface org.quartz.core.**RemotableQuartzScheduler**

**startDelayed(int)** - Method in class org.quartz.impl.**RemoteMBeanScheduler**
Calls the equivalent method on the 'proxied' QuartzScheduler.

**startDelayed(int)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**startDelayed(int)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**startDelayed(int)** - Method in interface org.quartz.Scheduler
Calls `{#start()}` after the indicated number of seconds.

**startMonitoring()** - Method in class
org.quartz.locality.constraint.evaluator.EvaluatorTimerTask
Starts the monitoring

**startNow()** - Method in class org.quartz.TriggerBuilder
Set the time the Trigger should start at to the current moment - the trigger may or may not fire at this time - depending upon the schedule configured for the Trigger.

**startService()** - Method in class org.quartz.ee.jmx.jboss.QuartzService

**STATE_ACQUIRED** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**STATE_BLOCKED** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**STATE_COMPLETE** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**STATE_DELETED** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**STATE_ERROR** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**STATE_EXECUTING** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**STATE_MISFIRED** - Static variable in interface
org.quartz.impl.jdbcjobstore.Constants

**Deprecation**: Whether a trigger has misfired is no longer a state, but rather now identified dynamically by whether the trigger's next fire time is more than the misfire threshold time in the past.
**STATE_PAUSED** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{Constants}

**STATE_PAUSED_BLOCKED** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{Constants}

**STATE_WAITING** - Static variable in interface org.quartz.impl.jdbcjobstore.\texttt{Constants}

**StatefulJob** - Interface in \texttt{org.quartz}
   
   \texttt{Deprecated. use DisallowConcurrentExecution and/or PersistJobDataAfterExecution annotations instead.}

**StdJDBCConstants** - Interface in org.quartz.impl.jdbcjobstore
   
   This interface extends \texttt{Constants} to include the query string constants in use by the \texttt{StdJDBCDelegate} class.

**StdJDBCDelegate** - Class in org.quartz.impl.jdbcjobstore
   
   This is meant to be an abstract base class for most, if not all, \texttt{DriverDelegate} implementations.

**StdJDBCDelegate(Logger, String, String, String, ClassLoadHelper)** - Constructor for class org.quartz.impl.jdbcjobstore.\texttt{StdJDBCDelegate}
   
   Create new \texttt{StdJDBCDelegate} instance.

**StdJDBCDelegate(Logger, String, String, String, String, ClassLoadHelper, Boolean)** - Constructor for class org.quartz.impl.jdbcjobstore.\texttt{StdJDBCDelegate}
   
   Create new \texttt{StdJDBCDelegate} instance.

**StdJobRunShellFactory** - Class in org.quartz.impl
   
   Responsible for creating the instances of \texttt{JobRunShell} to be used within the \texttt{QuartzScheduler} instance.

**StdJobRunShellFactory()** - Constructor for class org.quartz.impl.\texttt{StdJobRunShellFactory}

**StdRowLockSemaphore** - Class in org.quartz.impl.jdbcjobstore
   
   Internal database based lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.

**StdRowLockSemaphore(String, String, String)** - Constructor for class org.quartz.impl.jdbcjobstore.\texttt{StdRowLockSemaphore}

**StdScheduler** - Class in org.quartz.impl
   
   An implementation of the \texttt{Scheduler} interface that directly proxies all
method calls to the equivalent call on a given QuartzScheduler instance.

**StdScheduler(QuartzScheduler)** - Constructor for class org.quartz.impl.StdScheduler

Construct a StdScheduler instance to proxy the given QuartzScheduler instance, and with the given SchedulingContext.

**StdSchedulerFactory** - Class in org.quartz.impl

An implementation of SchedulerFactory that does all of its work of creating a QuartzScheduler instance based on the contents of a Properties file.

**StdSchedulerFactory()** - Constructor for class org.quartz.impl.StdSchedulerFactory

Create an uninitialized StdSchedulerFactory.

**StdSchedulerFactory(Properties)** - Constructor for class org.quartz.impl.StdSchedulerFactory

Create a StdSchedulerFactory that has been initialized via StdSchedulerFactory.initialize(Properties).

**StdSchedulerFactory(String)** - Constructor for class org.quartz.impl.StdSchedulerFactory

Create a StdSchedulerFactory that has been initialized via StdSchedulerFactory.initialize(String).

**stopService()** - Method in class org.quartz.ee.jmx.jboss.QuartzService

**storeCalendar(String, Calendar, boolean, boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

Store the given Calendar.

**storeCalendar(Connection, String, Calendar, boolean, boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**storeCalendar(String, Calendar, boolean, boolean)** - Method in class org.quartz.simpl.RAMJobStore

Store the given Calendar.

**storeDurably()** - Method in class org.quartz.JobBuilder

Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

**storeDurably(boolean)** - Method in class org.quartz.JobBuilder

Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

**storeExpressionVals(int, String, int)** - Method in class org.quartz.CronExpression
**storeJob(JobDetail, boolean)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
- Store the given JobDetail.

**storeJob(Connection, JobDetail, boolean)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
- Insert or update a job.

**storeJob(JobDetail, boolean)** - Method in class `org.quartz.simpl.RAMJobStore`
- Store the given Job.

**storeJobAndTrigger(JobDetail, OperableTrigger)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
- Store the given JobDetail and Trigger.

**storeJobAndTrigger(JobDetail, OperableTrigger)** - Method in class `org.quartz.simpl.RAMJobStore`
- Store the given JobDetail and Trigger.

**storeJobsAndTriggers(Map<JobDetail, List<Trigger>>, boolean)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`

**storeJobsAndTriggers(Map<JobDetail, List<Trigger>>, boolean)** - Method in class `org.quartz.simpl.RAMJobStore`

**storeTrigger(OperableTrigger, boolean)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
- Store the given Trigger.

**storeTrigger(Connection, OperableTrigger, JobDetail, boolean, String, boolean, boolean)** - Method in class `org.quartz.impl.jdbcjobstore.JobStoreSupport`
- Insert or update a trigger.

**storeTrigger(OperableTrigger, boolean)** - Method in class `org.quartz.simpl.RAMJobStore`
- Store the given Trigger.

**StringKeyDirtyFlagMap** - Class in `org.quartz.utils`
- An implementation of Map that wraps another Map and flags itself 'dirty' when it is modified, enforces that all keys are Strings.

**StringKeyDirtyFlagMap()** - Constructor for class `org.quartz.utils.StringKeyDirtyFlagMap`

**StringKeyDirtyFlagMap(int)** - Constructor for class `org.quartz.utils.StringKeyDirtyFlagMap`
**StringKeyDirtyFlagMap(int, float)** - Constructor for class org.quartz.util.StringKeyDirtyFlagMap

**StringMatcher<T extends Key>** - Class in org.quartz.impl.matchers
   An abstract base class for some types of matchers.

**StringMatcher(String, StringMatcher.StringOperatorName)** - Constructor for class org.quartz.impl.matchers.StringMatcher

**StringMatcher.StringOperatorName** - Enum in org.quartz.impl.matchers

**suitedNodes(CpuConstraint)** - Method in class org.quartz.locality.constraint.evaluator.CpuEvaluator

**suitedNodes(T)** - Method in interface org.quartz.locality.constraint.evaluator.Evaluator
   Returns a list of best suited node (instanceId) based on the constraint

**suitedNodes(MemoryConstraint)** - Method in class org.quartz.locality.constraint.evaluator.MemoryEvaluator
   Returns a list of best suited node (instanceId) based on the constraint

**suitedNodes(NodeGroupConstraint)** - Method in class org.quartz.locality.constraint.evaluator.NodeGroupEvaluator
   Returns a list of best suited node (instanceId) based on the constraint

**suitedNodes(OsConstraint)** - Method in class org.terracotta.modules.ehcache.store.EhcacheEvaluator
   Returns a list of best suited node (instanceId) based on the constraint

**suitedNodes(EhcacheConstraint)** - Method in class org.terracotta.modules.ehcache.store.EhcacheEvaluator
   Returns a list of best suited node (instanceId) based on the constraint

**SUNDAY** - Static variable in class org.quartz.DateBuilder

**supportsPersistence()** - Method in class org.quartz.core.QuartzScheduler

**supportsPersistence()** - Method in interface org.quartz.core.RemovableQuartzScheduler

**supportsPersistence()** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
supportsPersistence() - Method in class org.quartz.simpl.RAMJobStore

SybaseDelegate - Class in org.quartz.impl.jdbcjobstore
This is a driver delegate for the Sybase database.
SybaseDelegate(Logger, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore. SybaseDelegate
Create new SybaseDelegate instance.
SybaseDelegate(Logger, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore. SybaseDelegate

SYSTEM_PROPERTY - Static variable in class org.quartz.simpl.SystemPropertyInstanceIdGenerator
System property to read the instanceId from
SYSTEM_PROPERTY AS_INSTANCE_ID - Static variable in class org.quartz.impl.StdSchedulerFactory

SystemPropertyInstanceIdGenerator - Class in org.quartz.simpl
InstanceIdGenerator that will use a system property to configure the scheduler.
SystemPropertyInstanceIdGenerator() - Constructor for class org.quartz.simpl. SystemPropertyInstanceIdGenerator
TABLE_BLOB_TRIGGERS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_CALENDARS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_CRON_TRIGGERS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_FIRED_TRIGGERS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_JOB_DETAILS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_LOCKS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_PAUSED_TRIGGERS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_PREFIX_SUBST - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCCConstants

TABLE_SCHEDULER_STATE - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_SIMPLE_PROPERTIES_TRIGGERS - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

TABLE_SIMPLE_TRIGGERS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TABLE_TRIGGERS - Static variable in interface org.quartz.impl.jdbcjobstore.Constants
**tablePrefix** - Variable in class
org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

**tablePrefix** - Variable in class org.quartz.impl.jdbcjobstore.JobStoreSupport

**tablePrefix** - Variable in class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**tablePrefix** - Variable in class
org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

**tablePrefix** - Variable in class
org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**TablePrefixAware** - Interface in org.quartz.impl.jdbcjobstore
Interface for Quartz objects that need to know what the table prefix of the
tables used by a JDBC JobStore is.

**TABULAR_TYPE** - Static variable in class
org.quartz.core.jmx.JobDataMapSupport

**threadAcquired()** - Method in class
org.quartz.locality.constraint.evaluator.CpuEvaluator

**ThreadContextClassLoadHelper** - Class in org.quartz.simpl
A ClassLoadHelper that uses either the current thread's context class loader
(Thread.currentThread().getContextClassLoader().loadClass( ..

**ThreadContextClassLoadHelper()** - Constructor for class
org.quartz.simpl.ThreadContextClassLoadHelper

**threadReleased()** - Method in class
org.quartz.locality.constraint.evaluator.CpuEvaluator

**threadsAvailableAtLeast(int)** - Static method in class
org.quartz.locality.constraint.CpuConstraint
    Creates a constraint that requires the node to have a least an amount of
    threads available

**THURSDAY** - Static variable in class org.quartz.DateBuilder

**TimeStampedCounterValue** - Class in org.quartz.utils.counter.sampled
A counter value at a particular time instance

**TimeStampedCounterValue(long, long)** - Constructor for class org.quartz.utils.counter.sampled. TimeStampedCounterValue

Constructor accepting the value of both timestamp and the counter value.

**timeTriggers** - Variable in class org.quartz.simpl. RAMJobStore

**toArray(T[])** - Method in class org.quartz.utils. CircularLossyQueue

Returns an array of the current elements in the queue.

**toBoolean(boolean)** - Method in class org.quartz.impl. RemoteMBeanScheduler

**toBytes(long)** - Method in enum org.quartz.locality.constraint. MemoryConstraint.Unit

**toCompositeData(CronTrigger)** - Static method in class org.quartz.core.jmx. CronTriggerSupport

**toCompositeData(String, String)** - Static method in class org.quartz.core.jmx. JobDataMapSupport

**toCompositeData(JobDetail)** - Static method in class org.quartz.core.jmx. JobDetailSupport

**toCompositeData(JobExecutionContext)** - Static method in class org.quartz.core.jmx. JobExecutionContextSupport

**toCompositeData(SimpleTrigger)** - Static method in class org.quartz.core.jmx. SimpleTriggerSupport

**toCompositeData(Trigger)** - Static method in class org.quartz.core.jmx. TriggerSupport

**toCompositeList(List<? extends Trigger>)** - Static method in class org.quartz.core.jmx. TriggerSupport

**todayAt(int, int, int)** - Static method in class org.quartz. DateBuilder

Get a Date object that represents the given time, on today's date (equivalent to DateBuilder.dateOf(int, int, int)).

**tomorrowAt(int, int, int)** - Static method in class org.quartz. DateBuilder

Get a Date object that represents the given time, on tomorrow's date.
**toSqlLikeClause**(*GroupMatcher*) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**toString()** - Method in class org.quartz.CronExpression
Returns the string representation of the CronExpression

**toString()** - Method in class org.quartz.impl.calendar.CronCalendar
Returns a string representing the properties of the CronCalendar

**toString()** - Method in class org.quartz.impl.calendar.DailyCalendar
Returns a string representing the properties of the DailyCalendar

**toString()** - Method in class org.quartz.impl.jdbcjobstore.TriggerStatus
Return the string representation of the TriggerStatus.

**toString()** - Method in class org.quartz.impl.JobDetailImpl
Return a simple string representation of this object.

**toString()** - Method in class org.quartz.impl.JobExecutionContextImpl

**toString()** - Method in class org.quartz.jobs.ee.mail.SendMailJob.MailInfo

**toString()** - Method in class org.quartz.locality.constraint.CpuConstraint

**toString()** - Method in class exception org.quartz.SchedulerException

**toString()** - Method in class org.quartz.SchedulerMetaData
Return a simple string representation of this object.

**toString()** - Method in class org.quartz.utils.counter.sampled.TimeStampedCounterValue

**toString()** - Method in class org.quartz.utils.Key
Return the string representation of the key.

**toTabularData(List<? extends CronTrigger>)** - Static method in class org.quartz.core.jmx.CronTriggerSupport

**toTabularData(JobDataMap)** - Static method in class org.quartz.core.jmx.JobDataMapSupport

**toTabularData(JobDetail[])** - Static method in class org.quartz.core.jmx.JobDetailSupport

**toTabularData(List<JobExecutionContext>)** - Static method in class
org.quartz.core.jmx.JobExecutionContextSupport

toTabularData(List<? extends SimpleTrigger>) - Static method in class org.quartz.core.jmx.SimpleTriggerSupport

toTabularData(List<? extends Trigger>) - Static method in class org.quartz.core.jmx.TriggerSupport

translateTime(Date, TimeZone, TimeZone) - Static method in class org.quartz.core.jmx.DateBuilder

Translate a date & time from a users time zone to the another (probably server) time zone to assist in creating a simple trigger with the right date & time.

**Trigger** - Interface in org.quartz

The base interface with properties common to all Triggers - use **TriggerBuilder** to instantiate an actual Trigger.

**Trigger.CompletedExecutionInstruction** - Enum in org.quartz

NOOP Instructs the Scheduler that the Trigger has no further instructions.

**Trigger.TriggerState** - Enum in org.quartz

**Trigger.TriggerTimeComparator** - Class in org.quartz

A Comparator that compares trigger's next fire times, or in other words, sorts them according to earliest next fire time.

**Trigger.TriggerTimeComparator()** - Constructor for class org.quartz.Trigger.TriggerTimeComparator

**TRIGGER_FINALIZED** - Static variable in interface org.quartz.core.jmx.QuartzSchedulerMBean

**TriggerBuilder<T** extends Trigger> - Class in org.quartz

TriggerBuilder is used to instantiate Trigger.

**triggerComplete(Trigger, JobExecutionContext, Trigger.CompletedExecutionInstruction)** - Method in class org.quartz.listeners.BroadcastTriggerListener

**triggerComplete(Trigger, JobExecutionContext, Trigger.CompletedExecutionInstruction)** - Method in class org.quartz.listeners.TriggerListenerSupport
**triggerComplete(Trigger, JobExecutionContext, Trigger.CompletedExecutionInstruction)** - Method in class org.quartz.plugins.history.**LoggingTriggerHistoryPlugin**

**triggerComplete(Trigger, JobExecutionContext, Trigger.CompletedExecutionInstruction)** - Method in interface org.quartz.**TriggerListener**

Called by the Scheduler when a Trigger has fired, its associated JobDetail has been executed, and its triggered(xx) method has been called.

**triggered(Calendar)** - Method in class org.quartz.impl.triggers.**AbstractTrigger**

This method should not be used by the Quartz client.

**triggered(Calendar)** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).

**triggered(Calendar)** - Method in class org.quartz.impl.triggers.**CronTriggerImpl**

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).

**triggered(Calendar)** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).

**triggered(Calendar)** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

**triggeredJobComplete(OperableTrigger, JobDetail, Trigger.CompletedExecutionInstruction)** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

Inform the JobStore that the scheduler has completed the firing of the given Trigger (and the execution its associated Job), and that the JobDataMap in the given JobDetail should be updated if the Job is stateful.

**triggeredJobComplete(Connection, OperableTrigger, JobDetail, Trigger.CompletedExecutionInstruction)** - Method in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**
triggeredJobComplete(OperableTrigger, JobDetail, Trigger.CompletedExecutionInstruction) - Method in class org.quartz.simpl.RAMJobStore
Inform the JobStore that the scheduler has completed the firing of the given Trigger (and the execution its associated Job), and that the JobDataMap in the given JobDetail should be updated if the Job is stateful.

triggerExists(Connection, TriggerKey) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Check whether or not a trigger exists.

triggerExists(Connection, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
Check existence of a given trigger.

triggerExists(Connection, TriggerKey) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Check whether or not a trigger exists.

triggerFinalized(Trigger) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl
Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.

triggerFired(Connection, OperableTrigger) - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

triggerFired(Trigger, JobExecutionContext) - Method in class org.quartz.listeners.BroadcastTriggerListener

triggerFired(Trigger, JobExecutionContext) - Method in class org.quartz.listeners.TriggerListenerSupport

triggerFired(Trigger, JobExecutionContext) - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin
triggerFired(Trigger, JobExecutionContext) - Method in interface org.quartz.TriggerListener
   Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.

triggerGroupsToDelete - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

triggerGroupsToNeverDelete - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

triggerJob(String, String, Map<String, String>) - Method in interface org.quartz.core.jmx.QuartzSchedulerMBean

triggerJob(JobKey, JobDataMap) - Method in class org.quartz.core.QuartzScheduler
   Trigger the identified Job (execute it now) - with a non-volatile trigger.

triggerJob(OperableTrigger) - Method in class org.quartz.core.QuartzScheduler
   Store and schedule the identified OperableTrigger

triggerJob(String, String, Map<String, String>) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

triggerJob(CompositeData) - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

triggerJob(JobKey, JobDataMap) - Method in interface org.quartz.core.RemotableQuartzScheduler

triggerJob(OperableTrigger) - Method in interface org.quartz.core.RemotableQuartzScheduler

triggerJob(JobKey) - Method in class org.quartz.impl.RemoteMBeanScheduler
   Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

triggerJob(JobKey, JobDataMap) - Method in class org.quartz.impl.RemoteMBeanScheduler
   Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

triggerJob(JobKey) - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**triggerJob(JobKey, JobDataMap)** - Method in class org.quartz.impl.RemoteScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**triggerJob(JobKey)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**triggerJob(JobKey, JobDataMap)** - Method in class org.quartz.impl.StdScheduler
Calls the equivalent method on the 'proxied' QuartzScheduler.

**triggerJob(JobKey)** - Method in interface org.quartz.Scheduler
Trigger the identified JobDetail (execute it now).

**triggerJob(JobKey, JobDataMap)** - Method in interface org.quartz.Scheduler
Trigger the identified JobDetail (execute it now).

**TriggerKey** - Class in org.quartz
Uniquely identifies a Trigger.

**TriggerKey(String)** - Constructor for class org.quartz.TriggerKey

**TriggerKey(String, String)** - Constructor for class org.quartz.TriggerKey

**triggerKey(String)** - Static method in class org.quartz.TriggerKey

**triggerKey(String, String)** - Static method in class org.quartz.TriggerKey

**TriggerListener** - Interface in org.quartz
The interface to be implemented by classes that want to be informed when a Trigger fires.

**TriggerListenerSupport** - Class in org.quartz.listeners
A helpful abstract base class for implementors of TriggerListener.

**TriggerListenerSupport()** - Constructor for class org.quartz.listeners.TriggerListenerSupport

**triggerMisfired(Trigger)** - Method in class org.quartz.listeners.BroadcastTriggerListener

**triggerMisfired(Trigger)** - Method in class org.quartz.listeners.TriggerListenerSupport

**triggerMisfired(Trigger)** - Method in class org.quartz.plugins.history.LoggingTriggerHistoryPlugin
**triggerMisfired(Trigger)** - Method in interface org.quartz.TriggerListener
   Called by the Scheduler when a Trigger has misfired.

**triggerPaused(TriggerKey)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**triggerPaused(TriggerKey)** - Method in class org.quartz.listeners.BroadcastSchedulerListener

**triggerPaused(TriggerKey)** - Method in class org.quartz.listeners.SchedulerListenerSupport

**triggerPaused(TriggerKey)** - Method in interface org.quartz.SchedulerListener
   Called by the Scheduler when a Trigger has been paused.

**TriggerPersistenceDelegate** - Interface in org.quartz.impl.jdbcjobstore
   An interface which provides an implementation for storing a particular type of Trigger's extended properties.

**TriggerPersistenceDelegate.TriggerPropertyBundle** - Class in org.quartz.impl.jdbcjobstore

**TriggerPersistenceDelegate.TriggerPropertyBundle(ScheduleBuilder, String[], Object[])** - Constructor for class org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPropertyBundle

**triggerPersistenceDelegates** - Variable in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

**triggerResumed(TriggerKey)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**triggerResumed(TriggerKey)** - Method in class org.quartz.listeners.BroadcastSchedulerListener

**triggerResumed(TriggerKey)** - Method in class org.quartz.listeners.SchedulerListenerSupport

**triggerResumed(TriggerKey)** - Method in interface org.quartz.SchedulerListener
   Called by the Scheduler when a Trigger has been un-paused.
**triggers** - Variable in class org.quartz.simpl.RAMJobStore

**TRIGGERS_PAUSED** - Static variable in interface org.quartz.core.jmx.QuartzSchedulerMBean

**TRIGGERS_RESUMED** - Static variable in interface org.quartz.core.jmx.QuartzSchedulerMBean

**triggersByGroup** - Variable in class org.quartz.simpl.RAMJobStore

**triggersByKey** - Variable in class org.quartz.simpl.RAMJobStore

**triggersFired(List<OperableTrigger>)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport
   Inform the JobStore that the scheduler is now firing the given Trigger (executing its associated Job), that it had previously acquired (reserved).

**triggersFired(List<OperableTrigger>)** - Method in class org.quartz.simpl.RAMJobStore
   Inform the JobStore that the scheduler is now firing the given Trigger (executing its associated Job), that it had previously acquired (reserved).

**triggersPaused(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**triggersPaused(String)** - Method in class org.quartz.listeners.BroadcastSchedulerListener

**triggersPaused(String)** - Method in class org.quartz.listeners.SchedulerListenerSupport

**triggersPaused(String)** - Method in interface org.quartz.SchedulerListener
   Called by the Scheduler when a group of Triggers has been paused.

**triggersResumed(String)** - Method in class org.quartz.core.QuartzSchedulerMBeanImpl

**triggersResumed(String)** - Method in class org.quartz.listeners.BroadcastSchedulerListener

**triggersResumed(String)** - Method in class org.quartz.listeners.SchedulerListenerSupport
triggersResumed(String) - Method in interface org.quartz.SchedulerListener
Called by the Scheduler when a group of Triggers has been un-paused.

TriggerStatus - Class in org.quartz.impl.jdbcjobstore
Object representing a job or trigger key.

TriggerStatus(String, Date) - Constructor for class org.quartz.impl.jdbcjobstore.TriggerStatus
Construct a new TriggerStatus with the status name and nextFireTime.

triggersToDelete - Variable in class org.quartz.xml.XMLSchedulingDataProcessor

TriggerSupport - Class in org.quartz.core.jmx

TriggerSupport() - Constructor for class org.quartz.core.jmx.TriggerSupport

TriggerUtils - Class in org.quartz
Convenience and utility methods for working with Triggers.

TTYPE_BLOB - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TTYPE_CAL_INT - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TTYPE_CRON - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TTYPE_SIMPLE - Static variable in interface org.quartz.impl.jdbcjobstore.Constants

TUESDAY - Static variable in class org.quartz.DateBuilder
UnableToInterruptJobException - Exception in org.quartz

An exception that is thrown to indicate that a call to
InterruptableJob.interrupt() failed without interrupting the Job.

UnableToInterruptJobException(String) - Constructor for exception
org.quartz.UnableToInterruptJobException

Create a UnableToInterruptJobException with the given message.

UnableToInterruptJobException(Throwables) - Constructor for exception
org.quartz.UnableToInterruptJobException

Create a UnableToInterruptJobException with the given cause.

unscheduleAllTriggers() - Method in exception
org.quartz.JobExecutionException

unscheduleFiringTriggers() - Method in exception
org.quartz.JobExecutionException

unscheduleJob(String, String) - Method in interface
org.quartz.core.jmx.QuartzSchedulerMBean

unscheduleJob(TriggerKey) - Method in class org.quartz.core.QuartzScheduler

Remove the indicated Trigger from the scheduler.

unscheduleJob(TriggerKey, String) - Method in class
org.quartz.core.QuartzSchedulerMBeanImpl

unscheduleJob(TriggerKey) - Method in interface
org.quartz.core.RemotableQuartzScheduler

unscheduleJob(TriggerKey) - Method in class
org.quartz.impl.RemoteMBeanScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

unscheduleJob(TriggerKey) - Method in class
org.quartz.impl.RemoteScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.

unscheduleJob(TriggerKey) - Method in class org.quartz.impl.StdScheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.
**unscheduleJob(TriggerKey)** - Method in interface org.quartz.Scheduler
Remove the indicated Trigger from the scheduler.

**unscheduleJobs(List<TriggerKey>)** - Method in class org.quartz.core.QuartzScheduler

**unscheduleJobs(List<TriggerKey>)** - Method in interface org.quartz.core.RemotableQuartzScheduler

**unscheduleJobs(List<TriggerKey>)** - Method in class org.quartz.impl.RemoteMBeanScheduler

**unscheduleJobs(List<TriggerKey>)** - Method in class org.quartz.impl.RemoteScheduler

**unscheduleJobs(List<TriggerKey>)** - Method in interface org.quartz.core.RemotableQuartzScheduler

- Remove all of the indicated Triggers from the scheduler.

**UPDATE_BLOB_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_CALENDAR** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_CRON_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_FIRED_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_FOR_LOCK** - Static variable in class org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore

**UPDATE_INSTANCES_FIRED_TRIGGER_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_JOB_DATA** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
**UPDATE_JOB_DETAIL** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_JOB_TRIGGER_STATES** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_ORACLE_CALENDAR_BLOB** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

**UPDATE_ORACLE_JOB_DETAIL** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

**UPDATE_ORACLE_JOB_DETAIL_BLOB** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

**UPDATE_ORACLE_TRIGGER** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

**UPDATE_ORACLE_TRIGGER_JOB_DETAIL_BLOB** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

**UPDATE_ORACLE_TRIGGER_JOB_DETAIL_EMPTY_BLOB** - Static variable in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

**UPDATE_SCHEDULER_STATE** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_SIMPLE_PROPS_TRIGGER** - Static variable in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**UPDATE_SIMPLE_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

**UPDATE_TRIGGER** - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

UPDATE_TRIGGER_GROUP_STATE_FROM_STATES - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

UPDATE_TRIGGER_SKIP_DATA - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

UPDATE_TRIGGER_STATE - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

UPDATE_TRIGGER_STATE_FROM_STATE - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

UPDATE_TRIGGER_STATE_FROM_STATES - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

UPDATE_TRIGGER_STATES_FROM_OTHER_STATES - Static variable in interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

updateAfterMisfire(Calendar) - Method in class org.quartz.impl.triggers.AbstractTrigger
This method should not be used by the Quartz client.

updateAfterMisfire(Calendar) - Method in class org.quartz.impl.triggers.CalendarIntervalTriggerImpl
Updates the DateIntervalTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the DateIntervalTrigger was created.

updateAfterMisfire(Calendar) - Method in class org.quartz.impl.triggers.CronTriggerImpl
Updates the CronTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the CronTrigger was created.

updateAfterMisfire(Calendar) - Method in class org.quartz.impl.triggers.SimpleTriggerImpl
Updates the SimpleTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the
SimpleTrigger was created.

**updateAfterMisfire(Calendar)** - Method in class org.quartz.locality.DelegatingLocalityTrigger

**updateBlobTrigger(Connection, OperableTrigger)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Update the blob trigger data.

**updateCalendar(Connection, String, Calendar)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Update a calendar.

**updateCalendar(Connection, String, Calendar)** - Method in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

**updateCalendar(Connection, String, Calendar)** - Method in class org.quartz.impl.jdbcjobstore.PointbaseDelegate

Update a calendar.

**updateCalendar(Connection, String, Calendar)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Update a calendar.

**UpdateChecker** - Class in org.quartz.utils

Check for updates and alert users if an update is available

**UpdateChecker()** - Constructor for class org.quartz.utils.UpdateChecker

**updateExtendedTriggerProperties(Connection, OperableTrigger, String, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

**updateExtendedTriggerProperties(Connection, OperableTrigger, String, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

**updateExtendedTriggerProperties(Connection, OperableTrigger, String, JobDetail)** - Method in interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate

**updateFiredTrigger(Connection, OperableTrigger, String, JobDetail)** -
Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
  Update a fired trigger record.
  **updateFiredTrigger(Connection, OperableTrigger, String, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  Update a fired trigger.
  **updateJobData(Connection, JobDetail)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
  Update the job data map for the given job.
  **updateJobData(Connection, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

  **updateJobData(Connection, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.PointbaseDelegate
  Update the job data map for the given job.
  **updateJobData(Connection, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  Update the job data map for the given job.
  **updateJobDetail(Connection, JobDetail)** - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
  Update the job detail record.
  **updateJobDetail(Connection, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

  **updateJobDetail(Connection, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.PointbaseDelegate
  Update the job detail record.
  **updateJobDetail(Connection, JobDetail)** - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  Update the job detail record.
  **UpdateLockRowSemaphore** - Class in org.quartz.impl.jdbcjobstore
  Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.
  **UpdateLockRowSemaphore()** - Constructor for class org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore

  **updateMisfiredTrigger(Connection, TriggerKey, String, boolean)** - Method in class org.quartz.impl.jdbcjobstore.JobStoreSupport

  **updateSchedulerState(Connection, String, long)** - Method in interface
org.quartz.impl.jdbcjobstore.DriverDelegate
   Update a scheduler-instance state record.
updateSchedulerState(Connection, String, long) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

updateTrigger(Connection, OperableTrigger, String, JobDetail) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
   Update the base trigger data.
updateTrigger(Connection, OperableTrigger, String, JobDetail) - Method in class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

updateTrigger(Connection, OperableTrigger, String, JobDetail) - Method in class org.quartz.impl.jdbcjobstore.PointbaseDelegate

updateTrigger(Connection, OperableTrigger, String, JobDetail) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   Update the base trigger data.
updateTriggerGroupStateFromOtherState(Connection, GroupMatcher<TriggerKey>, String, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
   Update all of the triggers of the given group to the given new state, if they are in the given old state.
updateTriggerGroupStateFromOtherState(Connection, GroupMatcher<TriggerKey>, String, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   Update all of the triggers of the given group to the given new state, if they are in the given old state.
updateTriggerGroupStateFromOtherStates(Connection, GroupMatcher<TriggerKey>, String, String, String, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
   Update all triggers in the given group to the given new state, if they are in one of the given old states.
updateTriggerGroupStateFromOtherStates(Connection, GroupMatcher<TriggerKey>, String, String, String, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   Update all triggers in the given group to the given new state, if they are in one of the given old states.
updateTriggerState(Connection, TriggerKey, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate
Update the state for a given trigger.

updateTriggerState(Connection, TriggerKey, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Update the state for a given trigger.

updateTriggerStateFromOtherState(Connection, TriggerKey, String, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Update the given trigger to the given new state, if it is in the given old state.

updateTriggerStateFromOtherStates(Connection, TriggerKey, String, String, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Update the given trigger to the given new state, if it is one of the given old states.

updateTriggerStatesForJob(Connection, JobKey, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Update the states of all triggers associated with the given job.

updateTriggerStatesForJob(Connection, JobKey, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

Update the states of all triggers associated with the given job.

updateTriggerStatesForJobFromOtherState(Connection, JobKey, String, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Update the states of any triggers associated with the given job, that are in the given current state.

updateTriggerStatesForJobFromOtherState(Connection, JobKey, String, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

updateTriggerStatesFromOtherStates(Connection, String, String, String) - Method in interface org.quartz.impl.jdbcjobstore.DriverDelegate

Update all triggers having one of the two given states, to the given new state.

updateTriggerStatesFromOtherStates(Connection, String, String, String) - Method in class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
Insert the job detail record.

**updateWithNewCalendar(Calendar, long)** - Method in class org.quartz.impl.triggers.**AbstractTrigger**

This method should not be used by the Quartz client.

**updateWithNewCalendar(Calendar, long)** - Method in class org.quartz.impl.triggers.**CalendarIntervalTriggerImpl**

**updateWithNewCalendar(Calendar, long)** - Method in class org.quartz.impl.triggers.**CronTriggerImpl**

**updateWithNewCalendar(Calendar, long)** - Method in class org.quartz.impl.triggers.**SimpleTriggerImpl**

**updateWithNewCalendar(Calendar, long)** - Method in class org.quartz.locality.**DelegatingLocalityTrigger**

useProperties - Variable in class org.quartz.impl.jdbcjobstore.**JobStoreSupport**

useProperties - Variable in class org.quartz.impl.jdbcjobstore.**StdJDBCDelegate**

UserTransactionHelper - Class in org.quartz.ee.jta

A helper for obtaining a handle to a UserTransaction...

useTransaction(JobDataMap) - Static method in class org.quartz.jobs.ee.jms.**JmsHelper**

usingJobData(String, String) - Method in class org.quartz.**JobBuilder**

Add the given key-value pair to the JobDetail's **JobDataMap**.

usingJobData(String, Integer) - Method in class org.quartz.**JobBuilder**

Add the given key-value pair to the JobDetail's **JobDataMap**.

usingJobData(String, Long) - Method in class org.quartz.**JobBuilder**

Add the given key-value pair to the JobDetail's **JobDataMap**.

usingJobData(String, Float) - Method in class org.quartz.**JobBuilder**

Add the given key-value pair to the JobDetail's **JobDataMap**.

usingJobData(String, Double) - Method in class org.quartz.**JobBuilder**

Add the given key-value pair to the JobDetail's **JobDataMap**.

usingJobData(String, Boolean) - Method in class org.quartz.**JobBuilder**

Add the given key-value pair to the JobDetail's **JobDataMap**.

usingJobData(JobDataMap) - Method in class org.quartz.**JobBuilder**

Set the JobDetail's **JobDataMap**, adding any values to it that were already set on this JobBuilder using any of the other 'usingJobData' methods.
**usingJobData(String, String)** - Method in class org.quartz.TriggerBuilder
Add the given key-value pair to the Trigger's JobDataMap.

**usingJobData(String, Integer)** - Method in class org.quartz.TriggerBuilder
Add the given key-value pair to the Trigger's JobDataMap.

**usingJobData(String, Long)** - Method in class org.quartz.TriggerBuilder
Add the given key-value pair to the Trigger's JobDataMap.

**usingJobData(String, Float)** - Method in class org.quartz.TriggerBuilder
Add the given key-value pair to the Trigger's JobDataMap.

**usingJobData(String, Double)** - Method in class org.quartz.TriggerBuilder
Add the given key-value pair to the Trigger's JobDataMap.

**usingJobData(String, Boolean)** - Method in class org.quartz.TriggerBuilder
Add the given key-value pair to the Trigger's JobDataMap.

**usingJobData(JobDataMap)** - Method in class org.quartz.TriggerBuilder
Set the Trigger's JobDataMap, adding any values to it that were already set on this TriggerBuilder using any of the other 'usingJobData' methods.

**Util** - Class in org.quartz.impl.jdbctemplate
This class contains utility functions for use in all delegate classes.
validate() - Method in class org.quartz.impl.triggers. AbstractTrigger
Validates whether the properties of the JobDetail are valid for submission into a Scheduler.

validate() - Method in class org.quartz.impl.triggers. CalendarIntervalTriggerImpl
Validates whether the properties of the JobDetail are valid for submission into a Scheduler.

validate() - Method in class org.quartz.impl.triggers. SimpleTriggerImpl
Validates whether the properties of the JobDetail are valid for submission into a Scheduler.

validateDayOfMonth(int) - Static method in class org.quartz. DateBuilder

validateDayOfWeek(int) - Static method in class org.quartz. DateBuilder

validateExpression(String) - Static method in class org.quartz. CronExpression

validateHour(int) - Static method in class org.quartz. DateBuilder

validateMinute(int) - Static method in class org.quartz. DateBuilder

validateMisfireInstruction(int) - Method in class org.quartz.impl.triggers. AbstractTrigger

validateMisfireInstruction(int) - Method in class org.quartz.impl.triggers. CalendarIntervalTriggerImpl

validateMisfireInstruction(int) - Method in class org.quartz.impl.triggers. CronTriggerImpl

validateMisfireInstruction(int) - Method in class org.quartz.impl.triggers. SimpleTriggerImpl

validateMonth(int) - Static method in class org.quartz. DateBuilder
**validateSecond(int)** - Static method in class org.quartz.DateBuilder

**validateState()** - Method in class org.quartz.core.QuartzScheduler

**validateYear(int)** - Static method in class org.quartz.DateBuilder

**ValidationException** - Exception in org.quartz.xml

Reports JobSchedulingDataLoader validation exceptions.

**ValidationException()** - Constructor for exception

org.quartz.xml.ValidationException

Constructor for ValidationException.

**ValidationException(String)** - Constructor for exception

org.quartz.xml.ValidationException

Constructor for ValidationException.

**ValidationException(Collection<Exception>)** - Constructor for exception

org.quartz.xml.ValidationException

Constructor for ValidationException.

**validationExceptions** - Variable in class

org.quartz.xml.XMLSchedulingDataProcessor

**valueOf(String)** - Static method in enum org.quartz.DateBuilder.IntervalUnit

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum

org.quartz.impl.matchers.StringMatcher.StringOperatorName

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum

org.quartz.locality.constraint.CpuConstraint.Operator

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum

org.quartz.locality.constraint.EhcacheConstraint.Operator

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in class

org.quartz.locality.constraint.evaluator.EvaluatorTimerTask.Status

**valueOf(String)** - Static method in enum

org.quartz.locality.constraint.evaluator.EvaluatorTimerTask.Status
Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum org.quartz.locality.constraint.**MemoryConstraint.Operator**

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum org.quartz.locality.constraint.**MemoryConstraint.Unit**

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum org.quartz.locality.constraint.**NodeGroupConstraint.Operator**

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum org.quartz.locality.constraint.**OsConstraint.Operator**

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum org.quartz.locality.constraint.**OsConstraint.OS**

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum org.quartz.**Trigger.CompletedExecutionInstruction**

Returns the enum constant of this type with the specified name.

**valueOf(String)** - Static method in enum org.quartz.**Trigger.TriggerState**

Returns the enum constant of this type with the specified name.

**values()** - Static method in enum org.quartz.**DateBuilder.IntervalUnit**

Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum org.quartz.impl.matchers.**StringMatcher.StringOperatorName**

Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum org.quartz.locality.constraint.**CpuConstraint.Operator**

Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum org.quartz.locality.constraint.**EhcacheConstraint.Operator**

Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum org.quartz.locality.constraint.evaluator.**EvaluatorTimerTask.Status**

Returns an array containing the constants of this enum type, in the order
they are declared.

**values()** - Static method in enum
org.quartz.locality.constraint.**MemoryConstraint.Operator**
Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum
org.quartz.locality.constraint.**MemoryConstraint.Unit**
Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum
org.quartz.locality.constraint.**NodeGroupConstraint.Operator**
Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum
org.quartz.locality.constraint.**OsConstraint.Operator**
Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum
org.quartz.locality.constraint.**OsConstraint.OS**
Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum
org.quartz.**Trigger.CompletedExecutionInstruction**
Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Static method in enum
org.quartz.**Trigger.TriggerState**
Returns an array containing the constants of this enum type, in the order they are declared.

**values()** - Method in class org.quartz.utils.**DirtyFlagMap**

**verify(CpuConstraint)** - Method in class
org.quartz.locality.constraint.evaluator.**CpuEvaluator**

**verify(T)** - Method in interface org.quartz.locality.constraint.evaluator.**Evaluator**
Verifies the validity of a constraint.

**verify(MemoryConstraint)** - Method in class
org.quartz.locality.constraint.evaluator.**MemoryEvaluator**
Always true

**verify(NodeGroupConstraint)** - Method in class
org.quartz.locality.constraint.evaluator.**NodeGroupEvaluator**
Verifies that the node group actually exists in the current configuration 
**verify(OsConstraint)** - Method in class 
org.quartz.locality.constraint.evaluator.OsEvaluator

Verifies the validity of a constraint. 
**verify(EhcacheConstraint)** - Method in class 
org.terracotta.modules.ehcache.store.EhcacheEvaluator

Doesn't do anything 
**VersionPrinter** - Class in org.quartz.helpers

Prints the version of Quartz on stdout. 
**vetoedJobRetryLoop(OperableTrigger, JobDetail, Trigger.CompletedExecutionInstruction)** - Method in class 
org.quartz.core.JobRunShell

**vetoJobExecution(Trigger, JobExecutionContext)** - Method in class 
org.quartz.listeners.BroadcastTriggerListener

**vetoJobExecution(Trigger, JobExecutionContext)** - Method in class 
org.quartz.listeners.TriggerListenerSupport

**vetoJobExecution(Trigger, JobExecutionContext)** - Method in class 
org.quartz.plugins.history.LoggingTriggerHistoryPlugin

**vetoJobExecution(Trigger, JobExecutionContext)** - Method in interface 
org.quartz.TriggerListener

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.
warning(SAXParseException) - Method in class org.quartz.xml.XMLSchedulingDataProcessor
   ErrorHandler interface.
WeblogicConnectionProvider - Class in org.quartz.utils.weblogic
   Provides connections via Weblogic's JTS driver.
WeblogicConnectionProvider(String) - Constructor for class org.quartz.utils.weblogic.WeblogicConnectionProvider
WebLogicDelegate - Class in org.quartz.impl.jdbcjobstore
   This is a driver delegate for the WebLogic JDBC driver.
WebLogicDelegate(Logger, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.WebLogicDelegate
   Create new WebLogicDelegate instance.
WebLogicDelegate(Logger, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore.WebLogicDelegate
   Create new WebLogicDelegate instance.
WebLogicOracleDelegate - Class in org.quartz.impl.jdbcjobstore.oracle.weblogic
   Handle Blobs correctly when Oracle is being used inside of Weblogic 8.1, as discussed at:
   http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705
WebLogicOracleDelegate(Logger, String, String, String, ClassLoadHelper) - Constructor for class org.quartz.impl.jdbcjobstore.oracle.weblogic.WebLogicOracleDelegate
   Create new WebLogicOracleDelegate instance.
WebLogicOracleDelegate(Logger, String, String, String, ClassLoadHelper, Boolean) - Constructor for class org.quartz.impl.jdbcjobstore.oracle.weblogic.WebLogicOracleDelegate
   Create new WebLogicOracleDelegate instance.
WEDNESDAY - Static variable in class org.quartz.DateBuilder
WeeklyCalendar - Class in org.quartz.impl.calendar
   This implementation of the Calendar excludes a set of days of the week.
WeeklyCalendar() - Constructor for class org.quartz.impl.calendar.WeeklyCalendar
**WeeklyCalendar(Calendar)** - Constructor for class org.quartz.impl.calendar.**WeeklyCalendar**

**WeeklyCalendar(TimeZone)** - Constructor for class org.quartz.impl.calendar.**WeeklyCalendar**

**WeeklyCalendar(Calendar, TimeZone)** - Constructor for class org.quartz.impl.calendar.**WeeklyCalendar**

**weeklyOnDayAndHourAndMinute(int, int, int)** - Static method in class org.quartz.**CronScheduleBuilder**
Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire one per week on the given day at the given time (hour and minute).

**where(NodeSpecBuilder)** - Method in class org.quartz.locality.**LocalityJobBuilder**
Setter to the **NodeSpecBuilder** that will be build at JobDetail **LocalityJobBuilder.build()** time

**where(NodeSpecBuilder)** - Method in class org.quartz.locality.**LocalityTriggerBuilder**
Setter to the **NodeSpecBuilder** that will be build at JobDetail **LocalityTriggerBuilder.build()** time

**willFireOn(Calendar)** - Method in class org.quartz.impl.triggers.**CronTriggerImpl**
Determines whether the date and (optionally) time of the given Calendar instance falls on a scheduled fire-time of this trigger.

**willFireOn(Calendar, boolean)** - Method in class org.quartz.impl.triggers.**CronTriggerImpl**
Determines whether the date and (optionally) time of the given Calendar instance falls on a scheduled fire-time of this trigger.

**WINDOWS** - Static variable in class org.quartz.locality.constraint.**OsConstraint**

**withDescription(String)** - Method in class org.quartz.**JobBuilder**
Set the given (human-meaningful) description of the Job.

**withDescription(String)** - Method in class org.quartz.**TriggerBuilder**
Set the given (human-meaningful) description of the Trigger.

**withIdentity(String)** - Method in class org.quartz.**JobBuilder**
Use a JobKey with the given name and default group to identify the JobDetail.
**withIdentity(String, String)** - Method in class org.quartz.JobBuilder
Use a JobKey with the given name and group to identify the JobDetail.

**withIdentity(JobKey)** - Method in class org.quartz.JobBuilder
Use a JobKey to identify the JobDetail.

**withIdentity(String)** - Method in class org.quartz.TriggerBuilder
Use a TriggerKey with the given name and default group to identify the Trigger.

**withIdentity(String, String)** - Method in class org.quartz.TriggerBuilder
Use a TriggerKey with the given name and group to identify the Trigger.

**withIdentity(String, String)** - Method in class org.quartz.TriggerBuilder
Use a TriggerKey with the given name and group to identify the Trigger.

**withIdentity(TriggerKey)** - Method in class org.quartz.TriggerBuilder
Use the given TriggerKey to identify the Trigger.

**withInterval(int, DateBuilder.IntervalUnit)** - Method in class org.quartz.CalendarIntervalScheduleBuilder
Specify the time unit and interval for the Trigger to be produced.

**withIntervalInDays(int)** - Method in class org.quartz.CalendarIntervalScheduleBuilder
Specify an interval in the IntervalUnit.DAY that the produced Trigger will repeat at.

**withIntervalInHours(int)** - Method in class org.quartz.CalendarIntervalScheduleBuilder
Specify an interval in the IntervalUnit.HOUR that the produced Trigger will repeat at.

**withIntervalInHours(int)** - Method in class org.quartz.SimpleScheduleBuilder
Specify a repeat interval in minutes - which will then be multiplied by 60 * 60 * 1000 to produce milliseconds.

**withIntervalInMilliseconds(long)** - Method in class org.quartz.SimpleScheduleBuilder
Specify a repeat interval in milliseconds.

**withIntervalInMinutes(int)** - Method in class org.quartz.CalendarIntervalScheduleBuilder
Specify an interval in the IntervalUnit.MINUTE that the produced Trigger will repeat at.

**withIntervalInMinutes(int)** - Method in class org.quartz.SimpleScheduleBuilder
Specify a repeat interval in minutes - which will then be multiplied by 60 * 1000 to produce milliseconds.

**withIntervalInMonths(int)** - Method in class org.quartz.CalendarIntervalScheduleBuilder
Specify an interval in the IntervalUnit.MONTH that the produced Trigger
will repeat at.

**withIntervalInSeconds(int)** - Method in class org.quartz.CalendarIntervalScheduleBuilder

Specify an interval in the IntervalUnit.SECOND that the produced Trigger will repeat at.

**withIntervalInWeeks(int)** - Method in class org.quartz.CalendarIntervalScheduleBuilder

Specify an interval in the IntervalUnit.WEEK that the produced Trigger will repeat at.

**withIntervalInYears(int)** - Method in class org.quartz.CalendarIntervalScheduleBuilder

Specify an interval in the IntervalUnit.YEAR that the produced Trigger will repeat at.

**withMisfireHandlingInstructionDoNothing()** - Method in class org.quartz.CalendarIntervalScheduleBuilder

If the Trigger misfires, use the CalendarIntervalTrigger.MISFIRE_INSTRUCTION_DO NOTHING instruction.

**withMisfireHandlingInstructionDoNothing()** - Method in class org.quartz.CronScheduleBuilder

If the Trigger misfires, use the CronTrigger.MISFIRE_INSTRUCTION_DO NOTHING instruction.

**withMisfireHandlingInstructionFireAndProceed()** - Method in class org.quartz.CalendarIntervalScheduleBuilder

If the Trigger misfires, use the CalendarIntervalTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW instruction.

**withMisfireHandlingInstructionFireAndProceed()** - Method in class org.quartz.CronScheduleBuilder

If the Trigger misfires, use the CronTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW instruction.

**withMisfireHandlingInstructionFireNow()** - Method in class org.quartz.SimpleScheduleBuilder

If the Trigger misfires, use the SimpleTrigger.MISFIRE_INSTRUCTION_FIRE_NOW instruction.
**withMisfireHandlingInstructionIgnoreMisfires()** - Method in class org.quartz.CalendarIntervalScheduleBuilder

If the Trigger misfires, use the `Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY` instruction.

**withMisfireHandlingInstructionIgnoreMisfires()** - Method in class org.quartz.CronScheduleBuilder

If the Trigger misfires, use the `Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY` instruction.

**withMisfireHandlingInstructionIgnoreMisfires()** - Method in class org.quartz.SimpleScheduleBuilder

If the Trigger misfires, use the `SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT` instruction.

**withMisfireHandlingInstructionNextWithExistingCount()** - Method in class org.quartz.SimpleScheduleBuilder

If the Trigger misfires, use the `SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT` instruction.

**withMisfireHandlingInstructionNextWithRemainingCount()** - Method in class org.quartz.SimpleScheduleBuilder

If the Trigger misfires, use the `SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT` instruction.

**withMisfireHandlingInstructionNowWithExistingCount()** - Method in class org.quartz.SimpleScheduleBuilder

If the Trigger misfires, use the `SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT` instruction.

**withMisfireHandlingInstructionNowWithRemainingCount()** - Method in class org.quartz.SimpleScheduleBuilder

If the Trigger misfires, use the `SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT` instruction.

**withPriority(int)** - Method in class org.quartz.TriggerBuilder

Set the Trigger's priority.

**withRepeatCount(int)** - Method in class org.quartz.SimpleScheduleBuilder

Specify a the number of time the trigger will repeat - total number of firings will be this number + 1.

**withSchedule(ScheduleBuilder<SBT>)** - Method in class org.quartz.TriggerBuilder
Set the `ScheduleBuilder` that will be used to define the Trigger's schedule.

`writeDataToBlob(ResultSet, int, byte[])` - Method in class `org.quartz.impl.jdbcjobstore.oracle.OracleDelegate`

`writeDataToBlob(ResultSet, int, byte[])` - Method in class `org.quartz.impl.jdbcjobstore.oracle.weblogic.WebLogicOracleDelegate`

Check for the Weblogic Blob wrapper, and handle accordingly...
XMLSchedulingDataProcessor - Class in org.quartz.xml
Parses an XML file that declares Jobs and their schedules (Triggers), and processes the related data.

XMLSchedulingDataProcessor(ClassLoadHelper) - Constructor for class org.quartz.xml.XMLSchedulingDataProcessor
Constructor for JobSchedulingDataLoader.

XMLSchedulingDataProcessorPlugin - Class in org.quartz.plugins.xml
This plugin loads XML file(s) to add jobs and schedule them with triggers as the scheduler is initialized, and can optionally periodically scan the file for changes.

XMLSchedulingDataProcessorPlugin() - Constructor for class org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin

XSD_DATE_FORMAT - Static variable in class org.quartz.xml.XMLSchedulingDataProcessor
XML Schema dateTime datatype format.
Y

**YEAR** - Static variable in class org.quartz.CronExpression

**YEAR_TO_GIVEUP_SCHEDULING_AT** - Static variable in class org.quartz.impl.triggers.CronTriggerImpl

**years** - Variable in class org.quartz.CronExpression
ZeroSizeThreadPool - Class in org.quartz.simpl
This is class is a simple implementation of a zero size thread pool, based on the ThreadPool interface.

ZeroSizeThreadPool() - Constructor for class org.quartz.simpl.ZeroSizeThreadPool
Create a new ZeroSizeThreadPool.
All Classes

Packages org.quartz
org.quartz.core
org.quartz.core.jmx
org.quartz.ee.jmx.jboss
org.quartz.ee.jta
org.quartz.ee.servlet
org.quartz.helpers
org.quartz.impl
org.quartz.impl.calendar
org.quartz.impl.jdbcjobstore
org.quartz.impl.jdbcjobstore.oracle
org.quartz.impl.jdbcjobstore.oracle.weblogic
org.quartz.impl.matchers
org.quartz.impl.triggers
org.quartz.jobs
org.quartz.jobs.ee.ejb
org.quartz.jobs.ee.jms
org.quartz.jobs.ee.jmx
org.quartz.jobs.ee.mail
org.quartz.listeners
org.quartz.locality
org.quartz.locality.constraint
org.quartz.locality.constraint.evaluator
org.quartz.plugins
org.quartz.plugins.history
org.quartz.plugins.management
org.quartz.plugins.xml
org.quartz.simpl
org.quartz.utils
org.quartz.utils.counter
org.quartz.utils.counter.sampled
org.quartz.utils.weblogic
org.quartz.xml
org.terracotta.modules.ehcache.store
# Quartz Enterprise Job Scheduler 2.0.1 API

## Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz</code></td>
<td>The main package of <strong>Quartz</strong>, containing the client-side interfaces.</td>
</tr>
<tr>
<td><code>org.quartz.core</code></td>
<td>Contains the core classes and interfaces for the <strong>Quartz</strong> job scheduler.</td>
</tr>
<tr>
<td><code>org.quartz.core.jmx</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.ee.jmx.jboss</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.ee.jta</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.ee.servlet</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.helpers</code></td>
<td>Contains helper classes to make working with Quartz easier.</td>
</tr>
<tr>
<td><code>org.quartz.impl</code></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><code>org.quartz.impl.calendar</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore.oracle</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore.oracle.weblogic</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.impl.matchers</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers</code></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td><code>org.quartz.jobs</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs.ee.ejb</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs.ee.jms</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs.ee.jmx</code></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>org.quartz.jobs.ee.mail</td>
<td></td>
</tr>
<tr>
<td>org.quartz.listeners</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality.constraint</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.history</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.management</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.xml</td>
<td></td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.utils</td>
<td></td>
</tr>
<tr>
<td>org.quartz.utils.counter</td>
<td></td>
</tr>
<tr>
<td>org.quartz.utils.counter.sampled</td>
<td></td>
</tr>
<tr>
<td>org.quartz.utils.weblogic</td>
<td></td>
</tr>
<tr>
<td>org.quartz.xml</td>
<td></td>
</tr>
<tr>
<td>org.terracotta.modules.ehcache.store</td>
<td></td>
</tr>
</tbody>
</table>
Hierarchy For All Packages

Package Hierarchies:

- org.quartz
- org.quartz.core
- org.quartz.core.jmx
- org.quartz.ee.jmx.jboss
- org.quartz.ee.jta
- org.quartz.ee.servlet
- org.quartz.helpers
- org.quartz.impl
- org.quartz.impl.calendar
- org.quartz.impl.jdbcjobstore
- org.quartz.impl.jdbcjobstore.oracle
- org.quartz.impl.jdbcjobstore.oracle.weblogic
- org.quartz.impl.matchers
- org.quartz.impl.triggers
- org.quartz.jobs
- org.quartz.jobs.ee.ejb
- org.quartz.jobs.ee.jms
- org.quartz.jobs.ee.jmx
- org.quartz.jobs.ee.mail
- org.quartz.listeners
- org.quartz.locality
- org.quartz.locality.constraint
- org.quartz.locality.constraint.evaluator
- org.quartz.plugins
- org.quartz.plugins.history
- org.quartz.plugins.management
- org.quartz.plugins.xml
- org.quartz.simpl
- org.quartz.utils
- org.quartz.utils.counter
- org.quartz.utils.counter.sampled
- org.quartz.utils.weblogic
- org.quartz.xml
- org.terracotta.modules.ehcache.store
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.triggers.**AbstractTrigger**<T> (implements org.quartz.spi.OperableTrigger)
    - org.quartz.impl.triggers.**CalendarIntervalTriggerImpl** (implements org.quartz.CalendarIntervalTrigger, org.quartz.impl.triggers.**CoreTrigger**)
    - org.quartz.impl.triggers.**CronTriggerImpl** (implements org.quartz.impl.triggers.**CoreTrigger**, org.quartz.**CronTrigger**)
    - org.quartz.impl.triggers.**SimpleTriggerImpl** (implements org.quartz.impl.triggers.**CoreTrigger**, org.quartz.**SimpleTrigger**)
  - org.quartz.impl.matchers.**AndMatcher**<T> (implements org.quartz.**Matcher**<T>)
  - org.quartz.impl.jdbcjobstore.**AttributeRestoringConnectionInvocationHandler** (implements java.lang.reflect.**InvocationHandler**)
  - org.quartz.impl.calendar.**BaseCalendar** (implements org.quartz.**Calendar**, java.lang.**Cloneable**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**AnnualCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**CronCalendar**
    - org.quartz.impl.calendar.**DailyCalendar**
    - org.quartz.impl.calendar.**HolidayCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**MonthlyCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**WeeklyCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)
  - org.quartz.listeners.**BroadcastJobListener** (implements org.quartz.**JobListener**)
  - org.quartz.listeners.**BroadcastSchedulerListener** (implements org.quartz.**SchedulerListener**)
  - org.quartz.listeners.**BroadcastTriggerListener** (implements org.quartz.**TriggerListener**)
  - org.quartz.simpl.**CascadingClassLoadHelper** (implements org.quartz.spi.ClassLoadHelper)
  - org.quartz.utils.**CircularLossyQueue**<T>
org.quartz.utils. ClassUtils
org.quartz.utils.counter. CounterConfig
  org.quartz.utils.counter.sampled. SampledCounterConfig
    org.quartz.utils.counter.sampled. SampledRateCounterConfig
org.quartz.utils.counter. CounterImpl (implements org.quartz.utils.counter.sampled. SampledCounter)
  org.quartz.utils.counter.sampled. SampledRateCounterImpl (implements org.quartz.utils.counter.sampled. SampledRateCounter)
org.quartz.utils.counter. CounterManagerImpl (implements org.quartz.utils.counter. CounterManager)
org.quartz. CronExpression (implements java.lang. Cloneable, java.io. Serializable)
org.quartz.core.jmx. CronTriggerSupport
org.quartz. DateBuilder
org.quartz.utils. DBConnectionManager
  org.quartz.impl.jdbcjobstore. StdRowLockSemaphore
  org.quartz.impl.jdbcjobstore. UpdateLockRowSemaphore
org.quartz.jobs. DirectoryScanJob (implements org.quartz. Job)
org.quartz.impl. DirectSchedulerFactory (implements org.quartz. SchedulerFactory)
• org.quartz.utils.**DirtyFlagMap** (implements java.lang.**Cloneable**, java.util.**Map**<K,V>, java.io.**Serializable**)
  • org.quartz.utils.**StringKeyDirtyFlagMap**
  • org.quartz..**JobDataMap** (implements java.io.**Serializable**)
  • org.quartz..**SchedulerContext** (implements java.io.**Serializable**)
• org.quartz.locality.constraint.**EhcacheConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
• org.quartz.locality.constraint.**EhcacheConstraint.Value** (implements java.io.**Serializable**)
• org.terracotta.modules.ehcache.store.**EhcacheEvaluator** (implements org.quartz.locality.constraint.evaluator.**Evaluator**<T>)
• org.quartz.jobs.ee.ejb.**EJBInvokerJob** (implements org.quartz..**Job**)
• org.quartz.impl.matchers.**EverythingMatcher**<T> (implements org.quartz..**Matcher**<T>)
• org.quartz.jobs.**FileSyncJob** (implements org.quartz..**Job**)
• org.quartz.impl.jdbcjobstore.**FiredTriggerRecord** (implements java.io.**Serializable**)
• javax.servlet.GenericServlet (implements java.io.**Serializable**, javax.servlet.Servlet, javax.servlet.ServletConfig)
  • javax.servlet.http.HttpServlet (implements java.io.**Serializable**)
  • org.quartz..**InitializerServlet**
• org.quartz.simpl.**HostnameInstanceIdGenerator** (implements org.quartz.spi.**InstanceIdGenerator**)
• org.quartz.simpl.**InitThreadContextClassLoaderHelper** (implements org.quartz.spi.**ClassLoaderHelper**)
• org.jboss.mx.util.JBossNotificationBroadcasterSupport (implements javax.management.**NotificationEmitter**)
  • org.jboss.system.ServiceMBeanSupport (implements javax.management.**MBeanRegistration**, org.jboss.system.ServiceMBean)
  • org.quartz.ee.jmx.jboss.**QuartzService** (implements org.quartz.ee.jmx.jboss.**QuartzServiceMBean**)
• org.quartz.jobs.ee.jms.**JmsHelper**
• org.quartz.jobs.ee.jmx.**JMXInvokerJob** (implements org.quartz..**Job**)
• org.quartz.utils.**JNDIConnectionProvider** (implements org.quartz.utils.**ConnectionProvider**)
• org.quartz..**JobBuilder**
• org.quartz.core.jmx.**JobDataMapSupport**
- `org.quartz.core.jmx.JobDetailSupport`
- `org.quartz.core.jmx.JobExecutionContextSupport`
  - `org.quartz.listeners.JobChainingJobListener`
  - `org.quartz.impl.jdbcjobstore.JobStoreCMT`
  - `org.quartz.impl.jdbcjobstore.JobStoreTX`
- `org.quartz.ee.jta.JTAAnnotationAwareJobRunShellFactory` (implements `org.quartz.core.JobRunShellFactory`)
- `org.quartz.ee.jta.JTAJobRunShellFactory` (implements `org.quartz.core.JobRunShellFactory`)
- `org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore` (implements `org.quartz.impl.jdbcjobstore.Semaphore`)
- `org.quartz.utils.Key<T>` (implements `java.lang.Comparable<T>`, `java.io.Serializable`)
  - `org.quartz.JobKey`
  - `org.quartz.TriggerKey`
- `org.quartz.impl.matchers.KeyMatcher<T>` (implements `org.quartz.Matcher<T>`)
- `org.quartz.core.ListenerManagerImpl` (implements `org.quartz.ListenerManager`)
- `org.quartz.simpl.LoadingLoaderClassLoadHelper` (implements `org.quartz.spi.ClassLoadHelper`)
- `org.quartz.locality.LocalityJobBuilder`
- `org.quartz.locality.LocalityTriggerBuilder`
- `org.quartz.plugins.history.LoggingTriggerHistoryPlugin` (implements `org.quartz.spi.SchedulerPlugin`, `org.quartz.TriggerListener`)
- org.quartz.locality.constraint.**NodeGroupConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
- org.quartz.locality.constraint.evaluator.**NodeGroupEvaluator** (implements org.quartz.locality.constraint.evaluator.**Evaluator**<T>)
- org.quartz.locality.**NodeSpecBuilder**
- org.quartz.jobs.**NoOpJob** (implements org.quartz.**Job**)
- org.quartz.impl.matchers.**NotMatcher**<T> (implements org.quartz.**Matcher**<T>)
- org.quartz.core.**NullSampledStatisticsImpl** (implements org.quartz.core.**SampledStatistics**)
- org.quartz.impl.matchers.**OrMatcher**<T> (implements org.quartz.**Matcher**<T>)
- org.quartz.locality.constraint.**OsConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
- org.quartz.locality.constraint.evaluator.**OsEvaluator** (implements org.quartz.locality.constraint.evaluator.**PersistentEvaluator**<T,V>)
- org.quartz.utils.**PoolingConnectionProvider** (implements org.quartz.utils.**ConnectionProvider**)
- org.quartz.utils.**PropertiesParser**
- org.quartz.ee.servlet.**QuartzInitializerListener** (implements javax.servlet.ServletContextListener)
- org.quartz.core.**QuartzScheduler** (implements org.quartz.core.**RemotableQuartzScheduler**)
- org.quartz.core.**QuartzSchedulerResources**
- org.quartz.simpl.**RAMJobStore** (implements org.quartz.spi.JobStore)
- org.quartz.impl.**RemoteMBeanScheduler** (implements org.quartz.**Scheduler**)
  - org.quartz.ee.jmx.jboss.**JBoss4RMIRemoteMBeanScheduler**
- org.quartz.impl.**RemoteScheduler** (implements org.quartz.**Scheduler**)
- org.quartz.**ScheduleBuilder**<T>
  - org.quartz.**CalendarIntervalScheduleBuilder**
  - org.quartz.**CronScheduleBuilder**
  - org.quartz.**SimpleScheduleBuilder**
- org.quartz.listeners.**SchedulerListenerSupport** (implements org.quartz.**SchedulerListener**)
  - org.quartz.core.**JobRunShell** (implements java.lang.**Runnable**)
    - org.quartz.ee.jta.**JTAJobRunShell**
  - org.quartz.impl.**QuartzServer**
  - org.quartz.core.**SampledStatisticsImpl** (implements
- org.quartz.SchedulerMetaData (implements java.io.Serializable)
  - org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin (implements org.quartz.jobs.FileScanListener)
- org.quartz.impl.SchedulerRepository
- org.quartz.core.SchedulerSignalerImpl (implements org.quartz.spi.SchedulerSignaler)
- org.quartz.impl.jdbcjobstore.SchedulerStateRecord (implements java.io.Serializable)
- org.quartz.jobs.ee.mail.SendMailJob (implements org.quartz.Job)
- org.quartz.jobs.ee.mail.SendMailJob.MailInfo
- org.quartz.plugins.management.ShutdownHookPlugin (implements org.quartz.spi.SchedulerPlugin)
- org.quartz.simpl.SimpleClassLoadHelper (implements org.quartz.spi.ClassLoadHelper)
- org.quartz.simpl.SimpleInstanceIdGenerator (implements org.quartz.spi.InstanceIdGenerator)
  - org.quartz.simpl.PropertySettingJobFactory
  - org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate
- org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties
- org.quartz.impl.jdbcjobstore.SimpleSemaphore (implements org.quartz.impl.jdbcjobstore.Semaphore)
- org.quartz.simpl.SimpleThreadPool (implements org.quartz.spi.ThreadPool)
- org.quartz.simpl.SimpleTimeBroker (implements
org.quartz.spi.TimeBroker)

- org.quartz.core.jmx.SimpleTriggerSupport
- javax.management.StandardMBean (implements javax.management.DynamicMBean, javax.management.MBeanRegistration)
  - org.quartz.impl.jdbcjobstore.CloudscapeDelegate
  - org.quartz.impl.jdbcjobstore.DB2v6Delegate
  - org.quartz.impl.jdbcjobstore.DB2v7Delegate
  - org.quartz.impl.jdbcjobstore.DB2v8Delegate
  - org.quartz.impl.jdbcjobstore.HSQLDBDelegate
  - org.quartz.impl.jdbcjobstore.MSSQLDelegate
  - org.quartz.impl.jdbcjobstore.PointbaseDelegate
  - org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
  - org.quartz.impl.jdbcjobstore.SysbaseDelegate
  - org.quartz.impl.jdbcjobstore.WebLogicDelegate
- org.quartz.impl.StdJobRunShellFactory (implements org.quartz.core.JobRunShellFactory)
- org.quartz.impl.StdScheduler (implements org.quartz.Scheduler)
- org.quartz.impl.StdSchedulerFactory (implements org.quartz.SchedulerFactory)
- org.quartz.impl.matchers.StringMatcher<T> (implements org.quartz.Matcher<T>)
  - org.quartz.impl.matchers.GroupMatcher<T>
  - org.quartz.impl.matchers.NameMatcher<T>
- org.quartz.simpl.SystemPropertyInstanceIdGenerator (implements org.quartz.spi.InstanceIdGenerator)
- java.lang.Thread (implements java.langRunnable)
- org.quartz.core.QuartzSchedulerThread
- org.quartz.simpl.ThreadContextClassLoader (implements org.quartz.spi.ClassLoaderHelper)
- java.lang Throwable (implements java.io.Serializable)
  - java.lang Exception
    - org.quartz.impl.jdbcjobstore.InvalidConfigurationException
    - org.quartz.SchedulerException
    - org.quartz.jobs.ee.jms.JmsJobException
    - org.quartz.JobExecutionException
    - org.quartz.JobPersistenceException
    - org.quartz.locality.LocalityException
    - org.quartz.impl.jdbcjobstore.LockException
    - org.quartz.impl.jdbcjobstore.NoSuchDelegateException
    - org.quartz.ObjectAlreadyExistsException
    - org.quartz.SchedulerConfigException
    - org.quartz.UnableToInterruptJobException
    - org.quartz.xml.ValidationException
- java.util.TimerTask (implements java.lang.Runnable)
  - org.quartz.locality.constraint.evaluator.EvaluatorTimerTask
  - org.quartz.locality.constraint.evaluator.CpuEvaluator (implements org.quartz.locality.constraint.evaluator.PersistentEvaluator<T>
  - org.quartz.locality.constraint.evaluator.MemoryEvaluator (implements org.quartz.locality.constraint.evaluator.PersistentEvaluator<T>
  - org.quartz.utils.UpdateChecker
  - org.quartz.utils.counter.sampled.TimeStampedCounterValue (implements java.io.Serializable)
  - org.quartz.TriggerBuilder<T>
  - org.quartz.listeners.TriggerListenerSupport (implements org.quartz.TriggerListener)
  - org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerProp
  - org.quartz.impl.jdbcjobstore.TriggerStatus
  - org.quartz.core.jmx.TriggerSupport
  - org.quartz.TriggerUtils
  - org.quartz.ee.jta.UserTransactionHelper
  - org.quartz.impl.jdbcjobstore.Util
- org.quartz.helpers.**VersionPrinter**
- org.quartz.utils.weblogic.**WeblogicConnectionProvider** (implements org.quartz.utils.**ConnectionProvider**)
- org.quartz.xml.**XMLSchedulingDataProcessor** (implements org.xml.sax.**ErrorHandler**)
- org.quartz.simpl.**ZeroSizeThreadPool** (implements org.quartz.spi.ThreadPool)
Interface Hierarchy

- java.lang.**Cloneable**
  - org.quartz.**Calendar** (also extends java.io.**Serializable**)
  - org.quartz.**CalendarIntervalTrigger**
  - org.quartz.impl.triggers.**CoreTrigger**
  - org.quartz.**CronTrigger**
  - org.quartz.**JobDetail** (also extends java.io.**Serializable**)
    - org.quartz.locality.**LocalityJobDetail** (also extends org.quartz.locality.**LocalityAware**)
  - org.quartz.locality.**LocalityJobDetail** (also extends org.quartz.**JobDetail**, org.quartz.locality.**LocalityAware**)
  - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.locality.**LocalityAware**, org.quartz.**Trigger**)
  - org.quartz.**SimpleTrigger**
  - org.quartz.**Trigger** (also extends java.lang.**Comparable**<T>, java.io.**Serializable**)
    - org.quartz.**CalendarIntervalTrigger**
    - org.quartz.impl.triggers.**CoreTrigger**
    - org.quartz.**CronTrigger**
    - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.locality.**LocalityAware**)
    - org.quartz.**SimpleTrigger**
  - java.lang.**Comparable**<T>
    - org.quartz.**CalendarIntervalTrigger**
    - org.quartz.impl.triggers.**CoreTrigger**
    - org.quartz.**CronTrigger**
    - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.locality.**LocalityAware**, org.quartz.**Trigger**)
    - org.quartz.**SimpleTrigger**
    - org.quartz.**Trigger** (also extends java.lang.**Cloneable**, java.io.**Serializable**)
      - org.quartz.**CalendarIntervalTrigger**
      - org.quartz.impl.triggers.**CoreTrigger**
      - org.quartz.**CronTrigger**
      - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.locality.**LocalityAware**)
- org.quartz.**SimpleTrigger**
- org.quartz.utils.**ConnectionProvider**
- org.quartz.impl.jdcjobstore.**Constants**
  - org.quartz.impl.jdcjobstore.**StdJDBCConstants**
- org.quartz.utils.counter.**Counter**
  - org.quartz.utils.counter.sampled.**SampledCounter**
    - org.quartz.utils.counter.sampled.**SampledRateCounter**
  - org.quartz.utils.counter.sampled.**SampledRateCounter**
- org.quartz.utils.counter.**CounterManager**
- org.quartz.jobs.**DirectoryScanListener**
- org.quartz.impl.jdcjobstore.**DriverDelegate**
- org.quartz.locality.constraint.evaluator.**Evaluator**<T>
  - org.quartz.locality.constraint.evaluator.**PersistentEvaluator**<T,V>
- org.quartz.jobs.**FileScanListener**
- org.quartz.jobs.ee.jms.**JmsMessageFactory**
- org.quartz.**Job**
  - org.quartz.**InterruptableJob**
  - org.quartz.**StatefulJob**
- org.quartz.**JobExecutionContext**
- org.quartz.**JobListener**
- org.quartz.core.**JobRunShellFactory**
- org.quartz.impl.jdcjobstore.**JobStoreSupport.TransactionCallback**
- org.quartz.impl.jdcjobstore.**JobStoreSupport.VoidTransactionCallback**
- org.quartz.**ListenerManager**
- org.quartz.locality.**LocalityAware**
  - org.quartz.locality.**LocalityJobDetail** (also extends org.quartz.**JobDetail**)
  - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.**Trigger**)
- org.quartz.locality.**NodeSpec**
- org.quartz.core.jmx.**QuartzSchedulerMBean**
- java.rmi.**Remote**
  - org.quartz.core.**RemotableQuartzScheduler**
- org.quartz.core.**SampledStatistics**
- org.quartz.**Scheduler**
- org.quartz.**SchedulerFactory**
- org.quartz.**SchedulerListener**
- org.quartz.impl.jdcjobstore.**Semaphore**
- java.io.**Serializable**
  - org.quartz.**Calendar** (also extends java.lang.**Cloneable**

- `org.quartz.CalendarIntervalTrigger`
- `org.quartz.locality.constraint.Constraint<OP,T>`
- `org.quartz.impl.triggers.CoreTrigger`
- `org.quartz.CronTrigger`
- `org.quartz.JobDetail` (also extends java.lang.Cloneable)
  - `org.quartz.locality.LocalityJobDetail` (also extends `org.quartz.locality.LocalityAware`)
- `org.quartz.locality.LocalityJobDetail` (also extends `org.quartz.JobDetail`, `org.quartz.locality.LocalityAware`)
- `org.quartz.locality.LocalityTrigger` (also extends `org.quartz.locality.LocalityAware`, `org.quartz.Trigger`)
- `org.quartz.Matcher<T>`
- `org.quartz.SimpleTrigger`
- `org.quartz.Trigger` (also extends java.lang.Cloneable, java.lang.Comparable<T>)
  - `org.quartz.CalendarIntervalTrigger`
  - `org.quartz.impl.triggers.CoreTrigger`
  - `org.quartz.CronTrigger`
  - `org.quartz.locality.LocalityTrigger` (also extends `org.quartz.locality.LocalityAware`)
  - `org.quartz.SimpleTrigger`
- `org.jboss.system.Service`
  - `org.quartz.ee.jmx.jboss.QuartzServiceMBean`
- `org.jboss.system.ServiceMBean`
  - `org.quartz.ee.jmx.jboss.QuartzServiceMBean`
- `org.quartz.impl.jdbcjobstore.TablePrefixAware`
- `org.quartz.TriggerListener`
- `org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate`
Annotation Type Hierarchy

- org.quartz.\texttt{PersistJobDataAfterExecution} (implements java.lang.annotation.\texttt{Annotation})
- org.quartz.\texttt{DisallowConcurrentExecution} (implements java.lang.annotation.\texttt{Annotation})
- org.quartz.\texttt{ExecuteInJTATransaction} (implements java.lang.annotation.\texttt{Annotation})
Enum Hierarchy

- java.lang.**Object**
  - java.lang.**Enum**<E> (implements java.lang.**Comparable**<T>, java.io.**Serializable**)
    - org.quartz.locality.constraint.**NodeGroupConstraint.Operator**
    - org.quartz.locality.constraint.**EhcacheConstraint.Operator**
    - org.quartz.locality.constraint.**MemoryConstraint.Operator**
    - org.quartz.locality.constraint.**MemoryConstraint.Unit**
    - org.quartz.locality.constraint.**OsConstraint.OS**
    - org.quartz.locality.constraint.**OsConstraint.Operator**
    - org.quartz.locality.constraint.evaluator.**EvaluatorTimerTask.Status**
    - org.quartz.impl.matchers.**CpuConstraint.Operator**
    - org.quartz.impl.matchers.**StringMatcher.StringOperatorName**
    - org.quartz.**Trigger.TriggerState**
    - org.quartz.**Trigger.CompletedExecutionInstruction**
    - org.quartz.**DateBuilder.IntervalUnit**

---

**Overview** Package Class Use Deprecated Index Help

Copyright 2001-2011, Terracotta, Inc.
Serialized Form

Package org.quartz

Class org.quartz.CronExpression extends Object
implements Serializable

serialVersionUID: 12423409423L

Serialization Methods

readObject

private void readObject(ObjectInputStream stream)
    throws IOException, ClassNotFoundException

    Throws:
        IOException
        ClassNotFoundException

Serialized Fields

cronExpression

String cronExpression

timeZone

TimeZone timeZone

Class org.quartz.DateIntervalTrigger extends AbstractTrigger
implements Serializable
serialVersionUID: -2635982274232850343L

### Serialized Fields

**startTime**

*Date* startTime

**endTime**

*Date* endTime

**nextFireTime**

*Date* nextFireTime

**previousFireTime**

*Date* previousFireTime

**repeatInterval**

*int* repeatInterval

**repeatIntervalUnit**

*org.quartz.DateIntervalTrigger.IntervalUnit* repeatIntervalUnit

**timesTriggered**

*int* timesTriggered
complete

boolean complete

<table>
<thead>
<tr>
<th>Class org.quartz.JobDataMap extends StringKeyDirtyFlagMap implements Serializable</th>
</tr>
</thead>
</table>

serialVersionUID: -6939901990106713909L

<table>
<thead>
<tr>
<th>Class org.quartz.JobExecutionException extends SchedulerException implements Serializable</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Serialized Fields</th>
</tr>
</thead>
</table>

refire

boolean refire

unscheduleTrigg

boolean unscheduleTrigg

unscheduleAllTriggs

boolean unscheduleAllTriggs

<table>
<thead>
<tr>
<th>Class org.quartz.JobKey extends Key&lt;JobKey&gt; implements Serializable</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Class org.quartz.JobPersistenceException</th>
</tr>
</thead>
</table>
SchedulerException implements Serializable

Class org.quartz.NthIncludedDayTrigger extends AbstractTrigger implements Serializable

serialVersionUID: 6267700049629328293L

Serialized Fields

startTime

Date startTime

endTime

Date endTime

previousFireTime

Date previousFireTime

nextFireTime

Date nextFireTime

calendar

Calendar calendar

n
int n

intervalType
int intervalType

fireAtHour
int fireAtHour

fireAtMinute
int fireAtMinute

fireAtSecond
int fireAtSecond

nextFireCutoffInterval
int nextFireCutoffInterval

timeZone
TimeZone timeZone

Class org.quartz.ObjectAlreadyExistsException
extends JobPersistenceException implements Serializable

Class org.quartz.SchedulerConfigException extends
SchedulerException implements Serializable

Class org.quartz.SchedulerContext extends StringKeyDirtyFlagMap implements Serializable

Class org.quartz.SchedulerException extends Exception implements Serializable

Class org.quartz.SchedulerMetaData extends Object implements Serializable

Serialized Fields

schedName

String schedName

schedInst

String schedInst

schedClass

Class<T> schedClass

isRemote

boolean isRemote

started
boolean started

isInStandbyMode

boolean isInStandbyMode

shutdown

boolean shutdown

startTime

Date startTime

numJobsExec

int numJobsExec

jsClass

Class<T> jsClass

jsPersistent

boolean jsPersistent

jsClustered

boolean jsClustered

tpClass
Class `<T> tpClass

```
tpSize
``` int tpSize

```
version
``` String version

```
Class org.quartz.Trigger.TriggerTimeComparator extends Object implements Serializable
```

```
Class org.quartz.TriggerKey extends Key<TriggerKey> implements Serializable
```

```
Class org.quartz.UnableToInterruptJobException extends SchedulerException implements Serializable
```

```
Package org.quartz.core
```

```
Class org.quartz.core.QuartzScheduler_Stub extends RemoteStub implements Serializable
``` serialVersionUID: 2L

```
Package org.quartz.ee.servlet
```

```
Class org.quartz.ee.servlet.QuartzInitializerServlet
```
extends javax.servlet.http.HttpServlet
implements Serializable

serialVersionUID: 1L

Serialized Fields

performShutdown

boolean performShutdown

waitOnShutdown

boolean waitOnShutdown

Package org.quartz.impl

Class org.quartz.impl.JobDetailImpl extends Object
implements Serializable

Serialized Fields

name

String name

group

String group

description
String description

jobClass

Class&lt;T&gt; jobClass

jobDataMap

JobDataMap jobDataMap

durability

boolean durability

shouldRecover

boolean shouldRecover

Class org.quartz.impl.JobExecutionContextImpl
extends Object implements Serializable

Serialized Fields

trigger

Trigger trigger

jobDetail

JobDetail jobDetail
jobDataMap

JobDataMap jobDataMap

calendar

Calendar calendar

recovering

boolean recovering

numRefires

int numRefires

fireTime

Date fireTime

scheduledFireTime

Date scheduledFireTime

prevFireTime

Date prevFireTime

nextFireTime

Date nextFireTime
jobRunTime

long jobRunTime

result

Object result

data

HashMap<K, V> data

Package org.quartz.impl.calendar

Class org.quartz.impl.calendar.AnnualCalendar extends BaseCalendar implements Serializable

serialVersionUID: 7346867105876610961L

Serialized Fields

excludeDays

ArrayList<E> excludeDays

dataSorted

boolean dataSorted

Class org.quartz.impl.calendar.BaseCalendar extends Object implements Serializable
**Serializable Fields**

**baseCalendar**

`Calendar baseCalendar`

**description**

`String description`

**timeZone**

`TimeZone timeZone`

---

**Class** `org.quartz.impl.calendar.CronCalendar`

Extends `BaseCalendar` implements `Serializable`  

**serialVersionUID:** `-8172103999750856831L`

**Serialized Fields**

**cronExpression**

`CronExpression cronExpression`

---

**Class** `org.quartz.impl.calendar.DailyCalendar`

Extends `BaseCalendar` implements `Serializable`  

**serialVersionUID:** `-7561220099904944039L`

**Serialized Fields**
rangeStartingHourOfDay
int rangeStartingHourOfDay

rangeStartingMinute
int rangeStartingMinute

rangeStartingSecond
int rangeStartingSecond

rangeStartingMillis
int rangeStartingMillis

rangeEndingHourOfDay
int rangeEndingHourOfDay

rangeEndingMinute
int rangeEndingMinute

rangeEndingSecond
int rangeEndingSecond

rangeEndingMillis
int rangeEndingMillis
invertTimeRange

boolean invertTimeRange

Class org.quartz.impl.calendar.HolidayCalendar extends BaseCalendar implements Serializable

serialVersionUID: -7590908752291814693L

Serialized Fields

dates

TreeSet<E> dates

Class org.quartz.impl.calendar.MonthlyCalendar extends BaseCalendar implements Serializable

serialVersionUID: 419164961091807944L

Serialized Fields

excludeDays

boolean[] excludeDays

excludeAll

boolean excludeAll

Class org.quartz.impl.calendar.WeeklyCalendar extends BaseCalendar implements Serializable
serialVersionUID: -6809298821229007586L

### Serialized Fields

**excludeDays**

boolean[] excludeDays

**excludeAll**

boolean excludeAll

---

#### Package org.quartz.impl.jdbcjobstore

#### Class

org.quartz.impl.jdbcjobstore.FiredTriggerRecord

extends Object

implements Serializable

### Serialized Fields

**fireInstanceId**

String fireInstanceId

**fireTimestamp**

long fireTimestamp

**schedulerInstanceId**

String schedulerInstanceId
triggerKey

triggerKey

fireInstanceState

fireInstanceState

jobKey

jobKey

jobDisallowsConcurrentExecution

jobDisallowsConcurrentExecution

jobRequestsRecovery

jobRequestsRecovery

priority

priority

Class

org.quartz.impl.jdbcjobstore.InvalidConfigurationException

extends Exception implements Serializable

Class org.quartz.impl.jdbcjobstore.LockException

extends JobPersistenceException implements
### Serializable

Class

`org.quartz.impl.jdbcjobstore.NoSuchDelegateException`

extends `JobPersistenceException`

implements `Serializable`

Class

`org.quartz.impl.jdbcjobstore.SchedulerStateRecord`

extends `Object`

implements `Serializable`

**Serialized Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedulerInstanceId</td>
<td>String</td>
</tr>
<tr>
<td>checkinTimestamp</td>
<td>long</td>
</tr>
<tr>
<td>checkinInterval</td>
<td>long</td>
</tr>
</tbody>
</table>

Package `org.quartz.impl.matchers`

Class `org.quartz.impl.matchers.AndMatcher`

extends `Object`

implements `Serializable`
Serialized Fields

leftOperand

Matcher\langle T \rangle extends Key > leftOperand

rightOperand

Matcher\langle T \rangle extends Key > rightOperand

Class org.quartz.impl.matchers.EverythingMatcher
extends Object implements Serializable

Class org.quartz.impl.matchers.GroupMatcher
extends StringMatcher\langle T \rangle extends Key > implements Serializable

Class org.quartz.impl.matchers.KeyMatcher
extends Object implements Serializable

Serialized Fields

c ompareTo

Key\langle T \rangle compareTo

Class org.quartz.impl.matchers.NameMatcher
extends StringMatcher\langle T \rangle extends Key > implements Serializable
Class `org.quartz.impl.matchers.NotMatcher` extends `Object` implements `Serializable`

**Serialized Fields**

`operand`

`Matcher<T extends Key> operand`

Class `org.quartz.impl.matchers.OrMatcher` extends `Object` implements `Serializable`

**Serialized Fields**

`leftOperand`

`Matcher<T extends Key> leftOperand`

`rightOperand`

`Matcher<T extends Key> rightOperand`

Class `org.quartz.impl.matchers.StringMatcher` extends `Object` implements `Serializable`

**Serialized Fields**

`compareTo`

`String compareTo`
compareWith

`StringMatcher.StringOperatorName` compareWith

---

**Package org.quartz.impl.triggers**

**Class** `org.quartz.impl.triggers.AbstractTrigger` extends `Object` implements `Serializable`

**serialVersionUID:** -3904243490805975570L

---

**Serialized Fields**

- **name**
  ```
  String name
  ```

- **group**
  ```
  String group
  ```

- **jobName**
  ```
  String jobName
  ```

- **jobGroup**
  ```
  String jobGroup
  ```

- **description**
  ```
  String description
  ```
jobDataMap

boolean volatility

calendarName

String fireInstanceId

int misfireInstruction

int priority

Class

org.quartz.impl.triggers.CalendarIntervalTriggerImpl extends AbstractTrigger implements Serializable

serialVersionUID: -2635982274232850343L

Serialized Fields
startTime

Date startTime

endTime

Date endTime

nextFireTime

Date nextFireTime

previousFireTime

Date previousFireTime

repeatInterval

int repeatInterval

repeatIntervalUnit

DateBuilder.IntervalUnit repeatIntervalUnit

timesTriggered

int timesTriggered

complete
boolean complete

Class org.quartz.impl.triggers.CronTriggerImpl extends AbstractTrigger<CronTrigger> implements Serializable

serialVersionUID: -8644953146451592766L

Serialized Fields

cronEx

CronExpression cronEx

startTime

Date startTime

endTime

Date endTime

nextFireTime

Date nextFireTime

previousFireTime

Date previousFireTime

Class org.quartz.impl.triggers.SimpleTriggerImpl extends AbstractTrigger<SimpleTrigger>
implements Serializable

serialVersionUID: -3735980074222850397L

| Serialized Fields |

startTime

Date startTime

endTime

Date endTime

nextFireTime

Date nextFireTime

previousFireTime

Date previousFireTime

repeatCount

int repeatCount

repeatInterval

long repeatInterval

timesTriggered
int timesTriggered

complete

boolean complete

Package org.quartz.jobs.ee.jms

Class org.quartz.jobs.ee.jms.JmsJobException
extends SchedulerException implements Serializable

serialVersionUID: 3045647075496522093L

Package org.quartz.locality

Class org.quartz.locality.DelegatingLocalityJobDetail
extends Object implements Serializable

Serialized Fields

jobDetail

JobDetail jobDetail

nodeSpec

NodeSpec nodeSpec

Class org.quartz.locality.DelegatingLocalityTrigger
extends Object implements Serializable
Serialized Fields

trigger

org.quartz.spi.OperableTrigger trigger

nodeSpec

NodeSpec nodeSpec

Class org.quartz.locality.LocalityException extends JobPersistenceException implements Serializable

Serialized Fields

constraint

Constraint<OP extends Enum,I> constraint

Package org.quartz.locality.constraint

Class org.quartz.locality.constraint.CpuConstraint extends Object implements Serializable

Serialized Fields

op

CpuConstraint.Operator op

value
int value

Class
org.quartz.locality.constraint.EhcacheConstraint
extends Object implements Serializable

Serialized Fields
operator

EhcacheConstraint.Operator operator

value

EhcacheConstraint.Value value

Class
org.quartz.locality.constraint.EhcacheConstraint.Value
extends Object implements Serializable

Serialized Fields
keys

Set\langle E\rangle keys

mgrName

String mgrName

cache
Class

```
org.quartz.locality.constraint.MemoryConstraint
```

extends `Object`

implements `Serializable`

Serialized Fields

- `op`
  - `MemoryConstraint.Operator`  `op`

- `value`
  - `long`  `value`

Class

```
org.quartz.locality.constraint.NodeGroupConstraint
```

extends `Object`

implements `Serializable`

Serialized Fields

- `operator`
  - `NodeGroupConstraint.Operator`  `operator`

- `targetNodeGroup`
  - `String`  `targetNodeGroup`

Class

```
org.quartz.locality.constraint.OsConstraint
```
<table>
<thead>
<tr>
<th>extends <strong>Object</strong> implements Serializable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Serialized Fields</strong></td>
</tr>
<tr>
<td>operator</td>
</tr>
<tr>
<td><strong>OsConstraint.Operator</strong> operator</td>
</tr>
<tr>
<td>targetedOs</td>
</tr>
<tr>
<td><strong>OsConstraint.OS</strong> targetedOs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Package <strong>org.quartz.utils</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong> <strong>org.quartz.utils.DirtyFlagMap</strong> extends <strong>Object</strong> implements Serializable</td>
</tr>
<tr>
<td><strong>serialVersionUID</strong>: 1433884852607126222L</td>
</tr>
<tr>
<td><strong>Serialized Fields</strong></td>
</tr>
<tr>
<td>dirty</td>
</tr>
<tr>
<td>boolean dirty</td>
</tr>
<tr>
<td>map</td>
</tr>
<tr>
<td><strong>Map&lt;K, V&gt;</strong> map</td>
</tr>
<tr>
<td><strong>Class</strong> <strong>org.quartz.utils.Key</strong> extends <strong>Object</strong> implements Serializable</td>
</tr>
</tbody>
</table>
Serialized Fields

name

String name

group

String group

Class **org.quartz.utils.StringKeyDirtyFlagMap** extends **DirtyFlagMap** implements **Serializable**

**serialVersionUID**: -9076749120524952280L

Serialized Fields

allowsTransientData

boolean allowsTransientData

**Deprecated.** JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

Package **org.quartz.utils.counter**

Class **org.quartz.utils.counter.CounterImpl** extends **Object** implements **Serializable**

Serialized Fields
value

**AtomicLong** value

---

**Package org.quartz.utils.counter.sampled**

**Class**

`org.quartz.utils.counter.sampled.SampledCounterImpl` extends `CounterImpl` implements `Serializable`

---

**Serialized Fields**

**history**

`CircularLossyQueue<T> history`

The history of this counter

---

**resetOnSample**

`boolean resetOnSample`

Should the counter reset on each sample?

---

**samplerTask**

`TimerTask samplerTask`

---

**intervalMillis**

`long intervalMillis`
Class 
`org.quartz.utils.counter.sampled.SampledRateCounterImpl` extends `SampledCounterImpl` implements Serializable

Serialized Fields

`numeratorValue`  long numeratorValue

`denominatorValue`  long denominatorValue

Class 
`org.quartz.utils.counter.sampled.TimeStampedCounterValue` extends `Object` implements Serializable

Serialized Fields

`counterValue`  long counterValue

`timestamp`  long timestamp

Package org.quartz.xml
Class **org.quartz.xml.ValidationException** extends **Exception** implements **Serializable**

**Serialized Fields**

**validationExceptions**

`Collection<E> validationExceptions`
org.quartz Interface Calendar

All Superinterfaces:
  Cloneable, Serializable

All Known Implementing Classes:
  AnnualCalendar, BaseCalendar, CronCalendar, DailyCalendar,
  HolidayCalendar, MonthlyCalendar, WeeklyCalendar

public interface Calendar
  extends Serializable, Cloneable

An interface to be implemented by objects that define spaces of time during
which an associated Trigger may (not) fire. Calendars do not define actual fire
times, but rather are used to limit a Trigger from firing on its normal schedule if
necessary. Most Calendars include all times by default and allow the user to
specify times to exclude.

As such, it is often useful to think of Calendars as being used to exclude a block
of time - as opposed to include a block of time. (i.e. the schedule "fire every five
minutes except on Sundays" could be implemented with a SimpleTrigger and a
WeeklyCalendar which excludes Sundays)

Implementations MUST take care of being properly cloneable and
Serializable.

Author:
  James House, Juergen Donnerstag

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int MONTH</td>
</tr>
</tbody>
</table>

| Method Summary |
Object clone()

Calendar getBaseCalendar()
    Get the base calendar.

String getDescription()
    Return the description given to the Calendar instance by its creator (if any).

long getNextIncludedTime(long timeStamp)
    Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.

boolean isTimeIncluded(long timeStamp)
    Determine whether the given time (in milliseconds) is 'included' by the Calendar.

void setBaseCalendar(Calendar baseCalendar)
    Set a new base calendar or remove the existing one.

void setDescription(String description)
    Set a description for the Calendar instance - may be useful for remembering/displaying the purpose of the calendar, though the description has no meaning to Quartz.

Field Detail

MONTH

static final int MONTH

See Also:
    Constant Field Values

Method Detail

setBaseCalendar

void setBaseCalendar(Calendar baseCalendar)
Set a new base calendar or remove the existing one.

---

**getBaseCalendar**

*Calendar* `getBaseCalendar()`

Get the base calendar. Will be null, if not set.

---

**isTimeIncluded**

`boolean isTimeIncluded(long timeStamp)`

Determine whether the given time (in milliseconds) is 'included' by the Calendar.

---

**getNextIncludedTime**

`long getNextIncludedTime(long timeStamp)`

Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.

---

**getDescription**

*String* `getDescription()`

Return the description given to the Calendar instance by its creator (if any).

**Returns:**

null if no description was set.

---

**setDescription**

`void setDescription(String description)`
Set a description for the Calendar instance - may be useful for remembering/displaying the purpose of the calendar, though the description has no meaning to Quartz.

clone

Object clone()
public class CalendarIntervalScheduleBuilder
extends ScheduleBuilder<CalendarIntervalTrigger>

CalendarIntervalScheduleBuilder is a ScheduleBuilder that defines calendar time (day, week, month, year) interval-based schedules for Triggers.

Quartz provides a builder-style API for constructing scheduling-related entities via a Domain-Specific Language (DSL). The DSL can best be utilized through the usage of static imports of the methods on the classes TriggerBuilder, JobBuilder, DateBuilder, JobKey, TriggerKey and the various ScheduleBuilder implementations.

Client code can then use the DSL to write code such as this:

```java
JobDetail job = newJob(MyJob.class)
    .withIdentity("myJob")
    .build();

Trigger trigger = newTrigger()
    .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
    .withSchedule(simpleSchedule()
        .withIntervalInHours(1)
        .repeatForever())
    .startAt(futureDate(10, MINUTES))
    .build();

scheduler.scheduleJob(job, trigger);
```
See Also:
CalendarIntervalTrigger, CronScheduleBuilder, ScheduleBuilder, SimpleScheduleBuilder, TriggerBuilder

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz.spi.MutableTrigger</strong></td>
</tr>
<tr>
<td><strong>build()</strong></td>
</tr>
<tr>
<td><strong>static CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>calendarIntervalSchedule()</strong></td>
</tr>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>withInterval(int interval, DateBuilder.IntervalUnit unit)</strong></td>
</tr>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>withIntervalInDays(int intervalInDays)</strong></td>
</tr>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>withIntervalInHours(int intervalInHours)</strong></td>
</tr>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>withIntervalInMinutes(int intervalInMinutes)</strong></td>
</tr>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>withIntervalInMonths(int intervalInMonths)</strong></td>
</tr>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>withIntervalInSeconds(int intervalInSeconds)</strong></td>
</tr>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
</tr>
<tr>
<td><strong>withIntervalInWeeks(int intervalInWeeks)</strong></td>
</tr>
</tbody>
</table>
that the produced Trigger will repeat.

```java
CalendarIntervalScheduleBuilder
```

**withIntervalInYears** (int intervalInYear)
Specify an interval in the IntervalUnit.YEAR that the produced Trigger will repeat.

```java
CalendarIntervalScheduleBuilder
```

**withMisfireHandlingInstructionDoNothing**
If the Trigger misfires, use CalendarIntervalTrigger.MISFIRE_INSTRUCTION_DO_NOTHING instruction.

```java
CalendarIntervalScheduleBuilder
```

**withMisfireHandlingInstructionFireAndProceed**
If the Trigger misfires, use CalendarIntervalTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW instruction.

```java
CalendarIntervalScheduleBuilder
```

**withMisfireHandlingInstructionIgnoreMisfires**
If the Trigger misfires, use Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRES instruction.

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Method Detail

calendarIntervalSchedule

```java
public static CalendarIntervalScheduleBuilder calendarIntervalSchedu
```

Create a CalendarIntervalScheduleBuilder.
Returns:
the new CalendarIntervalScheduleBuilder

---

**build**

```java
public org.quartz.spi.MutableTrigger build()
```

Build the actual Trigger -- NOT intended to be invoked by end users but will rather be invoked by a TriggerBuilder which this ScheduleBuilder is given to.

**Specified by:**
build in class ScheduleBuilder<CalendarIntervalTrigger>

**See Also:**
`TriggerBuilder.withSchedule(ScheduleBuilder)`

---

**withInterval**
public CalendarIntervalScheduleBuilder withInterval(int interval, DateBuilder.IntervalUnit unit) {
    Specify the time unit and interval for the Trigger to be produced.

    Parameters:
    interval - the interval at which the trigger should repeat.
    unit - the time unit (IntervalUnit) of the interval.
    
    Returns:
    the updated CalendarIntervalScheduleBuilder

    See Also:
    CalendarIntervalTrigger.getRepeatInterval(),
    CalendarIntervalTrigger.getRepeatIntervalUnit()
}

withIntervalInSeconds

public CalendarIntervalScheduleBuilder withIntervalInSeconds(int intervalInSeconds) {
    Specify an interval in the IntervalUnit.SECOND that the produced Trigger will repeat at.

    Parameters:
    intervalInSeconds - the number of seconds at which the trigger should repeat.

    Returns:
withIntervalInMinutes

**withIntervalInMinutes**

```java
public CalendarIntervalScheduleBuilder withIntervalInMinutes(int intervalInMinutes)
```

Specify an interval in the IntervalUnit.MINUTE that the produced Trigger will repeat at.

**Parameters:**
- `intervalInMinutes` - the number of minutes at which the trigger should repeat.

**Returns:**
- the updated CalendarIntervalScheduleBuilder

**See Also:**
- `CalendarIntervalTrigger.getRepeatInterval()`,
- `CalendarIntervalTrigger.getRepeatIntervalUnit()`

---

withIntervalInHours

**withIntervalInHours**

```java
public CalendarIntervalScheduleBuilder withIntervalInHours(int intervalInHours)
```

Specify an interval in the IntervalUnit.HOUR that the produced Trigger will repeat at.

**Parameters:**
- `intervalInHours` - the number of hours at which the trigger should repeat.

**Returns:**
- the updated CalendarIntervalScheduleBuilder

**See Also:**
- `CalendarIntervalTrigger.getRepeatInterval()`,
- `CalendarIntervalTrigger.getRepeatIntervalUnit()`

---
public CalendarIntervalScheduleBuilder withIntervalInHours(int intervalInHours)

Specify an interval in the IntervalUnit.HOUR that the produced Trigger will repeat at.

Parameters:
intervalInHours - the number of hours at which the trigger should repeat.

Returns:
the updated CalendarIntervalScheduleBuilder

See Also:
CalendarIntervalTrigger.getRepeatInterval(),
CalendarIntervalTrigger.getRepeatIntervalUnit()

---

withIntervalInDays

public CalendarIntervalScheduleBuilder withIntervalInDays(int intervalInDays)

Specify an interval in the IntervalUnit.DAY that the produced Trigger will repeat at.

Parameters:
intervalInDays - the number of days at which the trigger should repeat.

Returns:
the updated CalendarIntervalScheduleBuilder

See Also:
CalendarIntervalTrigger.getRepeatInterval(),
withIntervalInWeeks

public CalendarIntervalScheduleBuilder withIntervalInWeeks(int intervalInWeeks)

Specify an interval in the IntervalUnit.WEEK that the produced Trigger will repeat at.

Parameters:
- intervalInWeeks - the number of weeks at which the trigger should repeat.

Returns:
- the updated CalendarIntervalScheduleBuilder

See Also:
- CalendarIntervalTrigger.getRepeatInterval(), CalendarIntervalTrigger.getRepeatIntervalUnit()
Specify an interval in the IntervalUnit.MONTH that the produced Trigger will repeat at.

Parameters:
   intervalInMonths - the number of months at which the trigger repeats.

Returns:
   the updated CalendarIntervalScheduleBuilder

See Also:
   CalendarIntervalTrigger.getRepeatInterval(),
   CalendarIntervalTrigger.getRepeatIntervalUnit()

withIntervalInYears

public CalendarIntervalScheduleBuilder withIntervalInYears(int intervalInYears)

Specify an interval in the IntervalUnit.YEAR that the produced Trigger will repeat at.

Parameters:
   intervalInYears - the number of years at which the trigger repeats.

Returns:
   the updated CalendarIntervalScheduleBuilder

See Also:
   CalendarIntervalTrigger.getRepeatInterval(),
   CalendarIntervalTrigger.getRepeatIntervalUnit()
withMisfireHandlingInstructionIgnoreMisfires

```java
public CalendarIntervalScheduleBuilder withMisfireHandlingInstructionIgnoreMisfires
```

If the Trigger misfires, use the

```
Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY
```

instruct.

**Returns:**
the updated CronScheduleBuilder

**See Also:**
```
Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY
```

withMisfireHandlingInstructionDoNothing

```java
public CalendarIntervalScheduleBuilder withMisfireHandlingInstructionDoNothing
```

If the Trigger misfires, use the

```
CalendarIntervalTrigger.MISFIRE_INSTRUCTION_DO NOTHING
```

instruct.
withMisfireHandlingInstructionFireAndProceed

public CalendarIntervalScheduleBuilder withMisfireHandlingInstructionFireAndProceed

If the Trigger misfires, use the
CalendarIntervalTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW instruction.

Returns:
the updated CalendarIntervalScheduleBuilder
See Also:
CalendarIntervalTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
org.quartz Interface CalendarIntervalTrigger

All Superinterfaces:
   Cloneable, Comparable<Trigger>, Serializable, Trigger

All Known Implementing Classes:
   CalendarIntervalTriggerImpl

public interface CalendarIntervalTrigger
extends Trigger

A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.

The trigger will fire every N (see #setRepeatInterval(int) ) units of calendar time (see #setRepeatIntervalUnit(IntervalUnit)) as specified in the trigger's definition. This trigger can achieve schedules that are not possible with SimpleTrigger (e.g because months are not a fixed number of seconds) or CronTrigger (e.g. because "every 5 months" is not an even divisor of 12).

If you use an interval unit of MONTH then care should be taken when setting a startTime value that is on a day near the end of the month. For example, if you choose a start time that occurs on January 31st, and have a trigger with unit MONTH and interval 1, then the next fire time will be February 28th, and the next time after that will be March 28th - and essentially each subsequent firing will occur on the 28th of the month, even if a 31st day exists. If you want a trigger that always fires on the last day of the month - regardless of the number of days in the month, you should use CronTrigger.

Author:
   James House

See Also:
   TriggerBuilder, CalendarIntervalScheduleBuilder, SimpleScheduleBuilder, CronScheduleBuilder
## Nested Class Summary

**Nested classes/interfaces inherited from interface org.quartz.Trigger**

- Trigger.CompletedExecutionInstruction
- Trigger.TriggerState
- Trigger.TriggerTimeComparator

## Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td><code>MISFIRE_INSTRUCTION_DO NOTHING</code></td>
<td>Instructs the Scheduler that upon a mis-fire situation, the CalendarIntervalTrigger wants to have its next-fire-time updated to the next time in the schedule after the current time (taking into account any associated calendar, but it does not want to be fired now.</td>
</tr>
<tr>
<td>static int</td>
<td><code>MISFIRE_INSTRUCTION_FIRE_ONCE_NOW</code></td>
<td>Instructs the Scheduler that upon a mis-fire situation, the CalendarIntervalTrigger wants to be fired now by Scheduler.</td>
</tr>
</tbody>
</table>

## Fields inherited from interface org.quartz.Trigger

- DEFAULT_PRIORITY, MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY, MISFIRE_INSTRUCTION_SMART_POLICY, serialVersionUID

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int <code>getRepeatInterval()</code></td>
<td>Get the the time interval that will be added to the DateIntervalTrigger's fire time (in the set repeat interval unit) in order to calculate the time of the next trigger repeat.</td>
</tr>
<tr>
<td>DateBuilder.IntervalUnit <code>getRepeatIntervalUnit()</code></td>
<td>Get the interval unit - the time unit on with the interval applies.</td>
</tr>
<tr>
<td>int <code>getTimesTriggered()</code></td>
<td>Get the number of times the DateIntervalTrigger has already fired.</td>
</tr>
</tbody>
</table>
getTriggerBuilder()  
Get a TriggerBuilder that is configured to produce a Trigger identical to this one.

Methods inherited from interface org.quartz.Trigger
compareTo, equals, getCalendarName, getDescription, getEndTime, getFinalFireTime, getFireTimeAfter, getJobDataMap, getJobKey, getKey, getMisfireInstruction, getNextFireTime, getPreviousFireTime, getPriority, getScheduleBuilder, getStartTime, mayFireAgain

Field Detail

MISFIRE_INSTRUCTION_FIRE_ONCE_NOW

static final int MISFIRE_INSTRUCTION_FIRE_ONCE_NOW

Instructs the Scheduler that upon a mis-fire situation, the CalendarIntervalTrigger wants to be fired now by Scheduler.

See Also:
Constant Field Values

MISFIRE_INSTRUCTION_DO NOTHING

static final int MISFIRE_INSTRUCTION_DO NOTHING

Instructs the Scheduler that upon a mis-fire situation, the CalendarIntervalTrigger wants to have its next-fire-time updated to the next time in the schedule after the current time (taking into account any associated Calendar, but it does not want to be fired now.

See Also:
Constant Field Values
getRepeatIntervalUnit

(DateBuilder.IntervalUnit) getRepeatIntervalUnit()

Get the interval unit - the time unit on with the interval applies.

getRepeatInterval

(int) getRepeatInterval()

Get the the time interval that will be added to the DateIntervalTrigger's fire time (in the set repeat interval unit) in order to calculate the time of the next trigger repeat.

getTimesTriggered

(int) getTimesTriggered()

Get the number of times the DateIntervalTrigger has already fired.

goingToTriggerBuilder

(TriggerBuilder<CalendarIntervalTrigger>) getTriggerBuilder()

Description copied from interface: Trigger
Get a TriggerBuilder that is configured to produce a Trigger identical to this one.

Specified by:
getTriggerBuilder in interface Trigger

See Also:
Trigger.getScheduleBuilder()
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
org.quartz Class CronExpression

java.lang.Object
   org.quartz.CronExpression

All Implemented Interfaces:
   Serializable, Cloneable

public class CronExpression
   extends Object
   implements Serializable, Cloneable

Provides a parser and evaluator for unix-like cron expressions. Cron expressions provide the ability to specify complex time combinations such as "At 8:00am every Monday through Friday" or "At 1:30am every last Friday of the month".

Cron expressions are comprised of 6 required fields and one optional field separated by white space. The fields respectively are described as follows:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Allowed Values</th>
<th>Allowed Special Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seconds</td>
<td>0-59</td>
<td>, - * /</td>
</tr>
<tr>
<td>Minutes</td>
<td>0-59</td>
<td>, - * /</td>
</tr>
<tr>
<td>Hours</td>
<td>0-23</td>
<td>, - * /</td>
</tr>
<tr>
<td>Day-of-month</td>
<td>1-31</td>
<td>, - * ? / L W</td>
</tr>
<tr>
<td>Month</td>
<td>1-12 or JAN-DEC</td>
<td>- * /</td>
</tr>
<tr>
<td>Day-of-Week</td>
<td>1-7 or SUN-SAT</td>
<td>, - * ? / L #</td>
</tr>
<tr>
<td>Year (Optional)</td>
<td>empty, 1970-2199</td>
<td>- * /</td>
</tr>
</tbody>
</table>

The '*' character is used to specify all values. For example, "*" in the minute field means "every minute".

The '?' character is allowed for the day-of-month and day-of-week fields. It is used to specify 'no specific value'. This is useful when you need to specify something in one of the two fields, but not the other.

The '-' character is used to specify ranges For example "10-12" in the hour field
means "the hours 10, 11 and 12".

The ',' character is used to specify additional values. For example "MON,WED,FRI" in the day-of-week field means "the days Monday, Wednesday, and Friday".

The '/' character is used to specify increments. For example "0/15" in the seconds field means "the seconds 0, 15, 30, and 45". And "5/15" in the seconds field means "the seconds 5, 20, 35, and 50". Specifying '*' before the '/' is equivalent to specifying 0 is the value to start with. Essentially, for each field in the expression, there is a set of numbers that can be turned on or off. For seconds and minutes, the numbers range from 0 to 59. For hours 0 to 23, for days of the month 0 to 31, and for months 1 to 12. The '/' character simply helps you turn on every "nth" value in the given set. Thus "7/6" in the month field only turns on month "7", it does NOT mean every 6th month, please note that subtlety.

The 'L' character is allowed for the day-of-month and day-of-week fields. This character is short-hand for "last", but it has different meaning in each of the two fields. For example, the value "L" in the day-of-month field means "the last day of the month" - day 31 for January, day 28 for February on non-leap years. If used in the day-of-week field by itself, it simply means "7" or "SAT". But if used in the day-of-week field after another value, it means "the last xxx day of the month" - for example "6L" means "the last friday of the month". You can also specify an offset from the last day of the month, such as "L-3" which would mean the third-to-last day of the calendar month. When using the 'L' option, it is important not to specify lists, or ranges of values, as you'll get confusing/unexpected results.

The 'W' character is allowed for the day-of-month field. This character is used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: "the nearest weekday to the 15th of the month". So if the 15th is a Saturday, the trigger will fire on Friday the 14th. If the 15th is a Sunday, the trigger will fire on Monday the 16th. If the 15th is a Tuesday, then it will fire on Tuesday the 15th. However if you specify "1W" as the value for day-of-month, and the 1st is a Saturday, the trigger will fire on Monday the 3rd, as it will not 'jump' over the boundary of a month's days. The 'W' character can only be specified when the day-of-month is a single day, not a range or list of days.
The 'L' and 'W' characters can also be combined for the day-of-month expression to yield 'LW', which translates to "last weekday of the month".

The '#' character is allowed for the day-of-week field. This character is used to specify "the nth" XXX day of the month. For example, the value of "6#3" in the day-of-week field means the third Friday of the month (day 6 = Friday and "#3" = the 3rd one in the month). Other examples: "2#1" = the first Monday of the month and "4#5" = the fifth Wednesday of the month. Note that if you specify "#5" and there is not 5 of the given day-of-week in the month, then no firing will occur that month. If the '#' character is used, there can only be one expression in the day-of-week field ("3#1,6#3" is not valid, since there are two expressions).

The legal characters and the names of months and days of the week are not case sensitive.

NOTES:

- Support for specifying both a day-of-week and a day-of-month value is not complete (you'll need to use the '?' character in one of these fields).
- Overflowing ranges is supported - that is, having a larger number on the left hand side than the right. You might do 22-2 to catch 10 o'clock at night until 2 o'clock in the morning, or you might have NOV-FEB. It is very important to note that overuse of overflowing ranges creates ranges that don't make sense and no effort has been made to determine which interpretation CronExpression chooses. An example would be "0 0 14-6 ? * FRI-MON".

Author:
Sharada Jambula, James House, Contributions from Mads Henderson, Refactoring from CronTrigger to CronExpression by Aaron Craven

See Also:
Serialized Form

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected static Integer</td>
</tr>
<tr>
<td>protected static int</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>static int</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
<tr>
<td>protected</td>
</tr>
</tbody>
</table>
### Constructor Summary

**CronExpression**(String cronExpression)

Constructs a new CronExpression based on the specified parameter.

### Method Summary

- **protected void** `addToSet`(int val, int end, int incr, int type)
- **protected void** `buildExpression`(String expression)
- **protected int** `checkNext`(int pos, String s, int val, int type)
- **Object** `clone`()
- **protected int** `findNextWhiteSpace`(int i, String s)
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String getCronExpression()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected int getDayOfWeekNumber(String s)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected String getExpressionSetSummary(ArrayList list)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected String getExpressionSetSummary(Set set)</code></td>
<td></td>
</tr>
<tr>
<td><code>String getExpressionSummary()</code></td>
<td></td>
</tr>
<tr>
<td><code>Date getFinalFireTime()</code></td>
<td>NOT YET IMPLEMENTED: Returns the final time that the CronExpression will match.</td>
</tr>
<tr>
<td><code>protected int getLastDayOfMonth(int monthNum, int year)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected int getMonthNumber(String s)</code></td>
<td></td>
</tr>
<tr>
<td><code>Date getNextInvalidTimeAfter(Date date)</code></td>
<td>Returns the next date/time after the given date/time which does <em>not</em> satisfy the expression.</td>
</tr>
<tr>
<td><code>Date getNextValidTimeAfter(Date date)</code></td>
<td>Returns the next date/time after the given date/time which satisfies the cron expression.</td>
</tr>
<tr>
<td><code>protected int getNumericValue(String s, int i)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected TreeSet&lt;Integer&gt; getSet(int type)</code></td>
<td></td>
</tr>
<tr>
<td><code>Date getTimeAfter(Date afterTime)</code></td>
<td></td>
</tr>
<tr>
<td><code>Date getTimeBefore(Date endTime)</code></td>
<td>NOT YET IMPLEMENTED: Returns the time before the given time that the CronExpression matches.</td>
</tr>
<tr>
<td><code>TimeZone getTimeZone()</code></td>
<td>Returns the time zone for which this CronExpression will be resolved.</td>
</tr>
<tr>
<td>Method</td>
<td>Signature</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>protected <code>getValue</code></td>
<td><code>getValue(int v, String s, int i)</code></td>
</tr>
<tr>
<td>protected boolean <code>isLeapYear</code></td>
<td><code>isLeapYear(int year)</code></td>
</tr>
<tr>
<td>boolean <code>isSatisfiedBy</code></td>
<td><code>isSatisfiedBy(Date date)</code></td>
</tr>
<tr>
<td>static boolean <code>isValidExpression</code></td>
<td><code>isValidExpression(String cronExpression)</code></td>
</tr>
<tr>
<td>protected void <code>setCalendarHour</code></td>
<td><code>setCalendarHour(Calendar cal, int hour)</code></td>
</tr>
<tr>
<td>void <code>setTimeZone</code></td>
<td><code>setTimeZone(TimeZone timeZone)</code></td>
</tr>
<tr>
<td>protected int <code>skipWhiteSpace</code></td>
<td><code>skipWhiteSpace(int i, String s)</code></td>
</tr>
<tr>
<td>protected int <code>storeExpressionVals</code></td>
<td><code>storeExpressionVals(int pos, String s, int type)</code></td>
</tr>
<tr>
<td>String <code>toString</code></td>
<td><code>toString()</code></td>
</tr>
<tr>
<td>static void <code>validateExpression</code></td>
<td><code>validateExpression(String cronExpression)</code></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object:

- `equals`
- `finalize`
- `getClass`
- `hashCode`
- `notify`
- `notifyAll`
- `wait`
- `wait`
- `wait`

Field Detail

SECOND
protected static final int SECOND

See Also:
Constant Field Values

MINUTE

protected static final int MINUTE

See Also:
Constant Field Values

HOUR

protected static final int HOUR

See Also:
Constant Field Values

DAY_OF_MONTH

protected static final int DAY_OF_MONTH

See Also:
Constant Field Values

MONTH

protected static final int MONTH

See Also:
Constant Field Values

DAY_OF_WEEK
protected static final int \texttt{DAY\_OF\_WEEK}

\textbf{See Also:}
\begin{verbatim}
  Constant Field Values
\end{verbatim}

\textbf{YEAR}

protected static final int \texttt{YEAR}

\textbf{See Also:}
\begin{verbatim}
  Constant Field Values
\end{verbatim}

\textbf{ALL\_SPEC\_INT}

protected static final int \texttt{ALL\_SPEC\_INT}

\textbf{See Also:}
\begin{verbatim}
  Constant Field Values
\end{verbatim}

\textbf{NO\_SPEC\_INT}

protected static final int \texttt{NO\_SPEC\_INT}

\textbf{See Also:}
\begin{verbatim}
  Constant Field Values
\end{verbatim}

\textbf{ALL\_SPEC}

protected static final \texttt{Integer} \texttt{ALL\_SPEC}

\textbf{NO\_SPEC}

protected static final \texttt{Integer} \texttt{NO\_SPEC}
monthMap
protected static final Map monthMap

dayMap
protected static final Map dayMap

seconds
protected transient TreeSet<Integer> seconds

minutes
protected transient TreeSet<Integer> minutes

hours
protected transient TreeSet<Integer> hours

daysOfMonth
protected transient TreeSet<Integer> daysOfMonth

months
protected transient TreeSet<Integer> months

daysOfWeek
protected transient TreeSet<Integer> daysOfWeek
years
protected transient TreeSet<Integer> years

lastdayOfWeek
protected transient boolean lastdayOfWeek

nthdayOfWeek
protected transient int nthdayOfWeek

lastdayOfMonth
protected transient boolean lastdayOfMonth

nearestWeekday
protected transient boolean nearestWeekday

lastdayOffset
protected transient int lastdayOffset

expressionParsed
protected transient boolean expressionParsed

MAX_YEAR
public static final int MAX_YEAR
### Constructor Detail

#### CronExpression

```java
public CronExpression(String cronExpression)
    throws ParseException
```

Constructs a new CronExpression based on the specified parameter.

**Parameters:**
- `cronExpression` - String representation of the cron expression the new object should represent

**Throws:**
- `ParseException` - if the string expression cannot be parsed into a valid CronExpression

### Method Detail

#### isSatisfiedBy

```java
public boolean isSatisfiedBy(Date date)
```

Indicates whether the given date satisfies the cron expression. Note that milliseconds are ignored, so two Dates falling on different milliseconds of the same second will always have the same result here.

**Parameters:**
- `date` - the date to evaluate

**Returns:**
- a boolean indicating whether the given date satisfies the cron expression

#### getNextValidTimeAfter

```java
public Date getNextValidTimeAfter(Date date)
```

Returns the next date/time after the given date/time which satisfies the cron
expression.

**Parameters:**
- date - the date/time at which to begin the search for the next valid date/time

**Returns:**
- the next valid date/time

---

**getNextInvalidTimeAfter**

public `Date getNextInvalidTimeAfter(Date date)`

Returns the next date/time `after` the given date/time which does *not* satisfy the expression

**Parameters:**
- date - the date/time at which to begin the search for the next invalid date/time

**Returns:**
- the next valid date/time

---

**getTimeZone**

public `TimeZone getTimeZone()`

Returns the time zone for which this CronExpression will be resolved.

---

**setTimeZone**

public void `setTimeZone(TimeZone timeZone)`

Sets the time zone for which this CronExpression will be resolved.

---

**toString**
public String toString()

Returns the string representation of the CronExpression

Overrides:

    toString in class Object

Returns:

    a string representation of the CronExpression

isValidExpression

public static boolean isValidExpression(String cronExpression)

Indicates whether the specified cron expression can be parsed into a valid cron expression

Parameters:

    cronExpression - the expression to evaluate

Returns:

    a boolean indicating whether the given expression is a valid cron expression

validateExpression

public static void validateExpression(String cronExpression)
throws ParseException

Throws:

    ParseException

buildExpression

protected void buildExpression(String expression)
throws ParseException

Throws:

    ParseException
storeExpressionVals

protected int storeExpressionVals(int pos, String s, int type)
    throws ParseException

    Throws:
    ParseException

checkNext

protected int checkNext(int pos, String s, int val, int type)
    throws ParseException

    Throws:
    ParseException

getcronExpression

public String getCronExpression()

getExpressionSummary

public String getExpressionSummary()

getExpressionSetSummary

protected String getExpressionSetSummary(Set set)

getExpressionSetSummary
protected String getExpressionSetSummary(ArrayList list)

skipWhiteSpace

protected int skipWhiteSpace(int i, String s)

findNextWhiteSpace

protected int findNextWhiteSpace(int i, String s)

addToSet

protected void addToSet(int val,
                        int end,
                        int incr,
                        int type)
                        throws ParseException

    Throws:
    ParseException

getSet

protected TreeSet<Integer> getSet(int type)

getValue

protected org.quartz.ValueSet getValue(int v,
                                       String s,
                                       int i)

getNumericValue
protected int getNumericValue(String s, int i)

getMonthNumber
protected int getMonthNumber(String s)

g.getDayOfWeekNumber
protected int getDayOfWeekNumber(String s)

g.getTimeAfter
public Date getTimeAfter(Date afterTime)

setCalendarHour
protected void setCalendarHour(Calendar cal, int hour)

Advance the calendar to the particular hour paying particular attention to daylight saving problems.

Parameters:
  cal -
  hour -

g.getTimeBefore
public Date getTimeBefore(Date endTime)

NOT YET IMPLEMENTED: Returns the time before the given time that the CronExpression matches.
getFinalFireTime

public Date getFinalFireTime()

NOT YET IMPLEMENTED: Returns the final time that the CronExpression will match.

isLeapYear

protected boolean isLeapYear(int year)

g getLastDayOfMonth

protected int getLastDayOfMonth(int monthNum, int year)

c lone

public Object clone()

Overrides:
clone in class Object

Copyright 2001-2011, Terracotta, Inc.
public class CronScheduleBuilder
extends ScheduleBuilder<CronTrigger>

CronScheduleBuilder is a ScheduleBuilder that defines CronExpression-based schedules for Triggers.

Quartz provides a builder-style API for constructing scheduling-related entities via a Domain-Specific Language (DSL). The DSL can best be utilized through the usage of static imports of the methods on the classes TriggerBuilder, JobBuilder, DateBuilder, JobKey, TriggerKey and the various ScheduleBuilder implementations.

Client code can then use the DSL to write code such as this:

```java
    JobDetail job = newJob(MyJob.class)
        .withIdentity("myJob")
        .build();

    Trigger trigger = newTrigger()
        .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
        .withSchedule(simpleSchedule()
            .withIntervalInHours(1)
            .repeatForever())
        .startAt(futureDate(10, MINUTES))
        .build();

    scheduler.scheduleJob(job, trigger);
```
### Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.spi.MutableTrigger</td>
<td><strong>build()</strong></td>
<td>Build the actual Trigger -- NOT intended to be invoked by end users, but will rather be invoked by a TriggerBuilder which the ScheduleBuilder is given to.</td>
</tr>
<tr>
<td>static CronScheduleBuilder</td>
<td><strong>cronSchedule(String cronExpression)</strong></td>
<td>Create a CronScheduleBuilder with a given cron-expression.</td>
</tr>
<tr>
<td>static CronScheduleBuilder</td>
<td><strong>dailyAtHourAndMinute(int hour, int minute)</strong></td>
<td>Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire every day at the given time (hour and minute).</td>
</tr>
<tr>
<td>static CronScheduleBuilder</td>
<td><strong>inTimeZone(TimeZone tz)</strong></td>
<td>The TimeZone in which to base the schedule.</td>
</tr>
<tr>
<td>static CronScheduleBuilder</td>
<td><strong>monthlyOnDayAndHourAndMinute(int dayOfMonth, int hour, int minute)</strong></td>
<td>Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire one per month on the given day of month at the given time (hour and minute).</td>
</tr>
<tr>
<td>static CronScheduleBuilder</td>
<td><strong>weeklyOnDayAndHourAndMinute(int dayOfWeek, int hour, int minute)</strong></td>
<td>Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire one per week on the given day at the given time.</td>
</tr>
</tbody>
</table>
(hour and minute).

<table>
<thead>
<tr>
<th>CronScheduleBuilder</th>
<th>withMisfireHandlingInstructionDoNothing()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the Trigger misfires, use the CronTrigger.MISFIRE_INSTRUCTION_DO NOTHING instruction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CronScheduleBuilder</th>
<th>withMisfireHandlingInstructionFireAndProceed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the Trigger misfires, use the CronTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW instruction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CronScheduleBuilder</th>
<th>withMisfireHandlingInstructionIgnoreMisfires</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the Trigger misfires, use the Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY instruction.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Method Detail

build

public org.quartz.spi.MutableTrigger build()

Build the actual Trigger -- NOT intended to be invoked by end us but will rather be invoked by a TriggerBuilder which this ScheduleBuilder is given to.
Specified by:
build in class ScheduleBuilder<CronTrigger>

See Also:
TriggerBuilder.withSchedule(ScheduleBuilder)

cronSchedule

public static CronScheduleBuilder cronSchedule(String cronExpression) throws ParseException

Create a CronScheduleBuilder with the given cron-expression.

Parameters:
cronExpression - the cron expression to base the schedule on

Returns:
the new CronScheduleBuilder

Throws:
ParseException

See Also:
CronExpression
**dailyAtHourAndMinute**

```java
public static CronScheduleBuilder dailyAtHourAndMinute(int hour, int minute)
```

Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire every day at the given time (hour and minute).

**Parameters:**
- `hour`: the hour of day to fire
- `minute`: the minute of the given hour to fire

**Returns:**
- the new CronScheduleBuilder

**Throws:**
- ParseException

**See Also:**
- CronExpression

---

**weeklyOnDayAndHourAndMinute**

```java
public static CronScheduleBuilder weeklyOnDayAndHourAndMinute(int dayOfWeek, int hour, int minute)
```

Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire one per week on the given day at the given time.
Create a CronScheduleBuilder with a cron-expression that sets the schedule to fire one per month on the given day of month at the time (hour and minute).
Parameters:
- dayOfMonth - the day of the month to fire
- hour - the hour of day to fire
- minute - the minute of the given hour to fire

Returns:
- the new CronScheduleBuilder

Throws:
- ParseException

See Also:
- CronExpression

---

inTimeZone

public CronScheduleBuilder inTimeZone(TimeZone tz)

The TimeZone in which to base the schedule.

Parameters:
- tz - the time-zone for the schedule.

Returns:
- the updated CronScheduleBuilder

See Also:
- CronExpression.getTimeZone()
withMisfireHandlingInstructionIgnoreMisfires

public CronScheduleBuilder withMisfireHandlingInstructionIgnoreMisfires

If the Trigger misfires, use the
    Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY instruction.

Returns:
    the updated CronScheduleBuilder
See Also:
    Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY

withMisfireHandlingInstructionDoNothing

public CronScheduleBuilder withMisfireHandlingInstructionDoNothing()

If the Trigger misfires, use the
    CronTrigger.MISFIRE_INSTRUCTION_DO NOTHING instruction.

Returns:
    the updated CronScheduleBuilder
See Also:
CronTrigger.MISFIRE_INSTRUCTION_DO NOTHING

withMisfireHandlingInstructionFireAndProceed

public CronScheduleBuilder withMisfireHandlingInstructionFireAndProceed

If the Trigger misfires, use the
CronTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW instruction.

Returns:
the updated CronScheduleBuilder

See Also:
CronTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED  FIELD  CONSTR  METHOD

DETAIL: FIELD  CONSTR  METHOD

FRAMES  NO FRAMES
Interface CronTrigger

All Superinterfaces: 
   Cloneable, Comparable<Trigger>, Serializable, Trigger

All Known Implementing Classes: 
   CronTriggerImpl

public interface CronTrigger
extends Trigger

The public interface for inspecting settings specific to a CronTrigger, which is used to fire a Job at given moments in time, defined with Unix 'cron-like' schedule definitions.

For those unfamiliar with "cron", this means being able to create a firing schedule such as: "At 8:00am every Monday through Friday" or "At 1:30am every last Friday of the month".

The format of a "Cron-Expression" string is documented on the CronExpression class.

Here are some full examples:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;0 0 12 * * ?&quot;</td>
<td>Fire at 12pm (noon) every day</td>
</tr>
<tr>
<td>&quot;0 15 10 ? * *&quot;</td>
<td>Fire at 10:15am every day</td>
</tr>
<tr>
<td>&quot;0 15 10 * * ?&quot;</td>
<td>Fire at 10:15am every day</td>
</tr>
<tr>
<td>&quot;0 15 10 * * ? *&quot;</td>
<td>Fire at 10:15am every day during the year 2005</td>
</tr>
<tr>
<td>&quot;0 * 14 *&quot;</td>
<td>Fire every minute starting at 2pm and ending at 2:59pm, every day</td>
</tr>
<tr>
<td>&quot;0 0/5 14&quot;</td>
<td>Fire every 5 minutes starting at 2pm and ending at 2:59pm, every day</td>
</tr>
</tbody>
</table>
* * ?" 2:55pm, every day

"0 0/5 14,18 * *" Fire every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day

"0 0-5 14 * * ?" Fire every minute starting at 2pm and ending at 2:05pm, every day

"0 10,44 14 ? 3 WED" Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.

"0 15 10 ? * MON-FRI" Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday

"0 15 10 15 * ?" Fire at 10:15am on the 15th day of every month

"0 15 10 L * ?" Fire at 10:15am on the last day of every month

"0 15 10 ? * 6L" Fire at 10:15am on the last Friday of every month

"0 15 10 ? * 6L" Fire at 10:15am on the last Friday of every month

"0 15 10 ? * 6L 2002-2005" Fire at 10:15am on every last Friday of every month during the years 2002, 2003, 2004 and 2005

"0 15 10 ? * 6#3" Fire at 10:15am on the third Friday of every month

Pay attention to the effects of '?' and '*' in the day-of-week and day-of-month fields!

NOTES:

- Support for specifying both a day-of-week and a day-of-month value is not complete (you'll need to use the '?' character in one of these fields).
- Be careful when setting fire times between mid-night and 1:00 AM - "daylight savings" can cause a skip or a repeat depending on whether the time moves back or jumps forward.

Author:
jhouse, Contributions from Mads Henderson

See Also:
CronScheduleBuilder, TriggerBuilder
## Nested Class Summary

**Nested classes/interfaces inherited from interface org.quartz.Trigger**
- Trigger.CompletedExecutionInstruction
- Trigger.TriggerState
- Trigger.TriggerTimeComparator

## Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int MISFIRE_INSTRUCTION.DO_NOTHING</td>
<td>Instructs the Scheduler that upon a mis-fire situation, the CronTrigger wants to have its next-fire-time updated to the next time in the schedule after the current time (taking into account any associated Calendar, but it does not want to be fired now.)</td>
</tr>
<tr>
<td>static int MISFIRE_INSTRUCTION_FIRE_ONCE_NOW</td>
<td>Instructs the Scheduler that upon a mis-fire situation, the CronTrigger wants to be fired now by Scheduler.</td>
</tr>
<tr>
<td>static long serialVersionUID</td>
<td></td>
</tr>
</tbody>
</table>

**Fields inherited from interface org.quartz.Trigger**
- DEFAULT_PRIORITY
- MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY
- MISFIRE_INSTRUCTION_SMART_POLICY

## Method Summary

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getCronExpression()</td>
<td></td>
</tr>
<tr>
<td>String getExpressionSummary()</td>
<td></td>
</tr>
<tr>
<td>TimeZone getTimeZone()</td>
<td>Returns the time zone for which the cronExpression of this CronTrigger will be resolved.</td>
</tr>
<tr>
<td>TriggerBuilder&lt;CronTrigger&gt; getTriggerBuilder()</td>
<td></td>
</tr>
</tbody>
</table>

Get a TriggerBuilder that is configured to produce a Trigger identical to this one.
Methods inherited from interface `org.quartz.Trigger`

`compareTo`, `equals`, `getCalendarName`, `getDescription`, `getEndTime`, `getFinalFireTime`, `getFireTimeAfter`, `getJobDataMap`, `getJobKey`, `getKey`, `getMisfireInstruction`, `getNextFireTime`, `getPreviousFireTime`, `getPriority`, `getScheduleBuilder`, `getStartTime`, `mayFireAgain`

Field Detail

`serialVersionUID`

static final long `serialVersionUID`

See Also:

`Constant Field Values`

`MISFIRE_INSTRUCTION_FIRE_ONCE_NOW`

static final int `MISFIRE_INSTRUCTION_FIRE_ONCE_NOW`

Instructs the Scheduler that upon a mis-fire situation, the CronTrigger wants to be fired now by Scheduler.

See Also:

`Constant Field Values`

`MISFIRE_INSTRUCTION_DO_NOTHING`

static final int `MISFIRE_INSTRUCTION_DO_NOTHING`

Instructs the Scheduler that upon a mis-fire situation, the CronTrigger wants to have it's next-fire-time updated to the next time in the schedule after the current time (taking into account any associated Calendar, but it does not want to be fired now.)
getCronExpression

`String getCronExpression()`

getTimeZone

`TimeZone getTimeZone()`

Returns the time zone for which the cronExpression of this CronTrigger will be resolved.

getExpressionSummary

`String getExpressionSummary()`

getTriggerBuilder

`TriggerBuilder<CronTrigger> getTriggerBuilder()`

Description copied from interface: `Trigger`
Get a `TriggerBuilder` that is configured to produce a Trigger identical to this one.

Specified by:
`getTriggerBuilder` in interface `Trigger`

See Also:
`Trigger.getScheduleBuilder()`
Copyright 2001-2011, Terracotta, Inc.
| SUMMARY: | FIELD | CONSTR | METHOD | DETAIL: | FIELD | CONSTR | METHOD | PREV CLASS | NEXT CLASS | FRAMES | NO FRAMES |
|----------|-------|--------|--------|---------|--------|--------|--------|----------|------------|----------|---------|-----------|
| Overview | Package | Use | Tree | Deprecated | Index | Help | Overview | Package | Use | Tree | Deprecated | Index | Help | Overview | Package | Use | Tree | Deprecated | Index | Help | Overview | Package | Use | Tree | Deprecated | Index | Help |
| PREV CLASS | NEXT CLASS | SUMMARY: | FIELD | CONSTR | METHOD | DETAIL: | FIELD | CONSTR | METHOD | PREV CLASS | NEXT CLASS | FRAMES | NO FRAMES |
org.quartz  Class DateBuilder

java.lang.Object
   ↓ org.quartz.DateBuilder

public class DateBuilder

extends Object

DateBuilder is used to conveniently create java.util.Date instances that meet particular criteria.

Quartz provides a builder-style API for constructing scheduling-related entities via a Domain-Specific Language (DSL). The DSL can best be utilized through the usage of static imports of the methods on the classes TriggerBuilder, JobBuilder, DateBuilder, JobKey, TriggerKey and the various ScheduleBuilder implementations.

Client code can then use the DSL to write code such as this:

```java
JobDetail job = newJob(MyJob.class)
   .withIdentity("myJob")
   .build();

Trigger trigger = newTrigger()
   .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
   .withSchedule(simpleSchedule()
      .withIntervalInHours(1)
      .repeatForever())
   .startAt(futureDate(10, MINUTES))
   .build();

scheduler.scheduleJob(job, trigger);
```

See Also:
### Nested Class Summary

| static class | DateBuilder.IntervalUnit |

### Field Summary

| static int | FRI
| static int | MON
| static int | SAT
| static long | SUN
| static int | THU
| static int | TUE
| static int | WED
| static long | MILLISEC
| static long | MILLISEC
| static long | MILLISEC
| static long | SECON

**TriggerBuilder**,
**JobBuilder**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DateBuilder</strong></td>
<td><strong>atHourMinuteAndSecond</strong>(int hour, int minute, int second)</td>
</tr>
<tr>
<td></td>
<td>Set the hour (0-23) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><strong>DateBuilder</strong></td>
<td><strong>atHourOfDay</strong>(int hour)</td>
</tr>
<tr>
<td></td>
<td>Set the hour (0-23) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><strong>DateBuilder</strong></td>
<td><strong>atMinute</strong>(int minute)</td>
</tr>
<tr>
<td></td>
<td>Set the minute (0-59) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><strong>DateBuilder</strong></td>
<td><strong>atSecond</strong>(int second)</td>
</tr>
<tr>
<td></td>
<td>Set the second (0-59) for the Date that will be built by this builder, and truncate the milliseconds to 000.</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td><strong>build</strong>()</td>
</tr>
<tr>
<td></td>
<td>Build the Date defined by this builder instance.</td>
</tr>
<tr>
<td><strong>static Date</strong></td>
<td><strong>dateOf</strong>(int hour, int minute, int second)</td>
</tr>
<tr>
<td></td>
<td>Get a Date object that represents the given time, on today's date (equivalent to todayAt(int, int, int)).</td>
</tr>
<tr>
<td><strong>static Date</strong></td>
<td><strong>dateOf</strong>(int hour, int minute, int second, int dayOfMonth, int month)</td>
</tr>
<tr>
<td></td>
<td>Get a Date object that represents the given time, on the given date.</td>
</tr>
<tr>
<td><strong>static Date</strong></td>
<td><strong>dateOf</strong>(int hour, int minute, int second, int dayOfMonth, int month, int year)</td>
</tr>
<tr>
<td></td>
<td>Get a Date object that represents the given time, on the given date.</td>
</tr>
<tr>
<td><strong>static Date</strong></td>
<td><strong>evenHourDate</strong>(Date date)</td>
</tr>
<tr>
<td></td>
<td>Returns a date that is rounded to the next even hour above the given date.</td>
</tr>
<tr>
<td><strong>static Date</strong></td>
<td><strong>evenHourDateAfterNow</strong>()</td>
</tr>
<tr>
<td></td>
<td>Returns a date that is rounded to the next even hour after the current time.</td>
</tr>
<tr>
<td><strong>static Date</strong></td>
<td><strong>evenHourDateBefore</strong>(Date date)</td>
</tr>
<tr>
<td></td>
<td>Returns a date that is rounded to the previous even hour below the given date.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>evenMinuteDate(Date date)</code></td>
<td>Returns a date that is rounded to the next even minute above the given date.</td>
</tr>
<tr>
<td><code>evenMinuteDateAfterNow()</code></td>
<td>Returns a date that is rounded to the next even minute after the current time.</td>
</tr>
<tr>
<td><code>evenMinuteDateBefore(Date date)</code></td>
<td>Returns a date that is rounded to the previous even minute below the given date.</td>
</tr>
<tr>
<td><code>evenSecondDate(Date date)</code></td>
<td>Returns a date that is rounded to the next even second above the given date.</td>
</tr>
<tr>
<td><code>evenSecondDateAfterNow()</code></td>
<td>Returns a date that is rounded to the next even second after the current time.</td>
</tr>
<tr>
<td><code>evenSecondDateBefore(Date date)</code></td>
<td>Returns a date that is rounded to the previous even second below the given date.</td>
</tr>
<tr>
<td><code>futureDate(int interval, DateBuilder.IntervalUnit unit)</code></td>
<td>Set the Locale for the Date that will be built by this builder (if &quot;null&quot;, system default will be used).</td>
</tr>
<tr>
<td><code>inLocale(Locale lc)</code></td>
<td>Set the month (1-12) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><code>inMonthOnDay(int month, int day)</code></td>
<td>Set the TimeZone for the Date that will be built by this builder (if &quot;null&quot;, system default will be used).</td>
</tr>
<tr>
<td><code>inTimeZone(TimeZone tz)</code></td>
<td>Set the year for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><code>inYear(int year)</code></td>
<td>Create a DateBuilder, with initial settings for the current date and time in the system default timezone.</td>
</tr>
<tr>
<td><code>newDate()</code></td>
<td>Create a DateBuilder, with initial settings for the current date and time in the system default timezone.</td>
</tr>
<tr>
<td><code>newDateInLocale(Locale lc)</code></td>
<td>Create a DateBuilder, with initial settings for the current date and time in the system default timezone.</td>
</tr>
</tbody>
</table>
settings for the current date and time in the given locale.

```java
static DateBuilder newDateInTimeZone(TimeZone tz)
    Create a DateBuilder, with initial settings for the current date and time in the given timezone.

static DateBuilder newDateInTimeZoneAndLocale(TimeZone tz, Locale lc)
    Create a DateBuilder, with initial settings for the current date and time in the given timezone and locale.

static Date nextGivenMinuteDate(Date date, int minuteBase)
    Returns a date that is rounded to the next even multiple of the given minute.

static Date nextGivenSecondDate(Date date, int secondBase)
    Returns a date that is rounded to the next even multiple of the given minute.

DateBuilder onDay(int day)
    Set the day of month (1-31) for the Date that will be built by this builder.

static Date todayAt(int hour, int minute, int second)
    Get a Date object that represents the given time, on today's date (equivalent to dateOf(int, int, int)).

static Date tomorrowAt(int hour, int minute, int second)
    Get a Date object that represents the given time, on tomorrow's date.

static Date translateTime(Date date, TimeZone src, TimeZone dest)
    Translate a date & time from a users time zone to the another (probably server) time zone to assist in creating a simple trigger with the right date & time.

static void validateDayOfMonth(int day)

static void validateDayOfWeek(int dayOfWeek)

static void validateHour(int hour)

static void validateMinute(int minute)

static void validateMonth(int month)
```
validateSecond(int second)

static void validateYear(int year)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SUNDAY

public static final int SUNDAY

See Also:
Constant Field Values

MONDAY
public static final int MONDAY

See Also:
Constant Field Values

TUESDAY

public static final int TUESDAY

See Also:
Constant Field Values

WEDNESDAY

public static final int WEDNESDAY

See Also:
Constant Field Values
THURSDAY

public static final int THURSDAY

See Also: Constant Field Values

FRIDAY

public static final int FRIDAY

See Also: Constant Field Values
SATURDAY

public static final int SATURDAY

See Also:
Constant Field Values

MILLISECONDS_IN_MINUTE

public static final long MILLISECONDS_IN_MINUTE

See Also:
Constant Field Values

MILLISECONDS_IN_HOUR

public static final long MILLISECONDS_IN_HOUR
See Also:
Constant Field Values

SECONDS_IN_MOST_DAYS

public static final long SECONDS_IN_MOST_DAYS

See Also:
Constant Field Values

MILLISECONDS_IN_DAY

public static final long MILLISECONDS_IN_DAY

See Also:
Constant Field Values
**newDate**

```java
public static DateBuilder newDate()
```

Create a DateBuilder, with initial settings for the current date.

---

**newDateInTimezone**

```java
public static DateBuilder newDateInTimezone(TimeZone tz)
```

Create a DateBuilder, with initial settings for the current date in the given timezone.
newDateInLocale

public static DateBuilder newDateInLocale(Locale lc)

Create a DateBuilder, with initial settings for the current date

newDateInTimeZoneAndLocale

public static DateBuilder newDateInTimeZoneAndLocale(TimeZone tz, Locale lc)

Create a DateBuilder, with initial settings for the current date
**build**

public Date build()

Build the Date defined by this builder instance.

**atHourOfDay**

public DateBuilder atHourOfDay(int hour)

Set the hour (0-23) for the Date that will be built by this build
atMinute

public DateBuilder atMinute(int minute)

Set the minute (0-59) for the Date that will be built by this builder.

atSecond

public DateBuilder atSecond(int second)

Set the second (0-59) for the Date that will be built by this builder.
public DateBuilder atHourMinuteAndSecond(int hour,
    int minute,
    int second)

onDay

public DateBuilder onDay(int day)

Set the day of month (1-31) for the Date that will be built by t

inMonth

public DateBuilder inMonth(int month)
Set the month (1-12) for the Date that will be built by this builder.

inMonthOnDay

public DateBuilder inMonthOnDay(int month,
                                  int day)

inYear

public DateBuilder inYear(int year)

Set the year for the Date that will be built by this builder.
public DateBuilder inTimeZone(TimeZone tz)

Set the TimeZone for the Date that will be built by this builder

public DateBuilder inLocale(Locale lc)

Set the Locale for the Date that will be built by this builder (}
**futureDate**

```java
public static Date futureDate(int interval,
                              DateBuilder.IntervalUnit unit)
```

**tomorrowAt**

```java
public static Date tomorrowAt(int hour,
                               int minute,
                               int second)
```

Get a Date object that represents the given time, on tomorrow's date.
todayAt

```java
public static Date todayAt(int hour,
    int minute,
    int second)
```

Get a Date object that represents the given time, on today's date (equivalent to `dateOf(int, int, int)`).
public static Date dateOf(int hour,
    int minute,
    int second)

Get a Date object that represents the given time, on today's date (equivalent to todayAt(int, int, int)).

**Parameters:**
- second - The value (0-59) to give the seconds field of the date
- minute - The value (0-59) to give the minutes field of the date
- hour - The value (0-23) to give the hours field of the date

**Returns:**
the new date
Get a Date object that represents the given time, on the given date.

Parameters:
- second - The value (0-59) to give the seconds field of the date
- minute - The value (0-59) to give the minutes field of the date
- hour - The value (0-23) to give the hours field of the date
- dayOfMonth - The value (1-31) to give the day of month field of the date
- month - The value (1-12) to give the month field of the date

Returns:
- the new date

---

dateOf

code:

```java
def public static Date dateOf(int hour,
    int minute,
    int second,
    int dayOfMonth,
    int month,
    int year)
```
Get a Date object that represents the given time, on the given date.

**Parameters:**
- second - The value (0-59) to give the seconds field of the date
- minute - The value (0-59) to give the minutes field of the date
- hour - The value (0-23) to give the hours field of the date
- dayOfMonth - The value (1-31) to give the day of month field of the date
- month - The value (1-12) to give the month field of the date
- year - The value (1970-2099) to give the year field of the date

**Returns:**
- the new date

---

**evenHourDateAfterNow**

public static Date evenHourDateAfterNow()

Returns a date that is rounded to the next even hour after the
For example a current time of 08:13:54 would result in a date with the time of 09:00:00. If the date's time is in the 23rd hour, the date's 'day' will be promoted, and the time will be set to 00:00.

Returns:
the new rounded date

---

evenHourDate

public static Date evenHourDate(Date date)

Returns a date that is rounded to the next even hour above the date.

For example an input date with a time of 08:13:54 would result with the time of 09:00:00. If the date's time is in the 23rd hour, the date's 'day' will be promoted, and the time will be set to 00:00.
Parameters:
  date - the Date to round, if null the current time will be used

Returns:
  the new rounded date

---

evenHourDateBefore

public static Date evenHourDateBefore(Date date)

Returns a date that is rounded to the previous even hour below date.

For example an input date with a time of 08:13:54 would result with the time of 08:00:00.

Parameters:
date - the Date to round, if null the current time will be used

**Returns:**
the new rounded date

evenMinuteDateAfterNow

public static Date evenMinuteDateAfterNow()

Returns a date that is rounded to the next even minute after the current time.

For example a current time of 08:13:54 would result in a date with the time of 08:14:00. If the date's time is in the 59th minute then the hour (and possibly the day) will be promoted.

**Returns:**
the new rounded date
evenMinuteDate

public static Date evenMinuteDate(Date date)

Returns a date that is rounded to the next even minute above the date.

For example an input date with a time of 08:13:54 would result with the time of 08:14:00. If the date's time is in the 59th minute then the hour (and possibly the day) will be promoted.

Parameters:
    date - the Date to round, if null the current time will be used

Returns:
    the new rounded date
evenMinuteDateBefore

public static Date evenMinuteDateBefore(Date date)

Returns a date that is rounded to the previous even minute below the given date.

For example an input date with a time of 08:13:54 would result with the time of 08:13:00.

Parameters:
   date - the Date to round, if null the current time will be used

Returns:
   the new rounded date
public static Date evenSecondDateAfterNow()

Returns a date that is rounded to the next even second after the current time.

Returns:  
the new rounded date

evenSecondDate

class

public static Date evenSecondDate(Date date)

Returns a date that is rounded to the next even second above the given date.

Parameters:

date - the Date to round, if null the current time will
evenSecondDateBefore

public static Date evenSecondDateBefore(Date date)

Returns a date that is rounded to the previous even second below the given date.

For example an input date with a time of 08:13:54.341 would result in a date with the time of 08:13:00.000.

Parameters:
  date - the Date to round, if null the current time will be used

Returns:
  the new rounded date
public static Date nextGivenMinuteDate(Date date, int minuteBase)

Returns a date that is rounded to the next even multiple of the minute.

For example an input date with a time of 08:13:54, and an input minute-base of 5 would result in a date with the time of 08:15:00. Same input date with an input minute-base of 10 would result in a date with the time of 08:20:00. But a date with the time 08:53:31 and an input minute-base of 45 would result in 09:00:00, because the 0 is the next 'base' for 45-minute intervals.

More examples:

<table>
<thead>
<tr>
<th>Input Time</th>
<th>Minute-Base</th>
<th>Result Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:16:41:20</td>
<td>20</td>
<td>11:20:00</td>
</tr>
<tr>
<td>11:36:41:20</td>
<td>20</td>
<td>11:40:00</td>
</tr>
<tr>
<td>11:46:41:20</td>
<td>20</td>
<td>12:00:00</td>
</tr>
<tr>
<td>11:26:41:30</td>
<td>30</td>
<td>11:30:00</td>
</tr>
</tbody>
</table>
Parameters:
  date - the Date to round, if null the current time will be used
  minuteBase - the base-minute to set the time on

Returns:
  the new rounded date

See Also:
  nextGivenSecondDate(Date, int)

---

nextGivenSecondDate

public static Date nextGivenSecondDate(Date date, int secondBase)

Returns a date that is rounded to the next even multiple of the minute.
The rules for calculating the second are the same as those for calculating the minute in the method
getNextGivenMinuteDate(..).

Parameters:
- date - the Date to round, if null the current time will be used
- secondBase - the base-second to set the time on

Returns:
- the new rounded date

See Also:
- nextGivenMinuteDate(Date, int)

---

**translateTime**

public static Date translateTime(Date date, TimeZone src, TimeZone dest)

Translate a date & time from a users time zone to the another (probably server) time zone to assist in creating a simple trigger with the right date & time.

Parameters:
- date - the date to translate
- src - the original time-zone
dest - the destination time-zone

Returns:
the translated date

validateDayOfWeek

public static void validateDayOfWeek(int dayOfWeek)

validateHour

public static void validateHour(int hour)
validateMinute

public static void validateMinute(int minute)

validateSecond

public static void validateSecond(int second)

validateDayOfMonth

public static void validateDayOfMonth(int day)
validateMonth

public static void validateMonth(int month)

validateYear

public static void validateYear(int year)
org.quartz Enum DateBuilder.IntervalUnit

java.lang.Object
  ▼ java.lang.Enum<DateBuilder.IntervalUnit>
  ▼ org.quartz.DateBuilder.IntervalUnit

All Implemented Interfaces:
  Serializable, Comparable<DateBuilder.IntervalUnit>

Enclosing class:
  DateBuilder

public static enum DateBuilder.IntervalUnit
  extends Enum<DateBuilder.IntervalUnit>

<table>
<thead>
<tr>
<th>Enum Constant Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAY</td>
</tr>
<tr>
<td>HOUR</td>
</tr>
<tr>
<td>MILLISECOND</td>
</tr>
<tr>
<td>MINUTE</td>
</tr>
<tr>
<td>MONTH</td>
</tr>
<tr>
<td>SECOND</td>
</tr>
<tr>
<td>WEEK</td>
</tr>
<tr>
<td>YEAR</td>
</tr>
</tbody>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>DateBuilder.IntervalUnit</code> <code>valueOf</code> (<code>String name</code>)</td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static <code>DateBuilder.IntervalUnit[]</code> <code>values</code> (``)</td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

## Methods inherited from class `java.lang.Enum`  
`clone`, `compareTo`, `equals`, `finalize`, `getDeclaringClass`, `hashCode`, `name`, `ordinal`, `toString`, `valueOf`

## Methods inherited from class `java.lang.Object`  
`getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

## Enum Constant Detail

### MILLISECOND

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final <code>DateBuilder.IntervalUnit MILLISECOND</code></td>
<td></td>
</tr>
</tbody>
</table>

### SECOND

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final <code>DateBuilder.IntervalUnit SECOND</code></td>
<td></td>
</tr>
</tbody>
</table>

### MINUTE

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final <code>DateBuilder.IntervalUnit MINUTE</code></td>
<td></td>
</tr>
</tbody>
</table>

### HOUR

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final <code>DateBuilder.IntervalUnit HOUR</code></td>
<td></td>
</tr>
</tbody>
</table>
DAY

public static final DayBuilder.IntervalUnit DAY

WEEK

public static final DayBuilder.IntervalUnit WEEK

MONTH

public static final DayBuilder.IntervalUnit MONTH

YEAR

public static final DayBuilder.IntervalUnit YEAR

Method Detail

values

public static DayBuilder.IntervalUnit[] values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

for (DateBuilder.IntervalUnit c : DateBuilder.IntervalUnit.values())
    System.out.println(c);

Returns:
    an array containing the constants of this enum type, in the order they are declared

valueOf
public static DateBuilder.IntervalUnit valueOf(String name)

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
   name - the name of the enum constant to be returned.

Returns:
   the enum constant with the specified name

Throws:
   IllegalArgumentException - if this enum type has no constant with the specified name
   NullPointerException - if the argument is null

Copyright 2001-2011, Terracotta, Inc.
An annotation that marks a Job class as one that must not have multiple instances executed concurrently (where instance is based-upon a JobDetail definition - or in other words based upon a JobKey).

This can be used in lieu of implementing the StatefulJob marker interface that was used prior to Quartz 2.0

Author:
jhouse

See Also:
PersistJobDataAfterExecution
An annotation that marks a `Job` class as one that will have its execution wrapped by a JTA Transaction.

If this annotation is present, Quartz will begin a JTA transaction before calling the `execute()` method, and will commit the transaction if the method does not throw an exception and the transaction has not had `setRollbackOnly()` called on it (otherwise the transaction will be rolled-back by Quartz).

This is essentially the same behavior as setting the configuration property `org.quartz.scheduler.wrapJobExecutionInUserTransaction` to `true` - except that it only affects the job that has the annotation, rather than all jobs (as the property does). If the property is set to `true` and the annotation is also set, then of course the annotation becomes redundant.

**Author:**

jhouse
org.quartz Interface InterruptableJob

All Superinterfaces:
Job

public interface InterruptableJob
extends Job

The interface to be implemented by Jobs that provide a mechanism for having their execution interrupted. It is NOT a requirement for jobs to implement this interface - in fact, for most people, none of their jobs will.

Interrupting a Job is very analogous in concept and challenge to normal interruption of a Thread in Java.

The means of actually interrupting the Job must be implemented within the Job itself (the interrupt() method of this interface is simply a means for the scheduler to inform the Job that a request has been made for it to be interrupted). The mechanism that your jobs use to interrupt themselves might vary between implementations. However the principle idea in any implementation should be to have the body of the job's execute(...) periodically check some flag to see if an interruption has been requested, and if the flag is set, somehow abort the performance of the rest of the job's work. An example of interrupting a job can be found in the java source for the class org.quartz.examples.DumbInterruptableJob. It is legal to use some combination of wait() and notify() synchronization within interrupt() and execute(...) in order to have the interrupt() method block until the execute(...) signals that it has noticed the set flag.

If the Job performs some form of blocking I/O or similar functions, you may want to consider having the Job.execute(...) method store a reference to the calling Thread as a member variable. Then the Implementation of this interfaces interrupt() method can call interrupt() on that Thread. Before attempting this, make sure that you fully understand what java.lang.Thread.interrupt() does and doesn't do. Also make sure that you clear the Job's member reference to the Thread when the execute(..) method exits (preferably in a finally block.)
See Example 7 (org.quartz.examples.example7.DumbInterruptableJob) for a simple implementation demonstration.

**Author:**
James House

**See Also:**
Job, StatefulJob, Scheduler.interrupt(JobKey)

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void [<strong>interrupt</strong>()]</td>
<td>Called by the Scheduler when a user interrupts the Job.</td>
</tr>
</tbody>
</table>

### Methods inherited from interface org.quartz.Job

- execute

### Method Detail

**interrupt**

```java
void interrupt() throws UnableToInterruptJobException
```

Called by the Scheduler when a user interrupts the Job.

**Throws:**

UnableToInterruptJobException - if there is an exception while interrupting the job.

---

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
public interface Job

The interface to be implemented by classes which represent a 'job' to be performed.

Instances of Job must have a public no-argument constructor.

JobDataMap provides a mechanism for 'instance member data' that may be required by some implementations of this interface.

Author: James House

See Also: JobDetail, JobBuilder, ExecuteInJTATransaction, DisallowConcurrentExecution, PersistJobDataAfterExecution, Trigger, Scheduler

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void execute(JobExecutionContext context)</td>
</tr>
<tr>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
</tbody>
</table>
execute

void execute(JobExecutionContext context)
        throws JobExecutionException

Called by the Scheduler when a Trigger fires that is associated with the Job.

The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Throws:

JobExecutionException - if there is an exception while executing the job.

Copyright 2001-2011, Terracotta, Inc.
public class JobBuilder

extends Object

JobBuilder is used to instantiate JobDetails.

Quartz provides a builder-style API for constructing scheduling-related entities via a Domain-Specific Language (DSL). The DSL can best be utilized through the usage of static imports of the methods on the classes TriggerBuilder, JobBuilder, DateBuilder, JobKey, TriggerKey and the various ScheduleBuilder implementations.

Client code can then use the DSL to write code such as this:

```java
JobDetail job = newJob(MyJob.class)
    .withIdentity("myJob")
    .build();

Trigger trigger = newTrigger()
    .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
    .withSchedule(simpleSchedule()
        .withIntervalInHours(1)
        .repeatForever())
    .startAt(futureDate(10, MINUTES))
    .build();

scheduler.scheduleJob(job, trigger);
```

See Also:
  TriggerBuilder,
### Method Summary

<table>
<thead>
<tr>
<th>JobDetail</th>
<th><code>build()</code></th>
<th>Produce the JobDetail instance defined by this JobBuilder.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static JobBuilder</td>
<td><code>newJob()</code></td>
<td>Create a JobBuilder with which to define a JobDetail.</td>
</tr>
<tr>
<td>static JobBuilder</td>
<td><code>newJob(Class&lt;? extends Job&gt; jobClass)</code></td>
<td>Create a JobBuilder with which to define a JobDetail, and set the class name of the Job to be executed.</td>
</tr>
<tr>
<td>JobBuilder</td>
<td><code>ofType(Class&lt;? extends Job&gt; jobClass)</code></td>
<td>Set the class which will be instantiated and executed when a Trigger fires that is associated with this JobDetail.</td>
</tr>
<tr>
<td>JobBuilder</td>
<td><code>requestRecovery()</code></td>
<td>Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.</td>
</tr>
<tr>
<td>JobBuilder</td>
<td><code>requestRecovery(boolean shouldRecover)</code></td>
<td>Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.</td>
</tr>
<tr>
<td>JobBuilder</td>
<td><code>storeDurably()</code></td>
<td>Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).</td>
</tr>
<tr>
<td>JobBuilder</td>
<td><code>storeDurably(boolean durability)</code></td>
<td>Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).</td>
</tr>
<tr>
<td>JobBuilder</td>
<td><code>usingJobData(JobDataMap newJobDataMap)</code></td>
<td>Set the JobDetail's <code>JobDataMap</code>, adding any</td>
</tr>
</tbody>
</table>
Values to it that were already set on this JobBuilder using any of the other 'usingJobData' methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobBuilder</strong> <strong>usingJobData</strong>(String key, Boolean value)</td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>usingJobData</strong>(String key, Double value)</td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>usingJobData</strong>(String key, Float value)</td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>usingJobData</strong>(String key, Integer value)</td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>usingJobData</strong>(String key, Long value)</td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>usingJobData</strong>(String key, String value)</td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobBuilder</strong> <strong>withDescription</strong>(String description)</td>
<td>Set the given (human-meaningful) description of the Job.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>withIdentity</strong>(JobKey key)</td>
<td>Use a JobKey to identify the JobDetail.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>withIdentity</strong>(String name)</td>
<td>Use a JobKey with the given name and default group to identify the JobDetail.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong> <strong>withIdentity</strong>(String name, String group)</td>
<td>Use a JobKey with the given name and group to identify the JobDetail.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
newJob

public static JobBuilder newJob()

Create a JobBuilder with which to define a JobDetail.

Returns:
  a new JobBuilder

newJob

public static JobBuilder newJob(Class<? extends Job> jobClass)

Create a JobBuilder with which to define a JobDetail, and set the class name of the Job to be executed.
Returns:
a new JobBuilder

build

public JobDetail build()

Produce the JobDetail instance defined by this JobBuilder.

Returns:
the defined JobDetail.

withIdentity

public JobBuilder withIdentity(String name)
Use a JobKey with the given name and default group to identify the JobDetail.

If none of the 'withIdentity' methods are set on the JobBuilder, then a random, unique JobKey will be generated.

Parameters:
name - the name element for the Job's JobKey

Returns:
the updated JobBuilder

See Also:
JobKey,
JobDetail.getKey()

withIdentity

public JobBuilder withIdentity(String name,
String group)

Use a JobKey with the given name and group to identify the JobDetail.

If none of the 'withIdentity' methods are set on the JobBuilder, then a random, unique JobKey will be generated.
Parameters:
name - the name element for the Job's JobKey
group - the group element for the Job's JobKey

Returns:
the updated JobBuilder

See Also:
JobKey,
JobDetail.getKey()

---

**withIdentity**

```java
public JobBuilder withIdentity(JobKey key)
```

Use a JobKey to identify the JobDetail.

If none of the 'withIdentity' methods are set on the JobBuilder, then a random, unique JobKey will be generated.

Parameters:
key - the Job's JobKey

Returns:
the updated JobBuilder

See Also:
JobKey,
JobDetail.getKey()
withDescription

public JobBuilder withDescription(String description)

Set the given (human-meaningful) description of the Job.

Parameters:
  description - the description for the Job

Returns:
  the updated JobBuilder

See Also:
  JobDetail.getDescription()

ofType

public JobBuilder ofType(Class<? extends Job> jobClass)

Set the class which will be instantiated and executed when a
Trigger fires that is associated with this JobDetail.

**Parameters:**
jobClass - a class implementing the Job interface.

**Returns:**
the updated JobBuilder

**See Also:**
JobDetail.getJobClass()

---

**requestRecovery**

public JobBuilder requestRecovery()

Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation encountered.

If not explicitly set, the default value is false.

**Returns:**
the updated JobBuilder

**See Also:**
requestRecovery

public JobBuilder requestRecovery(boolean shouldRecover)

Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation encountered.

If not explicitly set, the default value is false.

Parameters:
  shouldRecover -

Returns:
  the updated JobBuilder

---

storeDurably
public JobBuilder storeDurably()

Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

If not explicitly set, the default value is false.

Returns: the updated JobBuilder

See Also: JobDetail.isDurable()

storeDurably

public JobBuilder storeDurably(boolean durability)

Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).
If not explicitly set, the default value is false.

**Parameters:**
- durability - the value to set for the durability property.

**Returns:**
- the updated JobBuilder

**See Also:**
- JobDetail.isDurable()

---

**usingJobData**

```java
public JobBuilder usingJobData(String key,
                               String value)
```

Add the given key-value pair to the JobDetail's JobDataMap.

**Returns:**
- the updated JobBuilder

**See Also:**
- JobDetail.getJobDataMap()
**usingJobData**

public JobBuilder usingJobData(String key, Integer value)

Add the given key-value pair to the JobDetail's JobDataMap.

**Returns:**
the updated JobBuilder

**See Also:**
JobDetail.getJobDataMap()

**usingJobData**

public JobBuilder usingJobData(String key, Long value)

Add the given key-value pair to the JobDetail's JobDataMap.
Returns:
the updated JobBuilder

See Also:
JobDetail.getJobDataMap()

usingJobData

public JobBuilder usingJobData(String key, Float value)

Add the given key-value pair to the JobDetail's JobDataMap.

Returns:
the updated JobBuilder

See Also:
JobDetail.getJobDataMap()
Add the given key-value pair to the JobDetail's `JobDataMap`.

**Returns:**
the updated JobBuilder

**See Also:**
`JobDetail.getJobDataMap()`
public JobBuilder usingJobData(JobDataMap newJobDataMap)

Set the JobDetail's JobDataMap, adding any values to it that were already set on this JobBuilder using any of the other 'usingJobData' methods.

Returns:  
the updated JobBuilder

See Also:  
JobDetail.getJobDataMap()
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>

| SUMMARY: NESTED | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD |
**org.quartz** **Class JobDataMap**

```java
java.lang.Object
  └ org.quartz.util.DirtyFlagMap
      └ org.quartz.util.StringKeyDirtyFlagMap
          └ org.quartz.JobDataMap
```

**All Implemented Interfaces:**

`Serializable`, `Cloneable`, `Map`

---

```java
public class JobDataMap
  extends StringKeyDirtyFlagMap
  implements Serializable
```

Holds state information for Job instances.

JobDataMap instances are stored once when the Job is added to a scheduler. They are also re-persisted after every execution of StatefulJob instances.

JobDataMap instances can also be stored with a Trigger. This can be useful in the case where you have a Job that is stored in the scheduler for regular/repeated use by multiple Triggers, yet with each independent triggering, you want to supply the Job with different data inputs.

The JobExecutionContext passed to a Job at execution time also contains a convenience JobDataMap that is the result of merging the contents of the trigger's JobDataMap (if any) over the Job's JobDataMap (if any).

**Author:**

James House

**See Also:**

`Job`, `StatefulJob`, `Trigger`, `JobExecutionContext`, `Serialized Form`

---

**Nested Class Summary**
### Nested classes/interfaces inherited from interface java.util.Map

<table>
<thead>
<tr>
<th>Class/Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map.Entry&lt;K, V&gt;</td>
<td>Class representing a single entry in the map</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDataMap()</td>
<td>Create an empty JobDataMap.</td>
</tr>
<tr>
<td>JobDataMap(Map map)</td>
<td>Create a JobDataMap with the given data.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean getBooleanFromString(String key)</td>
<td>Retrieve the identified Boolean value from the JobDataMap.</td>
</tr>
<tr>
<td>boolean getBooleanValue(String key)</td>
<td>Retrieve the identified boolean value from the JobDataMap.</td>
</tr>
<tr>
<td>boolean getBooleanValueFromString(String key)</td>
<td>Retrieve the identified boolean value from the JobDataMap.</td>
</tr>
<tr>
<td>Character getCharacterFromString(String key)</td>
<td>Retrieve the identified Character value from the JobDataMap.</td>
</tr>
<tr>
<td>char getCharFromString(String key)</td>
<td>Retrieve the identified char value from the JobDataMap.</td>
</tr>
<tr>
<td>Double getDoubleFromString(String key)</td>
<td>Retrieve the identified Double value from the JobDataMap.</td>
</tr>
<tr>
<td>double getDoubleValue(String key)</td>
<td>Retrieve the identified double value from the JobDataMap.</td>
</tr>
<tr>
<td>double getDoubleValueFromString(String key)</td>
<td>Retrieve the identified double value from the JobDataMap.</td>
</tr>
<tr>
<td>Float getFloatFromString(String key)</td>
<td>Retrieve the identified Float value from the JobDataMap.</td>
</tr>
<tr>
<td>float getFloatValue(String key)</td>
<td>Retrieve the identified float value from the JobDataMap.</td>
</tr>
<tr>
<td>float getFloatValueFromString(String key)</td>
<td>Retrieve the identified float value from the JobDataMap.</td>
</tr>
<tr>
<td>Integer getIntegerFromString(String key)</td>
<td>Retrieve the identified Integer value from the JobDataMap.</td>
</tr>
</tbody>
</table>
Retrieve the identified `int` value from the JobDataMap.

```java
int getIntFromString(String key)
```

Retrieve the identified `int` value from the JobDataMap.

```java
int getIntValue(String key)
```

Retrieve the identified `Long` value from the JobDataMap.

```java
Long getLongFromString(String key)
```

Retrieve the identified `long` value from the JobDataMap.

```java
long getLongValue(String key)
```

Retrieve the identified `long` value from the JobDataMap.

```java
long getLongValueFromString(String key)
```

Adds the given `boolean` value as a string version to the Job's data map.

```java
void putAsString(String key, boolean value)
```

Adds the given `Boolean` value as a string version to the Job's data map.

```java
void putAsString(String key, Boolean value)
```

Adds the given `char` value as a string version to the Job's data map.

```java
void putAsString(String key, char value)
```

Adds the given `Character` value as a string version to the Job's data map.

```java
void putAsString(String key, Character value)
```

Adds the given `double` value as a string version to the Job's data map.

```java
void putAsString(String key, double value)
```

Adds the given `Double` value as a string version to the Job's data map.

```java
void putAsString(String key, Double value)
```

Adds the given `float` value as a string version to the Job's data map.

```java
void putAsString(String key, float value)
```

Adds the given `Float` value as a string version to the Job's data map.

```java
void putAsString(String key, Float value)
```
public JobDataMap()

    Create an empty JobDataMap.
JobDataMap

public JobDataMap(Map map)

Create a JobDataMap with the given data.

Method Detail

putAsString

public void putAsString(String key, boolean value)

Adds the given boolean value as a string version to the Job's data map.

putAsString

public void putAsString(String key, Boolean value)

Adds the given Boolean value as a string version to the Job's data map.

putAsString

public void putAsString(String key, char value)

Adds the given char value as a string version to the Job's data map.

putAsString

public void putAsString(String key, Character value)

Adds the given character value as a string version to the Job's data map.
putAsString

public void putAsString(String key, double value)

Adds the given double value as a string version to the Job's data map.

putAsString

public void putAsString(String key, Double value)

Adds the given Double value as a string version to the Job's data map.

putAsString

public void putAsString(String key, float value)

Adds the given float value as a string version to the Job's data map.

putAsString

public void putAsString(String key, Float value)

Adds the given Float value as a string version to the Job's data map.

putAsString

public void putAsString(String key, int value)

Adds the given int value as a string version to the Job's data map.
**putAsString**

```java
public void putAsString(String key, Integer value)
```

Adds the given Integer value as a string version to the Job's data map.

---

**putAsString**

```java
public void putAsString(String key, long value)
```

Adds the given long value as a string version to the Job's data map.

---

**putAsString**

```java
public void putAsString(String key, Long value)
```

Adds the given Long value as a string version to the Job's data map.

---

**getIntFromString**

```java
public int getIntFromString(String key)
```

Retrieve the identified int value from the JobDataMap.

**Throws:**

- `ClassCastException` - if the identified object is not a String.

---

**getIntValue**

```java
public int getIntValue(String key)
```

Retrieve the identified int value from the JobDataMap.
Throws:

ClassCastException - if the identified object is not a String or Integer.

---

getIntegerFromString

public Integer getIntegerFromString(String key)

Retrieve the identified int value from the JobDataMap.

Throws:

ClassCastException - if the identified object is not a String.

---

getBooleanValueFromString

public boolean getBooleanValueFromString(String key)

Retrieve the identified boolean value from the JobDataMap.

Throws:

ClassCastException - if the identified object is not a String.

---

getBooleanValue

public boolean getBooleanValue(String key)

Retrieve the identified boolean value from the JobDataMap.

Throws:

ClassCastException - if the identified object is not a String or Boolean.

---

getBooleanFromString

public Boolean getBooleanFromString(String key)
Retrieve the identified Boolean value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String.

---

**getCharFromString**

```java
public char getCharFromString(String key)
```

Retrieve the identified char value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String.

---

**getCharacterFromString**

```java
public Character getCharacterFromString(String key)
```

Retrieve the identified Character value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String.

---

**getDoubleValueFromString**

```java
public double getDoubleValueFromString(String key)
```

Retrieve the identified double value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String.

---

**getDoubleValue**

```java
public double getDoubleValue(String key)
```
Retrieve the identified double value from the JobDataMap.

Throws:
   ClassCastException - if the identified object is not a String or Double.

---

getDoubleFromString

class
  public Double getDoubleFromString(String key)

  Retrieve the identified Double value from the JobDataMap.

  Throws:
     ClassCastException - if the identified object is not a String.

---

getFloatValueFromString

class
  public float getFloatValueFromString(String key)

  Retrieve the identified float value from the JobDataMap.

  Throws:
     ClassCastException - if the identified object is not a String.

---

getFloatValue

class
  public float getFloatValue(String key)

  Retrieve the identified float value from the JobDataMap.

  Throws:
     ClassCastException - if the identified object is not a String or Float.

---

getFloatFromString

class
  public Float getFloatFromString(String key)
Retrieve the identified Float value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String.

---

**getLongValueFromString**

```java
public long getLongValueFromString(String key)
```

Retrieve the identified long value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String.

---

**getLongValue**

```java
public long getLongValue(String key)
```

Retrieve the identified long value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String or Long.

---

**getLongFromString**

```java
public Long getLongFromString(String key)
```

Retrieve the identified Long value from the JobDataMap.

**Throws:**

`ClassCastException` - if the identified object is not a String.
Copyright 2001-2011, Terracotta, Inc.
org.quartz Interface JobDetail

All Superinterfaces: Cloneable, Serializable

All Known Subinterfaces: LocalityJobDetail

All Known Implementing Classes: DelegatingLocalityJobDetail, JobDetailImpl

public interface JobDetail
extends Serializable, Cloneable

Conveys the detail properties of a given Job instance. JobDetails are to be created/defined with JobBuilder.

Quartz does not store an actual instance of a Job class, but instead allows you to define an instance of one, through the use of a JobDetail.

Jobs have a name and group associated with them, which should uniquely identify them within a single Scheduler.

Triggers are the 'mechanism' by which Jobs are scheduled. Many Triggers can point to the same Job, but a single Trigger can only point to one Job.

Author: James House

See Also: JobBuilder, Job, JobDataMap, Trigger

<table>
<thead>
<tr>
<th>Object</th>
<th>clone()</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>getDescription()</td>
</tr>
</tbody>
</table>
Return the description given to the Job instance by its creator (if any).

**jobBuilder**

`getJobBuilder()`  
Get a `JobBuilder` that is configured to produce a `JobDetail` identical to this one.

**Class<? extends Job>**

`getJobClass()`  
Get the instance of `Job` that will be executed.

**JobDataMap**

`getJobDataMap()`  
Get the `JobDataMap` that is associated with the `Job`.

**JobKey**

`getKey()`

**boolean**

`isConcurrentExectionDisallowed()`  
Whether or not the `Job` should remain stored after it is orphaned (no Triggers point to it).

`isDurable()`  
Whether or not the `Job` should remain stored after it is orphaned (no Triggers point to it).

`isPersistJobDataAfterExecution()`

`requestsRecovery()`  
Instructs the `Scheduler` whether or not the `Job` should be re-executed if a 'recovery' or 'fail-over' situation is encountered.

---

### Method Detail

**getKey**

`JobKey getKey()`

**getDescription**

`String getDescription()`  
Return the description given to the `Job` instance by its creator (if any).
Returns:
null if no description was set.

---

**getJobClass**

```
Class<? extends Job> getJobClass()
```

Get the instance of Job that will be executed.

---

**getJobDataMap**

```
JobDataMap getJobDataMap()
```

Get the JobDataMap that is associated with the Job.

---

**isDurable**

```
boolean isDurable()
```

Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

If not explicitly set, the default value is false.

Returns:
true if the Job should remain persisted after being orphaned.

---

**isPersistJobDataAfterExecution**

```
boolean isPersistJobDataAfterExecution()
```

Returns:
whether the associated Job class carries the PersistJobDataAfterExecution annotation.

See Also:
PersistJobDataAfterExecution
isConcurrentExectionDisallowed

boolean isConcurrentExectionDisallowed()

Returns:
whether the associated Job class carries the
DisallowConcurrentExecution annotation.

See Also:
DisallowConcurrentExecution

requestsRecovery

boolean requestsRecovery()

Instructs the Scheduler whether or not the Job should be re-executed if a
'recovery' or 'fail-over' situation is encountered.

If not explicitly set, the default value is false.

See Also:
JobExecutionContext.isRecovering()

clone

Object clone()

getJobBuilder

JobBuilder getJobBuilder()

Get a JobBuilder that is configured to produce a JobDetail identical to
this one.
Copyright 2001-2011, Terracotta, Inc.
public interface JobExecutionContext

A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.

The JobDataMap found on this object (via the getMergedJobDataMap() method) serves as a convenience - it is a merge of the JobDataMap found on the JobDetail and the one found on the Trigger, with the value in the latter overriding any same-named values in the former. *It is thus considered a 'best practice' that the execute code of a Job retrieve data from the JobDataMap found on this object* NOTE: Do not expect value 'set' into this JobDataMap to somehow be set back onto a StatefulJob's own JobDataMap.

JobExecutionContexts are also returned from the Scheduler.getCurrentlyExecutingJobs() method. These are the same instances as those passed into the jobs that are currently executing within the scheduler. The exception to this is when your application is using Quartz remotely (i.e. via RMI) - in which case you get a clone of the JobExecutionContexts, and their references to the Scheduler and Job instances have been lost (a clone of the JobDetail is still available - just not a handle to the job instance that is running).

Author:
James House

See Also:
getScheduler(), getMergedJobDataMap(), getJobDetail(), Job, Trigger, JobDataMap
### Object

**get**(Object key)
Get the value with the given key from the context's data map.

### Calendar

**getCalendar**( )
Get a handle to the Calendar referenced by the Trigger instance that fired the Job.

### Date

**getFireTime**( )
The actual time the trigger fired.

**getNextFireTime**( )
A Date representing the next time the trigger is scheduled to fire.

**getPreviousFireTime**( )
A Date representing the previous time the trigger fired.

### JobDetail

**getJobDetail**( )
Get the JobDetail associated with the Job.

### Job

**getJobInstance**( )
Get the instance of the Job that was created for this execution.

**getJobRunTime**( )
The amount of time the job ran for (in milliseconds).

### JobDataMap

**getMergedJobDataMap**( )
Returns a JobDataMap that contains the merged data for the current execution.

### Scheduler

**getScheduler**( )
Get a handle to the Scheduler instance that fired the Job.

### Trigger

**getTrigger**( )
Get a handle to the Trigger instance that fired the Job.

### boolean

**isRecovering**( )
If the Job is being re-executed because of a 'recovery' situation, this method will return true.

### void

**put**(Object key, Object value)
Put the value into the context's data map.
Put the specified value into the context's data map with the given key.

```java
void setResult(Object result)
```

Set the result (if any) of the Job's execution (the type of object set as the result is entirely up to the particular job).

### Method Detail

**getScheduler**

```java
Scheduler getScheduler()
```

Get a handle to the Scheduler instance that fired the Job.

**getTrigger**

```java
Trigger getTrigger()
```

Get a handle to the Trigger instance that fired the Job.

**getCalendar**

```java
Calendar getCalendar()
```

Get a handle to the Calendar referenced by the Trigger instance that fired the Job.

**isRecovering**

```java
boolean isRecovering()
```

If the Job is being re-executed because of a 'recovery' situation, this method will return true.
getRefireCount

```java
int getRefireCount()
```

getMergedJobDataMap

```java
JobDataMap getMergedJobDataMap()
```

Get the convenience JobDataMap of this execution context.

The JobDataMap found on this object serves as a convenience - it is a merge of the JobDataMap found on the JobDetail and the one found on the Trigger, with the value in the latter overriding any same-named values in the former. *It is thus considered a 'best practice' that the execute code of a Job retrieve data from the JobDataMap found on this object.*

NOTE: Do not expect value 'set' into this JobDataMap to somehow be set back onto a StatefulJob's own JobDataMap.

Attempts to change the contents of this map typically result in an IllegalStateException.

getJobDetail

```java
JobDetail getJobDetail()
```

Get the JobDetail associated with the Job.

getJobInstance

```java
Job getJobInstance()
```

Get the instance of the Job that was created for this execution.

Note: The Job instance is not available through remote scheduler interfaces.
getFireTime

```java
Date getFireTime()
```

The actual time the trigger fired. For instance the scheduled time may have been 10:00:00 but the actual fire time may have been 10:00:03 if the scheduler was too busy.

**Returns:**

- Returns the fireTime.

**See Also:**

- `getScheduledFireTime()`

---

getScheduledFireTime

```java
Date getScheduledFireTime()
```

The scheduled time the trigger fired for. For instance the scheduled time may have been 10:00:00 but the actual fire time may have been 10:00:03 if the scheduler was too busy.

**Returns:**

- Returns the scheduledFireTime.

**See Also:**

- `getFireTime()`

---

getPreviousFireTime

```java
Date getPreviousFireTime()
```

---

getNextFireTime

```java
Date getNextFireTime()
```

---

getResult
Object getResult()

Returns the result (if any) that the Job set before its execution completed (the type of object set as the result is entirely up to the particular job).

The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Returns:  
Returns the result.

---

void setResult(Object result)

Set the result (if any) of the Job's execution (the type of object set as the result is entirely up to the particular job).

The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

---

long getJobRunTime()

The amount of time the job ran for (in milliseconds). The returned value will be -1 until the job has actually completed (or thrown an exception), and is therefore generally only useful to JobListeners and TriggerListeners.

Returns:  
Returns the jobRunTime.

---

void put(Object key, Object value)
Put the specified value into the context's data map with the given key. Possibly useful for sharing data between listeners and jobs.

NOTE: this data is volatile - it is lost after the job execution completes, and all TriggerListeners and JobListeners have been notified.

**Parameters:**
- **key** -
- **value** -

---

**get**

Object get(Object key)

Get the value with the given key from the context's data map.

**Parameters:**
- **key** -
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAME</td>
</tr>
</tbody>
</table>
org.quartz Class JobExecutionException

java.lang.Object  
  java.lang.Throwable  
    java.lang.Exception  
      org.quartz.SchedulerException  
        org.quartz.JobExecutionException

All Implemented Interfaces:
    Serializable

public class JobExecutionException
  extends SchedulerException

An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.

Note that if the flag for 'refire immediately' is set, the flags for unscheduling the Job are ignored.

Author:
  James House

See Also:
  Job, JobExecutionContext, SchedulerException, Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobExecutionException()</td>
<td>Create a JobExecutionException, with the 're-fire immediately' flag set to false.</td>
</tr>
<tr>
<td>JobExecutionException(boolean refireImmediately)</td>
<td>Create a JobExecutionException with the 're-fire immediately' flag set to the given value.</td>
</tr>
<tr>
<td>JobExecutionException(String msg)</td>
<td></td>
</tr>
</tbody>
</table>
Create a JobExcuctionException, with the given message.

**JobExecutionException**(String msg, boolean refireImmediately)
Create a JobExcuctionException with the given message and the 're-fire immediately' flag set to the given value.

**JobExecutionException**(String msg, Throwable cause)
Create a JobExecutionException with the given message, and underlying exception.

**JobExecutionException**(String msg, Throwable cause, boolean refireImmediately)
Create a JobExecutionException with the given message, and underlying exception, and the 're-fire immediately' flag set to the given value.

**JobExecutionException**(Throwable cause)
Create a JobExecutionException, with the given cause.

**JobExecutionException**(Throwable cause, boolean refireImmediately)
Create a JobExecutionException with the given underlying exception, and the 're-fire immediately' flag set to the given value.

### Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>refireImmediately()</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>setRefireImmediately(boolean refire)</td>
</tr>
<tr>
<td>void</td>
<td>setUnscheduleAllTriggers(boolean unscheduleAllTriggs)</td>
</tr>
<tr>
<td>void</td>
<td>setUnscheduleFiringTrigger(boolean unscheduleTrigg)</td>
</tr>
<tr>
<td>boolean</td>
<td>unscheduleAllTriggers()</td>
</tr>
<tr>
<td>boolean</td>
<td>unscheduleFiringTrigger()</td>
</tr>
</tbody>
</table>

**Methods inherited from class org.quartz.SchedulerException**
getUnderlyingException, toString
Methods inherited from class java.lang.**Throwable**
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.**Object**
cloner, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

**Constructor Detail**

### JobExecutionException

public **JobExecutionException**()

Create a JobExcecutionException, with the 're-fire immediately' flag set to false.

### JobExecutionException

public **JobExecutionException**(Throwable cause)

Create a JobExcecutionException, with the given cause.

### JobExecutionException

public **JobExecutionException**(String msg)

Create a JobExcecutionException, with the given message.

### JobExecutionException

public **JobExecutionException**(boolean refireImmediately)
Create a JobExecutionException with the 're-fire immediately' flag set to the given value.

**JobExecutionException**

```java
public JobExecutionException(Throwable cause, boolean refireImmediately)
```

Create a JobExecutionException with the given underlying exception, and the 're-fire immediately' flag set to the given value.

**JobExecutionException**

```java
public JobExecutionException(String msg, Throwable cause)
```

Create a JobExecutionException with the given message, and underlying exception.

**JobExecutionException**

```java
public JobExecutionException(String msg, Throwable cause, boolean refireImmediately)
```

Create a JobExecutionException with the given message, and underlying exception, and the 're-fire immediately' flag set to the given value.

**JobExecutionException**

```java
public JobExecutionException(String msg, boolean refireImmediately)
```

Create a JobExecutionException with the given message and the 're-fire immediately' flag set to the given value.
Method Detail

setRefireImmediately

public void setRefireImmediately(boolean refire)

refireImmediately

public boolean refireImmediately()

set UnscheduleFiringTrigger

public void setUnscheduleFiringTrigger(boolean unscheduleTrigg)

unscheduleFiringTrigger

public boolean unscheduleFiringTrigger()

setUnscheduleAllTriggers

public void setUnscheduleAllTriggers(boolean unscheduleAllTriggs)

unscheduleAllTriggers

public boolean unscheduleAllTriggers()
Class JobKey

All Implemented Interfaces:
    Serializable, Comparable<Key>

public final class JobKey
    extends Key<JobKey>

Uniquely identifies a JobDetail.

Keys are composed of both a name and group, and the name must be unique within the group. If only a group is specified then the default group name will be used.

Quartz provides a builder-style API for constructing scheduling-related entities via a Domain-Specific Language (DSL). The DSL can best be utilized through the usage of static imports of the methods on the classes TriggerBuilder, JobBuilder, DateBuilder, JobKey, TriggerKey and the various ScheduleBuilder implementations.

Client code can then use the DSL to write code such as this:

```java
    JobDetail job = newJob(MyJob.class)
        .withIdentity("myJob")
        .build();

    Trigger trigger = newTrigger()
        .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
        .withSchedule(simpleSchedule()
            .withIntervalInHours(1)
            .repeatForever())
        .startAt(futureDate(10, MINUTES))
        .build();

    scheduler.scheduleJob(job, trigger);
```
See Also:
Job,
Key.DEFAULT_GROUP,
Serialized Form

Field Summary

Fields inherited from class org.quartz.utils.Key
DEFAULT_GROUP

Constructor Summary

JobKey(String name)

JobKey(String name, String group)

Method Summary

static JobKey
### Constructor Detail

**JobKey**

```java
public JobKey(String name)
```

**JobKey**

```java
public static JobKey jobKey(String name, String group)
```
public JobKey(String name, String group)

| Method Detail |

jobKey

public static JobKey jobKey(String name)

jobKey

public static JobKey jobKey(String name, String group)
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>DETAIL: FIELD</td>
</tr>
</tbody>
</table>
org.quartz Interface JobListener

All Known Implementing Classes:
   BroadcastJobListener, JobChainingJobListener, JobListenerSupport, LoggingJobHistoryPlugin, QuartzSchedulerMBeanImpl, SampledStatisticsImpl

public interface JobListener

The interface to be implemented by classes that want to be informed when a JobDetail executes. In general, applications that use a Scheduler will not have use for this mechanism.

Author:
   James House
See Also:
   Scheduler#addJobListener(JobListener, Matcher), Matcher, Job, JobExecutionContext, JobExecutionException, TriggerListener

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getName()</td>
</tr>
<tr>
<td>Get the name of the JobListener.</td>
</tr>
<tr>
<td>void jobExecutionVetoed(JobExecutionContext context)</td>
</tr>
<tr>
<td>Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.</td>
</tr>
<tr>
<td>void jobToBeExecuted(JobExecutionContext context)</td>
</tr>
<tr>
<td>Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).</td>
</tr>
<tr>
<td>void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)</td>
</tr>
<tr>
<td>Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.</td>
</tr>
</tbody>
</table>
Method Detail

getName

```java
String getName()
```

Get the name of the JobListener.

---

jobToBeExecuted

```java
void jobToBeExecuted(JobExecutionContext context)
```

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

This method will not be invoked if the execution of the Job was vetoed by a TriggerListener.

See Also:

`jobExecutionVetoed(JobExecutionContext)`

---

jobExecutionVetoed

```java
void jobExecutionVetoed(JobExecutionContext context)
```

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

See Also:

`jobToBeExecuted(JobExecutionContext)`

---

jobWasExecuted
void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

Copyright 2001-2011, Terracotta, Inc.
org.quartz Class JobPersistenceException

java.lang.Object
   ↓ java.lang.Throwable
      ↓ java.lang.Exception
         ↓ org.quartz.SchedulerException
            ↓ org.quartz.JobPersistenceException

All Implemented Interfaces:
   Serializable

Direct Known Subclasses:
   LocalityException, LockException, NoSuchDelegateException, ObjectAlreadyExistsException

public class JobPersistenceException

extends SchedulerException

An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.

Author:
   James House

See Also:
   Serialized Form

---

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobPersistenceException(String msg)</td>
</tr>
<tr>
<td>Create a JobPersistenceException with the given message.</td>
</tr>
<tr>
<td>JobPersistenceException(String msg, Throwable cause)</td>
</tr>
<tr>
<td>Create a JobPersistenceException with the given message and cause.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
</table>
Methods inherited from class org.quartz.SchedulerException
getUnderlyingException, toString

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

JobPersistenceException

public JobPersistenceException(String msg)

Create a JobPersistenceException with the given message.

JobPersistenceException

public JobPersistenceException(String msg, Throwable cause)

Create a JobPersistenceException with the given message and cause.

Copyright 2001-2011, Terracotta, Inc.
Client programs may be interested in the 'listener' interfaces that are available from Quartz. The JobListener interface provides notifications of Job executions. The TriggerListener interface provides notifications of Trigger firings. The SchedulerListener interface provides notifications of Scheduler events and errors. Listeners can be associated with local schedulers through the ListenerManager interface.

Since:
2.0 - previously listeners were managed directly on the Scheduler interface.

Author:
jhouse

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addJobListener(JobListener jobListener, List&lt;Matcher&lt;JobKey&gt;&gt; matchers)</td>
<td>Add the given JobListener to the Scheduler, and receive events for Jobs that are matched by ANY of the given Matchers.</td>
</tr>
<tr>
<td>void addJobListener(JobListener jobListener, Matcher&lt;JobKey&gt;... matchers)</td>
<td>Add the given JobListener to the Scheduler, and receive events for Jobs that are matched by ANY of the given Matchers.</td>
</tr>
<tr>
<td>boolean addJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)</td>
<td>Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td>void addSchedulerListener(SchedulerListener scheduleListener)</td>
<td>Add the given SchedulerListener to the set of listeners to which Scheduler events are sent.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void addTriggerListener(TriggerListener triggerListener, List&lt;Matcher&lt;TriggerKey&gt;&gt; matchers)</code></td>
<td>Add the given <code>TriggerListener</code> to the Scheduler to receive events for Triggers that are matched by ANY of the <code>Matchers</code>.</td>
</tr>
<tr>
<td><code>void addTriggerListener(TriggerListener triggerListener, Matcher&lt;TriggerKey&gt;... matchers)</code></td>
<td>Add the given <code>TriggerListener</code> to the Scheduler to receive events for Triggers that are matched by ANY of the <code>Matchers</code>.</td>
</tr>
<tr>
<td><code>boolean addTriggerListenerMatcher(String listenerName, Matcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Add the given <code>Matcher</code> to the set of matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td><code>JobListener getJobListener(String name)</code></td>
<td>Get the <code>JobListener</code> that has the given name.</td>
</tr>
<tr>
<td><code>List&lt;Matcher&lt;JobKey&gt;&gt; getJobListenerMatchers(String listenerName)</code></td>
<td>Get the set of <code>Matchers</code> for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td><code>List&lt;JobListener&gt; getJobListeners()</code></td>
<td>Get a List containing all of the <code>JobListeners</code> in the list.</td>
</tr>
<tr>
<td><code>List&lt;SchedulerListener&gt; getSchedulerListeners()</code></td>
<td>Get a List containing all of the <code>SchedulerListener</code>s with the Scheduler.</td>
</tr>
<tr>
<td><code>TriggerListener getTriggerListener(String name)</code></td>
<td>Get the <code>TriggerListener</code> that has the given name.</td>
</tr>
<tr>
<td><code>List&lt;Matcher&lt;TriggerKey&gt;&gt; getTriggerListenerMatchers(String listenerName)</code></td>
<td>Get the set of <code>Matchers</code> for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td><code>List&lt;TriggerListener&gt; getTriggerListeners()</code></td>
<td>Get a List containing all of the <code>TriggerListeners</code> with the Scheduler.</td>
</tr>
<tr>
<td><code>boolean removeJobListener(String name)</code></td>
<td>Remove the identified <code>JobListener</code> from the Scheduler.</td>
</tr>
<tr>
<td><code>boolean removeJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)</code></td>
<td>Remove the identified <code>Matcher</code> from the set of matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
</tbody>
</table>
Method Detail

addJobListener

void addJobListener(JobListener jobListener,
                    Matcher<JobKey>... matchers)

Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers. If no matchers are provided, the EverythingMatcher will be used.

See Also:  
Matcher, EverythingMatcher

addJobListener
void addJobListener(JobListener jobListener, List<Matcher<JobKey>> matchers)

Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers. If no matchers are provided, the EverythingMatcher will be used.

See Also:
Matcher, EverythingMatcher

addJobListenerMatcher

boolean addJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)

Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

Parameters:
listenerName - the name of the listener to add the matcher to
matcher - the additional matcher to apply for selecting events

Returns:
true if the identified listener was found and updated

Throws:
SchedulerException

removeJobListenerMatcher

boolean removeJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

Parameters:
listenerName - the name of the listener to add the matcher to
matcher - the additional matcher to apply for selecting events

Returns:
true if the given matcher was found and removed from the listener's list of matchers

**Throws:**

`SchedulerException`

---

### setJobListenerMatchers

```java
boolean setJobListenerMatchers(String listenerName,
                               List<Matcher<JobKey>> matchers)
```

Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

Removes any existing matchers for the identified listener!

**Parameters:**

- `listenerName` - the name of the listener to add the matcher to
- `matchers` - the matchers to apply for selecting events

**Returns:**

true if the given matcher was found and removed from the listener's list of matchers

**Throws:**

`SchedulerException`

---

### getJobListenerMatchers

```java
List<Matcher<JobKey>> getJobListenerMatchers(String listenerName)
```

Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

**Parameters:**

- `listenerName` - the name of the listener to add the matcher to

**Returns:**

the matchers registered for selecting events for the identified listener

**Throws:**

`SchedulerException`
**removeJobListener**

boolean \texttt{removeJobListener(String name)}

Remove the identified JobListener from the Scheduler.

**Returns:**
true if the identified listener was found in the list, and removed.

---

**getJobListeners**

\texttt{List\textless JobListener\textgreater getJobListeners()}

Get a List containing all of the JobListeners in the Scheduler.

---

**getJobListener**

\texttt{JobListener getJobListener(String name)}

Get the JobListener that has the given name.

---

**addTriggerListener**

void \texttt{addTriggerListener(TriggerListener triggerListener, Matcher\textless TriggerKey\textgreater ... matchers)}

Add the given TriggerListener to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers. If no matcher is provided, the EverythingMatcher will be used.

**See Also:**
Matcher, EverythingMatcher

---

**addTriggerListener**

void \texttt{addTriggerListener(TriggerListener triggerListener,}
Add the given TriggerListener to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers. If no matcher is provided, the EverythingMatcher will be used.

See Also: Matcher, EverythingMatcher

addTriggerListenerMatcher

boolean addTriggerListenerMatcher(String listenerName, Matcher<TriggerKey> matcher)

Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

Parameters:
   listenerName - the name of the listener to add the matcher to
   matcher - the additional matcher to apply for selecting events

Returns:
   true if the identified listener was found and updated

Throws:
   SchedulerException

removeTriggerListenerMatcher

boolean removeTriggerListenerMatcher(String listenerName, Matcher<TriggerKey> matcher)

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

Parameters:
   listenerName - the name of the listener to add the matcher to
   matcher - the additional matcher to apply for selecting events

Returns:
   true if the given matcher was found and removed from the listener's
setTriggerListenerMatchers

boolean setTriggerListenerMatchers(String listenerName, List<Matcher<TriggerKey>> matcher)

Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

Removes any existing matchers for the identified listener!

Parameters:
- listenerName - the name of the listener to add the matcher to
- matchers - the matchers to apply for selecting events

Returns:
- true if the given matcher was found and removed from the listener's list of matchers

Throws:
- SchedulerException

getTriggerListenerMatchers

List<Matcher<TriggerKey>> getTriggerListenerMatchers(String listenerName)

Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

Parameters:
- listenerName - the name of the listener to add the matcher to

Returns:
- the matchers registered for selecting events for the identified listener

Throws:
- SchedulerException
**removeTriggerListener**

boolean **removeTriggerListener**(String name)

Remove the identified TriggerListener from the Scheduler.

**Returns:**
true if the identified listener was found in the list, and removed.

---

**getTriggerListeners**

List<TriggerListener> **getTriggerListeners**()

Get a List containing all of the TriggerListeners in the Scheduler.

---

**getTriggerListener**

TriggerListener **getTriggerListener**(String name)

Get the TriggerListener that has the given name.

---

**addSchedulerListener**

void **addSchedulerListener**(SchedulerListener schedulerListener)

Register the given SchedulerListener with the Scheduler.

---

**removeSchedulerListener**

boolean **removeSchedulerListener**(SchedulerListener schedulerListener)

Remove the given SchedulerListener from the Scheduler.

**Returns:**
true if the identified listener was found in the list, and removed.
getSchedulerListeners

`List<SchedulerListener> getSchedulerListeners()`

Get a List containing all of the `SchedulerListener` objects registered with the `Scheduler`.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
org.quartz Interface Matcher<T extends Key>

All Superinterfaces:
Serializable

All Known Implementing Classes:
AndMatcher, EverythingMatcher, GroupMatcher, KeyMatcher, NameMatcher, NotMatcher, OrMatcher, StringMatcher

public interface Matcher<T extends Key>
extends Serializable

Matchers can be used in various Scheduler API methods to select the entities that should be operated upon.

Since:
2.0
Author:
jhouse

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals(Object)</td>
<td>compares keys</td>
</tr>
<tr>
<td>hashCode()</td>
<td>computes hash code</td>
</tr>
<tr>
<td>isMatch(T)</td>
<td>checks if a match</td>
</tr>
</tbody>
</table>

Method Detail

isMatch
boolean isMatch(T key)

hashCode

int hashCode()

Overrides:
    hashCode in class Object

equals

boolean equals(Object obj)

Overrides:
    equals in class Object

Copyright 2001-2011, Terracotta, Inc.
Class ObjectAlreadyExistsException

All Implemented Interfaces:
    Serializable

public class ObjectAlreadyExistsException
    extends JobPersistenceException

An exception that is thrown to indicate that an attempt to store a new object (i.e. JobDetail, Trigger or Calendar) in a Scheduler failed, because one with the same name & group already exists.

Author:
    James House

See Also:
    Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor</td>
<td>Create a ObjectAlreadyExistsException and auto-generate a message using the name/group from the given JobDetail.</td>
</tr>
<tr>
<td>Constructor</td>
<td>Create a ObjectAlreadyExistsException with the given message.</td>
</tr>
<tr>
<td>Constructor</td>
<td>Create a ObjectAlreadyExistsException and auto-generate a message using the name/group from the given Trigger.</td>
</tr>
</tbody>
</table>
### Method Summary

#### Methods inherited from class org.quartz.SchedulerException
- getUnderlyingException, toString

#### Methods inherited from class java.lang Throwable
- fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace
- printStackTrace, setStackTrace

#### Methods inherited from class java.lang Object
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

### Constructor Detail

#### ObjectAlreadyExistsException

public `ObjectAlreadyExistsException(String msg)`

Create a ObjectAlreadyExistsException with the given message.

#### ObjectAlreadyExistsException

public `ObjectAlreadyExistsException(JobDetail offendingJob)`

Create a ObjectAlreadyExistsException and auto-generate a message using the name/group from the given JobDetail.

The message will read:
"Unable to store Job with name: '__' and group: '__', because one already exists with this identification."
public ObjectAlreadyExistsException(Trigger offendingTrigger)

Create a ObjectAlreadyExistsException and auto-generate a message using the name/group from the given Trigger.

The message will read:
"Unable to store Trigger with name: '__' and group: '__', because one already exists with this identification."
org.quartz

Interfaces
- Calendar
- CalendarIntervalTrigger
- CronTrigger
- InterruptableJob
- Job
- JobDetail
- JobExecutionContext
- JobListener
- ListenerManager
- Matcher
- Scheduler
- SchedulerFactory
- SchedulerListener
- SimpleTrigger
- StatefulJob
- Trigger
- TriggerListener

Classes
- CalendarIntervalScheduleBuilder
- CronExpression
- CronScheduleBuilder
- DateBuilder
- JobBuilder
- JobDataMap
- JobKey
- ScheduleBuilder
- SchedulerContext
- SchedulerMetaData
- SimpleScheduleBuilder
- Trigger.TriggerTimeComparator
- TriggerBuilder
- TriggerKey
- TriggerUtils

Enums
- DateBuilder.IntervalUnit
- Trigger.CompletedExecutionInstruction
- Trigger.TriggerState

Exceptions
- JobExecutionException
- JobPersistenceException
- ObjectAlreadyExistsException
- SchedulerConfigException
- SchedulerException
- UnableToInterruptJobException
Annotation Types

DisallowConcurrentExecution
ExecuteInJTATransaction
PersistJobDataAfterExecution
**Package org.quartz**

The main package of Quartz, containing the client-side interfaces.

**See:**  
Description

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Calendar</strong></td>
<td>An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.</td>
</tr>
<tr>
<td><strong>CalendarIntervalTrigger</strong></td>
<td>A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.</td>
</tr>
<tr>
<td><strong>CronTrigger</strong></td>
<td>The public interface for inspecting settings specific to a CronTrigger.</td>
</tr>
<tr>
<td><strong>InterruptableJob</strong></td>
<td>The interface to be implemented by Jobs that provide a mechanism for having their execution interrupted.</td>
</tr>
<tr>
<td><strong>Job</strong></td>
<td>The interface to be implemented by classes which represent a 'job' to be performed.</td>
</tr>
<tr>
<td><strong>JobDetail</strong></td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td><strong>JobExecutionContext</strong></td>
<td>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</td>
</tr>
<tr>
<td><strong>JobListener</strong></td>
<td>The interface to be implemented by classes that want to be informed when a JobDetail executes.</td>
</tr>
<tr>
<td><strong>ListenerManager</strong></td>
<td>Client programs may be interested in the 'listener' interfaces that are available from Quartz.</td>
</tr>
<tr>
<td><strong>Matcher&lt;T extends Key&gt;</strong></td>
<td>Matchers can be used in various Scheduler API methods to select the entities that should be operated upon.</td>
</tr>
<tr>
<td><strong>Scheduler</strong></td>
<td>This is the main interface of a Quartz Scheduler.</td>
</tr>
<tr>
<td></td>
<td>Provides a mechanism for obtaining client-usable</td>
</tr>
</tbody>
</table>
### SchedulerFactory
Handles to Scheduler instances.

### SchedulerListener
The interface to be implemented by classes that want to be informed of major Scheduler events.

### SimpleTrigger
A Trigger that is used to fire a Job at a given moment in time, and optionally repeated at a specified interval.

### StatefulJob
**Deprecated.** use DisallowConcurrentExecution and/or PersistJobDataAfterExecution annotations instead.

### Trigger
The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.

### TriggerListener
The interface to be implemented by classes that want to be informed when a Trigger fires.

---

### Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CalendarIntervalScheduleBuilder</strong></td>
<td>CalendarIntervalScheduleBuilder is a ScheduleBuilder that defines calendar time (day, week, month, year) interval-based schedules for Triggers.</td>
</tr>
<tr>
<td><strong>CronExpression</strong></td>
<td>Provides a parser and evaluator for unix-like cron expressions.</td>
</tr>
<tr>
<td><strong>CronScheduleBuilder</strong></td>
<td>CronScheduleBuilder is a ScheduleBuilder that defines CronExpression-based schedules for Triggers.</td>
</tr>
<tr>
<td><strong>DateBuilder</strong></td>
<td>DateBuilder is used to conveniently create java.util.Date instances that meet particular criteria.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong></td>
<td>JobBuilder is used to instantiate JobDetails.</td>
</tr>
<tr>
<td><strong>JobDataMap</strong></td>
<td>Holds state information for Job instances.</td>
</tr>
<tr>
<td><strong>JobKey</strong></td>
<td>Uniquely identifies a JobDetail.</td>
</tr>
<tr>
<td><strong>ScheduleBuilder&lt;T extends Trigger&gt;</strong></td>
<td>Holds context/environment data that can be made available to Jobs as they are executed.</td>
</tr>
<tr>
<td><strong>SchedulerContext</strong></td>
<td>Describes the settings and capabilities of a given Scheduler instance.</td>
</tr>
<tr>
<td><strong>SchedulerMetaData</strong></td>
<td>SimpleScheduleBuilder is a ScheduleBuilder that defines strict/literal interval-based schedules for Triggers.</td>
</tr>
<tr>
<td><strong>SimpleScheduleBuilder</strong></td>
<td>A Comparator that compares trigger's next fire times, or in other words, sorts them according to earliest next fire time.</td>
</tr>
<tr>
<td><strong>Trigger.TriggerTimeComparator</strong></td>
<td>TriggerBuilder is used to instantiate Triggers.</td>
</tr>
<tr>
<td><strong>TriggerBuilder&lt;T extends Trigger&gt;</strong></td>
<td>Uniquely identifies a Trigger.</td>
</tr>
<tr>
<td><strong>TriggerKey</strong></td>
<td>Convenience and utility methods for working with Triggers.</td>
</tr>
<tr>
<td><strong>TriggerUtils</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Enum Summary**

| **DateBuilder.IntervalUnit** | |
| **Trigger.CompletedExecutionInstruction** | NOOP Instructs the Scheduler that the Trigger has no further instructions. |
| **Trigger.TriggerState** | |

**Exception Summary**

<p>| <strong>JobExecutionException</strong> | An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be refired immediately (using the same |</p>
<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobExecutionContext</td>
<td>JobExecutionContext, or whether it wants to be unscheduled.</td>
</tr>
<tr>
<td>JobPersistenceException</td>
<td>An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.</td>
</tr>
<tr>
<td>ObjectAlreadyExistsException</td>
<td>An exception that is thrown to indicate that an attempt to store a new object (i.e.</td>
</tr>
<tr>
<td>SchedulerConfigException</td>
<td>An exception that is thrown to indicate that there is a misconfiguration of the SchedulerFactory - or one of the components it configures.</td>
</tr>
<tr>
<td>SchedulerException</td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td>UnableToInterruptJobException</td>
<td>An exception that is thrown to indicate that a call to InterruptableJob.interrupt() failed without interrupting the Job.</td>
</tr>
</tbody>
</table>

### Annotation Types Summary

<table>
<thead>
<tr>
<th>Annotation Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disallow Concurrent Execution</td>
<td>An annotation that marks a <code>Job</code> class as one that must not have multiple instances executed concurrently (where instance is based-upon a <code>JobDetail</code> definition - or in other words based upon a <code>JobKey</code>).</td>
</tr>
<tr>
<td>Execute In JTA Transaction</td>
<td>An annotation that marks a <code>Job</code> class as one that will have its execution wrapped by a JTA Transaction.</td>
</tr>
<tr>
<td>Persist Job Data After Execution</td>
<td>An annotation that marks a <code>Job</code> class as one that makes updates to its <code>JobDataMap</code> during execution, and wishes the scheduler to restore the <code>JobDataMap</code> when execution completes.</td>
</tr>
</tbody>
</table>
Package org.quartz Description

The main package of Quartz, containing the client-side interfaces.

See the Quartz project for more information.
Hierarchy For Package org.quartz

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.**CronExpression** (implements java.lang.**Cloneable**, java.io.**Serializable**)
  - org.quartz.**DateBuilder**
  - org.quartz.util.**DirtyFlagMap** (implements java.lang.**Cloneable**, java.util.**Map**<K,V>, java.io.**Serializable**)
    - org.quartz.util.**StringKeyDirtyFlagMap**
    - org.quartz.**JobDataMap** (implements java.io.**Serializable**)
    - org.quartz.**SchedulerContext** (implements java.io.**Serializable**)
  - org.quartz.**JobBuilder**
  - org.quartz.util.**Key**<T> (implements java.lang.**Comparable**<T>, java.io.**Serializable**)
    - org.quartz.**JobKey**
    - org.quartz.**TriggerKey**
  - org.quartz.**ScheduleBuilder**<T>
    - org.quartz.**CalendarIntervalScheduleBuilder**
    - org.quartz.**CronScheduleBuilder**
    - org.quartz.**SimpleScheduleBuilder**
  - org.quartz.**SchedulerMetaData** (implements java.io.**Serializable**)
  - java.lang.**Runnable** (implements java.io.**Serializable**)
  - java.lang.**Exception**
    - org.quartz.**SchedulerException**
    - org.quartz.**JobExecutionException**
    - org.quartz.**JobPersistenceException**
    - org.quartz.**ObjectAlreadyExistsException**
    - org.quartz.**SchedulerConfigException**
    - org.quartz.**UnableToInterruptJobException**
  - org.quartz.**Trigger.****TriggerTimeComparator** (implements java.util.**Comparator**<T>, java.io.**Serializable**)
  - org.quartz.**TriggerBuilder**<T>
  - org.quartz.**TriggerUtils**
Interface Hierarchy

- java.lang.**Cloneable**
  - org.quartz.**Calendar** (also extends java.io.**Serializable**)
  - org.quartz.**JobDetail** (also extends java.io.**Serializable**)
  - org.quartz.**Trigger** (also extends java.lang.**Comparable**<T>, java.io.**Serializable**)
    - org.quartz.**CalendarIntervalTrigger**
    - org.quartz.**CronTrigger**
    - org.quartz.**SimpleTrigger**
- java.lang.**Comparable**<T>
  - org.quartz.**Trigger** (also extends java.lang.**Cloneable**, java.io.**Serializable**)
    - org.quartz.**CalendarIntervalTrigger**
    - org.quartz.**CronTrigger**
    - org.quartz.**SimpleTrigger**
- org.quartz.**Job**
  - org.quartz.**InterruptableJob**
  - org.quartz.**StatefulJob**
- org.quartz.**JobExecutionContext**
- org.quartz.**JobListener**
- org.quartz.**ListenerManager**
- org.quartz.**Scheduler**
- org.quartz.**SchedulerFactory**
- org.quartz.**SchedulerListener**
- java.io.**Serializable**
  - org.quartz.**Calendar** (also extends java.lang.**Cloneable**)
  - org.quartz.**JobDetail** (also extends java.lang.**Cloneable**)
  - org.quartz.**Matcher**<T>
  - org.quartz.**Trigger** (also extends java.lang.**Cloneable**, java.lang.**Comparable**<T>)
    - org.quartz.**CalendarIntervalTrigger**
    - org.quartz.**CronTrigger**
    - org.quartz.**SimpleTrigger**
- org.quartz.**TriggerListener**
Annotation Type Hierarchy

- org.quartz.**DisallowConcurrentExecution** (implements java.lang.annotation.**Annotation**)
- org.quartz.**ExecuteInJTATransaction** (implements java.lang.annotation.**Annotation**)
- org.quartz.**PersistJobDataAfterExecution** (implements java.lang.annotation.**Annotation**)


Enum Hierarchy

- java.lang.**Object**
  - java.lang.**Enum**<E> (implements java.lang.**Comparable<T>,
    java.io.**Serializable**)
    - org.quartz.**Trigger.CompletedExecutionInstruction**
    - org.quartz.**DateBuilder.IntervalUnit**
    - org.quartz.**Trigger.TriggerState**

Copyright 2001-2011, Terracotta, Inc.
# Uses of Package

**org.quartz**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of <strong>Quartz</strong>, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
<td>Contains the core classes and interfaces for the <strong>Quartz</strong> job scheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.core.jmx</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.ee.jmx.jboss</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.ee.jta</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.calendar</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore.oracle</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.matchers</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.triggers</strong></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td><strong>org.quartz.jobs</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.jobs.ee.ejb</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.jobs.ee.jms</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.jobs.ee.jmx</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.jobs.ee.mail</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.listeners</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.plugins</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.plugins.history</strong></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>org.quartz.plugins.management</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.plugins.xml</td>
<td></td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td></td>
</tr>
<tr>
<td>org.quartz.utils</td>
<td></td>
</tr>
<tr>
<td>org.quartz.xml</td>
<td></td>
</tr>
</tbody>
</table>

### Classes in `org.quartz` used by `org.quartz`

**Calendar**

An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.

**CalendarIntervalScheduleBuilder**

CalendarIntervalScheduleBuilder is a ScheduleBuilder that defines calendar time (day, week, month, year) interval-based schedules for Triggers.

**CalendarIntervalTrigger**

A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.

**CronScheduleBuilder**

CronScheduleBuilder is a ScheduleBuilder that defines CronExpression-based schedules for Triggers.

**CronTrigger**

The public interface for inspecting settings specific to a CronTrigger.

**DateBuilder**

DateBuilder is used to conveniently create java.util.Date instances that meet particular criteria.

**DateBuilder.IntervalUnit**

**DisallowConcurrentExecution**

An annotation that marks a Job class as one that must not have multiple instances executed concurrently (where instance is based-upon a JobDetail definition - or in other words based upon a JobKey).
<table>
<thead>
<tr>
<th><strong>Job</strong></th>
<th>The interface to be implemented by classes which represent a 'job' to be performed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobBuilder</strong></td>
<td>JobBuilder is used to instantiate <strong>JobDetail</strong>s.</td>
</tr>
<tr>
<td><strong>JobDataMap</strong></td>
<td>Holds state information for Job instances.</td>
</tr>
<tr>
<td><strong>JobDetail</strong></td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td><strong>JobExecutionContext</strong></td>
<td>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</td>
</tr>
<tr>
<td><strong>JobExecutionException</strong></td>
<td>An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.</td>
</tr>
<tr>
<td><strong>JobKey</strong></td>
<td>Uniquely identifies a <strong>JobDetail</strong>.</td>
</tr>
<tr>
<td><strong>JobListener</strong></td>
<td>The interface to be implemented by classes that want to be informed when a JobDetail executes.</td>
</tr>
<tr>
<td><strong>JobPersistenceException</strong></td>
<td>An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.</td>
</tr>
<tr>
<td><strong>ListenerManager</strong></td>
<td>Client programs may be interested in the 'listener' interfaces that are available from Quartz.</td>
</tr>
<tr>
<td><strong>Matcher</strong></td>
<td>Matchers can be used in various Scheduler API methods to select the entities that should be operated upon.</td>
</tr>
<tr>
<td><strong>PersistJobDataAfterExecution</strong></td>
<td>An annotation that marks a <strong>Job</strong> class as one that makes updates to its <strong>JobDataMap</strong> during execution, and wishes the scheduler to re-store the</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JobDataMap</td>
<td>When execution completes.</td>
</tr>
<tr>
<td>ScheduleBuilder</td>
<td>This is the main interface of a Quartz Scheduler.</td>
</tr>
<tr>
<td>SchedulerContext</td>
<td>Holds context/environment data that can be made available to Jobs as they are executed.</td>
</tr>
<tr>
<td>SchedulerException</td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td>SchedulerListener</td>
<td>The interface to be implemented by classes that want to be informed of major Scheduler events.</td>
</tr>
<tr>
<td>SchedulerMetaData</td>
<td>Describes the settings and capabilities of a given Scheduler instance.</td>
</tr>
<tr>
<td>SimpleScheduleBuilder</td>
<td>SimpleScheduleBuilder is a ScheduleBuilder that defines strict/literal interval-based schedules for Triggers.</td>
</tr>
<tr>
<td>SimpleTrigger</td>
<td>A Trigger that is used to fire a Job at a given moment in time, and optionally repeated at a specified interval.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.</td>
</tr>
<tr>
<td>Trigger.CompletedExecutionInstruction</td>
<td>NOOP Instructs the Scheduler that the Trigger has no further instructions.</td>
</tr>
<tr>
<td>Trigger.TriggerState</td>
<td></td>
</tr>
<tr>
<td>TriggerBuilder</td>
<td>TriggerBuilder is used to instantiate Triggers.</td>
</tr>
<tr>
<td>TriggerKey</td>
<td>Uniquely identifies a Trigger.</td>
</tr>
<tr>
<td>TriggerListener</td>
<td>The interface to be implemented by classes that want to be informed</td>
</tr>
</tbody>
</table>
when a Trigger fires.

**UnableToInterruptJobException**
An exception that is thrown to indicate that a call to InterruptableJob.interrupt() failed without interrupting the Job.

---

### Classes in org.quartz used by org.quartz.core

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.</td>
</tr>
<tr>
<td>JobDataMap</td>
<td>Holds state information for Job instances.</td>
</tr>
<tr>
<td>JobDetail</td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td>JobExecutionContext</td>
<td>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</td>
</tr>
<tr>
<td>JobExecutionException</td>
<td>An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.</td>
</tr>
<tr>
<td>JobKey</td>
<td>Uniquely identifies a JobDetail.</td>
</tr>
<tr>
<td>JobListener</td>
<td>The interface to be implemented by classes that want to be informed when a JobDetail executes.</td>
</tr>
<tr>
<td>JobPersistenceException</td>
<td>An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.</td>
</tr>
<tr>
<td>ListenerManager</td>
<td>Client programs may be interested in the 'listener' interfaces that are available from Quartz.</td>
</tr>
<tr>
<td>Matcher</td>
<td></td>
</tr>
</tbody>
</table>
Matchers can be used in various Scheduler API methods to select the entities that should be operated upon.

<table>
<thead>
<tr>
<th><strong>Scheduler</strong></th>
<th>This is the main interface of a Quartz Scheduler.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SchedulerConfigException</strong></td>
<td>An exception that is thrown to indicate that there is a misconfiguration of the SchedulerFactory- or one of the components it configures.</td>
</tr>
<tr>
<td><strong>SchedulerContext</strong></td>
<td>Holds context/environment data that can be made available to Jobs as they are executed.</td>
</tr>
<tr>
<td><strong>SchedulerException</strong></td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>SchedulerListener</strong></td>
<td>The interface to be implemented by classes that want to be informed of major Scheduler events.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.</td>
</tr>
<tr>
<td><strong>Trigger.CompletedExecutionInstruction</strong></td>
<td>NOOP Instructs the Scheduler that the Trigger has no further instructions.</td>
</tr>
<tr>
<td><strong>Trigger.TriggerState</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TriggerKey</strong></td>
<td>Uniquely identifies a Trigger.</td>
</tr>
<tr>
<td><strong>TriggerListener</strong></td>
<td>The interface to be implemented by classes that want to be informed when a Trigger fires.</td>
</tr>
<tr>
<td><strong>UnableToInterruptJobException</strong></td>
<td>An exception that is thrown to indicate that a call to InterruptableJob.interrupt() failed without interrupting the Job.</td>
</tr>
</tbody>
</table>

Classes in org.quartz used by org.quartz.core.jmx

| **CronTrigger** | |
The public interface for inspecting settings specific to a CronTrigger.

<table>
<thead>
<tr>
<th>JobDataMap</th>
<th>Holds state information for Job instances.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDetail</td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td>JobExecutionContext</td>
<td>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</td>
</tr>
<tr>
<td>SchedulerException</td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td>SimpleTrigger</td>
<td>A Trigger that is used to fire a Job at a given moment in time, and optionally repeated at a specified interval.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.</td>
</tr>
</tbody>
</table>

### Classes in org.quartz used by org.quartz.ee.jmx.jboss

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>This is the main interface of a Quartz Scheduler.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerException</td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
</tbody>
</table>

### Classes in org.quartz used by org.quartz.ee.jta

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>This is the main interface of a Quartz Scheduler.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerConfigException</td>
<td>An exception that is thrown to indicate that there is a misconfiguration of the SchedulerFactory- or one of the components it configures.</td>
</tr>
<tr>
<td>SchedulerException</td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
</tbody>
</table>
**SchedulerListener**

The interface to be implemented by classes that want to be informed of major Scheduler events.

---

### Classes in org.quartz used by org.quartz.impl

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Calendar</strong></td>
<td>An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.</td>
</tr>
<tr>
<td><strong>Job</strong></td>
<td>The interface to be implemented by classes which represent a 'job' to be performed.</td>
</tr>
<tr>
<td><strong>JobBuilder</strong></td>
<td>JobBuilder is used to instantiate JobDetails.</td>
</tr>
<tr>
<td><strong>JobDataMap</strong></td>
<td>Holds state information for Job instances.</td>
</tr>
<tr>
<td><strong>JobDetail</strong></td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td><strong>JobExecutionContext</strong></td>
<td>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</td>
</tr>
<tr>
<td><strong>JobKey</strong></td>
<td>Uniquely identifies a JobDetail.</td>
</tr>
<tr>
<td><strong>ListenerManager</strong></td>
<td>Client programs may be interested in the 'listener' interfaces that are available from Quartz.</td>
</tr>
<tr>
<td><strong>Scheduler</strong></td>
<td>This is the main interface of a Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>SchedulerContext</strong></td>
<td>Holds context/environment data that can be made available to Jobs as they are executed.</td>
</tr>
<tr>
<td><strong>SchedulerException</strong></td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>SchedulerFactory</strong></td>
<td></td>
</tr>
</tbody>
</table>
Provides a mechanism for obtaining client-usurable handles to Scheduler instances.

**SchedulerListener**
The interface to be implemented by classes that want to be informed of major Scheduler events.

**SchedulerMetaData**
Describes the settings and capabilities of a given Scheduler instance.

**Trigger**
The base interface with properties common to all Triggers - use `TriggerBuilder` to instantiate an actual Trigger.

**Trigger.TriggerState**

**TriggerKey**
Uniquely identifies a Trigger.

**UnableToInterruptJobException**
An exception that is thrown to indicate that a call to `InterruptableJob.interrupt()` failed without interrupting the Job.

---

### Classes in `org.quartz` used by `org.quartz.impl.calendar`

**Calendar**
An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.

**CronExpression**
Provides a parser and evaluator for unix-like cron expressions.

### Classes in `org.quartz` used by `org.quartz.impl.jdbcjobstore`

**Calendar**
An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.

**JobDataMap**
<table>
<thead>
<tr>
<th>Holds state information for Job instances.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobDetail</strong></td>
</tr>
<tr>
<td><strong>JobKey</strong></td>
</tr>
<tr>
<td><strong>JobPersistenceException</strong></td>
</tr>
<tr>
<td><strong>ObjectAlreadyExistsException</strong></td>
</tr>
<tr>
<td><strong>SchedulerConfigException</strong></td>
</tr>
<tr>
<td><strong>SchedulerException</strong></td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
</tr>
<tr>
<td><strong>Trigger.CompletedExecutionInstruction</strong></td>
</tr>
<tr>
<td><strong>Trigger.TriggerState</strong></td>
</tr>
<tr>
<td><strong>TriggerKey</strong></td>
</tr>
</tbody>
</table>

### Classes in [org.quartz](#) used by [org.quartz.impl.jdbcjobstore.oracle](#)

| **Calendar** | An interface to be implemented by objects that define spaces of time |
during which an associated Trigger may (not) fire.

<table>
<thead>
<tr>
<th><strong>JobDetail</strong></th>
<th>Conveys the detail properties of a given Job instance.</th>
</tr>
</thead>
</table>

### Classes in `org.quartz` used by `org.quartz.impl.matchers`

<table>
<thead>
<tr>
<th><strong>JobKey</strong></th>
<th>Uniquely identifies a <strong>JobDetail</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matcher</strong></td>
<td>Matchers can be used in various <strong>Scheduler</strong> API methods to select the entities that should be operated upon.</td>
</tr>
<tr>
<td><strong>TriggerKey</strong></td>
<td>Uniquely identifies a <strong>Trigger</strong>.</td>
</tr>
</tbody>
</table>

### Classes in `org.quartz` used by `org.quartz.impl.triggers`

<table>
<thead>
<tr>
<th><strong>Calendar</strong></th>
<th>An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CalendarIntervalTrigger</strong></td>
<td>A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.</td>
</tr>
<tr>
<td><strong>CronExpression</strong></td>
<td>Provides a parser and evaluator for unix-like cron expressions.</td>
</tr>
<tr>
<td><strong>CronTrigger</strong></td>
<td>The public interface for inspecting settings specific to a CronTrigger.</td>
</tr>
<tr>
<td><strong>DateBuilder.IntervalUnit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>JobDataMap</strong></td>
<td>Holds state information for Job instances.</td>
</tr>
<tr>
<td><strong>JobExecutionContext</strong></td>
<td></td>
</tr>
</tbody>
</table>
A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.

### JobExecutionException
An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.

### JobKey
Uniquely identifies a JobDetail.

### ScheduleBuilder

### SchedulerException
Base class for exceptions thrown by the Quartz Scheduler.

### SimpleTrigger
A Trigger that is used to fire a Job at a given moment in time, and optionally repeated at a specified interval.

### Trigger
The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.

### Trigger.CompletedExecutionInstruction
NOOP Instructs the Scheduler that the Trigger has no further instructions.

### TriggerBuilder
TriggerBuilder is used to instantiate Triggers.

### TriggerKey
Uniquely identifies a Trigger.

## Classes in org.quartz used by org.quartz.jobs

### DisallowConcurrentExecution
An annotation that marks a Job class as one that must not have multiple instances executed concurrently (where instance is based-upon a JobDetail definition - or in other words based upon a JobKey).

### Job
The interface to be implemented by classes which represent a 'job' to be performed.

**JobExecutionContext**
A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.

**JobExecutionException**
An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.

**PersistJobDataAfterExecution**
An annotation that marks a Job class as one that makes updates to its JobDataMap during execution, and wishes the scheduler to re-store the JobDataMap when execution completes.

---

**Classes in org.quartz used by org.quartz.jobs.ee.ejb**

**Job**
The interface to be implemented by classes which represent a 'job' to be performed.

**JobExecutionContext**
A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.

**JobExecutionException**
An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.

---

**Classes in org.quartz used by org.quartz.jobs.ee.jms**

**Job**
The interface to be implemented by classes which represent a 'job' to be
<table>
<thead>
<tr>
<th><strong>JobDataMap</strong></th>
<th>Holds state information for Job instances.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>JobExecutionContext</strong></th>
<th>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>JobExecutionContext</strong></th>
<th>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>JobExecutionException</strong></th>
<th>An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>SchedulerException</strong></th>
<th>Base class for exceptions thrown by the Quartz Scheduler.</th>
</tr>
</thead>
</table>

---

**Classes in org.quartz used by org.quartz.jobs.ee.jmx**

<table>
<thead>
<tr>
<th><strong>Job</strong></th>
<th>The interface to be implemented by classes which represent a 'job' to be performed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>JobExecutionContext</strong></th>
<th>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>JobExecutionContext</strong></th>
<th>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>JobExecutionException</strong></th>
<th>An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.</th>
</tr>
</thead>
</table>

**Classes in org.quartz used by org.quartz.jobs.ee.mail**

<table>
<thead>
<tr>
<th><strong>Job</strong></th>
<th>The interface to be implemented by classes which represent a 'job' to be performed.</th>
</tr>
</thead>
</table>
### JobDataMap
Holds state information for Job instances.

### JobExecutionContext
A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.

### JobExecutionException
An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.

### Classes in org.quartz used by org.quartz.listeners

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDetail</td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td>JobExecutionContext</td>
<td>A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.</td>
</tr>
<tr>
<td>JobExecutionException</td>
<td>An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.</td>
</tr>
<tr>
<td>JobKey</td>
<td>Uniquely identifies a JobDetail.</td>
</tr>
<tr>
<td>JobListener</td>
<td>The interface to be implemented by classes that want to be informed when a JobDetail executes.</td>
</tr>
<tr>
<td>SchedulerException</td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td>SchedulerListener</td>
<td>The interface to be implemented by classes that want to be informed of major Scheduler events.</td>
</tr>
</tbody>
</table>
### Trigger
The base interface with properties common to all Triggers - use `TriggerBuilder` to instantiate an actual Trigger.

#### Trigger.CompletedExecutionInstruction
- **NOOP** Instructs the Scheduler that the Trigger has no further instructions.

#### TriggerKey
Uniquely identifies a Trigger.

#### TriggerListener
The interface to be implemented by classes that want to be informed when a Trigger fires.

---

### Classes in `org.quartz` used by `org.quartz.locality`

#### Calendar
An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.

#### Job
The interface to be implemented by classes which represent a 'job' to be performed.

#### JobBuilder
JobBuilder is used to instantiate `JobDetail`.

#### JobDataMap
Holds state information for Job instances.

#### JobDetail
Conveys the detail properties of a given Job instance.

#### JobExecutionContext
A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes.

#### JobExecutionException
An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.)
<table>
<thead>
<tr>
<th><strong>JobKey</strong></th>
<th>Uniquely identifies a <a href="#">JobDetail</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobPersistenceException</strong></td>
<td>An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.</td>
</tr>
<tr>
<td><strong>ScheduleBuilder</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SchedulerException</strong></td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>The base interface with properties common to all Triggers - use <a href="#">TriggerBuilder</a> to instantiate an actual Trigger.</td>
</tr>
<tr>
<td><strong>Trigger.CompletedExecutionInstruction</strong></td>
<td>NOOP Instructs the Scheduler that the Trigger has no further instructions.</td>
</tr>
<tr>
<td><strong>TriggerBuilder</strong></td>
<td>TriggerBuilder is used to instantiate <a href="#">Triggers</a>.</td>
</tr>
<tr>
<td><strong>TriggerKey</strong></td>
<td>Uniquely identifies a <a href="#">Trigger</a>.</td>
</tr>
</tbody>
</table>

---

### Classes in `org.quartz` used by `org.quartz.plugins`

<table>
<thead>
<tr>
<th><strong>Scheduler</strong></th>
<th>This is the main interface of a Quartz Scheduler.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SchedulerException</strong></td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
</tbody>
</table>

---

### Classes in `org.quartz` used by `org.quartz.plugins.history`

<p>| <strong>JobExecutionContext</strong> | A context bundle containing handles to various environment information, that is given to a JobDetail instance as it is executed, and to a Trigger instance after the execution completes. |</p>
<table>
<thead>
<tr>
<th>JobExecutionException</th>
<th>An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same JobExecutionContext, or whether it wants to be unscheduled.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobListener</td>
<td>The interface to be implemented by classes that want to be informed when a JobDetail executes.</td>
</tr>
<tr>
<td>Scheduler</td>
<td>This is the main interface of a Quartz Scheduler.</td>
</tr>
<tr>
<td>SchedulerException</td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.</td>
</tr>
<tr>
<td>Trigger.CompletedExecutionInstruction</td>
<td>NOOP Instructs the Scheduler that the Trigger has no further instructions.</td>
</tr>
<tr>
<td>TriggerListener</td>
<td>The interface to be implemented by classes that want to be informed when a Trigger fires.</td>
</tr>
</tbody>
</table>

### Classes in org.quartz used by org.quartz.plugins.management

| Scheduler             | This is the main interface of a Quartz Scheduler.                                                          |
| SchedulerException    | Base class for exceptions thrown by the Quartz Scheduler.                                                  |

### Classes in org.quartz used by org.quartz.plugins.xml

<p>| Scheduler             | This is the main interface of a Quartz Scheduler.                                                          |</p>
<table>
<thead>
<tr>
<th><strong>SchedulerException</strong></th>
<th>Base class for exceptions thrown by the Quartz Scheduler.</th>
</tr>
</thead>
</table>

### Classes in `org.quartz` used by `org.quartz.simpl`

<table>
<thead>
<tr>
<th><strong>Calendar</strong></th>
<th>An interface to be implemented by objects that define spaces of time during which an associated Trigger may (not) fire.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Job</strong></td>
<td>The interface to be implemented by classes which represent a 'job' to be performed.</td>
</tr>
<tr>
<td><strong>JobDataMap</strong></td>
<td>Holds state information for Job instances.</td>
</tr>
<tr>
<td><strong>JobDetail</strong></td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td><strong>JobKey</strong></td>
<td>Uniquely identifies a JobDetail.</td>
</tr>
<tr>
<td><strong>JobPersistenceException</strong></td>
<td>An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.</td>
</tr>
<tr>
<td><strong>ObjectAlreadyExistsException</strong></td>
<td>An exception that is thrown to indicate that an attempt to store a new object (i.e.</td>
</tr>
<tr>
<td><strong>Scheduler</strong></td>
<td>This is the main interface of a Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>SchedulerConfigException</strong></td>
<td>An exception that is thrown to indicate that there is a misconfiguration of the SchedulerFactory- or one of the components it configures.</td>
</tr>
<tr>
<td><strong>SchedulerException</strong></td>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.</td>
</tr>
<tr>
<td><strong>Trigger.CompletedExecutionInstruction</strong></td>
<td>NOOP Instructs the Scheduler that the Trigger has no further</td>
</tr>
</tbody>
</table>
### Classes in `org.quartz` used by `org.quartz.utils`

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SchedulerException</strong></td>
</tr>
<tr>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
</tbody>
</table>

### Classes in `org.quartz` used by `org.quartz.xml`

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobDetail</strong></td>
</tr>
<tr>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td><strong>JobKey</strong></td>
</tr>
<tr>
<td>Uniquely identifies a <strong>JobDetail</strong>.</td>
</tr>
<tr>
<td><strong>Scheduler</strong></td>
</tr>
<tr>
<td>This is the main interface of a Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>SchedulerException</strong></td>
</tr>
<tr>
<td>Base class for exceptions thrown by the Quartz Scheduler.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
</tr>
<tr>
<td>The base interface with properties common to all Triggers - use <strong>TriggerBuilder</strong> to instantiate an actual Trigger.</td>
</tr>
<tr>
<td><strong>TriggerKey</strong></td>
</tr>
<tr>
<td>Uniquely identifies a <strong>Trigger</strong>.</td>
</tr>
</tbody>
</table>
An annotation that marks a Job class as one that makes updates to its JobDataMap during execution, and wishes the scheduler to re-store the JobDataMap when execution completes.

Jobs that are marked with this annotation should also seriously consider using the DisallowConcurrentExecution annotation, to avoid data storage race conditions with concurrently executing job instances.

This can be used in lieu of implementing the StatefulJob marker interface that was used prior to Quartz 2.0

Author:
   jhouse

See Also:
   DisallowConcurrentExecution
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
**Class ScheduleBuilder<T extends Trigger>**

**java.lang.Object**
  └ org.quartz.ScheduleBuilder<T>

**Direct Known Subclasses:**
  - CalendarIntervalScheduleBuilder
  - CronScheduleBuilder
  - SimpleScheduleBuilder

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScheduleBuilder()</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected abstract org.quartz.spi.MutableTrigger build()</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

**Constructor Detail**

**ScheduleBuilder**

public ScheduleBuilder()
Method Detail

build

protected abstract org.quartz.spi.MutableTrigger build()
**Interface Scheduler**

**All Known Implementing Classes:**
- JBoss4RMIRemoteMBeanScheduler, RemoteMBeanScheduler, RemoteScheduler, StdScheduler

---

```java
class Scheduler
```

This is the main interface of a Quartz Scheduler.

A Scheduler maintains a registry of JobDetails and Triggers. Once registered, the Scheduler is responsible for executing Job s when their associated Trigger s fire (when their scheduled time arrives).

Scheduler instances are produced by a SchedulerFactory. A scheduler that has already been created/initialized can be found and used through the same factory that produced it. After a Scheduler has been created, it is in "stand-by" mode, and must have its start() method called before it will fire any Jobs.

Job s are to be created by the 'client program', by defining a class that implements the Job interface. JobDetail objects are then created (also by the client) to define a individual instances of the Job. JobDetail instances can then be registered with the Scheduler via the scheduleJob(JobDetail, Trigger) or addJob(JobDetail, boolean) method.

Trigger s can then be defined to fire individual Job instances based on given schedules. SimpleTrigger s are most useful for one-time firings, or firing at an exact moment in time, with N repeats with a given delay between them. CronTrigger s allow scheduling based on time of day, day of week, day of month, and month of year.

Job s and Trigger s have a name and group associated with them, which should uniquely identify them within a single Scheduler. The 'group' feature may be useful for creating logical groupings or categorizations of Jobs s and Triggerss. If you don't have need for assigning a group to a given Jobs of Triggers, then you can use the DEFAULT_GROUP constant defined on this interface.
Stored jobs can also be 'manually' triggered through the use of the `triggerJob(String jobName, String jobGroup)` function.

Client programs may also be interested in the 'listener' interfaces that are available from Quartz. The `JobListener` interface provides notifications of Job executions. The `TriggerListener` interface provides notifications of Trigger firings. The `SchedulerListener` interface provides notifications of Scheduler events and errors. Listeners can be associated with local schedulers through the `ListenerManager` interface.

The setup/configuration of a Scheduler instance is very customizable. Please consult the documentation distributed with Quartz.

**Author:**
James House, Sharada Jambula

**See Also:**
Job, JobDetail, JobBuilder, Trigger, TriggerBuilder, JobListener, TriggerListener, SchedulerListener

---

### Field Summary

<table>
<thead>
<tr>
<th>Static String</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEFAULT_FAIL_OVER_GROUP</strong></td>
<td>A constant Trigger group name used internally by the scheduler - clients should not use the value of this constant (&quot;FAILED_OVER_JOBS&quot;) for the name of a Trigger's group.</td>
</tr>
<tr>
<td><strong>DEFAULT_GROUP</strong></td>
<td>A (possibly) useful constant that can be used for specifying the group that Job and Trigger instances belong to.</td>
</tr>
<tr>
<td><strong>DEFAULT_RECOVERY_GROUP</strong></td>
<td>A constant Trigger group name used internally by the scheduler - clients should not use the value of this constant (&quot;RECOVERING_JOBS&quot;) for the name of a Trigger's group.</td>
</tr>
<tr>
<td><strong>FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS</strong></td>
<td>A constant JobDataMap key that can be used to retrieve the scheduled fire time of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.</td>
</tr>
</tbody>
</table>
**FAILED_JOB_ORIGINAL_TRIGGER_GROUP**

A constant JobDataMap key that can be used to retrieve the group of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

**FAILED_JOB_ORIGINAL_TRIGGER_NAME**

A constant JobDataMap key that can be used to retrieve the name of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong> addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)**</td>
<td>Add (register) the given Calendar to the Scheduler.</td>
</tr>
<tr>
<td><strong>void</strong> addJob(JobDetail jobDetail, boolean replace)**</td>
<td>Add the given Job to the Scheduler - with no associated Trigger.</td>
</tr>
<tr>
<td><strong>boolean</strong> checkExists(JobKey jobKey)**</td>
<td>Determine whether a Job with the given identifier exists within the scheduler.</td>
</tr>
<tr>
<td><strong>boolean</strong> checkExists(TriggerKey triggerKey)**</td>
<td>Determine whether a Trigger with the given identifier exists within the scheduler.</td>
</tr>
<tr>
<td><strong>void</strong> clear()**</td>
<td>Clears (deletes!) all scheduling data - all Jobs, Trigger Calendars.</td>
</tr>
<tr>
<td><strong>boolean</strong> deleteCalendar(String calName)**</td>
<td>Delete the identified Calendar from the Scheduler</td>
</tr>
<tr>
<td><strong>boolean</strong> deleteJob(JobKey jobKey)**</td>
<td>Delete the identified Job from the Scheduler - and associated Triggers.</td>
</tr>
<tr>
<td><strong>boolean</strong> deleteJobs(List&lt;JobKey&gt; jobKeys)**</td>
<td>Delete the identified Jobs from the Scheduler - and associated Triggers.</td>
</tr>
<tr>
<td><strong>Calendar</strong> getCalendar(String calName)**</td>
<td>Get the Calendar instance with the given name.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>List&lt;String&gt; getCalendarNames()</td>
<td>Get the names of all registered calendars.</td>
</tr>
<tr>
<td>SchedulerContext getSchedulerContext()</td>
<td>Returns the SchedulerContext of the Scheduler.</td>
</tr>
<tr>
<td>List&lt;JobExecutionContext&gt; getCurrentlyExecutingJobs()</td>
<td>Return a list of JobExecutionContext objects that currently executing Jobs in this Scheduler instance.</td>
</tr>
<tr>
<td>JobDetail getJobDetail(JobKey jobKey)</td>
<td>Get the JobDetail for the Job instance with the given key.</td>
</tr>
<tr>
<td>List&lt;String&gt; getJobGroupNames()</td>
<td>Get the names of all known JobDetail groups.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
<td>Get the keys of all the JobDetails in the matching group.</td>
</tr>
<tr>
<td>ListenerManager getListenerManager()</td>
<td>Get a reference to the scheduler's ListenerManager which listeners may be registered.</td>
</tr>
<tr>
<td>SchedulerMetaData getSchedulerMetaData()</td>
<td>Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.</td>
</tr>
<tr>
<td>Set&lt;String&gt; getPausedTriggerGroups()</td>
<td>Get the names of all Trigger groups that are paused.</td>
</tr>
<tr>
<td>String getSchedulerInstanceId()</td>
<td>Returns the instance Id of the Scheduler.</td>
</tr>
<tr>
<td>String getSchedulerName()</td>
<td>Returns the name of the Scheduler.</td>
</tr>
<tr>
<td>Trigger getTrigger(TriggerKey triggerKey)</td>
<td>Get the Trigger instance with the given key.</td>
</tr>
<tr>
<td>List&lt;String&gt; getTriggerGroupNames()</td>
<td>Get the names of all known Trigger groups.</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt; getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Get the names of all the Triggers in the given group.</td>
</tr>
<tr>
<td>List&lt;? extends Trigger&gt; getTriggersOfJob(JobKey jobKey)</td>
<td>Get all Triggers that are associated with the identified JobDetail.</td>
</tr>
<tr>
<td>Trigger.TriggerState getTriggerState(TriggerKey triggerKey)</td>
<td></td>
</tr>
</tbody>
</table>
Get the current state of the identified Trigger.

```java
boolean interrupt(JobKey jobKey)
```
Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which implementor of the InterruptableJob interface.

```java
boolean isInStandbyMode()
```
Reports whether the Scheduler is in stand-by mode.

```java
boolean isShutdown()
```
Reports whether the Scheduler has been shutdown.

```java
boolean isStarted()
```
Whether the scheduler has been started.

```java
void pauseAll()
```
Pause all triggers - similar to calling `pauseTriggerGroup(group)` on every group, however, a `this method resumeAll()` must be called to clear the state of 'remembering' that all new triggers will be paused as they are added.

```java
void pauseJob(JobKey jobKey)
```
Pause the JobDetail with the given key - by pausing all current Triggers.

```java
void pauseJobs(GroupMatcher<JobKey> matcher)
```
Pause all of the JobDetails in the matching group - by pausing all of their Triggers.

```java
void pauseTrigger(TriggerKey triggerKey)
```
Pause the Trigger with the given key.

```java
void pauseTriggers(GroupMatcher<TriggerKey> matcher)
```
Pause all of the Triggers in the groups matching.

```java
Date rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)
```
Remove (delete) the Trigger with the given key, and store the new given one - which must be associated with the same Job (the new trigger must have the job name & group specified) - however, the new trigger need not have the same name as the old trigger.

```java
void resumeAll()
```
Resume (un-pause) all triggers - similar to calling `resumeTriggerGroup(group)` on every group.

```java
void resumeJob(JobKey jobKey)
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
<td>Resume (un-pause) all of the JobDetails in matching groups.</td>
</tr>
<tr>
<td>resumeTrigger(TriggerKey triggerKey)</td>
<td>Resume (un-pause) the Trigger with the given key.</td>
</tr>
<tr>
<td>resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Resume (un-pause) all of the Triggers in matching groups.</td>
</tr>
<tr>
<td>scheduleJob(JobDetail jobDetail, Trigger trigger)</td>
<td>Add the given JobDetail to the Scheduler, and associate the given Trigger with it.</td>
</tr>
<tr>
<td>scheduleJob(Trigger trigger)</td>
<td>Schedule the given Trigger with the Job identified by the Trigger's settings.</td>
</tr>
<tr>
<td>scheduleJobs(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; triggersAndJobs, boolean replace)</td>
<td>Schedule all of the given jobs with the related set of Triggers.</td>
</tr>
<tr>
<td>setJobFactory(org.quartz.spi.JobFactory factory)</td>
<td>Set the JobFactory that will be responsible for producing instances of Job classes.</td>
</tr>
<tr>
<td>shutdown()</td>
<td>Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler.</td>
</tr>
<tr>
<td>shutdown(boolean waitForJobsToComplete)</td>
<td>Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler.</td>
</tr>
<tr>
<td>standby()</td>
<td>Temporarily halts the Scheduler's firing of Triggers.</td>
</tr>
<tr>
<td>start()</td>
<td>Starts the Scheduler's threads that fire Triggers.</td>
</tr>
<tr>
<td>startDelayed(int seconds)</td>
<td>Calls {#start()} after the indicated number of seconds.</td>
</tr>
<tr>
<td>triggerJob(JobKey jobKey)</td>
<td>Trigger the identified JobDetail (execute it now).</td>
</tr>
<tr>
<td>triggerJob(JobKey jobKey, JobDataMap data)</td>
<td>Trigger the identified JobDetail (execute it now).</td>
</tr>
</tbody>
</table>
boolean unscheduleJob(TriggerKey triggerKey)
   Remove the indicated Trigger from the scheduler.

boolean unscheduleJobs(List<TriggerKey> triggerKeys)
   Remove all of the indicated Triggers from the scheduler.

Field Detail

DEFAULT_GROUP

static final String DEFAULT_GROUP

   A (possibly) useful constant that can be used for specifying the group that
   Job and Trigger instances belong to.

   See Also:
   Constant Field Values

DEFAULT_RECOVERY_GROUP

static final String DEFAULT_RECOVERY_GROUP

   A constant Trigger group name used internally by the scheduler - clients
   should not use the value of this constant ("RECOVERING_JOBS") for the
   name of a Trigger's group.

   See Also:
   JobDetail.requestsRecovery(), Constant Field Values

DEFAULT_FAIL_OVER_GROUP

static final String DEFAULT_FAIL_OVER_GROUP

   A constant Trigger group name used internally by the scheduler - clients
   should not use the value of this constant ("FAILED_OVER_JOBS") for the
   name of a Trigger's group.
See Also:
JobDetail.requestsRecovery(), Constant Field Values

### FAILED_JOB_ORIGINAL_TRIGGER_NAME

static final String FAILED_JOB_ORIGINAL_TRIGGER_NAME

A constant JobDataMap key that can be used to retrieve the name of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

See Also:
JobDetail.requestsRecovery(), Constant Field Values

### FAILED_JOB_ORIGINAL_TRIGGER_GROUP

static final String FAILED_JOB_ORIGINAL_TRIGGER_GROUP

A constant JobDataMap key that can be used to retrieve the group of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

See Also:
JobDetail.requestsRecovery(), Constant Field Values

### FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS

static final String FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS

A constant JobDataMap key that can be used to retrieve the scheduled fire time of the original Trigger from a recovery trigger's data map in the case of a job recovering after a failed scheduler instance.

See Also:
JobDetail.requestsRecovery(), Constant Field Values
getSchedulerName

```java
String getSchedulerName() throws SchedulerException
```

Returns the name of the Scheduler.

**Throws:**

SchedulerException

getSchedulerInstanceId

```java
String getSchedulerInstanceId() throws SchedulerException
```

Returns the instance Id of the Scheduler.

**Throws:**

SchedulerException

getContext

```java
SchedulerContext getContext() throws SchedulerException
```

Returns the SchedulerContext of the Scheduler.

**Throws:**

SchedulerException

start

```java
void start() throws SchedulerException
```

Starts the Scheduler's threads that fire Triggers. When a scheduler is first created it is in "stand-by" mode, and will not fire triggers. The scheduler can also be put into stand-by mode by calling the standby() method.

The misfire/recovery process will be started, if it is the initial call to this method on this scheduler instance.

**Throws:**

`SchedulerException` - if shutdown() has been called, or there is an error within the Scheduler.

**See Also:**

`startDelayed(int), standby(), shutdown()`

---

### startDelayed

`void startDelayed(int seconds)`

Throws `SchedulerException`

Calls `{#start()}` after the indicated number of seconds. (This call does not block). This can be useful within applications that have initializers that create the scheduler immediately, before the resources needed by the executing jobs have been fully initialized.

**Throws:**

`SchedulerException` - if shutdown() has been called, or there is an error within the Scheduler.

**See Also:**

`start(), standby(), shutdown()`

---

### isStarted

`boolean isStarted()`

Throws `SchedulerException`

Whether the scheduler has been started.

Note: This only reflects whether start() has ever been called on this Scheduler, so it will return `true` even if the Scheduler is currently in
standby mode or has been since shutdown.

**Throws:**

`SchedulerException`

**See Also:**

`start(), isShutdown(), isInStandbyMode()`

---

### standby

**void** `standby()`

**throws** `SchedulerException`

Temporarily halts the Scheduler's firing of Triggers.

When `start()` is called (to bring the scheduler out of stand-by mode), trigger misfire instructions will NOT be applied during the execution of the `start()` method - any misfires will be detected immediately afterward (by the JobStore's normal process).

The scheduler is not destroyed, and can be re-started at any time.

**Throws:**

`SchedulerException`

**See Also:**

`start(), pauseAll()`

---

### isInStandbyMode

**boolean** `isInStandbyMode()`

**throws** `SchedulerException`

Reports whether the Scheduler is in stand-by mode.

**Throws:**

`SchedulerException`

**See Also:**

`standby(), start()`
### shutdown

```java
class Scheduler
{
    void shutdown() throws SchedulerException
    {
        Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler. Equivalent to `shutdown(false)`.
        The scheduler cannot be re-started.
    }

   Throws:
        SchedulerException

    See Also:
        `shutdown(boolean)`
}
```

### shutdown

```java
class Scheduler
{
    void shutdown(boolean waitForJobsToComplete) throws SchedulerException
    {
        Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler.
        The scheduler cannot be re-started.

        Parameters:
            waitForJobsToComplete - if true the scheduler will not allow this method to return until all currently executing jobs have completed.

        Throws:
            SchedulerException

        See Also:
            `shutdown()`
}
```

### isShutdown

```java
class Scheduler
{
    boolean isShutdown() throws SchedulerException
    {
        Reports whether the Scheduler has been shutdown.
    }
}```
Throws:  
SchedulerException

getMetaData

SchedulerMetaData getMetaData()  
throws SchedulerException

Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.

Note that the data returned is an 'instantaneous' snap-shot, and that as soon as it's returned, the meta data values may be different.

Throws:  
SchedulerException

getCurrentlyExecutingJobs

List<JobExecutionContext> getCurrentlyExecutingJobs()  
throws SchedulerException

Return a list of JobExecutionContext objects that represent all currently executing Jobs in this Scheduler instance.

This method is not cluster aware. That is, it will only return Jobs currently executing in this Scheduler instance, not across the entire cluster.

Note that the list returned is an 'instantaneous' snap-shot, and that as soon as it's returned, the true list of executing jobs may be different. Also please read the doc associated with JobExecutionContext- especially if you're using RMI.

Throws:  
SchedulerException

See Also:  
JobExecutionContext
setJobFactory

```java
void setJobFactory(org.quartz.spi.JobFactory factory)
    throws SchedulerException
```

Set the JobFactory that will be responsible for producing instances of Job classes.

JobFactories may be of use to those wishing to have their application produce Job instances via some special mechanism, such as to give the opportunity for dependency injection.

**Throws:**
- SchedulerException

**See Also:**
- JobFactory

getListenerManager

```java
ListenerManager getListenerManager()
    throws SchedulerException
```

Get a reference to the scheduler's ListenerManager, through which listeners may be registered.

**Returns:**
- the scheduler's ListenerManager

**Throws:**
- SchedulerException - if the scheduler is not local

**See Also:**
- ListenerManager, JobListener, TriggerListener, SchedulerListener

scheduleJob

```java
Date scheduleJob(JobDetail jobDetail,
                 Trigger trigger)
    throws SchedulerException
```
Add the given JobDetail to the Scheduler, and associate the given Trigger with it.

If the given Trigger does not reference any Job, then it will be set to reference the Job passed with it into this method.

**Throws:**
SchedulesException - if the Job or Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

---

**scheduleJob**

```java
date scheduleJob(Trigger trigger)
throws SchedulesException
```

Schedule the given Trigger with the Job identified by the Trigger's settings.

**Throws:**
SchedulesException - if the indicated Job does not exist, or the Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

---

**scheduleJobs**

```java
void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs,
                  boolean replace)
throws SchedulesException
```

Schedule all of the given jobs with the related set of triggers.

If any of the given jobs or triggers already exist (or more specifically, if the keys are not unique) and the replace parameter is not set to true then an exception will be thrown.

**Throws:**
ObjectAlreadyExistsException - if the job/trigger keys are not unique and the replace flag is not set to true.
unscheduleJob

boolean unscheduleJob(TriggerKey triggerKey) throws SchedulerException

Remove the indicated Trigger from the scheduler.

If the related job does not have any other triggers, and the job is not durable, then the job will also be deleted.

Throws:  SchedulerException

unscheduleJobs

boolean unscheduleJobs(List<TriggerKey> triggerKeys) throws SchedulerException

Remove all of the indicated Triggers from the scheduler.

If the related job does not have any other triggers, and the job is not durable, then the job will also be deleted.

Note that while this bulk operation is likely more efficient than invoking unscheduleJob(TriggerKey triggerKey) several times, it may have the adverse affect of holding data locks for a single long duration of time (rather than lots of small durations of time).

Throws:  SchedulerException

rescheduleJob

Date rescheduleJob(TriggerKey triggerKey, Trigger newTrigger) throws SchedulerException


Remove (delete) the Trigger with the given key, and store the new given one - which must be associated with the same job (the new trigger must have the job name & group specified) - however, the new trigger need not have the same name as the old trigger.

**Parameters:**
- triggerKey - identity of the trigger to replace
- newTrigger - The new Trigger to be stored.

**Returns:**
- null if a Trigger with the given name & group was not found and removed from the store, otherwise the first fire time of the newly scheduled trigger.

**Throws:**
- SchedulerException

---

**addJob**

```java
void addJob(JobDetail jobDetail,
            boolean replace)
            throws SchedulerException
```

Add the given Job to the Scheduler - with no associated Trigger. The Job will be 'dormant' until it is scheduled with a Trigger, or Scheduler.triggerJob() is called for it.

The Job must by definition be 'durable', if it is not, SchedulerException will be thrown.

**Throws:**
- SchedulerException - if there is an internal Scheduler error, or if the Job is not durable, or a Job with the same name already exists, and replace is false.

---

**deleteJob**

```java
boolean deleteJob(JobKey jobKey)
            throws SchedulerException
```

---
Delete the identified Job from the Scheduler - and any associated Triggers.

**Returns:**
true if the Job was found and deleted.

**Throws:**
SchedulerException - if there is an internal Scheduler error.

---

### deleteJobs

```java
boolean deleteJobs(List<JobKey> jobKeys)
throws SchedulerException
```

Delete the identified Jobs from the Scheduler - and any associated Triggers.

Note that while this bulk operation is likely more efficient than invoking deleteJob(JobKey jobKey) several times, it may have the adverse affect of holding data locks for a single long duration of time (rather than lots of small durations of time).

**Returns:**
true if all of the Jobs were found and deleted, false if one or more were not deleted.

**Throws:**
SchedulerException - if there is an internal Scheduler error.

---

### triggerJob

```java
void triggerJob(JobKey jobKey)
throws SchedulerException
```

Trigger the identified JobDetail (execute it now).

**Throws:**
SchedulerException

---

### triggerJob
void triggerJob(JobKey jobKey, JobDataMap data) throws SchedulerException

Trigger the identified JobDetail (execute it now).

**Parameters:**
- data - the (possibly null) JobDataMap to be associated with the trigger that fires the job immediately.

**Throws:**
SchedulerException

---

void pauseJob(JobKey jobKey) throws SchedulerException

Pause the JobDetail with the given key - by pausing all of its current Triggers.

**Throws:**
SchedulerException

**See Also:**
resumeJob(JobKey)

---

void pauseJobs(GroupMatcher<JobKey> matcher) throws SchedulerException

Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.

The Scheduler will "remember" the groups paused, and impose the pause on any new jobs that are added to any of those groups until it is resumed.

**Parameters:**
- matcher - The matcher to evaluate against know groups

**Throws:**
SchedulerException - On error

See Also:
resumeJobs(org.quartz.impl.matchers.GroupMatcher)

---

**pauseTrigger**

void **pauseTrigger**(TriggerKey triggerKey)

throws SchedulerException

Pause the Trigger with the given key.

**Throws:**
SchedulerException

See Also:
resumeTrigger(TriggerKey)

---

**pauseTriggers**

void **pauseTriggers**(Matcher<TriggerKey> matcher)

throws SchedulerException

Pause all of the Triggers in the groups matching.

The Scheduler will "remember" all the groups paused, and impose the pause on any new triggers that are added to any of those groups until it is resumed.

**Parameters:**
matcher - The matcher to evaluate against know groups

**Throws:**
SchedulerException

See Also:
resumeTriggers(org.quartz.impl.matchers.GroupMatcher)

---

**resumeJob**

void **resumeJob**(JobKey jobKey)

throws SchedulerException
Resume (un-pause) the JobDetail with the given key.

If any of the Job's Triggers missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Throws:**
 SchedulerException

**See Also:**
 pauseJob(JobKey)

---

**resumeJobs**

```java
void resumeJobs(GroupMatcher<JobKey> matcher)
    throws SchedulerException
```

Resume (un-pause) all of the JobDetails in matching groups.

If any of the Jobs had Triggers that missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Parameters:**
 matcher - The matcher to evaluate against known paused groups

**Throws:**
 SchedulerException - On error

**See Also:**
 pauseJobs(GroupMatcher)

---

**resumeTrigger**

```java
void resumeTrigger(TriggerKey triggerKey)
    throws SchedulerException
```

Resume (un-pause) the Trigger with the given key.

If the Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Throws:**
 SchedulerException
resumeTriggers

```java
void resumeTriggers(GroupMatcher<TriggerKey> matcher)
    throws SchedulerException
```

Resume (un-pause) all of the Triggers in matching groups.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Parameters:**
- matcher - The matcher to evaluate against know paused groups

**Throws:**
- `SchedulerException` - On error

**See Also:**
- `pauseTriggers(org.quartz.impl.matchers.GroupMatcher)`

pauseAll

```java
void pauseAll()
    throws SchedulerException
```

Pause all triggers - similar to calling `pauseTriggerGroup(group)` on every group, however, after using this method `pauseAll()` must be called to clear the scheduler's state of 'remembering' that all new triggers will be paused as they are added.

When `pauseAll()` is called (to un-pause), trigger misfire instructions WILL be applied.

**Throws:**
- `SchedulerException`

**See Also:**
- `resumeAll()`,
- `pauseTriggers(org.quartz.impl.matchers.GroupMatcher)`,
- `pauseTrigger(TriggerKey)`
standby()

---

**resumeAll**

```java
to float resumeAll() throws SchedulerException
```

Resume (un-pause) all triggers - similar to calling resumeTriggerGroup(group) on every group.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Throws:**

SchedulerException

**See Also:**

pauseAll()

---

**getJobGroupNames**

```java
List<String> getJobGroupNames() throws SchedulerException
```

Get the names of all known JobDetail groups.

**Throws:**

SchedulerException

---

**getJobKeys**

```java
Set<JobKey> getJobKeys(GroupMatcher<JobKey> matcher) throws SchedulerException
```

Get the keys of all the JobDetails in the matching groups.

**Parameters:**

matcher - Matcher to evaluate against known groups

**Returns:**
Set of all keys matching

**Throws:**

`SchedulerException` - On error

---

**getTriggersOfJob**

```
List<? extends Trigger> getTriggersOfJob(JobKey jobKey)
```

Get all Trigger s that are associated with the identified JobDetail.

The returned Trigger objects will be snap-shots of the actual stored triggers. If you wish to modify a trigger, you must re-store the trigger afterward (e.g. see `rescheduleJob(TriggerKey, Trigger)`).

**Throws:**

`SchedulerException`

---

**getTriggerGroupNames**

```
List<String> getTriggerGroupNames()
```

Get the names of all known Trigger groups.

**Throws:**

`SchedulerException`

---

**getTriggerKeys**

```
Set<TriggerKey> getTriggerKeys(GroupMatcher<TriggerKey> matcher)
```

Get the names of all the Triggers in the given group.

**Parameters:**

`matcher` - Matcher to evaluate against known groups
Returns:
List of all keys matching

Throws:
SchedulerException - On error

getPausedTriggerGroups

Set<String> getPausedTriggerGroups() throws SchedulerException

Get the names of all Trigger groups that are paused.

Throws:
SchedulerException

getJobDetail

JobDetail getJobDetail(JobKey jobKey) throws SchedulerException

Get the JobDetail for the Job instance with the given key.

The returned JobDetail object will be a snap-shot of the actual stored JobDetail. If you wish to modify the JobDetail, you must re-store the JobDetail afterward (e.g. see addJob(JobDetail, boolean)).

Throws:
SchedulerException

getTrigger

Trigger getTrigger(TriggerKey triggerKey) throws SchedulerException

Get the Trigger instance with the given key.

The returned Trigger object will be a snap-shot of the actual stored trigger.
If you wish to modify the trigger, you must re-store the trigger afterward (e.g. see `rescheduleJob(TriggerKey, Trigger)`).

**Throws:**
- `SchedulerException`

---

**getTriggerState**

```java
Trigger.TriggerState getTriggerState(TriggerKey triggerKey)
throws SchedulerException
```

Get the current state of the identified Trigger.

**Throws:**
- `SchedulerException`

**See Also:**
- `Trigger.TriggerState`

---

**addCalendar**

```java
void addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)
throws SchedulerException
```

Add (register) the given calendar to the Scheduler.

**Parameters:**
- `updateTriggers` - whether or not to update existing triggers that referenced the already existing calendar so that they are 'correct' based on the new trigger.

**Throws:**
- `SchedulerException` - if there is an internal Scheduler error, or a Calendar with the same name already exists, and replace is false.

---

**deleteCalendar**
boolean deleteCalendar(String calName)
    throws SchedulerException

Delete the identified Calendar from the Scheduler.

Returns:
true if the Calendar was found and deleted.

Throws:
SchedulerException - if there is an internal Scheduler error.

---

getCalendar

Calendar getCalendar(String calName)
    throws SchedulerException

Get the Calendar instance with the given name.

Throws:
SchedulerException

---

getCalendarNames

List<String> getCalendarNames()
    throws SchedulerException

Get the names of all registered Calendars.

Throws:
SchedulerException

---

interrupt

boolean interrupt(JobKey jobKey)
    throws UnableToInterruptJobException

Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which must be an implementor of the InterruptableJob interface.
If more than one instance of the identified job is currently executing, the `InterruptableJob#interrupt()` method will be called on each instance. However, there is a limitation that in the case that `interrupt()` on one instances throws an exception, all remaining instances (that have not yet been interrupted) will not have their `interrupt()` method called.

If you wish to interrupt a specific instance of a job (when more than one is executing) you can do so by calling `getCurrentlyExecutingJobs()` to obtain a handle to the job instance, and then invoke `interrupt()` on it yourself.

This method is not cluster aware. That is, it will only interrupt instances of the identified `InterruptableJob` currently executing in this Scheduler instance, not across the entire cluster.

**Returns:**
true is at least one instance of the identified job was found and interrupted.

**Throws:**
`UnableToInterruptJobException` - if the job does not implement `InterruptableJob`, or there is an exception while interrupting the job.

**See Also:**
`InterruptableJob.interrupt()`, `getCurrentlyExecutingJobs()`

---

**checkExists**

```java
def checkExists(JobKey jobKey) throws SchedulerException
```

Determine whether a `Job` with the given identifier already exists within the scheduler.

**Parameters:**
jobKey - the identifier to check for

**Returns:**
true if a Job exists with the given identifier

**Throws:**
`SchedulerException`
checkExists

boolean checkExists(TriggerKey triggerKey)
    throws SchedulerException

Determine whether a Trigger with the given identifier already exists within the scheduler.

Parameters:
    triggerKey - the identifier to check for

Returns:
    true if a Trigger exists with the given identifier

Throws:
    SchedulerException

clear

void clear()
    throws SchedulerException

Clears (deletes!) all scheduling data - all Jobs, Triggers Calendars.

Throws:
    SchedulerException

Copyright 2001-2011, Terracotta, Inc.
**Class SchedulerConfigException**

```java
org.quartz
java.lang.Object
   ↓ java.lang.Throwable
       ↓ java.lang.Exception
           ↓ org.quartz.SchedulerException
               ↓ org.quartz.SchedulerConfigException
```

All Implemented Interfaces:
   Serializable

---

public class SchedulerConfigException

extends SchedulerException

An exception that is thrown to indicate that there is a misconfiguration of the SchedulerFactory- or one of the components it configures.

**Author:**
James House

**See Also:**
   Serialized Form

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SchedulerConfigException(String msg)</code></td>
<td>Create a JobPersistenceException with the given message.</td>
</tr>
<tr>
<td><code>SchedulerConfigException(String msg, Throwable cause)</code></td>
<td>Create a JobPersistenceException with the given message and cause.</td>
</tr>
</tbody>
</table>

**Method Summary**

Methods inherited from class org.quartz.SchedulerException:
   getUnderlyingException, toString
Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

SchedulerConfigException

public SchedulerConfigException(String msg)

Create a JobPersistenceException with the given message.

SchedulerConfigException

public SchedulerConfigException(String msg, Throwable cause)

Create a JobPersistenceException with the given message and cause.

Copyright 2001-2011, Terracotta, Inc.
### org.quartz Class SchedulerContext

**java.lang.Object**
- org.quartz.utils.DirtyFlagMap
  - org.quartz.utils.StringKeyDirtyFlagMap
- org.quartz.SchedulerContext

**All Implemented Interfaces:**
- Serializable, Cloneable, Map

```java
public class SchedulerContext extends StringKeyDirtyFlagMap implements Serializable
```

Holds context/environment data that can be made available to Jobs as they are executed. This feature is much like the ServletContext feature when working with J2EE servlets.

Future versions of Quartz may make distinctions on how it propagates data in `SchedulerContext` between instances of proxies to a single scheduler instance - i.e. if Quartz is being used via RMI.

**Author:**
James House

**See Also:**
- `Scheduler.getContext()`, Serialized Form

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from interface java.util.Map</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Map.Entry&lt;K, V&gt;</code></td>
</tr>
</tbody>
</table>

### Constructor Summary

- `SchedulerContext()`
Create an empty SchedulerContext.

## Method Summary

### Methods inherited from class org.quartz.util.StringKeyDirtyFlagMap
- `containsTransientData`, `equals`, `getAllowsTransientData`, `getBoolean`, `getChar`, `getDouble`, `getFloat`, `getInt`, `getKeys`, `getLong`, `getString`, `hashCode`, `put`, `put`, `put`, `put`, `put`, `put`, `putAll`, `removeTransientData`, `setAllowsTransientData`

### Methods inherited from class org.quartz.util.DirtyFlagMap
- `clear`, `clearDirtyFlag`, `clone`, `containsKey`, `containsValue`, `entrySet`, `get`, `getWrappedMap`, `isDirty`, `isEmpty`, `keySet`, `remove`, `size`, `values`

### Methods inherited from class java.lang.Object
- `finalize`, `getClass`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

## Constructor Detail

### SchedulerContext

```java
public SchedulerContext()
```

Create an empty SchedulerContext.

### SchedulerContext

```java
public SchedulerContext(Map map)
```

Create a SchedulerContext with the given data.
Copyright 2001-2011, Terracotta, Inc.
org.quartz Class SchedulerException

java.lang.Object
  └ java.lang.Throwable
    └ java.lang.Exception
      └ org.quartz.SchedulerException

All Implemented Interfaces:
  Serializable

Direct Known Subclasses:
  JmsJobException, JobExecutionException, JobPersistenceException,
  SchedulerConfigException, UnableToInterruptJobException

public class SchedulerException
  extends Exception

Base class for exceptions thrown by the Quartz Scheduler.

SchedulerExceptions may contain a reference to another Exception, which was
the underlying cause of the SchedulerException.

Author:
  James House

See Also:
  Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerException()</td>
</tr>
<tr>
<td>SchedulerException(String msg)</td>
</tr>
<tr>
<td>SchedulerException(String msg, Throwable cause)</td>
</tr>
</tbody>
</table>


### SchedulerException (Throwable cause)

#### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throwable-method</td>
<td>getUnderlyingException()</td>
</tr>
<tr>
<td>String-method</td>
<td>toString()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Throwable:
- fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.Object:
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

### Constructor Detail

**SchedulerException**

public SchedulerException()

**SchedulerException**

public SchedulerException(String msg)

**SchedulerException**

public SchedulerException(Throwable cause)
SchedulerException

public SchedulerException(String msg, Throwable cause)

Method Detail

getUnderlyingException

public Throwable getUnderlyingException()

Return the exception that is the underlying cause of this exception.
This may be used to find more detail about the cause of the error.

Returns:
the underlying exception, or null if there is not one.

toString

public String toString()

Overrides:
toString in class Throwable

Overview  Package  Use  Tree  Deprecated  Index  Help
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
org.quartz Interface SchedulerFactory

All Known Implementing Classes:  
DirectSchedulerFactory, StdSchedulerFactory

```
public interface SchedulerFactory

Provides a mechanism for obtaining client-readable handles to Scheduler instances.

Author:  
James House

See Also:  
Scheduler, StdSchedulerFactory
```

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getAllSchedulers()</code></td>
<td>Returns handles to all known Schedulers (made by any SchedulerFactory within this jvm.).</td>
</tr>
<tr>
<td><code>getScheduler()</code></td>
<td>Returns a client-readable handle to a Scheduler.</td>
</tr>
<tr>
<td><code>getScheduler(String schedName)</code></td>
<td>Returns a handle to the Scheduler with the given name, if it exists.</td>
</tr>
</tbody>
</table>

### Method Detail

**getScheduler**

```
Scheduler getScheduler()  
throws SchedulerException

Returns a client-readable handle to a Scheduler.
```
Throws:

SchedulerException - if there is a problem with the underlying Scheduler.

getScheduler

Scheduler getScheduler(String schedName) throws SchedulerException

Returns a handle to the Scheduler with the given name, if it exists.

Throws:

SchedulerException

getAllSchedulers

Collection<Scheduler> getAllSchedulers() throws SchedulerException

Returns handles to all known Schedulers (made by any SchedulerFactory within this jvm.).

Throws:

SchedulerException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAME</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
public interface SchedulerListener

The interface to be implemented by classes that want to be informed of major Scheduler events.

Author:
  James House

See Also:
  Scheduler, JobListener, TriggerListener

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobAdded</td>
<td>void jobAdded(JobDetail jobDetail)</td>
<td>Called by the Scheduler when a JobDetail has been added.</td>
</tr>
<tr>
<td>jobDeleted</td>
<td>void jobDeleted(JobKey jobKey)</td>
<td>Called by the Scheduler when a JobDetail has been deleted.</td>
</tr>
<tr>
<td>jobPaused</td>
<td>void jobPaused(JobKey jobKey)</td>
<td>Called by the Scheduler when a JobDetail has been paused.</td>
</tr>
<tr>
<td>jobResumed</td>
<td>void jobResumed(JobKey jobKey)</td>
<td>Called by the Scheduler when a JobDetail has been un-paused.</td>
</tr>
<tr>
<td>jobScheduled</td>
<td>void jobScheduled(Trigger trigger)</td>
<td>Called by the Scheduler when a JobDetail is scheduled.</td>
</tr>
<tr>
<td>jobsPaused</td>
<td>void jobsPaused(String jobGroup)</td>
<td>Called by the Scheduler when a group of JobDetails has been paused.</td>
</tr>
<tr>
<td>jobsResumed</td>
<td>void jobsResumed(String jobGroup)</td>
<td>Called by the Scheduler when a group of JobDetails has been un-</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>void jobUnscheduled(TriggerKey triggerKey)</td>
<td>Called by the Scheduler when a JobDetail is unscheduled.</td>
<td></td>
</tr>
<tr>
<td>void schedulerError(String msg, SchedulerException cause)</td>
<td>Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.</td>
<td></td>
</tr>
<tr>
<td>void schedulerInStandbyMode()</td>
<td>Called by the Scheduler to inform the listener that it has moved to standby mode.</td>
<td></td>
</tr>
<tr>
<td>void schedulerShutdown()</td>
<td>Called by the Scheduler to inform the listener that it has shutdown.</td>
<td></td>
</tr>
<tr>
<td>void schedulerShuttingdown()</td>
<td>Called by the Scheduler to inform the listener that it has begun the shutdown sequence.</td>
<td></td>
</tr>
<tr>
<td>void schedulerStarted()</td>
<td>Called by the Scheduler to inform the listener that it has started.</td>
<td></td>
</tr>
<tr>
<td>void schedulingDataCleared()</td>
<td>Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.</td>
<td></td>
</tr>
<tr>
<td>void triggerFinalized(Trigger trigger)</td>
<td>Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.</td>
<td></td>
</tr>
<tr>
<td>void triggerPaused(TriggerKey triggerKey)</td>
<td>Called by the Scheduler when a Trigger has been paused.</td>
<td></td>
</tr>
<tr>
<td>void triggerResumed(TriggerKey triggerKey)</td>
<td>Called by the Scheduler when a Trigger has been un-paused.</td>
<td></td>
</tr>
<tr>
<td>void triggersPaused(String triggerGroup)</td>
<td>Called by the Scheduler when a group of Triggers has been paused.</td>
<td></td>
</tr>
<tr>
<td>void triggersResumed(String triggerGroup)</td>
<td>Called by the Scheduler when a group of Triggers has been un-paused.</td>
<td></td>
</tr>
</tbody>
</table>
Method Detail

jobScheduled

```java
void jobScheduled(Trigger trigger)
```

Called by the Scheduler when a JobDetail is scheduled.

---

jobUnscheduled

```java
void jobUnscheduled(TriggerKey triggerKey)
```

Called by the Scheduler when a JobDetail is unscheduled.

See Also:  
  schedulingDataCleared()

---

triggerFinalized

```java
void triggerFinalized(Trigger trigger)
```

Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.

---

triggerPaused

```java
void triggerPaused(TriggerKey triggerKey)
```

Called by the Scheduler when a Trigger has been paused.

---

triggersPaused

```java
void triggersPaused(String triggerGroup)
```

Called by the Scheduler when a group of Triggers has been paused.
If all groups were paused then triggerGroup will be null

**Parameters:**

- triggerGroup - the paused group, or null if all were paused

---

**triggerResumed**

```java
void triggerResumed(TriggerKey triggerKey)
```

Called by the Scheduler when a Trigger has been un-paused.

---

**triggersResumed**

```java
void triggersResumed(String triggerGroup)
```

Called by the Scheduler when a group of Triggers has been un-paused.

---

**jobAdded**

```java
void jobAdded(JobDetail jobDetail)
```

Called by the Scheduler when a JobDetail has been added.

---

**jobDeleted**

```java
void jobDeleted(JobKey jobKey)
```

Called by the Scheduler when a JobDetail has been deleted.

---

**jobPaused**

```java
void jobPaused(JobKey jobKey)
```

Called by the Scheduler when a JobDetail has been paused.
jobsPaused

void jobsPaused(String jobGroup)

Called by the Scheduler when a group of JobDetails has been paused.

Parameters:
jobGroup - the paused group, or null if all were paused

jobResumed

void jobResumed(JobKey jobKey)

Called by the Scheduler when a JobDetail has been un-paused.

jobsResumed

void jobsResumed(String jobGroup)

Called by the Scheduler when a group of JobDetails has been un-paused.

schedulerError

void schedulerError(String msg, SchedulerException cause)

Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

The getErrorCode() method of the given SchedulerException can be used to determine more specific information about the type of error that was encountered.
schedulerInStandbyMode

void schedulerInStandbyMode()

Called by the Scheduler to inform the listener that it has move to standby mode.

schedulerStarted

void schedulerStarted()

Called by the Scheduler to inform the listener that it has started.

schedulerShutdown

void schedulerShutdown()

Called by the Scheduler to inform the listener that it has shutdown.

schedulerShuttingdown

void schedulerShuttingdown()

Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

schedulingDataCleared

void schedulingDataCleared()

Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD |
FRAMES | NO FRAMES
DETAIL: FIELD | CONSTR | METHOD |
org.quartz  Class SchedulerMetaData

java.lang.Object
   ├ org.quartz.SchedulerMetaData

All Implemented Interfaces:
   Serializable

public class SchedulerMetaData
  extends Object
  implements Serializable

Describes the settings and capabilities of a given Scheduler instance.

Author:  
  James House
See Also:  
  Serialized Form

Constructor Summary

SchedulerMetaData(String schedName, String schedInst, 
  Class schedClass, boolean isRemote, boolean started, 
  boolean isInStandbyMode, boolean shutdown, Date startTime, 
  int numJobsExec, Class jsClass, boolean jsPersistent, 
  boolean jsClustered, Class tpClass, int tpSize, String version)

Method Summary

Class getJobStoreClass()
   Returns the class-name of the JobStore instance that is being 
   used by the Scheduler.

int getNumberOfJobsExecuted()
   Returns the number of jobs executed since the Scheduler 
   started..
### `Date getRunningSince()`
- Returns the date at which the Scheduler started running.

### `Class getSchedulerClass()`
- Returns the class-name of the Scheduler instance.

### `String getSchedulerInstanceId()`
- Returns the instance Id of the Scheduler.

### `String getSchedulerName()`
- Returns the name of the Scheduler.

### `String getSummary()`
- Returns a formatted (human readable) String describing all the Scheduler's meta-data values.

### `Class getThreadPoolClass()`
- Returns the class-name of the ThreadPool instance that is being used by the Scheduler.

### `int getThreadPoolSize()`
- Returns the number of threads currently in the Scheduler's ThreadPool.

### `String getVersion()`
- Returns the version of Quartz that is running.

### `boolean isInStandbyMode()`
- Reports whether the Scheduler is in standby mode.

### `boolean isJobStoreClustered()`
- Returns whether or not the Scheduler's JobStore is clustered.

### `boolean isJobStoreSupportsPersistence()`
- Returns whether or not the Scheduler's JobStore instance supports persistence.

### `boolean isSchedulerRemote()`
- Returns whether the Scheduler is being used remotely (via RMI).

### `boolean isShutdown()`
- Reports whether the Scheduler has been shutdown.

### `boolean isStarted()`
- Returns whether the scheduler has been started.

### `boolean jobStoreSupportsPersistence()`
- Deprecated.
String **toString()**

Return a simple string representation of this object.

Methods inherited from class java.lang.**Object**
**clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait**

**Constructor Detail**

**SchedulerMetaData**

public **SchedulerMetaData**(**String** schedName,
**String** schedInst,
**Class** schedClass,
boolean isRemote,
boolean started,
boolean isInStandbyMode,
boolean shutdown,
**Date** startTime,
int numJobsExec,
**Class** jsClass,
boolean jsPersistent,
boolean jsClustered,
**Class** tpClass,
int tpSize,
**String** version)

**Method Detail**

**getSchedulerName**

public **String** **getSchedulerName()**

Returns the name of the Scheduler.

**getSchedulerInstanceId**
public String getSchedulerInstanceId()

    Returns the instance Id of the Scheduler.

getSchedulerClass

public Class getSchedulerClass()

    Returns the class-name of the Scheduler instance.

getRunningSince

public Date getRunningSince()

    Returns the date at which the Scheduler started running.

    Returns:
    null if the scheduler has not been started.

getNumberOfJobsExecuted

public int getNumberOfJobsExecuted()

    Returns the number of jobs executed since the Scheduler started.

isSchedulerRemote

public boolean isSchedulerRemote()

    Returns whether the Scheduler is being used remotely (via RMI).

isStarted

public boolean isStarted()
Returns whether the scheduler has been started.

Note: `isStarted()` may return true even if `isInStandbyMode()` returns true.

---

**isInStandbyMode**

```java
public boolean isInStandbyMode()
```

Reports whether the Scheduler is in standby mode.

---

**isShutdown**

```java
public boolean isShutdown()
```

Reports whether the Scheduler has been shutdown.

---

**getJobStoreClass**

```java
public Class getJobStoreClass()
```

Returns the class-name of the JobStore instance that is being used by the Scheduler.

---

**jobStoreSupportsPersistence**

```java
public boolean jobStoreSupportsPersistence()
```

Deprecated.

Returns whether or not the Scheduler's JobStore instance supports persistence.

See Also:

`isJobStoreSupportsPersistence()`
isJobStoreSupportsPersistence

public boolean isJobStoreSupportsPersistence()

Returns whether or not the Scheduler's JobStore instance supports persistence.

isJobStoreClustered

public boolean isJobStoreClustered()

Returns whether or not the Scheduler's JobStore is clustered.

getThreadPoolClass

public Class getThreadPoolClass()

Returns the class-name of the ThreadPool instance that is being used by the Scheduler.

getThreadPoolSize

public int getThreadPoolSize()

Returns the number of threads currently in the Scheduler's ThreadPool.

getVersion

public String getVersion()

Returns the version of Quartz that is running.
public String toString()

Return a simple string representation of this object.

Overrides:
　toString in class Object

getSummary

public String getSummary()
throws SchedulerException

Returns a formatted (human readable) String describing all the Scheduler's meta-data values.

The format of the String looks something like this:

Quartz Scheduler 'SchedulerName' with instanceId 'SchedulerInstanceId'
Scheduler class: 'org.quartz.impl.StdSchedulerImpl'
- with '8' threads
Using job-store 'org.quartz.impl.JDBCJobStore'
- which supports persistence.

Throws:
　SchedulerException
public class SimpleScheduleBuilder
extends ScheduleBuilder\<SimpleTrigger\

SimpleScheduleBuilder is a ScheduleBuilder that defines strict/literal
interval-based schedules for Triggers.

Quartz provides a builder-style API for constructing scheduling-related entities
via a Domain-Specific Language (DSL). The DSL can best be utilized through
the usage of static imports of the methods on the classes TriggerBuilder,
JobBuilder, DateBuilder, JobKey, TriggerKey and the various
ScheduleBuilder implementations.

Client code can then use the DSL to write code such as this:

```
JobDetail job = newJob(MyJob.class)
    .withIdentity("myJob")
    .build();

Trigger trigger = newTrigger()
    .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
    .withSchedule(simpleSchedule()
        .withIntervalInHours(1)
        .repeatForever())
    .startAt(futureDate(10, MINUTES))
    .build();

scheduler.scheduleJob(job, trigger);
```
**Method Summary**

<table>
<thead>
<tr>
<th>Class/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.spi.MutableTrigger.build()</td>
<td>Build the actual Trigger -- NOT intended to be invoked by end users, but will rather be invoked by a TriggerBuilder which this ScheduleBuilder is given to.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatForever()</td>
<td>Specify that the trigger will repeat indefinitely.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatHourlyForever()</td>
<td>Create a SimpleScheduleBuilder set to repeat with an hourly interval.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatHourlyForever(int hours)</td>
<td>Create a SimpleScheduleBuilder set to repeat with an interval of the given number of hours.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatForTotalCount(int count)</td>
<td>Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 hour interval.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatForTotalCount(int count, int hours)</td>
<td>Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of hours.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatMinutelyForever()</td>
<td>Create a SimpleScheduleBuilder set to repeat with a minute interval.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatMinutelyForever(int minutes)</td>
<td>Create a SimpleScheduleBuilder set to repeat with an interval of the given number of minutes.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatForTotalCount(int count)</td>
<td>Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 minute interval.</td>
</tr>
</tbody>
</table>
times - 1 with a 1 minute interval.

static SimpleScheduleBuilder repeatMinutelyForTotalCount(int count, int minutes)
  Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of minutes.

static SimpleScheduleBuilder repeatSecondlyForever()
  Create a SimpleScheduleBuilder set to repeat forever with a 1 second interval.

static SimpleScheduleBuilder repeatSecondlyForever(int seconds)
  Create a SimpleScheduleBuilder set to repeat forever with an interval of the given number of seconds.

static SimpleScheduleBuilder repeatSecondlyForTotalCount(int count)
  Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 second interval.

static SimpleScheduleBuilder repeatSecondlyForTotalCount(int count, int seconds)
  Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of seconds.

static SimpleScheduleBuilder simpleSchedule()
  Create a SimpleScheduleBuilder.

  SimpleScheduleBuilder withIntervalInHours(int intervalInHours)
    Specify a repeat interval in minutes - which will then be multiplied by 60 * 60 * 1000 to produce milliseconds.

  SimpleScheduleBuilder withIntervalInMilliseconds(long intervalInMilliseconds)
    Specify a repeat interval in milliseconds.

  SimpleScheduleBuilder withIntervalInMinutes(int intervalInMinutes)
    Specify a repeat interval in minutes - which will then be multiplied by 60 * 1000 to produce milliseconds.

  SimpleScheduleBuilder withIntervalInSeconds(int intervalInSeconds)
    Specify a repeat interval in seconds - which will then be multiplied by 1000 to produce milliseconds.

  SimpleScheduleBuilder withMisfireHandlingInstructionFireNow()
    If the Trigger misfires, use the SimpleTrigger.MISFIRE_INSTRUCTION_FIRE_NOW instruction.

  SimpleScheduleBuilder withMisfireHandlingInstructionIgnoreMisfires()
    If the Trigger misfires, use the Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY instruction.

  SimpleScheduleBuilder withMisfireHandlingInstructionNextWithExistingCount()
    If the Trigger misfires, use the SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT instruction.

  SimpleScheduleBuilder withMisfireHandlingInstructionNextWithRemainingCount()
    If the Trigger misfires, use the SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT instruction.
SimpleScheduleBuilder

withMisfireHandlingInstructionNowWithExistingCount

If the Trigger misfires, use the SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE
instruction.

SimpleScheduleBuilder

withMisfireHandlingInstructionNowWithRemainingCount

If the Trigger misfires, use the SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE
instruction.

SimpleScheduleBuilder

withRepeatCount(int repeatCount)

Specify a the number of time the t
number of firings will be this number + 1.

---

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Method Detail

simpleSchedule

public static SimpleScheduleBuilder simpleSchedule()

Create a SimpleScheduleBuilder.
Returns:
the new SimpleScheduleBuilder

repeatMinutelyForever

public static SimpleScheduleBuilder repeatMinutelyForever()

Create a SimpleScheduleBuilder set to repeat forever with a 1 mi

Returns:
the new SimpleScheduleBuilder

repeatMinutelyForever

public static SimpleScheduleBuilder repeatMinutelyForever(int minute

Create a SimpleScheduleBuilder set to repeat forever with an int of the given number of minutes.
**Returns:**
the new SimpleScheduleBuilder

---

**repeatSecondlyForever**

public static SimpleScheduleBuilder repeatSecondlyForever()

Create a SimpleScheduleBuilder set to repeat forever with a 1 second interval.

**Returns:**
the new SimpleScheduleBuilder

---

**repeatSecondlyForever**

public static SimpleScheduleBuilder repeatSecondlyForever(int second
Create a SimpleScheduleBuilder set to repeat forever with an int of the given number of seconds.

**Returns:**
the new SimpleScheduleBuilder

---

**repeatHourlyForever**

public static SimpleScheduleBuilder repeatHourlyForever()

Create a SimpleScheduleBuilder set to repeat forever with a 1 hr

**Returns:**
the new SimpleScheduleBuilder

---

**repeatHourlyForever**
public static SimpleScheduleBuilder repeatHourlyForever(int hours)

Create a SimpleScheduleBuilder set to repeat forever with an int of the given number of hours.

Returns:
the new SimpleScheduleBuilder

repeatMinutelyForTotalCount

public static SimpleScheduleBuilder repeatMinutelyForTotalCount(int count)

Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 minute interval.

Note: Total count = 1 (at start time) + repeat count

Returns:
the new SimpleScheduleBuilder
**repeatMinutelyForTotalCount**

public static SimpleScheduleBuilder repeatMinutelyForTotalCount(int count, int minutes)

Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of minutes.

Note: Total count = 1 (at start time) + repeat count

**Returns:**
the new SimpleScheduleBuilder

**repeatSecondlyForTotalCount**

public static SimpleScheduleBuilder repeatSecondlyForTotalCount(int count)
Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 second interval.

Note: Total count = 1 (at start time) + repeat count

Returns:
the new SimpleScheduleBuilder

repeatSecondlyForTotalCount

public static SimpleScheduleBuilder repeatSecondlyForTotalCount(int count, int seconds)

Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of seconds.

Note: Total count = 1 (at start time) + repeat count

Returns:
repeatHourlyForTotalCount

public static SimpleScheduleBuilder repeatHourlyForTotalCount(int count)

Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with a 1 hour interval.

Note: Total count = 1 (at start time) + repeat count

Returns:
the new SimpleScheduleBuilder
Create a SimpleScheduleBuilder set to repeat the given number of times - 1 with an interval of the given number of hours.

Note: Total count = 1 (at start time) + repeat count

Returns:
the new SimpleScheduleBuilder

build

public org.quartz.spi.MutableTrigger build()
withIntervalInMilliseconds

```
public SimpleScheduleBuilder withIntervalInMilliseconds(long intervalInMillis)
```

Specify a repeat interval in milliseconds.

**Parameters:**
- `intervalInMillis` - the number of seconds at which the trigger should repeat.

**Returns:**
- the updated SimpleScheduleBuilder

See Also:
- `SimpleTrigger.getRepeatInterval()`,
- `withRepeatCount(int)`

withIntervalInSeconds

```
public SimpleScheduleBuilder withIntervalInSeconds(int intervalInSeconds)
```

See Also:
- `TriggerBuilder.withSchedule(ScheduleBuilder)`
Specify a repeat interval in seconds - which will then be multiplied by 1000 to produce milliseconds.

**Parameters:**
intervalInSeconds - the number of seconds at which the trigger should repeat.

**Returns:**
the updated SimpleScheduleBuilder

**See Also:**
SimpleTrigger.getRepeatInterval(),
withRepeatCount(int)

---

**withIntervalInMinutes**

public SimpleScheduleBuilder withIntervalInMinutes(int intervalInMinutes)

Specify a repeat interval in minutes - which will then be multiplied by 60 * 1000 to produce milliseconds.

**Parameters:**
intervalInMinutes - the number of seconds at which the trigger should repeat.

**Returns:**
the updated SimpleScheduleBuilder

**See Also:**
SimpleTrigger.getRepeatInterval(),
withRepeatCount(int)
withIntervalInHours

public SimpleScheduleBuilder withIntervalInHours(int intervalInHours)

Specify a repeat interval in minutes - which will then be multiplied by \(60 \times 60 \times 1000\) to produce milliseconds.

Parameters:
intervalInHours - the number of seconds at which the trigger should repeat.

Returns:
the updated SimpleScheduleBuilder

See Also:
SimpleTrigger.getRepeatInterval(), withRepeatCount(int)

withRepeatCount

public SimpleScheduleBuilder withRepeatCount(int repeatCount)
Specify a the number of time the trigger will repeat - total num firings will be this number + 1.

**Parameters:**
- repeatCount - the number of seconds at which the trigger sho

**Returns:**
the updated SimpleScheduleBuilder

**See Also:**
- SimpleTrigger.getRepeatCount()
- repeatForever()

---

**repeatForever**

```java
public SimpleScheduleBuilder repeatForever()
```

Specify that the trigger will repeat indefinitely.

**Returns:**
the updated SimpleScheduleBuilder

**See Also:**
- SimpleTrigger.getRepeatCount()
- SimpleTrigger.REPEAT_INDEFINITELY
- withIntervalInMilliseconds(long)
- withIntervalInSeconds(int)
- withIntervalInMinutes(int)
- withIntervalInHours(int)
withMisfireHandlingInstructionIgnoreMisfires

public SimpleScheduleBuilder withMisfireHandlingInstructionIgnoreMisfires()

If the Trigger misfires, use the
Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY instruction.

Returns:
the updated CronScheduleBuilder

See Also:
Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY

withMisfireHandlingInstructionFireNow

public SimpleScheduleBuilder withMisfireHandlingInstructionFireNow()

If the Trigger misfires, use the
SimpleTrigger.MISFIRE_INSTRUCTION_FIRE_NOW instruction.
Returns:
the updated SimpleScheduleBuilder

See Also:
SimpleTrigger.MISFIRE_INSTRUCTION_FIRE_NOW

withMisfireHandlingInstructionNextWithExistingCount

public SimpleScheduleBuilder withMisfireHandlingInstructionNextWithExistingCount

If the Trigger misfires, use the
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING

Returns:
the updated SimpleScheduleBuilder

See Also:
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXIS

withMisfireHandlingInstructionNextWithRemainingCount
public SimpleScheduleBuilder withMisfireHandlingInstructionNextWithRemainingCount

If the Trigger misfires, use the
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT

Returns:
the updated SimpleScheduleBuilder
See Also:
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT

withMisfireHandlingInstructionNowWithExistingCount

public SimpleScheduleBuilder withMisfireHandlingInstructionNowWithExistingCount

If the Trigger misfires, use the
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING

Returns:
the updated SimpleScheduleBuilder
See Also:
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING
withMisfireHandlingInstructionNowWithRemainingCount

public SimpleScheduleBuilder withMisfireHandlingInstructionNowWithRemainingCount

If the Trigger misfires, use the
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING

Returns:
the updated SimpleScheduleBuilder

See Also:
SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING
Copyright 2001-2011, Terracotta, Inc.
org.quartz Interface SimpleTrigger

All Superinterfaces:
   Cloneable, Comparable<Trigger>, Serializable, Trigger

All Known Implementing Classes:
   SimpleTriggerImpl

public interface SimpleTrigger
   extends Trigger

A Trigger that is used to fire a Job at a given moment in time, and optionally repeated at a specified interval.

Author:
   James House, contributions by Lieven Govaerts of Ebitec Nv, Belgium.

See Also:
   TriggerBuilder, SimpleScheduleBuilder

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from interface org.quartz.Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger.CompletedExecutionInstruction, Trigger.TriggerState,</td>
</tr>
<tr>
<td>Trigger.TriggerTimeComparator</td>
</tr>
</tbody>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>static int MISFIRE_INSTRUCTION_FIRE_NOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructs the Scheduler that upon a mis-fire situation, the</td>
</tr>
<tr>
<td>SimpleTrigger wants to be fired now by Scheduler.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int MISFIRE_INSTRUCTION_RESC SCHEDULE_NEXT WITH EXISTING COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructs the Scheduler that upon a mis-fire situation, the</td>
</tr>
<tr>
<td>SimpleTrigger wants to be re-scheduled to the next scheduled time at</td>
</tr>
<tr>
<td>'now' - taking into account any associated Calendar, and with the re-</td>
</tr>
</tbody>
</table>
count left unchanged.

static int MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT
    Instructs the Scheduler that upon a mis-fire situation, the
    SimpleTrigger wants to be re-scheduled to the next scheduled time at
    'now' - taking into account any associated calendar, and with the repeat
    count set to what it would be, if it had not missed any firings.

static int MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT
    Instructs the Scheduler that upon a mis-fire situation, the
    SimpleTrigger wants to be re-scheduled to 'now' (even if the associat
    Calendar excludes 'now') with the repeat count left as-is.

static int MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT
    Instructs the Scheduler that upon a mis-fire situation, the
    SimpleTrigger wants to be re-scheduled to 'now' (even if the associat
    Calendar excludes 'now') with the repeat count set to what it would be
    it had not missed any firings.

static int REPEAT_INDEFINITELY
    Used to indicate the 'repeat count' of the trigger is indefinite.

static long serialVersionUID

Fields inherited from interface org.quartz.Trigger
DEFAULT_PRIORITY, MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY,
MISFIRE_INSTRUCTION_SMART_POLICY

Method Summary

int getRepeatCount()
    Get the the number of times the
    SimpleTrigger should repeat, after which it will be
    automatically deleted.

long getRepeatInterval()
    Get the the time interval (in milliseconds) at
    which the SimpleTrigger should repeat.

int getTimesTriggered()
    Get the number of times the SimpleTrigger
    has already fired.
getTriggerBuilder()

Get a TriggerBuilder that is configured to produce a Trigger identical to this one.

Methods inherited from interface org.quartz.Trigger

compareTo, equals, getCalendarName, getDescription, getEndTime, getFinalFireTime, getFireTimeAfter, getJobDataMap, getJobKey, getKey, getMisfireInstruction, getNextFireTime, getPreviousFireTime, getPriority, getScheduleBuilder, getStartTime, mayFireAgain

Field Detail

serialVersionUID

static final long serialVersionUID

See Also:

Constant Field Values

MISFIRE_INSTRUCTION_FIRE_NOW

static final int MISFIRE_INSTRUCTION_FIRE_NOW

Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be fired now by Scheduler.

NOTE: This instruction should typically only be used for 'one-shot' (non-repeating) Triggers. If it is used on a trigger with a repeat count > 0 then it is equivalent to the instruction MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT.

See Also:

Constant Field Values
**MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT**

```java
static final int MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT
```

Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to 'now' (even if the associated Calendar excludes 'now') with the repeat count left as-is. This does obey the Trigger end-time however, so if 'now' is after the end-time the Trigger will not fire again.

**NOTE:** Use of this instruction causes the trigger to 'forget' the start-time and repeat-count that it was originally setup with (this is only an issue if you for some reason wanted to be able to tell what the original values were at some later time).

**See Also:**

[Constant Field Values](#)

---

**MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT**

```java
static final int MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT
```

Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to 'now' (even if the associated Calendar excludes 'now') with the repeat count set to what it would be, if it had not missed any firings. This does obey the Trigger end-time however, so if 'now' is after the end-time the Trigger will not fire again.

**NOTE:** Use of this instruction causes the trigger to 'forget' the start-time and repeat-count that it was originally setup with. Instead, the repeat count on the trigger will be changed to whatever the remaining repeat count is (this is only an issue if you for some reason wanted to be able to tell what the original values were at some later time).

**NOTE:** This instruction could cause the Trigger to go to the 'COMPLETE' state after firing 'now', if all the repeat-fire-times where missed.

**See Also:**

[Constant Field Values](#)
MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT

static final int MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT

Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to the next scheduled time after 'now' - taking into account any associated Calendar, and with the repeat count set to what it would be, if it had not missed any firings.

NOTE/WARNING: This instruction could cause the Trigger to go directly to the 'COMPLETE' state if all fire-times where missed.

See Also:
Constant Field Values

MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITHEXISTING_COUNT

static final int MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITHEXISTING_COUNT

Instructs the Scheduler that upon a mis-fire situation, the SimpleTrigger wants to be re-scheduled to the next scheduled time after 'now' - taking into account any associated Calendar, and with the repeat count left unchanged.

NOTE/WARNING: This instruction could cause the Trigger to go directly to the 'COMPLETE' state if the end-time of the trigger has arrived.

See Also:
Constant Field Values

REPEAT_INDEFINITELY

static final int REPEAT_INDEFINITELY

Used to indicate the 'repeat count' of the trigger is indefinite. Or in other words, the trigger should repeat continually until the trigger's ending timestamp.

See Also:
Constant Field Values

Method Detail

getRepeatCount

int getRepeatCount()

Get the the number of times the SimpleTrigger should repeat, after which it will be automatically deleted.

See Also:
REPEAT_INDEFINITELY

getRepeatInterval

long getRepeatInterval()

Get the the time interval (in milliseconds) at which the SimpleTrigger should repeat.

getTimesTriggered

int getTimesTriggered()

Get the number of times the SimpleTrigger has already fired.

getTriggerBuilder

TriggerBuilder<SimpleTrigger> getTriggerBuilder()

Description copied from interface: Trigger
Get a TriggerBuilder that is configured to produce a Trigger identical to this one.

Specified by:
getTriggerBuilder in interface Trigger

See Also:
 Trigger.getScheduleBuilder()
org.quartz Interface StatefulJob

All Superinterfaces:
   Job

**Deprecated.** use DisallowConcurrentExecution and/or PersistJobDataAfterExecution annotations instead.

```java
@PersistJobDataAfterExecution
@DisallowConcurrentExecution
public interface StatefulJob
```

extends Job

A marker interface for JobDetail s that wish to have their state maintained between executions.

StatefulJob instances follow slightly different rules from regular Job instances. The key difference is that their associated JobDataMap is re-persisted after every execution of the job, thus preserving state for the next execution. The other difference is that stateful jobs are not allowed to execute concurrently, which means new triggers that occur before the completion of the execute(xx) method will be delayed.

**Author:**
James House

**See Also:**
DisallowConcurrentExecution, PersistJobDataAfterExecution, Job, JobDetail, JobDataMap, Scheduler

---

# Method Summary

<table>
<thead>
<tr>
<th>Methods inherited from interface org.quartz.Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute</td>
</tr>
</tbody>
</table>

---
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>ENUM CONSTANTS</td>
<td>FIELD</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL</td>
<td>ENUM CONSTANTS</td>
<td>FIELD</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| | | | | | | |
| | | | | | | |
org.quartz Enum Trigger.CompletedExecutionInstruction

java.lang.Object
   \java.lang.Enum<Trigger.CompletedExecutionInstruction>
   \org.quartz.Trigger.CompletedExecutionInstruction

All Implemented Interfaces:
   Serializable, Comparable<Trigger.CompletedExecutionInstruction>

Enclosing interface:
   Trigger

public static enum Trigger.CompletedExecutionInstruction
extends Enum<Trigger.CompletedExecutionInstruction>

NOOP Instructs the Scheduler that the Trigger has no further instructions.

RE_EXECUTE_JOB Instructs the Scheduler that the Trigger wants the JobDetail to re-execute immediately. If not in a 'RECOVERING' or 'FAILED_OVER' situation, the execution context will be re-used (giving the Job the ability to 'see' anything placed in the context by its last execution).

SET_TRIGGER_COMPLETE Instructs the Scheduler that the Trigger should be put in the COMPLETE state.

DELETE_TRIGGER Instructs the Scheduler that the Trigger wants itself deleted.

SET_ALL_JOB_TRIGGERS_COMPLETE Instructs the Scheduler that all Triggers referencing the same JobDetail as this one should be put in the COMPLETE state.

SET_TRIGGER_ERROR Instructs the Scheduler that all Triggers referencing the same JobDetail as this one should be put in the ERROR state.

SET_ALL_JOB_TRIGGERS_ERROR Instructs the Scheduler that the Trigger should be put in the ERROR state.
## Enum Constant Summary

<table>
<thead>
<tr>
<th>Constant</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE_TRIGGER</td>
<td></td>
</tr>
<tr>
<td>NOOP</td>
<td></td>
</tr>
<tr>
<td>RE_EXECUTE_JOB</td>
<td></td>
</tr>
<tr>
<td>SET_ALL_JOB_TRIGGERS_COMPLETE</td>
<td></td>
</tr>
<tr>
<td>SET_ALL_JOB_TRIGGERS_ERROR</td>
<td></td>
</tr>
<tr>
<td>SET_TRIGGER_COMPLETE</td>
<td></td>
</tr>
<tr>
<td>SET_TRIGGER_ERROR</td>
<td></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Trigger.CompletedExecutionInstruction.valueOf(String name)</td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static Trigger.CompletedExecutionInstruction[] values()</td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

## Methods inherited from class java.lang.Enum

- clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

## Methods inherited from class java.lang.Object

- getClass, notify, notifyAll, wait, wait, wait
### Enum Constant Detail

<table>
<thead>
<tr>
<th>NOOP</th>
<th>public static final Trigger.CompletedExecutionInstruction NOOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE_EXECUTE_JOB</td>
<td>public static final Trigger.CompletedExecutionInstruction RE_EXECUTE</td>
</tr>
<tr>
<td>SET_TRIGGER_COMPLETE</td>
<td>public static final Trigger.CompletedExecutionInstruction SET_TRIGGER</td>
</tr>
<tr>
<td>DELETE_TRIGGER</td>
<td>public static final Trigger.CompletedExecutionInstruction DELETE_TRIGGER</td>
</tr>
<tr>
<td>SET_ALL_JOB_TRIGGERS_COMPLETE</td>
<td>public static final Trigger.CompletedExecutionInstruction SET_ALL_JOB</td>
</tr>
<tr>
<td>SET_TRIGGER_ERROR</td>
<td>public static final Trigger.CompletedExecutionInstruction SET_TRIGGER</td>
</tr>
<tr>
<td>SET_ALL_JOB_TRIGGERS_ERROR</td>
<td>public static final Trigger.CompletedExecutionInstruction SET_ALL_JOB</td>
</tr>
</tbody>
</table>

### Method Detail
values

public static Trigger.CompletedExecutionInstruction[] values()

    Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

    for (Trigger.CompletedExecutionInstruction c : Trigger.CompletedExecutionInstruction.values())
        System.out.println(c);

    Returns:
    an array containing the constants of this enum type, in the order they are declared

valueOf

public static Trigger.CompletedExecutionInstruction valueOf(String name)

    Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

    Parameters:
    name - the name of the enum constant to be returned.

    Returns:
    the enum constant with the specified name

    Throws:
    IllegalArgumentException - if this enum type has no constant with the specified name
    NullPointerException - if the argument is null
| Overview | Package | Use | Tree | Deprecated | Index | Help | PREV CLASS | NEXT CLASS | SUMMARY: NESTED | FIELD | CONSTR | METHOD | FRAMES | NO FRAMES | DETAIL: FIELD | CONSTR | METHOD |
org.quartz Interface Trigger

All Superinterfaces: Cloneable, Comparable<Trigger>, Serializable

All Known Subinterfaces: CalendarIntervalTrigger, CoreTrigger, CronTrigger, LocalityTrigger, SimpleTrigger

All Known Implementing Classes: AbstractTrigger, CalendarIntervalTriggerImpl, CronTriggerImpl, DelegatingLocalityTrigger, SimpleTriggerImpl

public interface Trigger

extends Serializable, Cloneable, Comparable<Trigger>

The base interface with properties common to all Triggers - use TriggerBuilder to instantiate an actual Trigger.

Triggers have a TriggerKey associated with them, which should uniquely identify them within a single Scheduler.

Triggers are the 'mechanism' by which Jobs are scheduled. Many Triggers can point to the same Job, but a single Trigger can only point to one Job.

Triggers can 'send' parameters/data to Jobs by placing contents into the JobDataMap on the Trigger.

Author: James House

See Also: TriggerBuilder, JobDataMap, JobExecutionContext, TriggerUtils, SimpleTrigger, CronTrigger, CalendarIntervalTrigger, NthIncludedDayTrigger
### Nested Class Summary

<table>
<thead>
<tr>
<th>static class</th>
<th>Trigger.CompletedExecutionInstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOOP</td>
</tr>
<tr>
<td></td>
<td>Instructs the Scheduler that the Trigger has no further instructions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static class</th>
<th>Trigger.TriggerState</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>static class</th>
<th>Trigger.TriggerTimeComparator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A Comparator that compares trigger's next fire times, or in other words, sorts them according to earliest next fire time.</td>
</tr>
</tbody>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>DEFAULT_PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The default value for priority.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Instructs the Scheduler that the Trigger will never be evaluated for a misfire situation, and that the scheduler will simply try to fire it as soon as it can, and then update the Trigger as if it had fired at the proper time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static int</th>
<th>MISFIRE_INSTRUCTION_SMART_POLICY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Instructs the Scheduler that upon a mis-fire situation, the updateAfterMisfire() method will be called on the Trigger to determine the mis-fire instruction, which logic will be trigger-implementation-dependent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static long</th>
<th>serialVersionUID</th>
</tr>
</thead>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>int compareTo(Trigger other)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean equals(Object other)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger equality is based upon the equality of the TriggerKey.</td>
</tr>
</tbody>
</table>
### String `getCalendarName()`
Get the name of the calendar associated with this Trigger.

### String `getDescription()`
Return the description given to the Trigger instance by its creator (if any).

### Date `getEndTime()`
Get the time at which the Trigger should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).

### Date `getFinalFireTime()`
Returns the last time at which the Trigger will fire, if the Trigger will repeat indefinitely, null will be returned.

### Date `getFireTimeAfter(Date afterTime)`
Returns the next time at which the Trigger will fire, after the given time.

### JobDataMap `getJobDataMap()`
Get the JobDataMap that is associated with the Trigger.

### JobKey `getJobKey()`

### TriggerKey `getKey()`

### int `getMisfireInstruction()`
Get the instruction the Scheduler should be given for handling misfire situations for this Trigger- the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be set as this property's value.

### Date `getNextFireTime()`
Returns the next time at which the Trigger is scheduled to fire.

### Date `getPreviousFireTime()`
Returns the previous time at which the Trigger fired.

### int `getPriority()`
The priority of a Trigger acts as a tiebreaker such that if two Triggers have the same scheduled fire time, then the
one with the higher priority will get first access to a worker thread.

### ScheduleBuilder<?
extends Trigger> getScheduleBuilder()

Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

### Date getStartTime()

Get the time at which the Trigger should occur.

### TriggerBuilder<?
extends Trigger> getTriggerBuilder()

Get a TriggerBuilder that is configured to produce a Trigger identical to this one.

### boolean mayFireAgain()

Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

---

### Field Detail

#### serialVersionUID

static final long serialVersionUID

#### See Also:

Constant Field Values

---

### MISFIRE_INSTRUCTION_SMART_POLICY

static final int MISFIRE_INSTRUCTION_SMART_POLICY

Instructs the Scheduler that upon a mis-fire situation, the updateAfterMisfire() method will be called on the Trigger to determine the mis-fire instruction, which logic will be trigger-implementation-dependent.

In order to see if this instruction fits your needs, you should look at the documentation for the getSmartMisfirePolicy() method on the particular Trigger implementation you are using.
See Also:
Constant Field Values

MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY

static final int MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY

Instructs the Scheduler that the Trigger will never be evaluated for a
misfire situation, and that the scheduler will simply try to fire it as soon as it
can, and then update the Trigger as if it had fired at the proper time.

NOTE: if a trigger uses this instruction, and it has missed several of its
scheduled firings, then

See Also:
Constant Field Values

DEFAULT_PRIORITY

static final int DEFAULT_PRIORITY

The default value for priority.

See Also:
Constant Field Values

Method Detail

getKey

TriggerKey getKey()

getJobKey

JobKey getJobKey()
**get\(\text{Description}\)**

\texttt{String get\textit{Description}()}  

Return the description given to the \texttt{Trigger} instance by its creator (if any).

**Returns:**
null if no description was set.

**get\(\text{Calendar\textit{Name}}\)**

\texttt{String get\textit{Calendar\textit{Name}()}  

Get the name of the \texttt{calendar} associated with this \texttt{Trigger}.

**Returns:**
null if there is no associated \texttt{Calendar}.

**get\(\text{Job\textit{Data\textit{Map}}}\)**

\texttt{Job\textit{Data\textit{Map} get\textit{Job\textit{Data\textit{Map}()}}}

Get the \texttt{Job\textit{Data\textit{Map}}} that is associated with the \texttt{Trigger}.

Changes made to this map during job execution are not re-persisted, and in fact typically result in an \texttt{IllegalStateException}.

**get\(\text{Priority}\)**

\texttt{int get\textit{Priority}()}  

The priority of a \texttt{Trigger} acts as a tiebreaker such that if two \texttt{Triggers} have the same scheduled fire time, then the one with the higher priority will get first access to a worker thread.

If not explicitly set, the default value is 5.
mayFireAgain

boolean mayFireAgain()

Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

If the returned value is false then the Scheduler may remove the Trigger from the JobStore.

getStartTime

Date getStartTime()

Get the time at which the Trigger should occur.

getEndTime

Date getEndTime()

Get the time at which the Trigger should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).

See Also:

getFinalFireTime()

g getNextFireTime

Date getNextFireTime()

Returns the next time at which the Trigger is scheduled to fire. If the trigger will not fire again, null will be returned. Note that the time returned can possibly be in the past, if the time that was computed for the trigger to
next fire has already arrived, but the scheduler has not yet been able to fire the trigger (which would likely be due to lack of resources e.g. threads).

The value returned is not guaranteed to be valid until after the Trigger has been added to the scheduler.

See Also:
TriggerUtils#computeFireTimesBetween(Trigger, Calendar, Date, Date)

getPreviousFireTime

Date getPreviousFireTime()

Returns the previous time at which the Trigger fired. If the trigger has not yet fired, null will be returned.

getFireTimeAfter

Date getFireTimeAfter(Date afterTime)

Returns the next time at which the Trigger will fire, after the given time. If the trigger will not fire after the given time, null will be returned.

getFinalFireTime

Date getFinalFireTime()

Returns the last time at which the Trigger will fire, if the Trigger will repeat indefinitely, null will be returned.

Note that the return time *may* be in the past.

getMisfireInstruction

int getMisfireInstruction()
Get the instruction the Scheduler should be given for handling misfire situations for this Trigger - the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be set as this property's value.

If not explicitly set, the default value is MISFIRE_INSTRUCTION_SMART_POLICY.

See Also:

MISFIRE_INSTRUCTION_SMART_POLICY,
#updateAfterMisfire(Calendar), SimpleTrigger, CronTrigger

getTriggerBuilder

TriggerBuilder<? extends Trigger> getTriggerBuilder()

Get a TriggerBuilder that is configured to produce a Trigger identical to this one.

See Also:

getScheduleBuilder()

getScheduleBuilder

ScheduleBuilder<? extends Trigger> getScheduleBuilder()

Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

See Also:

getTriggerBuilder()

equals

boolean equals(Object other)

Trigger equality is based upon the equality of the TriggerKey.
Overrides:

equals in class Object

Returns:

true if the key of this Trigger equals that of the given Trigger.

**compareTo**

```java
int compareTo(Trigger other)
```

Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e. alphabetical) order of their keys.

Specified by:

compareTo in interface Comparable<Trigger>

---

Copyright 2001-2011, Terracotta, Inc.
**org.quartz** *Enum Trigger.TriggerState*

```java
java.lang.Object
    └ java.lang.Enum<Trigger.TriggerState>
        └ org.quartz.Trigger.TriggerState
```

**All Implemented Interfaces:**
- Serializable
- Comparable<Trigger.TriggerState>

**Enclosing interface:**
- Trigger

```java
public static enum Trigger.TriggerState extends Enum<Trigger.TriggerState>
```

### Enum Constant Summary

<table>
<thead>
<tr>
<th>Enum Constant</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocked</td>
<td></td>
</tr>
<tr>
<td>complete</td>
<td></td>
</tr>
<tr>
<td>error</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td></td>
</tr>
<tr>
<td>normal</td>
<td></td>
</tr>
<tr>
<td>paused</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

```java
static Trigger.TriggerState.valueOf(String name)
```

Returns the enum constant of this type with

```java
valueOf(String name)
```
static Trigger.TriggerState[] values()  
Returns an array containing the constants of this enum type, in the order they are declared.

| Methods inherited from class java.lang.Enum
| clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf |

| Methods inherited from class java.lang.Object
| getClass, notify, notifyAll, wait, wait, wait |

## Enum Constant Detail

### NONE

```java
public static final Trigger.TriggerState NONE
```

### NORMAL

```java
public static final Trigger.TriggerState NORMAL
```

### PAUSED

```java
public static final Trigger.TriggerState PAUSED
```

### COMPLETE

```java
public static final Trigger.TriggerState COMPLETE
```

### ERROR
public static final Trigger.TriggerState ERROR

BLOCKED

public static final Trigger.TriggerState BLOCKED

### Method Detail

#### values

```java
public static Trigger.TriggerState[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```java
for (Trigger.TriggerState c : Trigger.TriggerState.values())
    System.out.println(c);
```

**Returns:**
- an array containing the constants of this enum type, in the order they are declared

#### valueOf

```java
public static Trigger.TriggerState valueOf(String name)
```

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

**Parameters:**
- `name` - the name of the enum constant to be returned.

**Returns:**
- the enum constant with the specified name

**Throws:**
- `IllegalArgumentException` - if this enum type has no constant with
the specified name

`NullPointerException` - if the argument is null
public static class Trigger.TriggerTimeComparator

extends Object
implements Comparator<Trigger>, Serializable

A Comparator that compares trigger's next fire times, or in other words, sorts them according to earliest next fire time. If the fire times are the same, then the triggers are sorted according to priority (highest value first), if the priorities are the same, then they are sorted by key.

See Also:
Serialized Form

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger.TriggerTimeComparator()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>int compare(Trigger trig1, Trigger trig2)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
**Constructors**

**Trigger.TriggerTimeComparator**

```java
public Trigger.TriggerTimeComparator()
```

**Methods**

**compare**

```java
public int compare(Trigger trig1, Trigger trig2)
```

Specified by:

`compare` in interface `Comparator<Trigger>`
org.quartz Class TriggerBuilder<T extends Trigger>

java.lang.Object
    └ org.quartz.TriggerBuilder<T>

public class TriggerBuilder<T extends Trigger> extends Object

TriggerBuilder is used to instantiate Triggers.

Quartz provides a builder-style API for constructing scheduling-related entities via a Domain-Specific Language (DSL). The DSL can best be utilized through the usage of static imports of the methods on the classes TriggerBuilder, JobBuilder, DateBuilder, JobKey, TriggerKey and the various ScheduleBuilder implementations.

Client code can then use the DSL to write code such as this:

```java
    JobDetail job = newJob(MyJob.class)
        .withIdentity("myJob")
        .build();

    Trigger trigger = newTrigger()
        .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
        .withSchedule(simpleSchedule()
            .withIntervalInHours(1)
            .repeatForever())
        .startAt(futureDate(10, MINUTES))
        .build();

    scheduler.scheduleJob(job, trigger);
```

See Also:
    JobBuilder,
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>T build()</code></td>
<td>Produce the Trigger.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt; endAt(Date endTime)</code></td>
<td>Set the time at which the Trigger will no longer fire - even if it's schedule has remaining repeats.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt; forJob(JobDetail jobDetail)</code></td>
<td>Set the identity of the Job which should be fired by the produced Trigger, by extracting the JobKey from the given job.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt; forJob(JobKey jobKey)</code></td>
<td>Set the identity of the Job which should be fired by the produced Trigger.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt; forJob(String jobName)</code></td>
<td>Set the identity of the Job which should be fired by the produced Trigger - a JobKey will be produced with the given name and default group.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt; forJob(String jobName, String jobGroup)</code></td>
<td>Set the identity of the Job which should be fired by the produced Trigger - a JobKey will be produced with the given name and group.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt; modifiedByCalendar(String calendarName)</code></td>
<td>Set the name of the Calendar that should be applied to this Trigger's schedule.</td>
</tr>
<tr>
<td><code>static TriggerBuilder&lt;Trigger&gt; newTrigger()</code></td>
<td>Create a new TriggerBuilder with</td>
</tr>
</tbody>
</table>
which to define a specification for a Trigger.

**TriggerBuilder**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>startAt(Date startTime)</strong></td>
<td>Set the time the Trigger should start at - the trigger may or may not fire at this time - depending upon the schedule configured for the Trigger.</td>
</tr>
<tr>
<td><strong>startNow()</strong></td>
<td>Set the time the Trigger should start at to the current moment - the trigger may or may not fire at this time - depending upon the schedule configured for the Trigger.</td>
</tr>
<tr>
<td><strong>usingJobData(JobDataMap newJobDataMap)</strong></td>
<td>Set the Trigger's JobDataMap, adding any values to it that were already set on this TriggerBuilder using any of the other 'usingJobData' methods.</td>
</tr>
<tr>
<td><strong>usingJobData(String key, Boolean value)</strong></td>
<td>Add the given key-value pair to the Trigger's JobDataMap.</td>
</tr>
<tr>
<td><strong>usingJobData(String key, Double value)</strong></td>
<td>Add the given key-value pair to the Trigger's JobDataMap.</td>
</tr>
<tr>
<td><strong>usingJobData(String key, Float value)</strong></td>
<td>Add the given key-value pair to the Trigger's JobDataMap.</td>
</tr>
<tr>
<td><strong>usingJobData(String key, Integer value)</strong></td>
<td>Add the given key-value pair to the Trigger's JobDataMap.</td>
</tr>
<tr>
<td><strong>usingJobData(String key, Long value)</strong></td>
<td>Add the given key-value pair to the Trigger's JobDataMap.</td>
</tr>
<tr>
<td><strong>usingJobData(String key, String value)</strong></td>
<td>Add the given key-value pair to the Trigger's JobDataMap.</td>
</tr>
<tr>
<td><strong>withDescription(String description)</strong></td>
<td>Set the given (human-meaningful) description of the Trigger.</td>
</tr>
<tr>
<td><strong>withIdentity(String name)</strong></td>
<td>Use a TriggerKey with the given name and default group to identify the Trigger.</td>
</tr>
</tbody>
</table>
TriggerBuilder<T> withIdentity(String name, String group)
Use a TriggerKey with the given name and group to identify the Trigger.

TriggerBuilder<T> withIdentity(TriggerKey key)
Use the given TriggerKey to identify the Trigger.

TriggerBuilder<T> withPriority(int priority)
Set the Trigger's priority.

<SBT extends T>
TriggerBuilder<SBT> withSchedule(ScheduleBuilder<SBT> scheduleBuilder)
Set the ScheduleBuilder that will be used to define the Trigger's schedule.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

class newTrigger

public static TriggerBuilder<Trigger> newTrigger()
Create a new TriggerBuilder with which to define a specification for a Trigger.

**Returns:**
the new TriggerBuilder

---

**build**

public I build()

Produce the Trigger.

**Returns:**
a Trigger that meets the specifications of the builder.

---

**withIdentity**
public TriggerBuilder\<T\> withIdentity(String name)

Use a TriggerKey with the given name and default group to identify the Trigger.

If none of the 'withIdentity' methods are set on the TriggerBuilder then a random, unique TriggerKey will be generated.

Parameters:
name - the name element for the Trigger's TriggerKey

Returns:
the updated TriggerBuilder

See Also:
TriggerKey,
Trigger.getKey()
If none of the 'withIdentity' methods are set on the TriggerBuilder, then a random, unique TriggerKey will be generated.

**Parameters:**
- **name** - the name element for the Trigger's TriggerKey
- **group** - the group element for the Trigger's TriggerKey

**Returns:**
- the updated TriggerBuilder

**See Also:**
- TriggerKey, Trigger.getKey()
withDescription

public TriggerBuilder\<T\> withDescription(String description)

Set the given (human-meaningful) description of the Trigger.

Parameters:
- description - the description for the Trigger

Returns:
the updated TriggerBuilder

See Also:
Trigger.getDescription()
Set the Trigger's priority. When more than one Trigger have the fire time, the scheduler will fire the one with the highest priority first.

**Parameters:**
- priority - the priority for the Trigger

**Returns:**
- the updated TriggerBuilder

**See Also:**
- `Trigger.DEFAULT_PRIORITY`
- `Trigger.getPriority()`

---

**modifiedByCalendar**

```java
public TriggerBuilder<T> modifiedByCalendar(String calendarName)
```

Set the name of the Calendar that should be applied to this Trigger's schedule.

**Parameters:**
- calendarName - the name of the Calendar to reference.

**Returns:**
- the updated TriggerBuilder

**See Also:**
- `Calendar`
- `Trigger.getCalendarName()`
**startAt**

```java
public TriggerBuilder<T> startAt(Date startTime)
```

Set the time the Trigger should start at - the trigger may or may not fire at this time - depending upon the schedule configured the Trigger. However the Trigger will NOT fire before this time, regardless of the Trigger's schedule.

**Parameters:**
- `startTime`: the start time for the Trigger.

**Returns:**
- the updated TriggerBuilder

**See Also:**
- `Trigger.getStartTime()`,
- `DateBuilder`

---

**startNow**

```java
public TriggerBuilder<T> startNow()
```
Set the time the Trigger should start at to the current moment - the trigger may or may not fire at this time - depending upon the schedule configured for the Trigger.

**Returns:**
the updated TriggerBuilder

**See Also:**
Trigger.getStartTime()

---

**endAt**

```java
public TriggerBuilder<T> endAt(Date endTime)
```

Set the time at which the Trigger will no longer fire - even if schedule has remaining repeats.

**Parameters:**
- **endTime** - the end time for the Trigger. If null, the end time is indefinite.

**Returns:**
the updated TriggerBuilder

**See Also:**
Trigger.getEndTime(), DateBuilder
withSchedule

public <SBT extends T> TriggerBuilder<T> withSchedule(ScheduleBuilder scheduleBuilder)

Set the ScheduleBuilder that will be used to define the Trigger's schedule.

The particular SchedulerBuilder used will dictate the concrete type of Trigger that is produced by the TriggerBuilder.

Parameters:
   scheduleBuilder - the SchedulerBuilder to use.

Returns:
   the updated TriggerBuilder

See Also:
   ScheduleBuilder,
   SimpleScheduleBuilder,
   CronScheduleBuilder,
   CalendarIntervalScheduleBuilder

forJob
public TriggerBuilder<T> forJob(JobKey jobKey)

Set the identity of the Job which should be fired by the produced Trigger.

**Parameters:**
jobKey - the identity of the Job to fire.

**Returns:**
the updated TriggerBuilder

**See Also:**
Trigger.getJobKey()

---

forJob

public TriggerBuilder<T> forJob(String jobName)

Set the identity of the Job which should be fired by the produced Trigger - a JobKey will be produced with the given name and default group.

**Parameters:**
jobName - the name of the job (in default group) to fire.

**Returns:**
the updated TriggerBuilder
public TriggerBuilder<T> forJob(String jobName, String jobGroup)

Set the identity of the Job which should be fired by the produced Trigger - a JobKey will be produced with the given name and group.

Parameters:
  - jobName - the name of the job to fire.
  - jobGroup - the group of the job to fire.

Returns:
  - the updated TriggerBuilder

See Also:
  - Trigger.getJobKey()
public TriggerBuilder<T> forJob(JobDetail jobDetail)

Set the identity of the Job which should be fired by the produced Trigger, by extracting the JobKey from the given job.

Parameters:
  jobDetail - the Job to fire.
Returns:
  the updated TriggerBuilder
See Also:
  Trigger.getJobKey()

usingJobData

public TriggerBuilder<T> usingJobData(String key, String value)

Add the given key-value pair to the Trigger's JobDataMap.

Returns:
  the updated TriggerBuilder
See Also:
  Trigger.getJobDataMap()
usingJobData

public TriggerBuilder<T> usingJobData(String key, Integer value)

Add the given key-value pair to the Trigger's JobDataMap.

Returns:
the updated TriggerBuilder

See Also:
Trigger.getJobDataMap()
usingJobData

public TriggerBuilder<T> usingJobData(String key, Float value)

Add the given key-value pair to the Trigger's JobDataMap.

Returns:
the updated TriggerBuilder
See Also:
Trigger.getJobDataMap()
public TriggerBuilder<T> usingJobData(String key, Double value)

Add the given key-value pair to the Trigger's JobDataMap.

Returns:
the updated TriggerBuilder
See Also:
Trigger.getJobDataMap()

---

usingJobData

public TriggerBuilder<T> usingJobData(String key, Boolean value)

Add the given key-value pair to the Trigger's JobDataMap.

Returns:
the updated TriggerBuilder
See Also:
Trigger.getJobDataMap()
public TriggerBuilder<T> usingJobData(JobDataMap newJobDataMap)

Set the Trigger's JobDataMap, adding any values to it that were already set on this TriggerBuilder using any of the other 'usingJobData' methods.

Returns:
the updated TriggerBuilder

See Also:
Trigger.getJobDataMap()
**Class TriggerKey**

```java
public final class TriggerKey
    extends Key<TriggerKey>
```

Uniquely identifies a `Trigger`.

Keys are composed of both a name and group, and the name must be unique within the group. If only a group is specified then the default group name will be used.

Quartz provides a builder-style API for constructing scheduling-related entities via a Domain-Specific Language (DSL). The DSL can best be utilized through the usage of static imports of the methods on the classes `TriggerBuilder`, `JobBuilder`, `DateBuilder`, `JobKey`, `TriggerKey` and the various `ScheduleBuilder` implementations.

Client code can then use the DSL to write code such as this:

```java
JobDetail job = newJob(MyJob.class)
    .withIdentity("myJob")
    .build();

Trigger trigger = newTrigger()
    .withIdentity(triggerKey("myTrigger", "myTriggerGroup")
    .withSchedule(simpleSchedule()
        .withIntervalInHours(1)
        .repeatForever())
    .startAt(futureDate(10, MINUTES))
    .build();

scheduler.scheduleJob(job, trigger);
```
Field Summary

Fields inherited from class org.quartz.utils.Key
DEFAULT_GROUP

Constructor Summary

TriggerKey(String name)

TriggerKey(String name, String group)

Method Summary

static TriggerKey
triggerKey(String name)

static TriggerKey triggerKey(String name, String group)

Methods inherited from class org.quartz.utils.Key
compareTo, createUniqueName, equals, getGroup, getName, hashCode, toString

Methods inherited from class java.lang.Object
clon, finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

TriggerKey

public TriggerKey(String name)
public TriggerKey(String name, String group)

### Method Detail

**triggerKey**

**triggerKey**

public static TriggerKey triggerKey(String name)
org.quartz Interface TriggerListener

All Known Implementing Classes:
BroadcastTriggerListener, LoggingTriggerHistoryPlugin, TriggerListenerSupport

public interface TriggerListener

The interface to be implemented by classes that want to be informed when a Trigger fires. In general, applications that use a Scheduler will not have use for this mechanism.

Author:
James House

See Also:
Scheduler#addTriggerListener(TriggerListener, Matcher), Matcher, Trigger, JobListener, JobExecutionContext

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getName()</td>
</tr>
<tr>
<td>Get the name of the TriggerListener.</td>
</tr>
<tr>
<td>void triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</td>
</tr>
<tr>
<td>Called by the Scheduler when a Trigger has fired, its associated JobDetail has been executed, and its triggered(xx) method has been called.</td>
</tr>
<tr>
<td>void triggerFired(Trigger trigger, JobExecutionContext context)</td>
</tr>
<tr>
<td>Called by the Scheduler when a Trigger has fired, and its associated JobDetail is about to be executed.</td>
</tr>
<tr>
<td>void triggerMisfired(Trigger trigger)</td>
</tr>
<tr>
<td>Called by the Scheduler when a Trigger has misfired.</td>
</tr>
<tr>
<td>boolean vetoJobExecution(Trigger trigger, JobExecutionContext context)</td>
</tr>
</tbody>
</table>
| Called by the Scheduler when a Trigger has fired, and its
JobDetail is about to be executed.

### Method Detail

**getName**

```java
String getName()
```

Get the name of the TriggerListener.

**triggerFired**

```java
void triggerFired(Trigger trigger,
                 JobExecutionContext context)
```

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.

It is called before the vetoJobExecution(..) method of this interface.

**Parameters:**
- `trigger` - The Trigger that has fired.
- `context` - The JobExecutionContext that will be passed to the Job's execute(xx) method.

**vetoJobExecution**

```java
boolean vetoJobExecution(Trigger trigger,
                         JobExecutionContext context)
```

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed. If the implementation vetos the execution (via returning true), the job's execute method will not be called.

It is called after the triggerFired(..) method of this interface.
**Parameters:**
- trigger - The Trigger that has fired.
- context - The JobExecutionContext that will be passed to the Job's execute(xx) method.

---

**triggerMisfired**

```java
void triggerMisfired(Trigger trigger)
```

Called by the Scheduler when a Trigger has misfired.

Consideration should be given to how much time is spent in this method, as it will affect all triggers that are misfiring. If you have lots of triggers misfiring at once, it could be an issue if this method does a lot.

**Parameters:**
- trigger - The Trigger that has misfired.

---

**triggerComplete**

```java
void triggerComplete(Trigger trigger,
                     JobExecutionContext context,
                     Trigger.CompletedExecutionInstruction triggerInstructionCode)
```

Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered(xx) method has been called.

**Parameters:**
- trigger - The Trigger that was fired.
- context - The JobExecutionContext that was passed to the Job's execute(xx) method.
- triggerInstructionCode - the result of the call on the Trigger's triggered(xx) method.
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
</tr>
</tbody>
</table>
public class TriggerUtils
extends Object

Convenience and utility methods for working with Triggers.

Author:
James House

See Also:
CronTrigger, SimpleTrigger, DateBuilder

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Date computeEndTimeToAllowParticularNumberOfFirings</code>(org.quartz.spi.OperableTrigger trigg, Calendar cal, int numTimes)</td>
<td>Compute the date that is 1 second after the Nth firing of the associated Calendar into consideration.</td>
</tr>
<tr>
<td>static <code>List&lt;Date&gt; computeFireTimes</code>(org.quartz.spi.OperableTrigger trigg, Date from, Date to)</td>
<td>Returns a list of Dates that are the next fire times of a Trigger range.</td>
</tr>
<tr>
<td>static <code>List&lt;Date&gt; computeFireTimesBetween</code>(org.quartz.spi.OperableTrigger trigg, Date from, Date to)</td>
<td>Returns a list of Dates that are the next fire times of a Trigger range.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Method Detail
computeFireTimes

```java
public static List<Date> computeFireTimes(org.quartz.spi.OperableTrigger trigg,
                                         Calendar cal,
                                         int numTimes)
```

Returns a list of Dates that are the next fire times of a Trigger. The input trigger will be cloned before any work is done, so you need not worry about its state being altered by this method.

**Parameters:**
- `trigg` - The trigger upon which to do the work
- `cal` - The calendar to apply to the trigger's schedule
- `numTimes` - The number of next fire times to produce

**Returns:**
List of java.util.Date objects

computeEndTimeToAllowParticularNumberOfFirings

```java
public static Date computeEndTimeToAllowParticularNumberOfFirings(org.quartz.spi.OperableTrigger trigg,
                                                                  Calendar cal,
                                                                  int numTimes)
```

Compute the Date that is 1 second after the Nth firing of the given Trigger, taking the trigger's associated calendar into consideration. The input trigger will be cloned before any work is done, so you need not worry about its state being altered by this method.

**Parameters:**
- `trigg` - The trigger upon which to do the work
- `cal` - The calendar to apply to the trigger's schedule
- `numTimes` - The number of next fire times to produce

**Returns:**
the computed Date, or null if the trigger (as configured) will not fire that many times.

computeFireTimesBetween
public static List<Date> computeFireTimesBetween(org.quartz.spi.OperableTrigger trigg, Calendar cal, Date from, Date to)

Returns a list of Dates that are the next fire times of a Trigger that fall within the given date range. The input trigger will be cloned before any work is done, so you need not worry about its state being altered by this method.

NOTE: if this is a trigger that has previously fired within the given date range, then firings which have already occurred will not be listed in the output List.

Parameters:
trigg - The trigger upon which to do the work
cal - The calendar to apply to the trigger's schedule
from - The starting date at which to find fire times
to - The ending date at which to stop finding fire times

Returns:
List of java.util.Date objects
org.quartz Class UnableToInterruptJobException

java.lang.Object  
   ▼ java.lang.Throwable  
       ▼ java.lang.Exception  
           ▼ org.quartz.SchedulerException  
               ▼ org.quartz.UnableToInterruptJobException

All Implemented Interfaces:
    Serializable

public class UnableToInterruptJobException
    extends SchedulerException

An exception that is thrown to indicate that a call to InterruptableJob.interrupt() failed without interrupting the Job.

Author:
    James House

See Also:
    InterruptableJob.interrupt(), Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnableToInterruptJobException(String msg)</td>
<td>Create a UnableToInterruptJobException with the given message.</td>
</tr>
<tr>
<td>UnableToInterruptJobException(Throwable cause)</td>
<td>Create a UnableToInterruptJobException with the given cause.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Methods inherited from class org.quartz.SchedulerException</th>
</tr>
</thead>
<tbody>
<tr>
<td>getUnderlyingException, toString</td>
</tr>
</tbody>
</table>


UnableToInterruptJobException

public UnableToInterruptJobException(String msg)

Create an UnableToInterruptJobException with the given message.

UnableToInterruptJobException

public UnableToInterruptJobException(Throwable cause)

Create an UnableToInterruptJobException with the given cause.
### Uses of Interface

**org.quartz.Calendar**

<table>
<thead>
<tr>
<th>Packages that use <a href="#">Calendar</a></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of <strong>Quartz</strong>, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
<td>Contains the core classes and interfaces for the <strong>Quartz</strong> job scheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.calendar</strong></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of [Calendar](#) in *org.quartz*

### Methods in *org.quartz* that return [Calendar](#)

<table>
<thead>
<tr>
<th>return type</th>
<th>method name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>getBaseCalendar()</td>
<td>Get the base calendar.</td>
</tr>
<tr>
<td>Calendar</td>
<td>JobExecutionContext.getCalendar()</td>
<td></td>
</tr>
</tbody>
</table>
Get a handle to the calendar referenced by the Trigger instance that fired the Job.

```
Calendar scheduler.getCalendar(String calName)
Get the calendar instance with the given name.
```

### Methods in `org.quartz` with parameters of type `Calendar`

<table>
<thead>
<tr>
<th>void</th>
<th>Scheduler.addCalendar(String calName, Calendar calendar)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add (register) the given calendar to the Scheduler.</td>
</tr>
<tr>
<td>static Date</td>
<td>TriggerUtils.computeEndTimeToAllowParticularNumberOfFirings(Calendar cal, int numTimes)</td>
</tr>
<tr>
<td></td>
<td>Compute the date that is 1 second after the Nth firing of the Calendar into consideration.</td>
</tr>
<tr>
<td>static List&lt;Date&gt;</td>
<td>TriggerUtils.computeFireTimes(org.quartz.spi.OperableTrigger trigg)</td>
</tr>
<tr>
<td></td>
<td>Returns a list of Dates that are the next fire times of a Trigger.</td>
</tr>
<tr>
<td>static List&lt;Date&gt;</td>
<td>TriggerUtils.computeFireTimesBetween(org.quartz.spi.OperableTrigger trigg, Date from, Date to)</td>
</tr>
<tr>
<td></td>
<td>Returns a list of Dates that are the next fire times of a Trigger between two dates.</td>
</tr>
<tr>
<td>void</td>
<td>Calendar.setBaseCalendar(Calendar baseCalendar)</td>
</tr>
<tr>
<td></td>
<td>Set a new base calendar or remove the existing one.</td>
</tr>
</tbody>
</table>

### Uses of `Calendar` in `org.quartz.core`

### Methods in `org.quartz.core` that return `Calendar`

<table>
<thead>
<tr>
<th>Calendar</th>
<th>QuartzScheduler.getCalendar(String calName)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the calendar instance with the given name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Calendar</th>
<th>RemotableQuartzScheduler.getCalendar(String calName)</th>
</tr>
</thead>
</table>

### Methods in `org.quartz.core` with parameters of type `Calendar`

<table>
<thead>
<tr>
<th>void</th>
<th>QuartzScheduler.addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</th>
</tr>
</thead>
</table>
Add (register) the given calendar to the Scheduler.

```java
void RemotableQuartzScheduler.addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)
```

## Uses of `Calendar` in `org.quartz.impl`

### Methods in `org.quartz.impl` that return `Calendar`

<table>
<thead>
<tr>
<th>Calendar</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Calendar</code></td>
<td>JobExecutionContextImpl.getCalendar()</td>
<td></td>
</tr>
<tr>
<td><code>Calendar</code></td>
<td>RemoteScheduler.getCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>Calendar</code></td>
<td>StdScheduler.getCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>Calendar</code></td>
<td>RemoteMBeanScheduler.getCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl` with parameters of type `Calendar`

<table>
<thead>
<tr>
<th>void</th>
<th>RemoteScheduler.addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</th>
<th>Calls the equivalent method on the 'proxied' QuartzScheduler.</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>StdScheduler.addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td>RemoteMBeanScheduler.addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
</tbody>
</table>

## Uses of `Calendar` in `org.quartz.impl.calendar`
Classes in `org.quartz.impl.calendar` that implement `Calendar`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AnnualCalendar</code></td>
<td>This implementation of the Calendar excludes a set of days of the year.</td>
</tr>
<tr>
<td><code>BaseCalendar</code></td>
<td>This implementation of the Calendar may be used (you don't have to) as a base class for more sophisticated one's.</td>
</tr>
<tr>
<td><code>CronCalendar</code></td>
<td>This implementation of the Calendar excludes the set of times expressed by a given <code>CronExpression</code>.</td>
</tr>
<tr>
<td><code>DailyCalendar</code></td>
<td>This implementation of the Calendar excludes (or includes - see below) a specified time range each day.</td>
</tr>
<tr>
<td><code>HolidayCalendar</code></td>
<td>This implementation of the Calendar stores a list of holidays (full days that are excluded from scheduling).</td>
</tr>
<tr>
<td><code>MonthlyCalendar</code></td>
<td>This implementation of the Calendar excludes a set of days of the month.</td>
</tr>
<tr>
<td><code>WeeklyCalendar</code></td>
<td>This implementation of the Calendar excludes a set of days of the week.</td>
</tr>
</tbody>
</table>

Methods in `org.quartz.impl.calendar` that return `Calendar`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getBaseCalendar()</code></td>
<td>Get the base calendar.</td>
</tr>
</tbody>
</table>

Methods in `org.quartz.impl.calendar` with parameters of type `Calendar`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setBaseCalendar(Calendar baseCalendar)</code></td>
<td><code>Calendar</code> baseCalendar</td>
<td>Set a new base calendar or remove the existing one</td>
</tr>
</tbody>
</table>
Constructors in `org.quartz.impl.calendar` with parameters of type `Calendar`

**AnnualCalendar** *(Calendar baseCalendar)*

**AnnualCalendar** *(Calendar baseCalendar, TimeZone timeZone)*

**BaseCalendar** *(Calendar baseCalendar)*

**BaseCalendar** *(Calendar baseCalendar, TimeZone timeZone)*

**CronCalendar** *(Calendar baseCalendar, String expression)*

Create a CronCalendar with the given cron expression and baseCalendar.

**CronCalendar** *(Calendar baseCalendar, String expression, TimeZone timeZone)*

Create a CronCalendar with the given cron expression, baseCalendar, and TimeZone.

**DailyCalendar** *(Calendar baseCalendar, Calendar rangeStartingCalendar, Calendar rangeEndingCalendar)*

Create a DailyCalendar with a time range defined by the specified java.util.Calendar and the specified baseCalendar.

**DailyCalendar** *(Calendar baseCalendar, int rangeStartingHourOfDay, int rangeStartingMinute, int rangeStartingSecond, int rangeStartingMillis, int rangeEndingHourOfDay, int rangeEndingMinute, int rangeEndingSecond, int rangeEndingMillis)*

Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.

**DailyCalendar** *(Calendar baseCalendar, long rangeStartingTimeInMillis, long rangeEndingTimeInMillis)*

Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.

**DailyCalendar** *(Calendar baseCalendar, String rangeStartingTime, String rangeEndingTime)*

Create a DailyCalendar with a time range defined by the specified strings and the specified baseCalendar.

**DailyCalendar** *(Calendar baseCalendar, TimeZone timeZone,TimeZone timeZone)*
long rangeStartingTimeInMillis, long rangeEndingTimeInMillis
Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.

**HolidayCalendar** (Calendar baseCalendar)

**HolidayCalendar** (Calendar baseCalendar, TimeZone timeZone)

**MonthlyCalendar** (Calendar baseCalendar)

**MonthlyCalendar** (Calendar baseCalendar, TimeZone timeZone)

**WeeklyCalendar** (Calendar baseCalendar)

**WeeklyCalendar** (Calendar baseCalendar, TimeZone timeZone)

---

**Uses of Calendar in org.quartz.impl.jdbcjobstore**

**Fields in org.quartz.impl.jdbcjobstore with type parameters of type Calendar**

| protected HashMap<String, Calendar> | JobStoreSupport.calendarCache |

**Methods in org.quartz.impl.jdbcjobstore that return Calendar**

| protected Calendar | JobStoreSupport.retrieveCalendar (Connection conn, String calName) |

| Calendar | JobStoreSupport.retrieveCalendar (String calName) |

Retrieve the given Trigger.

| Calendar | StdJDBCDelegate.selectCalendar (Connection conn, String calendarName) |

Select a calendar.
Calendar

DriverDelegate.selectCalendar(Connection conn, String calendarName)
Select a calendar.

Methods in org.quartz.impl.jdbcjobstore with parameters of type Calendar

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int StdJDBCDelegate.insertCalendar</td>
<td>Connection conn, String calendarName, Calendar calendar</td>
<td>Insert a new calendar.</td>
</tr>
<tr>
<td>int DriverDelegate.insertCalendar</td>
<td>Connection conn, String calendarName, Calendar calendar</td>
<td>Insert a new calendar.</td>
</tr>
<tr>
<td>int PointbaseDelegate.insertCalendar</td>
<td>Connection conn, String calendarName, Calendar calendar</td>
<td>Insert a new calendar.</td>
</tr>
<tr>
<td>protected void JobStoreSupport.storeCalendar</td>
<td>Connection conn, String calName, Calendar calendar, boolean replaceExisting, boolean updateTriggers</td>
<td>Store the given Calendar.</td>
</tr>
<tr>
<td>void JobStoreSupport.storeCalendar</td>
<td>String calName, Calendar calendar, boolean replaceExisting, boolean updateTriggers</td>
<td>Store the given Calendar.</td>
</tr>
<tr>
<td>int StdJDBCDelegate.updateCalendar</td>
<td>Connection conn, String calendarName, Calendar calendar</td>
<td>Update a calendar.</td>
</tr>
<tr>
<td>int DriverDelegate.updateCalendar</td>
<td>Connection conn, String calendarName, Calendar calendar</td>
<td>Update a calendar.</td>
</tr>
<tr>
<td>int PointbaseDelegate.updateCalendar</td>
<td>Connection conn, String calendarName, Calendar calendar</td>
<td>Update a calendar.</td>
</tr>
</tbody>
</table>

Uses of Calendar in org.quartz.impl.jdbcjobstore.oracle
### Methods in `org.quartz.impl.jdbcjobstore.oracle` with parameters of type `Calendar`

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int OracleDelegate.insertCalendar(Connection conn, String calendarName, Calendar calendar)</code></td>
<td></td>
</tr>
<tr>
<td><code>int OracleDelegate.updateCalendar(Connection conn, String calendarName, Calendar calendar)</code></td>
<td></td>
</tr>
</tbody>
</table>

### Uses of `Calendar` in `org.quartz.impl.triggers`

### Methods in `org.quartz.impl.triggers` with parameters of type `Calendar`

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AbstractTrigger.computeFirstFireTime(Calendar calendar)</code></td>
<td>This method should not be used by the Quartz client.</td>
<td></td>
</tr>
<tr>
<td><code>CronTriggerImpl.computeFirstFireTime(Calendar calendar)</code></td>
<td>Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on associated calendar.</td>
<td></td>
</tr>
<tr>
<td><code>CalendarIntervalTriggerImpl.computeFirstFireTime(Calendar calendar)</code></td>
<td>Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on associated calendar.</td>
<td></td>
</tr>
<tr>
<td><code>SimpleTriggerImpl.computeFirstFireTime(Calendar calendar)</code></td>
<td>Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on associated calendar.</td>
<td></td>
</tr>
<tr>
<td><code>AbstractTrigger.triggered(Calendar calendar)</code></td>
<td>This method should not be used by the Quartz client.</td>
<td></td>
</tr>
<tr>
<td><code>CronTriggerImpl.triggered(Calendar calendar)</code></td>
<td>Called when the Scheduler has decided to 'fire' the trigger (execute associated Job), in order to give the Trigger a chance to update itself for triggering (if any).</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>void CalendarIntervalTriggerImpl::triggered(Calendar calendar)</code></td>
<td>Called when the Scheduler has decided to 'fire' the trigger (execute associated Job), in order to give the Trigger a chance to update itself for triggering (if any).</td>
<td></td>
</tr>
<tr>
<td><code>void SimpleTriggerImpl::triggered(Calendar calendar)</code></td>
<td>Called when the Scheduler has decided to 'fire' the trigger (execute associated Job), in order to give the Trigger a chance to update itself for triggering (if any).</td>
<td></td>
</tr>
<tr>
<td><code>abstract void AbstractTrigger::updateAfterMisfire(Calendar cal)</code></td>
<td>This method should not be used by the Quartz client.</td>
<td></td>
</tr>
<tr>
<td><code>void CronTriggerImpl::updateAfterMisfire(Calendar cal)</code></td>
<td>Updates the CronTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the CronTrigger was created.</td>
<td></td>
</tr>
<tr>
<td><code>void CalendarIntervalTriggerImpl::updateAfterMisfire(Calendar cal)</code></td>
<td>Updates the DateIntervalTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the DateIntervalTrigger was created.</td>
<td></td>
</tr>
<tr>
<td><code>void SimpleTriggerImpl::updateAfterMisfire(Calendar cal)</code></td>
<td>Updates the SimpleTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the SimpleTrigger was created.</td>
<td></td>
</tr>
<tr>
<td><code>abstract void AbstractTrigger::updateWithNewCalendar(Calendar cal, long misfireThreshold)</code></td>
<td>This method should not be used by the Quartz client.</td>
<td></td>
</tr>
<tr>
<td><code>void CronTriggerImpl::updateWithNewCalendar(Calendar calendar, long misfireThreshold)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>void CalendarIntervalTriggerImpl::updateWithNewCalendar(Calendar calendar, long misfireThreshold)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>void SimpleTriggerImpl::updateWithNewCalendar(Calendar calendar, long misfireThreshold)</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Uses of Calendar in org.quartz.locality

#### Methods in org.quartz.locality with parameters of type Calendar

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>DelegatingLocalityTrigger.computeFirstFireTime(Calendar calendar)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>DelegatingLocalityTrigger.triggered(Calendar calendar)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>DelegatingLocalityTrigger.updateAfterMisfire(Calendar cal)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>DelegatingLocalityTrigger.updateWithNewCalendar(Calendar cal, long misfireThreshold)</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of Calendar in org.quartz.simpl

#### Fields in org.quartz.simpl with type parameters of type Calendar

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>HashMap&lt;String, Calendar&gt;</td>
<td>RAMJobStore.calendarsByName</td>
</tr>
</tbody>
</table>

#### Methods in org.quartz.simpl that return Calendar

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>RAMJobStore.retrieveCalendar(String calName)</td>
<td>Retrieve the given Trigger.</td>
</tr>
</tbody>
</table>

#### Methods in org.quartz.simpl with parameters of type Calendar

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>RAMJobStore.storeCalendar(String name, Calendar calendar, boolean replaceExisting, boolean updateTriggers)</td>
<td>Store the given Calendar.</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
## Uses of Class

**org.quartz.CalendarIntervalScheduleBuilder**

### Packages that use **CalendarIntervalScheduleBuilder**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
</tbody>
</table>

### Uses of **CalendarIntervalScheduleBuilder** in **org.quartz**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CalendarIntervalScheduleBuilder</td>
<td>static CalendarIntervalScheduleBuilder.calendarIntervalSchedule()</td>
</tr>
<tr>
<td>Specify the time unit and interval for the Trigger to be produced.</td>
<td>CalendarIntervalScheduleBuilder.withInterval(DateBuilder.IntervalUnit unit)</td>
</tr>
<tr>
<td>Specify an interval in the IntervalUnit.DAY that the produced</td>
<td>CalendarIntervalScheduleBuilder.withIntervalInDays()</td>
</tr>
<tr>
<td>Specify an interval in the IntervalUnit.HOUR that the produced</td>
<td>CalendarIntervalScheduleBuilder.withIntervalInHours()</td>
</tr>
<tr>
<td>Specify an interval in the IntervalUnit.MINUTE that the produced</td>
<td>CalendarIntervalScheduleBuilder.withIntervalInMinutes()</td>
</tr>
<tr>
<td>Specify an interval in the IntervalUnit.MONTH that the produced</td>
<td>CalendarIntervalScheduleBuilder.withIntervalInMonths()</td>
</tr>
<tr>
<td>Specify an interval in the IntervalUnit.SECOND that the produced</td>
<td>CalendarIntervalScheduleBuilder.withIntervalInSeconds()</td>
</tr>
<tr>
<td>Specify an interval in the IntervalUnit.WEEK that the Produced</td>
<td>CalendarIntervalScheduleBuilder.withIntervalInWeeks()</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>withIntervalInYears</code></td>
<td>Specify an interval in the <code>IntervalUnit.YEAR</code> that the produced <code>CalendarIntervalScheduleBuilder</code> will use.</td>
</tr>
<tr>
<td><code>withMisfireHandlingInstructionDoNothing</code></td>
<td>If the Trigger misfires, use the <code>CalendarIntervalTrigger.MISFIRE_INSTRUCTION_DO_NOTHING</code> to handle the misfire.</td>
</tr>
<tr>
<td><code>withMisfireHandlingInstructionFireAndProceed</code></td>
<td>If the Trigger misfires, use the <code>CalendarIntervalTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW</code> to handle the misfire.</td>
</tr>
<tr>
<td><code>withMisfireHandlingInstructionIgnoreMisfires</code></td>
<td>If the Trigger misfires, use the <code>Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRES</code> to handle the misfire.</td>
</tr>
</tbody>
</table>
## Uses of Interface
### org.quartz.CalendarIntervalTrigger

<table>
<thead>
<tr>
<th>Packages that use</th>
<th>CalendarIntervalTrigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
</tbody>
</table>

### Uses of CalendarIntervalTrigger in org.quartz

| Methods in org.quartz that return types with arguments of type CalendarIntervalTrigger |
|-----------------------------------|---------------------------------|
| TriggerBuilder<CalendarIntervalTrigger> | CalendarIntervalTrigger.getTriggerBuilder |

### Uses of CalendarIntervalTrigger in org.quartz.impl.triggers

<table>
<thead>
<tr>
<th>Classes in org.quartz.impl.triggers that implement CalendarIntervalTrigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalendarIntervalTriggerImpl</td>
</tr>
</tbody>
</table>

A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.

| Methods in org.quartz.impl.triggers that return types with arguments of type CalendarIntervalTrigger |
CalendarIntervalTriggerImpl.<code>getSchedule</code>

Get a <code>ScheduleBuilder</code> that is configured to produce a schedule identical to this trigger.
Uses of Class
org.quartz.CronExpression

<table>
<thead>
<tr>
<th>Packages that use CronExpression</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.calendar</td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
</tr>
<tr>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
</tbody>
</table>

Uses of CronExpression in org.quartz.impl.calendar

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl.calendar that return CronExpression</th>
</tr>
</thead>
<tbody>
<tr>
<td>CronExpression CronCalendar. getCronExpression() (</td>
</tr>
<tr>
<td>Returns the object representation of the cron expression that</td>
</tr>
<tr>
<td>defines the dates and times this calendar excludes.</td>
</tr>
</tbody>
</table>

<p>| Methods in org.quartz.impl.calendar with parameters of type    |</p>
<table>
<thead>
<tr>
<th>CronExpression</th>
</tr>
</thead>
<tbody>
<tr>
<td>void CronCalendar. setCronExpression(CronExpression expression)</td>
</tr>
<tr>
<td>Sets the cron expression for the calendar to a new value</td>
</tr>
</tbody>
</table>

Uses of CronExpression in org.quartz.impl.triggers

<p>| Methods in org.quartz.impl.triggers with parameters of type    |</p>
<table>
<thead>
<tr>
<th>CronExpression</th>
</tr>
</thead>
<tbody>
<tr>
<td>void CronTriggerImpl. setCronExpression(CronExpression cronExpression)</td>
</tr>
<tr>
<td>Set the CronExpression to the given one.</td>
</tr>
</tbody>
</table>
Overview  Package  Class  Tree  Deprecated  Index  Help
# Uses of Class

**org.quartz.CronScheduleBuilder**

<table>
<thead>
<tr>
<th>Packages that use CronScheduleBuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
</tr>
</tbody>
</table>

## Uses of CronScheduleBuilder in org.quartz

<table>
<thead>
<tr>
<th>Methods in org.quartz that return CronScheduleBuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static</strong> CronScheduleBuilder</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>static</strong> CronScheduleBuilder</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>CronScheduleBuilder</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>static</strong> CronScheduleBuilder</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>static</strong> CronScheduleBuilder</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>CronScheduleBuilder</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>CronScheduleBuilder</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
CronScheduleBuilder

CronScheduleBuilder.withMisfireHandlingInstruction.

If the Trigger misfires, use the Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY.

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.CronTrigger

<table>
<thead>
<tr>
<th>Packages that use CronTrigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
</tr>
<tr>
<td>org.quartz.core.jmx</td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
</tr>
</tbody>
</table>

| Uses of CronTrigger in org.quartz |

<table>
<thead>
<tr>
<th>Methods in org.quartz that return types with arguments of type CronTrigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerBuilder&lt;CronTrigger&gt;</td>
</tr>
</tbody>
</table>

| Uses of CronTrigger in org.quartz.core.jmx |

<table>
<thead>
<tr>
<th>Methods in org.quartz.core.jmx with parameters of type CronTrigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>static CompositeData CronTriggerSupport.toCompositeData(CronTrigger trigger)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method parameters in org.quartz.core.jmx with type arguments of type CronTrigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>static TabularData CronTriggerSupport.toTabularData(List&lt;? extends CronTrigger&gt; triggers)</td>
</tr>
</tbody>
</table>
Uses of **CronTrigger** in **org.quartz.impl.triggers**

### Classes in **org.quartz.impl.triggers** that implement **CronTrigger**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CronTriggerImpl</td>
<td>A concrete Trigger that is used to fire a JobDetail at given moments in time, defined with Unix 'cron-like' definitions.</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz.impl.triggers** that return types with arguments of type **CronTrigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CronTriggerImpl.getScheduleBuilder()</td>
<td>Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.</td>
</tr>
</tbody>
</table>
# Uses of Class

**org.quartz.DateBuilder**

## Packages that use **DateBuilder**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of <strong>Quartz</strong>, containing the client-side interfaces.</td>
</tr>
</tbody>
</table>

## Uses of **DateBuilder** in org.quartz

### Methods in **org.quartz** that return **DateBuilder**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>atHourMinuteAndSecond</code></td>
<td>Set the hour (0-23) and minute (0-59) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><code>atHourOfDay</code></td>
<td>Set the hour (0-23) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><code>atMinute</code></td>
<td>Set the minute (0-59) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><code>atSecond</code></td>
<td>Set the second (0-59) for the Date that will be built by this builder, and truncate the milliseconds to 000.</td>
</tr>
<tr>
<td><code>inLocale</code></td>
<td>Set the Locale for the Date that will be built by this builder (if &quot;null&quot;, system default will be used)</td>
</tr>
<tr>
<td><code>inMonth</code></td>
<td>Set the month (1-12) for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><code>inMonthOnDay</code></td>
<td>Set the TimeZone for the Date that will be built by this builder.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>null</code></td>
<td>If &quot;null&quot;, system default will be used</td>
</tr>
<tr>
<td><code>inYear</code></td>
<td>Set the year for the Date that will be built by this builder.</td>
</tr>
<tr>
<td><code>newDate</code></td>
<td>Create a DateBuilder, with initial settings for the current date and time in the system default timezone.</td>
</tr>
<tr>
<td><code>newDateInLocale</code></td>
<td>Create a DateBuilder, with initial settings for the current date and time in the given locale.</td>
</tr>
<tr>
<td><code>newDateInTimezone</code></td>
<td>Create a DateBuilder, with initial settings for the current date and time in the given timezone.</td>
</tr>
<tr>
<td><code>newDateInTimeZoneAndLocale</code></td>
<td>Create a DateBuilder, with initial settings for the current date and time in the given timezone and locale.</td>
</tr>
<tr>
<td><code>onDay</code></td>
<td>Set the day of month (1-31) for the Date that will be built by this builder.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.DateBuilder.IntervalUnit**

## Packages that use `DateBuilder.IntervalUnit`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
</tbody>
</table>

## Uses of `DateBuilder.IntervalUnit` in `org.quartz`

## Methods in `org.quartz` that return `DateBuilder.IntervalUnit`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalendarIntervalTrigger.getRepeatIntervalUnit()</code></td>
<td>Get the interval unit - the time unit on which the interval applies.</td>
</tr>
<tr>
<td><code>static DateBuilder.IntervalUnit.valueOf(String name)</code></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td><code>static DateBuilder.IntervalUnit[] values()</code></td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

## Methods in `org.quartz` with parameters of type `DateBuilder.IntervalUnit`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DateBuilder.futureDate(int interval, DateBuilder.IntervalUnit unit)</code></td>
<td>Specify the time unit and interval for the Trigger to be produced.</td>
</tr>
<tr>
<td><code>CalendarIntervalScheduleBuilder.withInterval(DateBuilder.IntervalUnit unit)</code></td>
<td>Specify the time unit and interval for the Trigger to be produced.</td>
</tr>
</tbody>
</table>
Uses of `DateBuilder.IntervalUnit` in `org.quartz.impl.triggers`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.impl.triggers</code> that return <code>DateBuilder.IntervalUnit</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DateBuilder.IntervalUnit</code></td>
</tr>
<tr>
<td><code>CalendarIntervalTriggerImpl.getRepeatIntervalUnit</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.impl.triggers</code> with parameters of type <code>DateBuilder.IntervalUnit</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void CalendarIntervalTriggerImpl.setRepeatIntervalUnit(DateBuilder.IntervalUnit)</code></td>
</tr>
<tr>
<td>Set the interval unit - the time unit on with the interval applies.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructors in <code>org.quartz.impl.triggers</code> with parameters of type <code>DateBuilder.IntervalUnit</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalendarIntervalTriggerImpl(String name, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)</code></td>
</tr>
<tr>
<td>Create a <code>DateIntervalTrigger</code> that will occur immediately, and repeat at the given interval.</td>
</tr>
<tr>
<td><code>CalendarIntervalTriggerImpl(String name, Date startTime, Date endTime, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)</code></td>
</tr>
<tr>
<td>Create a <code>DateIntervalTrigger</code> that will occur at the given time, and repeat at the given interval until the given end time.</td>
</tr>
<tr>
<td><code>CalendarIntervalTriggerImpl(String name, String group, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)</code></td>
</tr>
<tr>
<td>Create a <code>DateIntervalTrigger</code> that will occur immediately, and repeat at the given interval.</td>
</tr>
<tr>
<td><code>CalendarIntervalTriggerImpl(String name, String group, Date startTime, Date endTime, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)</code></td>
</tr>
<tr>
<td>Create a <code>DateIntervalTrigger</code> that will occur at the given time, and repeat at the given interval until the given end time.</td>
</tr>
<tr>
<td><code>CalendarIntervalTriggerImpl(String name, String group, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)</code></td>
</tr>
<tr>
<td>Create a <code>DateIntervalTrigger</code> that will occur at the given time, and repeat at the given interval until the given end time.</td>
</tr>
<tr>
<td><code>CalendarIntervalTriggerImpl(String name, String group, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)</code></td>
</tr>
<tr>
<td>Create a <code>DateIntervalTrigger</code> that will occur at the given time, and repeat at the given interval until the given end time.</td>
</tr>
</tbody>
</table>
String jobName, String jobGroup, Date startTime, Date endTime, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)

Create a DateIntervalTrigger that will occur at the given time, fire the identified Job and repeat at the given interval until the given end time.
## Uses of Class

**org.quartz.DisallowConcurrentExecution**

### Packages that use *DisallowConcurrentExecution*

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz</code></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><code>org.quartz.jobs</code></td>
<td></td>
</tr>
</tbody>
</table>

### Uses of *DisallowConcurrentExecution* in `org.quartz`

### Classes in `org.quartz` with annotations of type *DisallowConcurrentExecution*

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StatefulJob</code></td>
<td>Deprecated. use <code>DisallowConcurrentExecution</code> and/or <code>PersistJobDataAfterExecution</code> annotations instead.</td>
</tr>
</tbody>
</table>

### Uses of *DisallowConcurrentExecution* in `org.quartz.jobs`

### Classes in `org.quartz.jobs` with annotations of type *DisallowConcurrentExecution*

- **DirectoryScanJob**
  - Inspects a directory and compares whether any files' "last modified dates" have changed since the last time it was inspected.

- **FileScanJob**
  - Inspects a file and compares whether it's "last modified date" has changed since the last time it was inspected.
Uses of Class
org.quartz.ExecuteInJTATransaction

No usage of org.quartz.ExecuteInJTATransaction

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.InterruptableJob

No usage of org.quartz.InterruptableJob

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.Job

### Packages that use Job

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.jobs</td>
<td>ƴ</td>
</tr>
<tr>
<td>org.quartz.jobs.ee.ejb</td>
<td>ƴ</td>
</tr>
<tr>
<td>org.quartz.jobs.ee.jms</td>
<td>ƴ</td>
</tr>
<tr>
<td>org.quartz.jobs.ee.jmx</td>
<td>ƴ</td>
</tr>
<tr>
<td>org.quartz.jobs.ee.mail</td>
<td>ƴ</td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td>ƴ</td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of Job in org.quartz

### Subinterfaces of Job in org.quartz

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterruptableJob</td>
<td>The interface to be implemented by Jobs that provide a mechanism for having their execution interrupted.</td>
</tr>
<tr>
<td>StatefulJob</td>
<td><strong>Deprecated.</strong> use DisallowConcurrentExecution and/or PersistJobDataAfterExecution annotations instead.</td>
</tr>
</tbody>
</table>
### Methods in `org.quartz` that return `Job`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobExecutionContext.getJobInstance()</code></td>
<td>Get the instance of the Job that was created for this execution.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz` that return types with arguments of type `Job`

<table>
<thead>
<tr>
<th>Class&lt;? extends Job&gt;</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>JobDetail.getJobClass()</code></td>
<td>Get the instance of Job that will be executed.</td>
</tr>
</tbody>
</table>

### Method parameters in `org.quartz` with type arguments of type `Job`

<table>
<thead>
<tr>
<th>Class&lt;? extends Job&gt;</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>JobBuilder.newJob(Class&lt;? extends Job&gt; jobClass)</code></td>
<td>Create a JobBuilder with which to define a JobDetail, and set the class name of the Job to be executed.</td>
</tr>
<tr>
<td></td>
<td><code>JobBuilder.ofType(Class&lt;? extends Job&gt; jobClass)</code></td>
<td>Set the class which will be instantiated and executed when a Trigger fires that is associated with this JobDetail.</td>
</tr>
</tbody>
</table>

### Uses of `Job` in `org.quartz.impl`

### Methods in `org.quartz.impl` that return `Job`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobExecutionContextImpl.getJobInstance()</code></td>
<td></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl` that return types with arguments of type `Job`

<table>
<thead>
<tr>
<th>Class&lt;? extends Job&gt;</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>JobDetailImpl.getJobClass()</code></td>
<td></td>
</tr>
</tbody>
</table>

### Method parameters in `org.quartz.impl` with type arguments of type `Job`

<table>
<thead>
<tr>
<th>Class&lt;? extends Job&gt;</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>JobDetailImpl.setJobClass(Class&lt;? extends Job&gt; jobClass)</code></td>
<td>Set the instance of Job that will be executed.</td>
</tr>
</tbody>
</table>
### Constructors in `org.quartz.impl` with parameters of type `Job`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobExecutionContextImpl</code></td>
<td><code>Scheduler</code> scheduler, <code>org.quartz.spi.TriggerFiredBundle</code> firedBundle, <code>Job</code> job</td>
</tr>
</tbody>
</table>

Create a `JobExecutionContext` with the given context data.

### Constructor parameters in `org.quartz.impl` with type arguments of type `Job`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobDetailImpl&lt;String name, Class&lt;? extends Job&gt; jobClass&gt;</code></td>
<td>Deprecated. use <code>JobBuilder</code></td>
</tr>
<tr>
<td><code>JobDetailImpl&lt;String name, String group, Class&lt;? extends Job&gt; jobClass&gt;</code></td>
<td>Deprecated. use <code>JobBuilder</code></td>
</tr>
<tr>
<td><code>JobDetailImpl&lt;String name, String group, Class&lt;? extends Job&gt; jobClass, boolean durability, boolean recover&gt;</code></td>
<td>Deprecated. use <code>JobBuilder</code></td>
</tr>
</tbody>
</table>

### Uses of `Job` in `org.quartz.jobs`

### Classes in `org.quartz.jobs` that implement `Job`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DirectoryScanJob</code></td>
<td>Inspects a directory and compares whether any files' &quot;last modified dates&quot; have changed since the last time it was inspected.</td>
</tr>
<tr>
<td><code>FileScanJob</code></td>
<td>Inspects a file and compares whether it's &quot;last modified date&quot; has changed since the last time it was inspected.</td>
</tr>
<tr>
<td><code>NativeJob</code></td>
<td>Built in job for executing native executables in a separate process.</td>
</tr>
<tr>
<td><code>NoOpJob</code></td>
<td>An implementation of Job, that does absolutely nothing - useful for system which only wish to use TriggerListeners and JobListeners,</td>
</tr>
</tbody>
</table>
rather than writing Jobs that perform work.

### Uses of Job in org.quartz.jobs.ee.ejb

**Classes in org.quartz.jobs.ee.ejb that implement Job**

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJBInvokerJob</td>
<td>A Job that invokes a method on an EJB.</td>
</tr>
</tbody>
</table>

### Uses of Job in org.quartz.jobs.ee.jms

**Classes in org.quartz.jobs.ee.jms that implement Job**

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendDestinationMessageJob</td>
<td>A Job that sends a javax.jms.Message to a javax.jms.Destination.</td>
</tr>
<tr>
<td>SendQueueMessageJob</td>
<td>A Job that sends a javax.jms.Message to a javax.jms.Queue</td>
</tr>
<tr>
<td>SendTopicMessageJob</td>
<td>A Job that sends a javax.jms.Message to a javax.jms.Topic</td>
</tr>
</tbody>
</table>

### Uses of Job in org.quartz.jobs.ee.jmx

**Classes in org.quartz.jobs.ee.jmx that implement Job**

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMXInvokerJob</td>
<td>Generic JMX invoker Job.</td>
</tr>
</tbody>
</table>

### Uses of Job in org.quartz.jobs.ee.mail
### Classes in `org.quartz.jobs.ee.mail` that implement `Job`

<table>
<thead>
<tr>
<th>Class</th>
<th>sendMailJob</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SendMailJob</td>
</tr>
</tbody>
</table>

A Job which sends an e-mail with the configured content to the configured recipient.

### Uses of `Job` in `org.quartz.locality`

### Methods in `org.quartz.locality` that return types with arguments of type `Job`

<table>
<thead>
<tr>
<th>Class</th>
<th>extends Job</th>
<th>DelegatingLocalityJobDetail.getJobClass()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Get the instance of Job that will be executed.</td>
</tr>
</tbody>
</table>

### Uses of `Job` in `org.quartz.simpl`

### Methods in `org.quartz.simpl` that return `Job`

<table>
<thead>
<tr>
<th>Job</th>
<th>PropertySettingJobFactory.newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler scheduler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job</td>
<td>SimpleJobFactory.newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler scheduler)</td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.JobBuilder

Packages that use JobBuilder

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and</td>
</tr>
<tr>
<td></td>
<td>other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
</tbody>
</table>

Uses of JobBuilder in org.quartz

Methods in org.quartz that return JobBuilder

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDetail.getJobBuilder()</td>
<td>Get a JobBuilder that is configured to produce a JobDetail identical to this one.</td>
</tr>
<tr>
<td>static JobBuilder.newJob()</td>
<td>Create a JobBuilder with which to define a JobDetail.</td>
</tr>
<tr>
<td>static JobBuilder.newJob(Class&lt;? extends Job&gt; jobClass)</td>
<td>Create a JobBuilder with which to define a JobDetail, and set the class name of the Job to be executed.</td>
</tr>
<tr>
<td>JobBuilder.ofType(Class&lt;? extends Job&gt; jobClass)</td>
<td>Set the class which will be instantiated and executed when a Trigger fires that is associated with this JobDetail.</td>
</tr>
<tr>
<td>JobBuilder.requestRecovery()</td>
<td>Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.</td>
</tr>
<tr>
<td>JobBuilder.requestRecovery(boolean shouldRecover)</td>
<td>Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>JobBuilder.storeDurably()</code></td>
<td>Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).</td>
</tr>
<tr>
<td><code>JobBuilder.storeDurably(boolean durability)</code></td>
<td>Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).</td>
</tr>
<tr>
<td><code>JobBuilder.usingJobData(JobDataMap newJobDataMap)</code></td>
<td>Set the JobDetail's JobDataMap, adding any values to it that were already set on this JobBuilder using any of the other 'usingJobData' methods.</td>
</tr>
<tr>
<td><code>JobBuilder.usingJobData(String key, Boolean value)</code></td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><code>JobBuilder.usingJobData(String key, Double value)</code></td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><code>JobBuilder.usingJobData(String key, Float value)</code></td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><code>JobBuilder.usingJobData(String key, Integer value)</code></td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><code>JobBuilder.usingJobData(String key, Long value)</code></td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><code>JobBuilder.usingJobData(String key, String value)</code></td>
<td>Add the given key-value pair to the JobDetail's JobDataMap.</td>
</tr>
<tr>
<td><code>JobBuilder.withDescription(String description)</code></td>
<td>Set the given (human-meaningful) description of the Job.</td>
</tr>
<tr>
<td><code>JobBuilder.withIdentity(JobKey key)</code></td>
<td>Use a JobKey to identify the JobDetail.</td>
</tr>
<tr>
<td><code>JobBuilder.withIdentity(String name)</code></td>
<td>Use a JobKey with the given name and default group to identify the JobDetail.</td>
</tr>
</tbody>
</table>
JobBuilder.withIdentity(String name, String group)
Use a JobKey with the given name and group to identify the JobDetail.

Uses of JobBuilder in org.quartz.impl

Methods in org.quartz.impl that return JobBuilder

<table>
<thead>
<tr>
<th>JobBuilder</th>
<th>JobDetailImpl.getJobBuilder()</th>
</tr>
</thead>
</table>

Uses of JobBuilder in org.quartz.locality

Methods in org.quartz.locality that return JobBuilder

<table>
<thead>
<tr>
<th>JobBuilder</th>
<th>DelegatingLocalityJobDetail.getJobBuilder()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get a JobBuilder that is configured to produce a JobDetail identical to this one.</td>
</tr>
</tbody>
</table>

Methods in org.quartz.locality with parameters of type JobBuilder

<table>
<thead>
<tr>
<th>static LocalityJobBuilder.locallJob(JobBuilder jobBuilder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory method to create builder to a wrapping LocalityAware JobDetailBuilder</td>
</tr>
</tbody>
</table>

Constructors in org.quartz.locality with parameters of type JobBuilder

<table>
<thead>
<tr>
<th>LocalityJobBuilder(JobBuilder jobBuilder)</th>
</tr>
</thead>
</table>

Overview  Package  Class  Tree  Deprecated  Index  Help
# Uses of Class

**org.quartz.JobDataMap**

<table>
<thead>
<tr>
<th>Packages that use <strong>JobDataMap</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.core.jmx</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.triggers</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.jobs.ee.jms</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.jobs.ee.mail</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.simpl</strong></td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Uses of **JobDataMap** in **org.quartz**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz</strong> that return <strong>JobDataMap</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobDataMap</strong></td>
</tr>
<tr>
<td>JobDetail. <strong>getJobDataMap</strong>()</td>
</tr>
<tr>
<td>Get the JobDataMap that is associated with the Job.</td>
</tr>
<tr>
<td><strong>JobDataMap</strong></td>
</tr>
<tr>
<td>Trigger. <strong>getJobDataMap</strong>()</td>
</tr>
</tbody>
</table>
Get the JobDataMap that is associated with the Trigger.

```java
JobDataMap jobExecutionContext.getMergedJobDataMap()
```

Get the convenience JobDataMap of this execution context.

### Methods in `org.quartz` with parameters of type `JobDataMap`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Scheduler.triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td></td>
<td>Trigger the identified JobDetail (execute it now).</td>
</tr>
<tr>
<td><code>JobBuilder USINGJobData(JobDataMap newJobDataMap)</code></td>
<td></td>
<td>Set the JobDetail's <code>JobDataMap</code>, adding any values to it that were already set on this JobBuilder using any of the other 'usingJobData' methods.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt; USINGJobData(JobDataMap newJobDataMap)</code></td>
<td></td>
<td>Set the Trigger's <code>JobDataMap</code>, adding any values to it that were already set on this TriggerBuilder using any of the other 'usingJobData' methods.</td>
</tr>
</tbody>
</table>

### Uses of `JobDataMap` in `org.quartz.core`

### Methods in `org.quartz.core` with parameters of type `JobDataMap`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzScheduler.triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td></td>
<td>Trigger the identified Job (execute it now) - with a non-volatile trigger.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Uses of `JobDataMap` in `org.quartz.core.jmx`

### Methods in `org.quartz.core.jmx` that return `JobDataMap`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobDataMapSupport.newJobDataMap(Map&lt;String, Object&gt; map)</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Methods in `org.quartz.core.jmx` with parameters of type `JobDataMap`

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>TabularData</code></td>
<td><code>JobDataMapSupport.toTabularData(JobDataMap jobDataMap)</code></td>
</tr>
<tr>
<td>static <code>JobDataMap</code></td>
<td><code>JobDataMapSupport.newJobDataMap(TabularData tabularData)</code></td>
</tr>
</tbody>
</table>

Uses of `JobDataMap` in `org.quartz.impl`

Methods in `org.quartz.impl` that return `JobDataMap`

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobDataMap</code></td>
<td><code>JobDetailImpl.getJobDataMap()</code></td>
</tr>
<tr>
<td><code>JobDataMap</code></td>
<td><code>JobExecutionContextImpl.getMergedJobDataMap()</code></td>
</tr>
</tbody>
</table>

Methods in `org.quartz.impl` with parameters of type `JobDataMap`

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>JobDetailImpl.setJobDataMap(JobDataMap jobDataMap)</code></td>
<td>Set the JobDataMap to be associated with the Job.</td>
</tr>
<tr>
<td>void <code>RemoteScheduler.triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void <code>StdScheduler.triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void <code>RemoteMBeanScheduler.triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
</tbody>
</table>

Uses of `JobDataMap` in `org.quartz.impl.jdbcjobstore`
Methods in `org.quartz.impl.jdbcjobstore` that return `JobDataMap`:

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>selectTriggerJobDataMap</code></td>
<td><code>JobDataMap StdJDBCDelegate.selectTriggerJobDataMap(Connection conn, String triggerName, String groupName)</code></td>
<td>Select a trigger's JobDataMap.</td>
</tr>
<tr>
<td><code>selectTriggerJobDataMap</code></td>
<td><code>JobDataMap DriverDelegate.selectTriggerJobDataMap(Connection conn, String triggerName, String groupName)</code></td>
<td>Select a trigger's JobDataMap.</td>
</tr>
</tbody>
</table>

Methods in `org.quartz.impl.jdbcjobstore` with parameters of type `JobDataMap`:

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>serializeJobData</code></td>
<td><code>JobDataMap protected StdJDBCDelegate.serializeJobData(JobDataMap data)</code></td>
<td>Remove the transient data from and then create a serialized java.util.ByteArrayOutputStream version of a JobDataMap.</td>
</tr>
</tbody>
</table>

Uses of `JobDataMap` in `org.quartz.impl.triggers`:

Methods in `org.quartz.impl.triggers` that return `JobDataMap`:

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getJobDataMap</code></td>
<td><code>JobDataMap AbstractTrigger.getJobDataMap()</code></td>
<td>Get the JobDataMap that is associated with the Trigger.</td>
</tr>
</tbody>
</table>

Methods in `org.quartz.impl.triggers` with parameters of type `JobDataMap`:

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setJobDataMap</code></td>
<td><code>void AbstractTrigger.setJobDataMap(JobDataMap jobDataMap)</code></td>
<td>Set the JobDataMap to be associated with the Trigger.</td>
</tr>
</tbody>
</table>

Uses of `JobDataMap` in `org.quartz.jobs.ee.jms`
### Methods in org.quartz.jobs.ee.jms with parameters of type JobDataMap

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static InitialContext JmsHelper.getInitialContext</td>
<td>Uses of JobDataMap in org.quartz.jobs.ee.mail</td>
</tr>
<tr>
<td>static boolean JmsHelper.isDestinationSecure</td>
<td></td>
</tr>
<tr>
<td>static boolean JmsHelper.useTransaction</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of JobDataMap in org.quartz.jobs.ee.mail

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected String SendMailJob.getOptionalParm</td>
<td>Get the JobDataMap that is associated with the Job.</td>
</tr>
<tr>
<td>protected String SendMailJob.getRequiredParm</td>
<td>Get the JobDataMap that is associated with the Trigger.</td>
</tr>
<tr>
<td>protected SendMailJob.MailInfo SendMailJob.populateMailInfo</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of JobDataMap in org.quartz.locality

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDataMap DelegatingLocalityJobDetail.getJobDataMap()</td>
<td></td>
</tr>
<tr>
<td>JobDataMap DelegatingLocalityTrigger.getJobDataMap()</td>
<td></td>
</tr>
</tbody>
</table>
Methods in `org.quartz.locality` with parameters of type `JobDataMap`

```java
void DelegatingLocalityTrigger.setJobDataMap(JobDataMap jobDataMap)
```

Uses of `JobDataMap` in `org.quartz.simpl`

Methods in `org.quartz.simpl` with parameters of type `JobDataMap`

```java
protected void PropertySettingJobFactory.setBeanProps(Object obj, JobDataMap data)
```

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.JobDetail

Packages that use JobDetail

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.core.jmx</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore.oracle</td>
<td></td>
</tr>
<tr>
<td>org.quartz.listeners</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td></td>
</tr>
<tr>
<td>org.quartz.xml</td>
<td></td>
</tr>
</tbody>
</table>

Uses of JobDetail in org.quartz

Methods in org.quartz that return JobDetail

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDetail</td>
<td>JobBuilder.\texttt{build}()</td>
</tr>
<tr>
<td></td>
<td>Produce the JobDetail instance defined by this JobBuilder.</td>
</tr>
<tr>
<td>JobDetail</td>
<td>JobExecutionContext.\texttt{getJobDetail}()</td>
</tr>
</tbody>
</table>
Get the JobDetail associated with the Job.

```
JobDetail scheduler.getJobDetail(JobKey jobKey)
```

Get the JobDetail for the Job instance with the given key.

## Methods in `org.quartz` with parameters of type `JobDetail`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>Scheduler.addJob(JobDetail jobDetail, boolean replace)</code> Add the given Job to the Scheduler - with no associated Trigger.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;T&gt;</code></td>
<td><code>TriggerBuilder.forJob(JobDetail jobDetail)</code> Set the identity of the Job which should be fired by the produced Trigger, by extracting the JobKey from the given job.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>SchedulerListener.jobAdded(JobDetail jobDetail)</code> Called by the Scheduler when a JobDetail has been added.</td>
</tr>
<tr>
<td><code>Date</code></td>
<td><code>Scheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</code> Add the given JobDetail to the Scheduler, and associate the given Trigger with it.</td>
</tr>
</tbody>
</table>

## Method parameters in `org.quartz` with type arguments of type `JobDetail`

```
void scheduler.scheduleJobs(Map<JobDetail,List<Trigger>> triggersAndJobs, boolean replace)
```

Schedule all of the given jobs with the related set of triggers.

## Constructors in `org.quartz` with parameters of type `JobDetail`

```
ObjectAlreadyExistsException(JobDetail offendingJob)
```

Create a `ObjectAlreadyExistsException` and auto-generate a message using the name/group from the given JobDetail.

## Uses of `JobDetail` in `org.quartz.core`
### Methods in `org.quartz.core` that return `JobDetail`

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDetail</td>
<td>QuartzScheduler.getJobDetail(JobKey jobKey)</td>
<td>Get the JobDetail for the Job instance with the given name and group.</td>
</tr>
<tr>
<td>JobDetail</td>
<td>RemotableQuartzScheduler.getJobDetail(JobKey jobKey)</td>
<td>Get the JobDetail for the Job instance with the given name and group.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.core` with parameters of type `JobDetail`

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>QuartzScheduler.addJob(JobDetail jobDetail, boolean replace)</td>
<td>Add the given Job to the Scheduler - with no associated Trigger.</td>
</tr>
<tr>
<td>void</td>
<td>RemotableQuartzScheduler.addJob(JobDetail jobDetail, boolean replace)</td>
<td>Add the given Job to the Scheduler - with no associated Trigger.</td>
</tr>
<tr>
<td>boolean</td>
<td>JobRunShell.completeTriggerRetryLoop(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)</td>
<td>Complete the trigger retry loop.</td>
</tr>
<tr>
<td>void</td>
<td>QuartzSchedulerMBeanImpl.jobAdded(JobDetail jobDetail)</td>
<td>Add the given Job to the Scheduler - with no associated Trigger.</td>
</tr>
<tr>
<td>void</td>
<td>SampledStatisticsImpl.jobAdded(JobDetail jobDetail)</td>
<td>Add the given Job to the Scheduler - with no associated Trigger.</td>
</tr>
<tr>
<td>void</td>
<td>QuartzScheduler.notifySchedulerListenersJobAdded(JobDetail jobDetail)</td>
<td>Notify the scheduler listeners job added.</td>
</tr>
<tr>
<td>Date</td>
<td>QuartzScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</td>
<td>Schedule the Job.</td>
</tr>
<tr>
<td>Date</td>
<td>RemotableQuartzScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</td>
<td>Schedule the Job.</td>
</tr>
</tbody>
</table>
JobRunShell. vetoedJobRetryLoop (org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)

Method parameters in org.quartz.core with type arguments of type JobDetail

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void QuartzScheduler.scheduleJobs</td>
<td>(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; triggersAndJobs, boolean replace)</td>
</tr>
<tr>
<td>void RemotableQuartzScheduler.scheduleJobs</td>
<td>(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; triggersAndJobs, boolean replace)</td>
</tr>
</tbody>
</table>

Uses of JobDetail in org.quartz.core.jmx

Methods in org.quartz.core.jmx that return JobDetail

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>static JobDetail JobDetailSupport.newJobDetail</td>
<td>(CompositeData cData)</td>
</tr>
<tr>
<td>static JobDetail JobDetailSupport.newJobDetail</td>
<td>(Map&lt;String, Object&gt; attrMap)</td>
</tr>
</tbody>
</table>

Methods in org.quartz.core.jmx with parameters of type JobDetail

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>static CompositeData JobDetailSupport.toCompositeData</td>
<td>(JobDetail jobDetail)</td>
</tr>
<tr>
<td>static TabularData JobDetailSupport.toTabularData</td>
<td>(JobDetail[] jobDetails)</td>
</tr>
</tbody>
</table>

Uses of JobDetail in org.quartz.impl

Classes in org.quartz.impl that implement JobDetail
### Class: JobDetailImpl

Conveys the detail properties of a given Job instance.

### Methods in org.quartz.impl that return JobDetail

<table>
<thead>
<tr>
<th>JobDetail</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JobExecutionContextImpl.getJobDetail()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td></td>
<td>RemoteScheduler.getJobDetail(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td></td>
<td>StdScheduler.getJobDetail(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td></td>
<td>RemoteMBeanScheduler.getJobDetail(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
</tbody>
</table>

### Methods in org.quartz.impl with parameters of type JobDetail

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>RemoteScheduler.addJob(JobDetail jobDetail, boolean replace)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td>StdScheduler.addJob(JobDetail jobDetail, boolean replace)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td>RemoteMBeanScheduler.addJob(JobDetail jobDetail, boolean replace)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>Date</td>
<td>StdScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteMBeanScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
</tbody>
</table>
### Method parameters in `org.quartz.impl` with type arguments of type `JobDetail`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RemoteScheduler.scheduleJobs</code></td>
<td><code>Map&lt;JobDetail,List&lt;Trigger&gt;&gt;</code> triggers and boolean replace</td>
</tr>
<tr>
<td><code>StdScheduler.scheduleJobs</code></td>
<td><code>Map&lt;JobDetail,List&lt;Trigger&gt;&gt;</code> triggers and boolean replace</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler.scheduleJobs</code></td>
<td><code>Map&lt;JobDetail,List&lt;Trigger&gt;&gt;</code> triggers and boolean replace</td>
</tr>
</tbody>
</table>

### Uses of `JobDetail` in `org.quartz.impl.jdbcjobstore`

### Methods in `org.quartz.impl.jdbcjobstore` that return `JobDetail`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobStoreSupport.retrieveJob</code></td>
<td><code>Connection conn, JobKey key</code></td>
</tr>
<tr>
<td><code>JobStoreSupport.retrieveJob</code></td>
<td><code>JobKey jobKey</code></td>
</tr>
<tr>
<td></td>
<td>Retrieve the JobDetail for the given Job.</td>
</tr>
<tr>
<td><code>StdJDBCDelegate.selectJobDetail</code></td>
<td><code>Connection conn, JobKey jobKey, org.quartz.spi.ClassLoadHelper loadHelper</code></td>
</tr>
<tr>
<td></td>
<td>Select the JobDetail object for a given job name / group name.</td>
</tr>
<tr>
<td><code>DriverDelegate.selectJobDetail</code></td>
<td><code>Connection conn, JobKey jobKey, org.quartz.spi.ClassLoadHelper loadHelper</code></td>
</tr>
<tr>
<td></td>
<td>Select the JobDetail object for a given job name / group name.</td>
</tr>
<tr>
<td><code>StdJDBCDelegate.selectJobForTrigger</code></td>
<td><code>Connection conn, org.quartz.spi.ClassLoadHelper loadHelper, TriggerKey triggerKey</code></td>
</tr>
<tr>
<td></td>
<td>Select the job to which the trigger is associated.</td>
</tr>
<tr>
<td><code>DriverDelegate.selectJobForTrigger</code></td>
<td><code>Connection conn, org.quartz.spi.ClassLoadHelper loadHelper, TriggerKey triggerKey</code></td>
</tr>
<tr>
<td></td>
<td>Select the job to which the trigger is associated.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl.jdbcjobstore` with parameters of type `JobDetail`
<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleTriggerPersistenceDelegate</td>
<td><code>insertExtendedTriggerProperties</code></td>
<td>Insert extended trigger properties</td>
</tr>
<tr>
<td>SimplePropertiesTriggerPersistenceDelegateSupport</td>
<td><code>insertExtendedTriggerProperties</code></td>
<td>Insert extended trigger properties</td>
</tr>
<tr>
<td>CronTriggerPersistenceDelegate</td>
<td><code>insertExtendedTriggerProperties</code></td>
<td>Insert extended trigger properties</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate</td>
<td><code>insertExtendedTriggerProperties</code></td>
<td>Insert extended trigger properties</td>
</tr>
<tr>
<td>StdJDBCDelegate</td>
<td><code>insertFiredTrigger</code></td>
<td>Insert a fired trigger</td>
</tr>
<tr>
<td>DriverDelegate</td>
<td><code>insertFiredTrigger</code></td>
<td>Insert a fired trigger</td>
</tr>
<tr>
<td>StdJDBCDelegate</td>
<td><code>insertJobDetail</code></td>
<td>Insert the job detail record</td>
</tr>
<tr>
<td>DriverDelegate</td>
<td><code>insertJobDetail</code></td>
<td>Insert the job detail record</td>
</tr>
<tr>
<td>PointbaseDelegate</td>
<td><code>insertJobDetail</code></td>
<td>Insert the job detail record</td>
</tr>
<tr>
<td>StdJDBCDelegate</td>
<td><code>insertTrigger</code></td>
<td>Insert the base trigger data</td>
</tr>
<tr>
<td>DriverDelegate</td>
<td><code>insertTrigger</code></td>
<td>Insert the base trigger data</td>
</tr>
<tr>
<td>PointbaseDelegate</td>
<td><code>insertTrigger</code></td>
<td>Insert the base trigger data</td>
</tr>
<tr>
<td>JobStoreSupport</td>
<td><code>storeJob</code></td>
<td>Insert or update a job</td>
</tr>
<tr>
<td>JobStoreSupport</td>
<td><code>storeJob</code></td>
<td>Store the given JobDetail</td>
</tr>
</tbody>
</table>

```java
protected void JobStoreSupport.storeJob(Connection conn, JobDetail newJob, boolean replaceExisting) {
    // Store the given JobDetail.
}
```
<table>
<thead>
<tr>
<th>void</th>
<th>JobStoreSupport.<strong>storeJobAndTrigger</strong>(JobDetail newJob, org.quartz.spi.OperableTrigger newTrigger)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Store the given JobDetail and Trigger.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected void</th>
<th>JobStoreSupport.<strong>storeTrigger</strong>(Connection conn, org.quartz.spi.OperableTrigger newTrigger, JobDetail job, boolean replaceExisting, String state, boolean forceState)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Insert or update a trigger.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected void</th>
<th>JobStoreSupport.<strong>triggeredJobComplete</strong>(Connection conn, org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction triggerInstCode)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inform the JobStore that the scheduler has completed the firing of the associated Job, and that the JobDataMap in the given JobDetail should be updated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>SimpleTriggerPersistenceDelegate.<strong>updateExtendedTriggerProperties</strong>(org.quartz.spi.OperableTrigger trigger, String state, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>SimplePropertiesTriggerPersistenceDelegateSupport.<strong>updateExtendedTriggerProperties</strong>(org.quartz.spi.OperableTrigger trigger, String state, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>CronTriggerPersistenceDelegate.<strong>updateExtendedTriggerProperties</strong>(org.quartz.spi.OperableTrigger trigger, String state, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>TriggerPersistenceDelegate.<strong>updateExtendedTriggerProperties</strong>(org.quartz.spi.OperableTrigger trigger, String state, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>StdJDBCDelegate.<strong>updateFiredTrigger</strong>(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Update a fired trigger.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>DriverDelegate.<strong>updateFiredTrigger</strong>(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Update a fired trigger record.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>StdJDBCDelegate.<strong>updateJobData</strong>(Connection conn, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Update the job data map for the given job.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>DriverDelegate.<strong>updateJobData</strong>(Connection conn, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Update the job data map for the given job.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>PointbaseDelegate.<strong>updateJobData</strong>(Connection conn, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Update the job data map for the given job.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>StdJDBCDelegate.<strong>updateJobDetail</strong>(Connection conn, JobDetail job)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Update the job data map for the given job.</td>
</tr>
</tbody>
</table>
Update the job detail record.

```java
int DriverDelegate.updateJobDetail(Connection conn, JobDetail job)
```

Update the job detail record.

```java
int PointbaseDelegate.updateJobDetail(Connection conn, JobDetail job)
```

Update the job detail record.

```java
int StdJDBCDelegate.updateTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
```

Update the base trigger data.

```java
int DriverDelegate.updateTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
```

Update the base trigger data.

```java
int PointbaseDelegate.updateTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
```

Method parameters in `org.quartz.impl.jdbcjobstore` with type arguments of `void`.

```java
JobStoreSupport.storeJobsAndTriggers(Map<JobDetail, List<Trigger>> jobsAndTriggers, boolean replace)
```

Uses of `JobDetail` in `org.quartz.impl.jdbcjobstore.oracle`.

Methods in `org.quartz.impl.jdbcjobstore.oracle` with parameters of type `JobDetail`.

```java
int OracleDelegate.insertJobDetail(Connection conn, JobDetail job)
```

```java
int OracleDelegate.insertTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
```

```java
int OracleDelegate.updateJobData(Connection conn, JobDetail job)
```
int OracleDelegate.updateJobDetail(Connection conn, JobDetail job)

int OracleDelegate.updateTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)

**Uses of JobDetail in org.quartz.listeners**

**Methods in** org.quartz.listeners **with parameters of type JobDetail**

void SchedulerListenerSupport.jobAdded(JobDetail jobDetail)

void BroadcastSchedulerListener.jobAdded(JobDetail jobDetail)

**Uses of JobDetail in org.quartz.locality**

**Subinterfaces of JobDetail in org.quartz.locality**

interface LocalityJobDetail

A specialized JobDetail that contains Quartz Where information

**Classes in** org.quartz.locality **that implement JobDetail**

class DelegatingLocalityJobDetail

Wrapping a JobDetail instance while adding the LocalityAware contract All JobDetail method calls will be delegated to the wrapped JobDetail instance

**Methods in** org.quartz.locality **with parameters of type JobDetail**
**static** LocalityJobBuilder

LocalityJobBuilder.**localJob**(JobDetail jobDetail)

Factory method to create builder to a wrapping LocalityAware JobDetail

<table>
<thead>
<tr>
<th>Constructors in org.quartz.locality with parameters of type JobDetail</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelegatingLocalityJobDetail(JobDetail jobDetail, NodeSpec nodeSpec)</td>
</tr>
</tbody>
</table>

Constructs a LocalityAware JobDetail, wrapping an existing JobDetail instance, with additional NodeSpec

<table>
<thead>
<tr>
<th>Uses of JobDetail in org.quartz.simpl</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.simpl with parameters of type JobDetail</th>
</tr>
</thead>
<tbody>
<tr>
<td>void RAMJobStore.<strong>storeJob</strong>(JobDetail newJob, boolean replaceExisting)</td>
</tr>
</tbody>
</table>

Store the given Job.

<table>
<thead>
<tr>
<th>Methods in org.quartz.simpl that return JobDetail</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDetail RAMJobStore.<strong>retrieveJob</strong>(JobKey jobKey)</td>
</tr>
</tbody>
</table>

Retrieve the JobDetail for the given Job.

<table>
<thead>
<tr>
<th>Methods in org.quartz.simpl with parameters of type JobDetail</th>
</tr>
</thead>
<tbody>
<tr>
<td>void RAMJobStore.<strong>storeJobAndTrigger</strong>(JobDetail newJob, org.quartz.spi.OperableTrigger newTrigger)</td>
</tr>
</tbody>
</table>

Store the given JobDetail and Trigger.

| void RAMJobStore.**triggeredJobComplete**(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction triggerInstCode) |

Inform the JobStore that the scheduler has completed the firing of the Trigger (and the execution its associated Job), and that the JobDataMap in the JobDetail should be updated if the Job is stateful.

<table>
<thead>
<tr>
<th>Method parameters in org.quartz.simpl with type arguments of type JobDetail</th>
</tr>
</thead>
<tbody>
<tr>
<td>void RAMJobStore.<strong>storeJobsAndTriggers</strong>(Map&lt;JobDetail, List&lt;Trigger&gt;&gt;)</td>
</tr>
</tbody>
</table>
boolean replace)

Uses of JobDetail in org.quartz.xml

Fields in org.quartz.xml with type parameters of type JobDetail

| protected List&lt;JobDetail&gt; | XMLSchedulingDataProcessor.<a>loadedJobs</a> |

Methods in org.quartz.xml that return types with arguments of type JobDetail

<table>
<thead>
<tr>
<th>protected List&lt;JobDetail&gt;</th>
<th>XMLSchedulingDataProcessor.&lt;a&gt;getLoadedJobs&lt;/a&gt;()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a List of jobs loaded from the xml file.</td>
</tr>
</tbody>
</table>

Methods in org.quartz.xml with parameters of type JobDetail

| protected void              | XMLSchedulingDataProcessor.<a>addJobToSchedule</a>(JobDetail job) |

Copyright 2001-2011, Terracotta, Inc.
### Packages that use `JobExecutionContext`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz</code></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><code>org.quartz.core</code></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><code>org.quartz.core.jmx</code></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><code>org.quartz.impl</code></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs.ee.ejb</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs.ee.jms</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs.ee.jmx</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.jobs.ee.mail</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.listeners</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.locality</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.plugins.history</code></td>
<td></td>
</tr>
</tbody>
</table>

### Uses of `JobExecutionContext` in `org.quartz`

### Methods in `org.quartz` that return types with arguments of type `JobExecutionContext`

- `Scheduler.getCurrentlyExecutingJobs()` returns a list of `JobExecutionContext` objects.
that represent all currently executing Jobs in this Scheduler instance.

### Methods in org.quartz with parameters of type JobExecutionContext

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Job.execute(JobExecutionContext context)</td>
<td></td>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
<tr>
<td>void JobListener.jobExecutionVetoed(JobExecutionContext context)</td>
<td></td>
<td>Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.</td>
</tr>
<tr>
<td>void JobListener.jobToBeExecuted(JobExecutionContext context)</td>
<td></td>
<td>Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).</td>
</tr>
<tr>
<td>void JobListener.jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)</td>
<td></td>
<td>Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.</td>
</tr>
<tr>
<td>void TriggerListener.triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</td>
<td></td>
<td>Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered(xx) method has been called.</td>
</tr>
<tr>
<td>void TriggerListener.triggerFired(Trigger trigger, JobExecutionContext context)</td>
<td></td>
<td>Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.</td>
</tr>
<tr>
<td>boolean TriggerListener.vetoJobExecution(Trigger trigger, JobExecutionContext context)</td>
<td></td>
<td>Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.</td>
</tr>
</tbody>
</table>

### Uses of JobExecutionContext in org.quartz.core
### Methods in `org.quartz.core` that return types with arguments of type `JobExecutionContext`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzScheduler.getCurrentlyExecutingJobs()</code></td>
<td>Return a list of <code>JobExecutionContext</code> objects that represent all currently executing Jobs in this Scheduler instance.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getCurrentlyExecutingJobs()</code></td>
<td></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.core` with parameters of type `JobExecutionContext`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzSchedulerMBeanImpl.jobExecutionVetoed(JobExecutionContext)</code></td>
<td></td>
</tr>
<tr>
<td><code>SampledStatisticsImpl.jobExecutionVetoed(JobExecutionContext)</code></td>
<td></td>
</tr>
<tr>
<td><code>QuartzSchedulerMBeanImpl.jobToBeExecuted(JobExecutionContext)</code></td>
<td></td>
</tr>
<tr>
<td><code>SampledStatisticsImpl.jobToBeExecuted(JobExecutionContext)</code></td>
<td></td>
</tr>
<tr>
<td><code>QuartzSchedulerMBeanImpl.jobWasExecuted(JobExecutionContext, JobExecutionException)</code></td>
<td></td>
</tr>
<tr>
<td><code>SampledStatisticsImpl.jobWasExecuted(JobExecutionContext, JobExecutionException)</code></td>
<td></td>
</tr>
<tr>
<td><code>QuartzScheduler.notifyJobListenersToBeExecuted(JobExecutionContext)</code></td>
<td></td>
</tr>
<tr>
<td><code>QuartzScheduler.notifyJobListenersWasExecuted(JobExecutionContext, JobExecutionException)</code></td>
<td></td>
</tr>
<tr>
<td><code>QuartzScheduler.notifyJobListenersWasVetoed(JobExecutionContext)</code></td>
<td></td>
</tr>
<tr>
<td><code>QuartzScheduler.notifyTriggerListenersComplete(Trigger.CompletedExecutionInstruction)</code></td>
<td></td>
</tr>
</tbody>
</table>
boolean QuartzScheduler.notifyTriggerListenersFired(JobExecutionContext)

**Uses of JobExecutionContext in org.quartz.core.jmx**

**Methods in org.quartz.core.jmx with parameters of type JobExecutionContext**

<table>
<thead>
<tr>
<th>static CompositeData</th>
<th>JobExecutionContextSupport.toCompositeData(JobExecutionContext)</th>
</tr>
</thead>
</table>

**Method parameters in org.quartz.core.jmx with type arguments of type JobExecutionContext**

<table>
<thead>
<tr>
<th>static TabularData</th>
<th>JobExecutionContextSupport.toTabularData(List&lt;JobExecutionContext&gt;)</th>
</tr>
</thead>
</table>

**Uses of JobExecutionContext in org.quartz.impl**

**Classes in org.quartz.impl that implement JobExecutionContext**

<table>
<thead>
<tr>
<th>class</th>
<th>JobExecutionContextImpl</th>
</tr>
</thead>
</table>

**Methods in org.quartz.impl that return types with arguments of type JobExecutionContext**

<table>
<thead>
<tr>
<th>List&lt;JobExecutionContext&gt;</th>
<th>StdScheduler.getCurrentlyExecutingJobs()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>List&lt;JobExecutionContext&gt;</td>
<td>RemoteMBeanScheduler.getCurrentlyExecutingJobs()</td>
</tr>
<tr>
<td></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
</tbody>
</table>
Uses of `JobExecutionContext` in `org.quartz.impl.triggers`

Methods in `org.quartz.impl.triggers` with parameters of type `JobExecutionContext`:
- `AbstractTrigger.executionComplete(JobExecutionContext context, JobExecutionException result)`
  - This method should not be used by the Quartz client.

Uses of `JobExecutionContext` in `org.quartz.jobs`

Methods in `org.quartz.jobs` with parameters of type `JobExecutionContext`:
- `void NoOpJob.execute(JobExecutionContext context)`
  - Do nothing.
- `void DirectoryScanJob.execute(JobExecutionContext context)`
- `void NativeJob.execute(JobExecutionContext context)`
- `void FileScanJob.execute(JobExecutionContext context)`

Uses of `JobExecutionContext` in `org.quartz.jobs.ee.ejb`

Methods in `org.quartz.jobs.ee.ejb` with parameters of type `JobExecutionContext`:
- `void EJBInvokerJob.execute(JobExecutionContext context)`
### Uses of JobExecutionContext in org.quartz.jobs.ee.jms

Methods in org.quartz.jobs.ee.jms with parameters of type JobExecutionContext

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SendQueueMessageJob.execute(JobExecutionContext context)</td>
<td></td>
</tr>
<tr>
<td>void SendDestinationMessageJob.execute(JobExecutionContext jobCtx)</td>
<td></td>
</tr>
<tr>
<td>void SendTopicMessageJob.execute(JobExecutionContext jobCtx)</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of JobExecutionContext in org.quartz.jobs.ee.jmx

Methods in org.quartz.jobs.ee.jmx with parameters of type JobExecutionContext

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void JMXInvokerJob.execute(JobExecutionContext context)</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of JobExecutionContext in org.quartz.jobs.ee.mail

Methods in org.quartz.jobs.ee.mail with parameters of type JobExecutionContext

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SendMailJob.execute(JobExecutionContext context)</td>
<td></td>
</tr>
</tbody>
</table>
Uses of **JobExecutionContext** in **org.quartz.listeners**

### Methods in **org.quartz.listeners** with parameters of type **JobExecutionContext**

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>JobListenerSupport.jobExecutionVetoed</td>
<td>(JobExecutionContext context)</td>
</tr>
<tr>
<td>void</td>
<td>BroadcastJobListener.jobExecutionVetoed</td>
<td>(JobExecutionContext context)</td>
</tr>
<tr>
<td>void</td>
<td>JobListenerSupport.jobToBeExecuted</td>
<td>(JobExecutionContext context)</td>
</tr>
<tr>
<td>void</td>
<td>BroadcastJobListener.jobToBeExecuted</td>
<td>(JobExecutionContext context)</td>
</tr>
<tr>
<td>void</td>
<td>JobChainingJobListener.jobWasExecuted</td>
<td>(JobExecutionContext context, JobExecutionException jobException)</td>
</tr>
<tr>
<td>void</td>
<td>JobListenerSupport.jobWasExecuted</td>
<td>(JobExecutionContext context, JobExecutionException jobException)</td>
</tr>
<tr>
<td>void</td>
<td>BroadcastJobListener.jobWasExecuted</td>
<td>(JobExecutionContext context, JobExecutionException jobException)</td>
</tr>
<tr>
<td>void</td>
<td>TriggerListenerSupport.triggerComplete</td>
<td>(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</td>
</tr>
<tr>
<td>void</td>
<td>BroadcastTriggerListener.triggerComplete</td>
<td>(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</td>
</tr>
<tr>
<td>void</td>
<td>TriggerListenerSupport.triggerFired</td>
<td>(Trigger trigger, JobExecutionContext context)</td>
</tr>
<tr>
<td>void</td>
<td>BroadcastTriggerListener.triggerFired</td>
<td>(Trigger trigger, JobExecutionContext context)</td>
</tr>
<tr>
<td>boolean</td>
<td>TriggerListenerSupport.vetoJobExecution</td>
<td>(Trigger trigger, JobExecutionContext context)</td>
</tr>
</tbody>
</table>
### Uses of `JobExecutionContext` in `org.quartz.locality`

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DelegatingLocalityTrigger.executionComplete</code></td>
<td><code>JobExecutionContext</code> context, <code>JobExecutionException</code> result</td>
</tr>
</tbody>
</table>

### Uses of `JobExecutionContext` in `org.quartz.plugins.history`

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LoggingJobHistoryPlugin.jobExecutionVetoed</code></td>
<td><code>JobExecutionContext</code> context</td>
</tr>
<tr>
<td><code>LoggingJobHistoryPlugin.jobToBeExecuted</code></td>
<td><code>JobExecutionContext</code> context</td>
</tr>
<tr>
<td><code>LoggingJobHistoryPlugin.jobWasExecuted</code></td>
<td><code>JobExecutionContext</code> context, <code>JobExecutionException</code> jobException</td>
</tr>
<tr>
<td><code>LoggingTriggerHistoryPlugin.triggerComplete</code></td>
<td><code>Trigger</code> trigger, <code>JobExecutionContext</code> context, <code>Trigger.CompletedExecutionInstruction</code> triggerInstructionCode</td>
</tr>
<tr>
<td><code>LoggingTriggerHistoryPlugin.triggerFired</code></td>
<td><code>Trigger</code> trigger, <code>JobExecutionContext</code> context</td>
</tr>
</tbody>
</table>

boolean `BroadcastTriggerListener.vetoJobExecution(Trigger trigger, JobExecutionContext context)`
LoggingTriggerHistoryPlugin. **vetoJobExecution** (Trigger trigger, JobExecutionContext context)
# Uses of Class

**org.quartz.JobExecutionException**

<table>
<thead>
<tr>
<th>Packages that use</th>
<th>JobExecutionException</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td>org.quartz.jobs</td>
<td></td>
</tr>
<tr>
<td>org.quartz.jobs.ee.ejb</td>
<td></td>
</tr>
<tr>
<td>org.quartz.jobs.ee.jms</td>
<td></td>
</tr>
<tr>
<td>org.quartz.jobs.ee.jmx</td>
<td></td>
</tr>
<tr>
<td>org.quartz.jobs.ee.mail</td>
<td></td>
</tr>
<tr>
<td>org.quartz.listeners</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.history</td>
<td></td>
</tr>
</tbody>
</table>

# Uses of JobExecutionException in org.quartz

<table>
<thead>
<tr>
<th>Methods in org.quartz with parameters of type JobExecutionException</th>
</tr>
</thead>
<tbody>
<tr>
<td>void JobListener.jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)</td>
</tr>
<tr>
<td>Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered() method has been called.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz that throw JobExecutionException</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
</tbody>
</table>
Job.execute(JobExecutionContext context)
Called by the Scheduler when a Trigger fires that is associated with the Job.

**Uses of JobExecutionException in org.quartz.core**

<table>
<thead>
<tr>
<th>Methods in org.quartz.core with parameters of type JobExecutionException</th>
</tr>
</thead>
<tbody>
<tr>
<td>void QuartzSchedulerMBeanImpl.jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)</td>
</tr>
<tr>
<td>void SampledStatisticsImpl.jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)</td>
</tr>
<tr>
<td>void QuartzScheduler.notifyJobListenersWasExecuted(JobExecutionException je)</td>
</tr>
</tbody>
</table>

**Uses of JobExecutionException in org.quartz.impl.triggers**

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl.triggers with parameters of type JobExecutionException</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Trigger.CompletedExecutionInstruction.jobWasExecuted(AbstractTrigger, JobExecutionException result)</td>
</tr>
<tr>
<td>This method should not be used by the</td>
</tr>
</tbody>
</table>

**Uses of JobExecutionException in org.quartz.jobs**

<table>
<thead>
<tr>
<th>Methods in org.quartz.jobs that throw JobExecutionException</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NoOpJob</th>
<th><code>execute(JobExecutionContext context)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Do nothing.</td>
</tr>
<tr>
<td>void</td>
<td>DirectoryScanJob.<code>execute(JobExecutionContext context)</code></td>
</tr>
<tr>
<td>void</td>
<td>NativeJob.<code>execute(JobExecutionContext context)</code></td>
</tr>
<tr>
<td>void</td>
<td>FileScanJob.<code>execute(JobExecutionContext context)</code></td>
</tr>
</tbody>
</table>

**Uses of JobExecutionException in org.quartz.jobs.ee.ejb**

| void | EJBInvokerJob.`execute(JobExecutionContext context)` |

**Methods in org.quartz.jobs.ee.ejb that throw JobExecutionException**

| void | SendQueueMessageJob.`execute(JobExecutionContext context)` |
|      | SendDestinationMessageJob.`execute(JobExecutionContext jobCtx)` |
| void | SendTopicMessageJob.`execute(JobExecutionContext jobCtx)` |

**Uses of JobExecutionException in org.quartz.jobs.ee.jms**

**Methods in org.quartz.jobs.ee.jms that throw JobExecutionException**

| void | SendQueueMessageJob.`execute(JobExecutionContext context)` |

**Uses of JobExecutionException in**
**org.quartz.jobs.ee.jmx**

Methods in **org.quartz.jobs.ee.jmx** that throw **JobExecutionException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void JMXInvokerJob.execute(JobExecutionContext context)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Uses of **JobExecutionException** in** **org.quartz.jobs.ee.mail**

Methods in **org.quartz.jobs.ee.mail** that throw **JobExecutionException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SendMailJob.execute(JobExecutionContext context)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Uses of **JobExecutionException** in** **org.quartz.listeners**

Methods in **org.quartz.listeners** with parameters of type **JobExecutionException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void JobChainingJobListener.jobWasExecuted(JobExecutionContext context, JobExecutionException exception)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void JobListenerSupport.jobWasExecuted(JobExecutionContext context, JobExecutionException exception)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void BroadcastJobListener.jobWasExecuted(JobExecutionContext context, JobExecutionException exception)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Uses of `JobExecutionException` in `org.quartz.locality`

Methods in `org.quartz.locality` with parameters of type `JobExecutionException`

- `Trigger.CompletedExecutionInstruction` in `org.quartz.locality`
- `DelegatingLocalityTrigger.executionComplete(JobExecutionException result)`

Uses of `JobExecutionException` in `org.quartz.plugins.history`

Methods in `org.quartz.plugins.history` with parameters of type `JobExecutionException`

- `void LoggingJobHistoryPlugin.jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)`

Overview  Package  Class  Tree  Deprecated  Index  Help
PREV  NEXT  Frames  No Frames

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.JobKey**

## Packages that use **JobKey**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.matchers</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.triggers</strong></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td><strong>org.quartz.listeners</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.simpl</strong></td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.xml</strong></td>
<td></td>
</tr>
</tbody>
</table>

## Uses of **JobKey** in **org.quartz**

## Methods in **org.quartz** that return **JobKey**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobKey</td>
<td>Trigger. <strong>getJobKey</strong>( )</td>
</tr>
<tr>
<td>JobKey</td>
<td>JobDetail. <strong>getJobKey</strong>( )</td>
</tr>
<tr>
<td>static JobKey</td>
<td>JobKey.&lt;code&gt;jobKey&lt;/code&gt;(String name)</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>static JobKey</td>
<td>JobKey.&lt;code&gt;jobKey&lt;/code&gt;(String name, String group)</td>
</tr>
</tbody>
</table>

**Methods in org.quartz that return types with arguments of type JobKey**

<table>
<thead>
<tr>
<th>Set&lt;JobKey&gt;</th>
<th>Scheduler.&lt;code&gt;getJobKeys&lt;/code&gt;(GroupMatcher&lt;JobKey&gt; matcher)</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;Matcher&lt;JobKey&gt;&gt;</td>
<td>ListenerManager.&lt;code&gt;getJobListenerMatchers&lt;/code&gt;(String listenerName)</td>
</tr>
</tbody>
</table>

Get the keys of all the JobDetails in the matching groups.

Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

**Methods in org.quartz with parameters of type JobKey**

<table>
<thead>
<tr>
<th>boolean</th>
<th>Scheduler.&lt;code&gt;checkExists&lt;/code&gt;(JobKey jobKey)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Determine whether a Job with the given identifier already exists within the scheduler.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>Scheduler.&lt;code&gt;deleteJob&lt;/code&gt;(JobKey jobKey)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Delete the identified Job from the Scheduler - and any associated Triggers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TriggerBuilder&lt;T&gt;</th>
<th>TriggerBuilder.&lt;code&gt;forJob&lt;/code&gt;(JobKey jobKey)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the identity of the Job which should be fired by the produced Trigger.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JobDetail</th>
<th>Scheduler.&lt;code&gt;getJobDetail&lt;/code&gt;(JobKey jobKey)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the JobDetail for the Job instance with the given key.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List&lt;? extends Trigger&gt;</th>
<th>Scheduler.&lt;code&gt;getTriggersOfJob&lt;/code&gt;(JobKey jobKey)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get all Triggers that are associated with the identified JobDetail.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>Scheduler.&lt;code&gt;interrupt&lt;/code&gt;(JobKey jobKey)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which must be an implementor of the InterruptableJob interface.</td>
</tr>
</tbody>
</table>
### void SchedulerListener.onActivityResult(JobKey jobKey)
Called by the Scheduler when a JobDetail has been deleted.

### void SchedulerListener.jobPaused(JobKey jobKey)
Called by the Scheduler when a JobDetail has been paused.

### void SchedulerListener.jobResumed(JobKey jobKey)
Called by the Scheduler when a JobDetail has been un-paused.

### void Scheduler.pauseJob(JobKey jobKey)
Pause the JobDetail with the given key - by pausing all of its current Triggers.

### void Scheduler.resumeJob(JobKey jobKey)
Resume (un-pause) the JobDetail with the given key.

### void Scheduler.triggerJob(JobKey jobKey)
Trigger the identified JobDetail (execute it now).

### void Scheduler.triggerJob(JobKey jobKey, JobDataMap data)
Trigger the identified JobDetail (execute it now).

### JobBuilder JobBuilder.withIdentity(JobKey key)
Use a JobKey to identify the JobDetail.

---

### Method parameters in org.quartz with type arguments of type JobKey

#### void ListenerManager.addJobListener(JobListener jobListener, List<Matcher<JobKey>> matchers)
Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers.

#### boolean ListenerManager.addJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)
Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

#### boolean Scheduler.deleteJobs(List<JobKey> jobKeys)
Delete the identified Jobs from the Scheduler - and any associated Triggers.

#### Set<JobKey> Scheduler.getJobKeys(GroupMatcher<JobKey> matcher)
Get the keys of all the JobDetails in the matching groups.
void Scheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)
    Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.

boolean ListenerManager.removeJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)
    Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

void Scheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)
    Resume (un-pause) all of the JobDetails in matching groups.

boolean ListenerManager.setJobListenerMatchers(String listenerName, List&lt;Matcher&lt;JobKey&gt;&gt; matchers)
    Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

---

### Uses of `JobKey` in `org.quartz.core`

#### Methods in `org.quartz.core` that return types with arguments of type `JobKey`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| `Set&lt;JobKey&gt;` | QuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)  
Get the names of all the Jobs in the matching groups. |
| `Set&lt;JobKey&gt;` | RemotableQuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)  
Get the names of all the Jobs in the matching groups. |
| `List&lt;Matcher&lt;JobKey&gt;&gt;` | ListenerManagerImpl.getJobListenerMatchers(String listenerName)  
Get the set of Matchers for which the listener will receive events if ANY of the matchers match. |

#### Methods in `org.quartz.core` with parameters of type `JobKey`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| boolean | QuartzScheduler.checkExists(JobKey jobKey)  
Determine whether a Job with the given identifier already exists in the scheduler. |
| boolean | RemotableQuartzScheduler.checkExists(JobKey jobKey)  
Determine whether a Job with the given identifier already exists in the scheduler. |
| boolean | QuartzScheduler.deleteJob(JobKey jobKey)  
Delete the identified Job from the Scheduler - and any associated Triggers. |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean RemotableQuartzScheduler. <strong>deleteJob</strong>(JobKey jobKey)</td>
<td>Delete a job from the scheduler</td>
</tr>
<tr>
<td>JobDetail QuartzScheduler. <strong>getJobDetail</strong>(JobKey jobKey)</td>
<td>Get the JobDetail for the Job instance with the given name and group</td>
</tr>
<tr>
<td>JobDetail RemotableQuartzScheduler. <strong>getJobDetail</strong>(JobKey jobKey)</td>
<td>Get the JobDetail for the Job instance with the given name and group</td>
</tr>
<tr>
<td>List&lt;? extends Trigger&gt; QuartzScheduler. <strong>getTriggersOfJob</strong>(JobKey jobKey)</td>
<td>Get all Triggers that are associated with the identified JobDetail</td>
</tr>
<tr>
<td>List&lt;? extends Trigger&gt; RemotableQuartzScheduler. <strong>getTriggersOfJob</strong>(JobKey jobKey)</td>
<td>Get all Triggers that are associated with the identified JobDetail</td>
</tr>
<tr>
<td>boolean QuartzScheduler. <strong>interrupt</strong>(JobKey jobKey)</td>
<td>Interrupt all instances of the identified InterruptableJob executing in the Scheduler instance</td>
</tr>
<tr>
<td>boolean RemotableQuartzScheduler. <strong>interrupt</strong>(JobKey jobKey)</td>
<td>Interrupt all instances of the identified InterruptableJob executing in the Scheduler instance</td>
</tr>
<tr>
<td>void QuartzSchedulerMBeanImpl. <strong>jobDeleted</strong>(JobKey jobKey)</td>
<td>Notify the SchedulerListeners of a Job being deleted</td>
</tr>
<tr>
<td>void QuartzSchedulerMBeanImpl. <strong>jobPaused</strong>(JobKey jobKey)</td>
<td>Notify the SchedulerListeners of a Job being paused</td>
</tr>
<tr>
<td>void QuartzSchedulerMBeanImpl. <strong>jobResumed</strong>(JobKey jobKey)</td>
<td>Notify the SchedulerListeners of a Job being resumed</td>
</tr>
<tr>
<td>void QuartzScheduler. <strong>notifySchedulerListenersJobDeleted</strong>(JobKey jobKey)</td>
<td>Notify the SchedulerListeners of a Job being deleted</td>
</tr>
<tr>
<td>void SchedulerSignalerImpl. <strong>notifySchedulerListenersJobDeleted</strong>(JobKey jobKey)</td>
<td>Notify the SchedulerListeners of a Job being deleted</td>
</tr>
<tr>
<td>void QuartzScheduler. <strong>notifySchedulerListenersPausedJob</strong>(JobKey jobKey)</td>
<td>Notify the SchedulerListeners of a Job being paused</td>
</tr>
<tr>
<td>void QuartzScheduler. <strong>notifySchedulerListenersResumedJob</strong>(JobKey jobKey)</td>
<td>Notify the SchedulerListeners of a Job being resumed</td>
</tr>
<tr>
<td>void QuartzScheduler. <strong>pauseJob</strong>(JobKey jobKey)</td>
<td>Pause the JobDetail with the given name - by pausing all of its Triggers</td>
</tr>
<tr>
<td>void RemotableQuartzScheduler. <strong>pauseJob</strong>(JobKey jobKey)</td>
<td>Pause the JobDetail with the given name - by pausing all of its Triggers</td>
</tr>
<tr>
<td>void QuartzScheduler. <strong>resumeJob</strong>(JobKey jobKey)</td>
<td>Resume the JobDetail with the given name - by resuming all of its Triggers</td>
</tr>
</tbody>
</table>
Resume (un-pause) the JobDetail with the given name.

```java
void RemotableQuartzScheduler.resumeJob(JobKey jobKey)
```

Trigger the identified Job (execute it now) - with a non-volatile trigger.

```java
void QuartzScheduler.triggerJob(JobKey jobKey, JobDataMap data)
```

```java
void RemotableQuartzScheduler.triggerJob(JobKey jobKey, JobDataMap data)
```

<table>
<thead>
<tr>
<th>Method parameters in org.quartz.core with type arguments of type JobKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>void ListenerManagerImpl.addJobListener(JobListener jobListener, List&lt;Matcher&lt;JobKey&gt;&gt; matchers)</td>
</tr>
<tr>
<td>boolean ListenerManagerImpl.addJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>boolean QuartzScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
</tr>
<tr>
<td>boolean RemotableQuartzScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; QuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>Get the names of all the Jobs in the matching groups.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; RemotableQuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void QuartzScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; groupMatcher)</td>
</tr>
<tr>
<td>Pause all of the JobDetails in the matching groups - by pausing their Triggers.</td>
</tr>
<tr>
<td>void RemotableQuartzScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>boolean ListenerManagerImpl.removeJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void QuartzScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>Resume (un-pause) all of the JobDetails in the matching groups.</td>
</tr>
<tr>
<td>void RemotableQuartzScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
</tbody>
</table>
boolean ListenerManagerImpl.setJobListenerMatchers(String listener List<Matcher<JobKey>> matchers)

### Uses of JobKey in org.quartz.impl

### Methods in org.quartz.impl that return JobKey

| JobKey | JobDetailImpl.getKey() |

### Methods in org.quartz.impl that return types with arguments of type JobKey

| Set<JobKey> | RemoteScheduler.getJobKeys(GroupMatcher<JobKey> matcher) |
| Set<JobKey> | StdScheduler.getJobKeys(GroupMatcher<JobKey> matcher) |
| Set<JobKey> | RemoteMBeanScheduler.getJobKeys(GroupMatcher<JobKey> matcher) |

Calls the equivalent method on the 'proxied' QuartzScheduler passing the SchedulingContext associated with this instance.

### Methods in org.quartz.impl with parameters of type JobKey

| boolean | RemoteScheduler.checkExists(JobKey jobKey) |
| boolean | StdScheduler.checkExists(JobKey jobKey) |
| boolean | RemoteMBeanScheduler.checkExists(JobKey jobKey) |
| boolean | RemoteScheduler.deleteJob(JobKey jobKey) |

Calls the equivalent method on the 'proxied' QuartzScheduler.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean QuartzScheduler. <strong>deleteJob</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean StdScheduler. <strong>deleteJob</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean RemoteMBeanScheduler. <strong>deleteJob</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>JobDetail RemoteScheduler. <strong>getJobDetail</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>JobDetail StdScheduler. <strong>getJobDetail</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>JobDetail RemoteMBeanScheduler. <strong>getJobDetail</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>List&lt;? extends Trigger&gt;* RemoteScheduler. <strong>getTriggersOfJob</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>List&lt;? extends Trigger&gt;* StdScheduler. <strong>getTriggersOfJob</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>List&lt;Trigger&gt;* RemoteMBeanScheduler. <strong>getTriggersOfJob</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>boolean RemoteScheduler. <strong>interrupt</strong> <em>(JobKey jobKey)</em></td>
<td></td>
</tr>
<tr>
<td>boolean StdScheduler. <strong>interrupt</strong> <em>(JobKey jobKey)</em></td>
<td></td>
</tr>
<tr>
<td>boolean RemoteMBeanScheduler. <strong>interrupt</strong> <em>(JobKey jobKey)</em></td>
<td></td>
</tr>
<tr>
<td>void RemoteScheduler. <strong>pauseJob</strong> <em>(JobKey jobKey)</em></td>
<td></td>
</tr>
</tbody>
</table>
void StdScheduler.pauseJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void RemoteMBeanScheduler.pauseJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler, passing the SchedulingContext associated
    with this instance.

void RemoteScheduler.resumeJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void StdScheduler.resumeJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void RemoteMBeanScheduler.resumeJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler, passing the SchedulingContext associated
    with this instance.

void JobDetailImpl.setKey(JobKey key)

void RemoteScheduler.triggerJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void StdScheduler.triggerJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void RemoteMBeanScheduler.triggerJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler, passing the SchedulingContext associated
    with this instance.

void RemoteScheduler.triggerJob(JobKey jobKey, JobDataMap data)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void StdScheduler.triggerJob(JobKey jobKey, JobDataMap data)
Calls the equivalent method on the 'proxied' QuartzScheduler.

```java
void RemoteMBeanScheduler.triggerJob(JobKey jobKey, JobDataMap data)
```

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

<table>
<thead>
<tr>
<th>Method parameters in <code>org.quartz.impl</code> with type arguments of type <code>JobKey</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean RemoteScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
</tr>
<tr>
<td>boolean StdScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
</tr>
<tr>
<td>boolean RemoteMBeanScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; RemoteScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; StdScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; RemoteMBeanScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void RemoteScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void StdScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void RemoteScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void StdScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
</tbody>
</table>
Calls the equivalent method on the 'proxied' QuartzScheduler passing the SchedulingContext associated with this instance.

## Uses of JobKey in org.quartz.impl.jdbcjobstore

### Methods in org.quartz.impl.jdbcjobstore that return JobKey

<table>
<thead>
<tr>
<th>JobKey</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBKEY</td>
<td>FiredTriggerRecord.getJobKey()</td>
</tr>
<tr>
<td>JOBKEY</td>
<td>TriggerStatus.getJobKey()</td>
</tr>
</tbody>
</table>

### Methods in org.quartz.impl.jdbcjobstore that return types with arguments of type JobKey

<table>
<thead>
<tr>
<th>JobKey</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBKEY</td>
<td>JobStoreSupport.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher) Get the names of all of the Jobs that matcher the given groupMatcher.</td>
</tr>
<tr>
<td>JOBKEY</td>
<td>JobStoreSupport.getJobNames(Connection conn, GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>JOBKEY</td>
<td>StdJDBCDelegate.selectJobsInGroup(Connection conn, GroupMatcher&lt;JobKey&gt; matcher) Select all of the jobs contained in a given group.</td>
</tr>
<tr>
<td>JOBKEY</td>
<td>DriverDelegate.selectJobsInGroup(Connection conn, GroupMatcher&lt;JobKey&gt; matcher) Select all of the jobs contained in a given group.</td>
</tr>
</tbody>
</table>

### Methods in org.quartz.impl.jdbcjobstore with parameters of type JobKey

<table>
<thead>
<tr>
<th>JobKey</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBKEY</td>
<td>JobStoreSupport.checkBlockedState(String currentState) Determines if a Trigger for the given job</td>
</tr>
<tr>
<td>JOBKEY</td>
<td>JobStoreSupport.checkExists(Connection conn)</td>
</tr>
</tbody>
</table>
boolean JobStoreSupport.checkExists(JobKey jobKey)
Determine whether a Job with the given identifier already exists.

int StdJDBCDelegate.deleteJobDetail(Connection conn)
Delete the job detail record for the given job.

int DriverDelegate.deleteJobDetail(Connection conn)
Delete the job detail record for the given job.

protected List<org.quartz.spi.OperableTrigger> JobStoreSupport.getTriggersForJob(Connection conn)
Get all of the Triggers that are associated with the given Job.

boolean StdJDBCDelegate.isJobNonConcurrent(Connection conn)
Check whether or not the given job is stateful.

boolean DriverDelegate.isJobNonConcurrent(Connection conn)
Check whether or not the given job disallows concurrent execution.

boolean StdJDBCDelegate.jobExists(Connection conn)
Check whether or not the given job exists.

boolean DriverDelegate.jobExists(Connection conn)
Check whether or not the given job exists.

protected boolean JobStoreSupport.jobExists(Connection conn)
Check existence of a given job.

void JobStoreSupport.pauseJob(JobKey jobKey)
Pause the Job with the given name.

protected boolean JobStoreSupport.removeJob(Connection conn, boolean activeDeleteSafe)
Remove (delete) the Job with the given identifier.

boolean JobStoreSupport.removeJob(JobKey jobKey)
Remove (delete) the Job with the given identifier.

void JobStoreSupport.resumeJob(JobKey jobKey)
Resume (un-pause) the Job with the given name.

protected JobDetail JobStoreSupport.retrieveJob(Connection conn)
Retrieve the JobDetail for the given JobKey.

JobDetail JobStoreSupport.retrieveJob(JobKey jobKey)
Retrieve the JobDetail for the given JobKey.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdJDBCDelegate.selectJobDetail(Connection org.quartz.spi.ClassLoadHelper loadHelper)</td>
<td>Select the JobDetail object for a given job name/group name.</td>
</tr>
<tr>
<td>DriverDelegate.selectJobDetail(Connection org.quartz.spi.ClassLoadHelper loadHelper)</td>
<td>Select the JobDetail object for a given job name/group name.</td>
</tr>
<tr>
<td>int StdJDBCDelegate.selectJobExecutionCount</td>
<td>Get the number instances of the identified job currently executing.</td>
</tr>
<tr>
<td>int DriverDelegate.selectJobExecutionCount</td>
<td>Get the number instances of the identified job currently executing.</td>
</tr>
<tr>
<td>int StdJDBCDelegate.selectNumTriggersForJob</td>
<td>Select the number of triggers associated with a given job.</td>
</tr>
<tr>
<td>int DriverDelegate.selectNumTriggersForJob</td>
<td>Select the number of triggers associated with a given job.</td>
</tr>
<tr>
<td>int DB2v6Delegate.selectNumTriggersForJob(Connection)</td>
<td>Select the number of triggers associated with a given job.</td>
</tr>
<tr>
<td>List&lt;TriggerKey&gt; StdJDBCDelegate.selectTriggerKeysForJob</td>
<td>Get the names of all of the triggers associated with a given job.</td>
</tr>
<tr>
<td>List&lt;TriggerKey&gt; DriverDelegate.selectTriggerKeysForJob</td>
<td>Get the names of all of the triggers associated with a given job.</td>
</tr>
<tr>
<td>List&lt;org.quartz.spi.OperableTrigger&gt; StdJDBCDelegate.selectTriggersForJob(Connection)</td>
<td>Select the triggers for a job.</td>
</tr>
<tr>
<td>List&lt;org.quartz.spi.OperableTrigger&gt; DriverDelegate.selectTriggersForJob(Connection)</td>
<td>Select the triggers for a job.</td>
</tr>
<tr>
<td>void FiredTriggerRecord.setJobKey(JobKey key)</td>
<td></td>
</tr>
<tr>
<td>void TriggerStatus.setJobKey(JobKey jobKey)</td>
<td></td>
</tr>
<tr>
<td>int StdJDBCDelegate.updateTriggerStatesForJob(JobKey jobKey, String state)</td>
<td>Update the states of all triggers associated with the given job.</td>
</tr>
<tr>
<td>int DriverDelegate.updateTriggerStatesForJob(String state)</td>
<td>Update the states of all triggers associated with the given job.</td>
</tr>
<tr>
<td>int StdJDBCDelegate.updateTriggerStatesForJobFromOtherState(JobKey jobKey, String state, String oldState)</td>
<td>Update the states of all triggers associated with the given job.</td>
</tr>
</tbody>
</table>
DriverDelegate.updateTriggerStatesForJob

JobKey jobKey, String state, String oldState

Update the states of any triggers associated with the given job given its current state.

Method parameters in org.quartz.impl.jdbcjobstore with type arguments of type JobKey

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;JobKey&gt;</td>
<td>JobStoreSupport.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher) Get the names of all Job s that match the given Matcher.</td>
</tr>
<tr>
<td>protected Set&lt;JobKey&gt;</td>
<td>JobStoreSupport.getJobNames(Connection conn, GroupMatcher&lt;JobKey&gt; matcher) Get the names of all the Jobs that match the given GroupMatcher.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher) Pause all of the Jobs matching the given GroupMatcher - by pausing all of their Triggers.</td>
</tr>
<tr>
<td>boolean</td>
<td>JobStoreSupport.removeJobs(List&lt;JobKey&gt; jobKeys) Remove (un-pause) all of the Jobs in the given group.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher) Resume (un-pause) all of the Jobs in the given group.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt;</td>
<td>StdJDBCDelegate.selectJobsInGroup(Connection conn, GroupMatcher&lt;JobKey&gt; matcher) Select all of the jobs contained in a given group.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt;</td>
<td>DriverDelegate.selectJobsInGroup(Connection conn, GroupMatcher&lt;JobKey&gt; matcher) Select all of the jobs contained in a given group.</td>
</tr>
</tbody>
</table>

Uses of JobKey in org.quartz.impl.matchers

Methods in org.quartz.impl.matchers that return types with arguments of type JobKey

static EverythingMatcher<JobKey> EverythingMatcher.allJobs()
Create an EverythingMatcher that matches all jobs.

### Uses of `JobKey` in `org.quartz.impl.triggers`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.impl.triggers</code> that return <code>JobKey</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobKey</code> AbstractTrigger. <strong>getJobKey</strong>()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.impl.triggers</code> with parameters of type <code>JobKey</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code> AbstractTrigger. <strong>setJobKey</strong>(<code>JobKey</code> key)</td>
</tr>
</tbody>
</table>

### Uses of `JobKey` in `org.quartz.listeners`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.listeners</code> with parameters of type <code>JobKey</code></th>
</tr>
</thead>
</table>
| `void` JobChainingJobListener. **addJobChainLink**(`JobKey` firstJob, `JobKey` secondJob)  
  Add a chain mapping - when the Job identified by the first key completes the job identified by the second key will be triggered. |
| `void` SchedulerListenerSupport. **jobDeleted**(`JobKey` jobKey) |
| `void` BroadcastSchedulerListener. **jobDeleted**(`JobKey` jobKey) |
| `void` SchedulerListenerSupport. **jobPaused**(`JobKey` jobKey) |
| `void` BroadcastSchedulerListener. **jobPaused**(`JobKey` key) |
| `void` SchedulerListenerSupport. **jobResumed**(`JobKey` jobKey) |
void BroadcastSchedulerListener.jobResumed(JobKey key)

<table>
<thead>
<tr>
<th>Uses of JobKey in org.quartz.locality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods in org.quartz.locality that return JobKey</td>
</tr>
<tr>
<td>JobKey DelegatingLocalityTrigger.getJobKey()</td>
</tr>
<tr>
<td>JobKey DelegatingLocalityJobDetail.getKey()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.locality with parameters of type JobKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>void DelegatingLocalityTrigger.setJobKey(JobKey key)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of JobKey in org.quartz.simpl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields in org.quartz.simpl with type parameters of type JobKey</td>
</tr>
<tr>
<td>protected HashSet&lt;JobKey&gt; RAMJobStore.blockedJobs</td>
</tr>
<tr>
<td>protected &lt;br&gt;HashMap&lt;String, &lt;br&gt;HashMap&lt;JobKey, org.quartz.simpl.JobWrapper&gt;&gt; RAMJobStore.jobsByGroup</td>
</tr>
<tr>
<td>protected &lt;br&gt;HashMap&lt;JobKey, org.quartz.simpl.JobWrapper&gt; RAMJobStore.jobsByKey</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.simpl that return types with arguments of type JobKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;JobKey&gt;</td>
</tr>
</tbody>
</table>
**Methods in** org.quartz.simpl **with parameters of type** JobKey

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean RAMJobStore.checkExists(JobKey jobKey)</td>
<td>Determine whether a Job with the given identifier already exists within the scheduler.</td>
</tr>
<tr>
<td>List&lt;org.quartz.spi.OperableTrigger&gt; RAMJobStore.getTriggersForJob(JobKey jobKey)</td>
<td>Get all of the Triggers that are associated with the given Job.</td>
</tr>
<tr>
<td>protected ArrayList&lt;org.quartz.simpl.TriggerWrapper&gt; RAMJobStore.getTriggerWrappersForJob(JobKey jobKey)</td>
<td></td>
</tr>
<tr>
<td>void RAMJobStore.pauseJob(JobKey jobKey)</td>
<td>Pause the JobDetail with the given JobKey and all of its current Triggers.</td>
</tr>
<tr>
<td>boolean RAMJobStore.removeJob(JobKey jobKey)</td>
<td>Remove (delete) the Job with the given JobID and the Triggers that reference it.</td>
</tr>
<tr>
<td>void RAMJobStore.resumeJob(JobKey jobKey)</td>
<td>Resume (un-pause) the JobDetail with the given JobKey.</td>
</tr>
<tr>
<td>JobDetail RAMJobStore.retrieveJob(JobKey jobKey)</td>
<td>Retrieve the JobDetail for the given JobKey.</td>
</tr>
<tr>
<td>protected void RAMJobStore.setAllTriggersOfJobToState(JobKey jobKey, int state)</td>
<td></td>
</tr>
</tbody>
</table>

**Method parameters in** org.quartz.simpl **with type arguments of type** JobKey

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;JobKey&gt;</td>
<td>RAMJobStore.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher) Get the names of all of the Jobs that match the given groupMatcher.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>RAMJobStore.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher) Pause all of the JobDetails in the given group - by...</td>
</tr>
</tbody>
</table>
pausing all of their Triggers.

<table>
<thead>
<tr>
<th>boolean</th>
<th>RAMJobStore.removeJobs(List&lt;JobKey&gt; jobKeys)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection&lt;String&gt;</td>
<td>RAMJobStore.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
</tbody>
</table>

Resume (un-pause) all of the JobDetails in the given group.

**Uses of JobKey in org.quartz.xml**

<table>
<thead>
<tr>
<th>Fields in org.quartz.xml with type parameters of type JobKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected List&lt;JobKey&gt;</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface org.quartz.JobListener

## Packages that use **JobListener**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.listeners</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.history</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of **JobListener** in org.quartz

## Methods in org.quartz that return **JobListener**

```
JobListener ListenerManager.getJobListener(String name)
```

Get the JobListener that has the given name.

## Methods in org.quartz that return types with arguments of type **JobListener**

```
List<JobListener> ListenerManager.getJobListeners()
```

Get a List containing all of the JobListeners in the Scheduler.

## Methods in org.quartz with parameters of type **JobListener**

```
void ListenerManager.addJobListener(JobListener jobListener, List<Matcher<JobKey>> matchers)
```

Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers.
Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers.

**Uses of JobListener in org.quartz.core**

<table>
<thead>
<tr>
<th>Classes in org.quartz.core that implement JobListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>class QuartzSchedulerMBEanImpl</td>
</tr>
<tr>
<td>class SampledStatisticsImpl</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.core that return JobListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobListener QuartzScheduler. getInternalJobListener(String name)</td>
</tr>
<tr>
<td>JobListener ListenerManagerImpl. getJobListener(String name)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.core that return types with arguments of type JobListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;JobListener&gt; QuartzScheduler. getInternalJobListeners()</td>
</tr>
<tr>
<td>List&lt;JobListener&gt; ListenerManagerImpl. getJobListeners()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.core with parameters of type JobListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>void QuartzScheduler. addInternalJobListener(JobListener jobListener)</td>
</tr>
<tr>
<td>void ListenerManagerImpl. addJobListener(JobListener jobListener,</td>
</tr>
</tbody>
</table>
ListenerManagerImpl.addJobListener(JobListener jobListener, Matcher<JobKey>... matchers)

Uses of JobListener in org.quartz.listeners

<table>
<thead>
<tr>
<th>Classes in org.quartz.listeners that implement JobListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>BroadcastJobListener</td>
</tr>
<tr>
<td>Holds a List of references to JobListener instances and broadcasts all events to them (in order).</td>
</tr>
<tr>
<td>JobChainingJobListener</td>
</tr>
<tr>
<td>Keeps a collection of mappings of which Job to trigger after the completion of a given job.</td>
</tr>
<tr>
<td>JobListenerSupport</td>
</tr>
<tr>
<td>A helpful abstract base class for implementors of JobListener.</td>
</tr>
</tbody>
</table>

Methods in org.quartz.listeners that return types with arguments of type JobListener

| List<JobListener> | BroadcastJobListener.getListeners() |

Methods in org.quartz.listeners with parameters of type JobListener

| void | BroadcastJobListener.addListener(JobListener listener) |
| boolean | BroadcastJobListener.removeListener(JobListener listener) |

Constructor parameters in org.quartz.listeners with type arguments of
type JobListener

BroadcastJobListener(String name, List<JobListener> listeners)

Construct an instance with the given name, and List of listeners.

Uses of JobListener in org.quartz.plugins.history

Classes in org.quartz.plugins.history that implement JobListener

class LoggingJobHistoryPlugin

Logs a history of all job executions (and execution vetos) via the Jakarta Commons-Logging framework.
Uses of Class
org.quartz.JobPersistenceException

<table>
<thead>
<tr>
<th>Packages that use JobPersistenceException</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
</tr>
<tr>
<td>org.quartz.core</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
</tr>
<tr>
<td>org.quartz.locality</td>
</tr>
<tr>
<td>org.quartz.simpl</td>
</tr>
</tbody>
</table>

Uses of JobPersistenceException in org.quartz

<table>
<thead>
<tr>
<th>Subclasses of JobPersistenceException in org.quartz</th>
</tr>
</thead>
<tbody>
<tr>
<td>class ObjectAlreadyExistsException</td>
</tr>
<tr>
<td>An exception that is thrown to indicate that an attempt to store a new object (i.e.</td>
</tr>
</tbody>
</table>

Uses of JobPersistenceException in org.quartz.core

<table>
<thead>
<tr>
<th>Methods in org.quartz.core that throw JobPersistenceException</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void QuartzScheduler.notifyJobStoreJobComplete(org.quartz.spi.OperableTrigger trigger, JobDetail detail, Trigger.CompletedExecutionInstruction inst</td>
</tr>
</tbody>
</table>
protected void QuartzScheduler.notifyJobStoreJobVetoed(org.quartz.spi.OperableTrigger trigger, JobDetail detail, Trigger.CompletedExecutionInstruction instCode)

Uses of JobPersistenceException in org.quartz.impl.jdbcjobstore

Subclasses of JobPersistenceException in org.quartz.impl.jdbcjobstore

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockException</td>
<td>Exception class for when there is a failure obtaining or releasing a resource lock.</td>
</tr>
<tr>
<td>NoSuchDelegateException</td>
<td>Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.</td>
</tr>
</tbody>
</table>

Methods in org.quartz.impl.jdbcjobstore that throw JobPersistenceException

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected List&lt;org.quartz.spi.OperableTrigger&gt; JobStoreSupport.acquireNextTrigger(int maxCount, long timeWindow)</td>
<td>Get a handle to the next N triggers to be fired, and mark them as 'reserved'. scheduler.</td>
</tr>
<tr>
<td>List&lt;org.quartz.spi.OperableTrigger&gt; JobStoreSupport.acquireNextTriggers(long noLaterThan, long timeWindow)</td>
<td>Get a handle to the next N triggers to be fired, and mark them as 'reserved'. scheduler.</td>
</tr>
<tr>
<td>protected boolean JobStoreSupport.calendarExists(Connection)</td>
<td>Determines if a Trigger for the given job should be blocked.</td>
</tr>
<tr>
<td>protected boolean JobStoreSupport.checkBlockedState(Connection, String currentState)</td>
<td>Determines if a Trigger for the given job should be blocked.</td>
</tr>
<tr>
<td>protected boolean JobStoreSupport.checkExists(Connection)</td>
<td>Determines if a Trigger for the given job should be blocked.</td>
</tr>
<tr>
<td>protected boolean JobStoreSupport.checkExists(Connection)</td>
<td>Determines if a Trigger for the given job should be blocked.</td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>boolean JobStoreSupport.checkExists(JobKey jobKey)</code></td>
<td></td>
</tr>
<tr>
<td>Determine whether a <code>Job</code> with the given identifier already exists.</td>
<td></td>
</tr>
<tr>
<td><code>void JobStoreSupport.clearAllSchedulingData()</code></td>
<td></td>
</tr>
<tr>
<td>Clear (delete!) all scheduling data.</td>
<td></td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.clearAllSchedulingData()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected List&lt;SchedulerStateRecord&gt; JobStoreSupport.clusterCheckIn(Connection)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.clusterCheckIn(Connection)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.clusterRecover(Connection, List&lt;SchedulerStateRecord&gt; failedInstances)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.commitConnection(Connection)</code></td>
<td></td>
</tr>
<tr>
<td>Commit the supplied connection.</td>
<td></td>
</tr>
<tr>
<td><code>protected boolean JobStoreSupport.doCheckin()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected JobStoreSupport.RecoverMisfiredJobsResult JobStoreSupport.doRecoverMisfires()</code></td>
<td></td>
</tr>
<tr>
<td><code>Object JobStoreSupport.TransactionCallback</code></td>
<td></td>
</tr>
<tr>
<td><code>void JobStoreSupport.VoidTransactionCallback</code></td>
<td></td>
</tr>
<tr>
<td><code>protected abstract Object JobStoreSupport.executeInLock(String)</code></td>
<td></td>
</tr>
<tr>
<td>Execute the given callback having acquired the given lock.</td>
<td></td>
</tr>
<tr>
<td><code>protected Object JobStoreTX.executeInLock(String)</code></td>
<td></td>
</tr>
<tr>
<td>Execute the given callback having acquired the given lock.</td>
<td></td>
</tr>
<tr>
<td><code>protected Object JobStoreCMT.executeInLock(String)</code></td>
<td></td>
</tr>
<tr>
<td>Execute the given callback having acquired the given lock.</td>
<td></td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.executeInLock(String)</code></td>
<td></td>
</tr>
<tr>
<td>Execute the given callback having acquired a lock.</td>
<td></td>
</tr>
<tr>
<td><code>protected Object JobStoreSupport.executeInNonManagedTXLock</code></td>
<td></td>
</tr>
<tr>
<td><code>Object JobStoreSupport.executeInNonManagedTXLock</code></td>
<td></td>
</tr>
<tr>
<td><code>Object JobStoreSupport.executeInNonManagedCMT</code></td>
<td></td>
</tr>
<tr>
<td>Execute the given callback having acquired a lock.</td>
<td></td>
</tr>
<tr>
<td><code>Object JobStoreSupport.executeInNonManaged</code></td>
<td></td>
</tr>
<tr>
<td>Execute the given callback having acquired a lock.</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>executeInNonManagedTXLock()</code></td>
<td>Execute the given callback having optionally acquired the given lock.</td>
</tr>
<tr>
<td><code>executeWithoutLock()</code></td>
<td>Execute the given callback in a transaction.</td>
</tr>
<tr>
<td><code>findFailedInstances()</code></td>
<td>Get a list of all scheduler instances in the cluster that may have failed.</td>
</tr>
<tr>
<td><code>getCalendarNames()</code></td>
<td>Get the names of all of the Calendar groups.</td>
</tr>
<tr>
<td><code>getJobGroupNames()</code></td>
<td>Get the names of all of the Job groups.</td>
</tr>
<tr>
<td><code>getJobKeys()</code></td>
<td>Get the names of all of the Jobs that match the given GroupMatcher.</td>
</tr>
<tr>
<td><code>getJobNames()</code></td>
<td>Get the names of all of the Jobs.</td>
</tr>
<tr>
<td><code>getNumberOfCalendars()</code></td>
<td>Get the number of Calendar groups that are active.</td>
</tr>
<tr>
<td><code>getNumberOfJobs()</code></td>
<td>Get the number of Jobs that are scheduled.</td>
</tr>
</tbody>
</table>
protected int JobStoreSupport.getNumberofJobs(Connection)

int JobStoreSupport.getNumberofTriggers

protected int JobStoreSupport.getNumberofTriggers

Set<String> JobStoreSupport.getPausedTriggerGroups

Set<String> JobStoreSupport.getPausedTriggerGroups

List<String> JobStoreSupport.getTriggerGroupName;

protected List<String> JobStoreSupport.getTriggerGroupName;

Set<TriggerKey> JobStoreSupport.getTriggerKeys(Group)

protected Set<TriggerKey> JobStoreSupport.getTriggerKeys:

protected List<TriggerKey> JobStoreSupport.getTriggerKeys:

protected List<TriggerKey> JobStoreSupport.getTriggersForJob:

List<org.quartz.spi.OperableTrigger> JobStoreSupport.getTriggersForJob:

Get all of the Triggers that are associated to the given Job.

Trigger.TriggerState JobStoreSupport.getTriggerState(Connection)

Trigger.TriggerState JobStoreSupport.getTriggerState:

Get the current state of the identified Trigger.

protected boolean JobStoreSupport.jobExists(Connection)

Check existence of a given job.

void JobStoreSupport.pauseAll()

Pause all triggers - equivalent of calling job.pauseAll().

void JobStoreSupport.pauseAll(Connection)

Pause all triggers - equivalent of calling job.pauseAll().

void JobStoreSupport.pauseJob(JobKey jobKey)

Pause the specified job.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pause the Job with the given name</strong></td>
<td></td>
</tr>
<tr>
<td><code>Set&lt;String&gt; JobStoreSupport.pauseJobs(GroupMatcher)</code></td>
<td>Pause all of the Jobs matching the given name.</td>
</tr>
<tr>
<td><code>void JobStoreSupport.pauseTrigger(Connection)</code></td>
<td>Pause the Trigger with the given name.</td>
</tr>
<tr>
<td><code>void JobStoreSupport.pauseTrigger(TriggerKey)</code></td>
<td>Pause the Trigger with the given name.</td>
</tr>
<tr>
<td><code>Set&lt;String&gt; JobStoreSupport.pauseTriggerGroup(Connection, GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Pause all of the Triggers matching the given name.</td>
</tr>
<tr>
<td><code>Set&lt;String&gt; JobStoreSupport.pauseTriggers(GroupMatcher)</code></td>
<td>Pause all of the Triggers matching the given name.</td>
</tr>
<tr>
<td><strong>protected void JobStoreSupport.recoverJobs</strong></td>
<td>Recover any failed or misfired jobs.</td>
</tr>
<tr>
<td><strong>protected void JobStoreSupport.recoverJobs(Connection)</strong></td>
<td>Will recover any failed or misfired jobs.</td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.releaseAcquiredTrigger(org.quartz.spi.OperableTrigger trigger)</code></td>
<td>Inform the JobStore that the scheduler no longer plans to use the trigger.</td>
</tr>
<tr>
<td><code>void JobStoreSupport.releaseAcquiredTrigger</code></td>
<td>Inform the JobStore that the scheduler no longer plans to use the trigger.</td>
</tr>
<tr>
<td><strong>protected boolean JobStoreSupport.removeCalendar(Connection)</strong></td>
<td>Remove (delete) the calendar with the given name.</td>
</tr>
<tr>
<td><code>boolean JobStoreSupport.removeCalendar(String)</code></td>
<td>Remove (delete) the calendar with the given name.</td>
</tr>
<tr>
<td><strong>protected boolean JobStoreSupport.removeJob(Connection, boolean activeDeleteSafe)</strong></td>
<td>Remove (delete) the Job with the given name.</td>
</tr>
<tr>
<td><code>boolean JobStoreSupport.removeJob(JobKey)</code></td>
<td>Remove (delete) the Job with the given name.</td>
</tr>
<tr>
<td><code>boolean JobStoreSupport.removeJobs(List&lt;Job&gt;)</code></td>
<td>Remove (delete) the Jobs with the given names.</td>
</tr>
<tr>
<td>Method</td>
<td>Code</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>protected boolean</td>
<td>JobStoreSupport.removeTrigger(ConnectionString)</td>
</tr>
<tr>
<td>boolean</td>
<td>JobStoreSupport.removeTrigger(TriggerKey)</td>
</tr>
<tr>
<td>boolean</td>
<td>JobStoreSupport.removeTriggers(List&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>protected boolean</td>
<td>JobStoreSupport.replaceTrigger(ConnectionString, OperableTrigger)</td>
</tr>
<tr>
<td>boolean</td>
<td>JobStoreSupport.replaceTrigger(TriggerKey, OperableTrigger)</td>
</tr>
<tr>
<td>void</td>
<td>JobStoreSupport.resumeAll()</td>
</tr>
<tr>
<td>void</td>
<td>JobStoreSupport.resumeAll(Connection)</td>
</tr>
<tr>
<td>void</td>
<td>JobStoreSupport.resumeJob(JobKey)</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.resumeJobs(GroupMatcher&lt;JobKey&gt;)</td>
</tr>
<tr>
<td>void</td>
<td>JobStoreSupport.resumeTrigger(Connection)</td>
</tr>
<tr>
<td>void</td>
<td>JobStoreSupport.resumeTrigger(TriggerKey)</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.resumeTriggerGroup(GroupMatcher&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.resumeTriggers(GroupMatcher&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>protected Calendar</td>
<td>JobStoreSupport.retrieveCalendar(Class&lt;Calendar&gt;)</td>
</tr>
<tr>
<td>Calendar</td>
<td>JobStoreSupport.retrieveCalendar(Class&lt;Calendar&gt;)</td>
</tr>
<tr>
<td>protected JobDetail</td>
<td>JobStoreSupport.retrieveJob(JobKey)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>JobStoreSupport.retrieveJob(JobKey jobKey)</code></td>
<td>Retrieve the JobDetail for the given jobKey.</td>
</tr>
<tr>
<td><code>JobStoreSupport.retrieveTrigger(TriggerKey triggerKey)</code></td>
<td>Retrieve the given Trigger.</td>
</tr>
<tr>
<td><code>StdJDBCDelegate.selectTrigger(Connection conn)</code></td>
<td>Select a trigger.</td>
</tr>
<tr>
<td><code>DriverDelegate.selectTrigger(Connection conn)</code></td>
<td>Select a trigger.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt;</code></td>
<td>Select the triggers for a calendar.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt;</code></td>
<td>Select the triggers for a job.</td>
</tr>
<tr>
<td><code>static void Util.setBeanProps(Object obj, String propertyName, String... propertyValues)</code></td>
<td>Set properties for an object.</td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.storeCalendar(Connection conn, boolean replaceExisting, boolean updateTriggers)</code></td>
<td>Store the given Calendar.</td>
</tr>
<tr>
<td><code>void JobStoreSupport.storeCalendar(String name, boolean replaceExisting, boolean updateTriggers)</code></td>
<td>Store the given Calendar.</td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.storeJob(Connection conn, boolean replaceExisting)</code></td>
<td>Insert or update a job.</td>
</tr>
<tr>
<td><code>void JobStoreSupport.storeJob(JobDetail newJob)</code></td>
<td>Store the given JobDetail.</td>
</tr>
<tr>
<td><code>void JobStoreSupport.storeJobAndTrigger(JobDetail newJob, org.quartz.spi.OperableTrigger newTrigger)</code></td>
<td>Store the given JobDetail and Trigger.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>JobStoreSupport.storeJobsAndTrigger(boolean replace)</code></td>
<td>Store jobs and triggers.</td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.storeTrigger(Connection conn, org.quartz.spi.OperableTrigger newTrigger, String state, boolean forceState, boolean recovering)</code></td>
<td>Insert or update a trigger.</td>
</tr>
<tr>
<td><code>void JobStoreSupport.storeTrigger(org.quartz.spi.OperableTrigger newTrigger, boolean replaceExisting)</code></td>
<td>Store the given Trigger.</td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.triggeredJobComplete(org.quartz.spi.OperableTrigger trigger, Trigger.CompletedExecutionInstruction instr)</code></td>
<td>Inform the JobStore that the scheduler has completed the execution its associated Job, and that it should be updated if the Job is stateful.</td>
</tr>
<tr>
<td><code>protected boolean JobStoreSupport.triggerExists(Connection conn)</code></td>
<td>Check existence of a given trigger.</td>
</tr>
<tr>
<td><code>protected org.quartz.spi.TriggerFiredBundle JobStoreSupport.triggerFired(Connection conn, org.quartz.spi.OperableTrigger trigger)</code></td>
<td>Inform the JobStore that the scheduler is now firing the associated Job, that it had previously acquired (reserved).</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.TriggerFiredResult&gt; JobStoreSupport.triggersFired(List&lt;org.quartz.spi.OperableTrigger&gt; triggers)</code></td>
<td>Inform the JobStore that the scheduler is now firing the associated Job, that it had previously acquired (reserved).</td>
</tr>
<tr>
<td><code>protected boolean JobStoreSupport.updateMisfiredTrigger(String newStateIfNotComplete, boolean forceState)</code></td>
<td>Uses of <code>org.quartz.locality.JobPersistenceException</code> in <code>org.quartz.locality</code></td>
</tr>
</tbody>
</table>

| Subclasses of `org.quartz.locality.JobPersistenceException` in `org.quartz.locality` | |

| Uses of `org.quartz.locality.JobPersistenceException` in `org.quartz.locality` | |

| Subclasses of `org.quartz.locality.JobPersistenceException` in `org.quartz.locality` | |
### Uses of **JobPersistenceException** in **org.quartz.simpl**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.simpl</strong> that throw <strong>JobPersistenceException</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>RAMJobStore.clearAllSchedulingData()</strong></td>
</tr>
<tr>
<td>Clear (delete!) all scheduling data - all Jobs, Triggers</td>
</tr>
<tr>
<td>Set <strong>RAMJobStore.pausedTriggerGroups()</strong></td>
</tr>
<tr>
<td>Trigger.TriggerState <strong>RAMJobStore.getTriggerState(TriggerKey triggerKey)</strong></td>
</tr>
<tr>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td>boolean <strong>RAMJobStore.removeCalendar(String calName)</strong></td>
</tr>
<tr>
<td>Remove (delete) the calendar with the given name.</td>
</tr>
<tr>
<td>boolean <strong>RAMJobStore.removeJobs(List&lt;JobKey&gt; jobKeys)</strong></td>
</tr>
<tr>
<td>boolean <strong>RAMJobStore.removeTriggers(List&lt;TriggerKey&gt; triggerKeys)</strong></td>
</tr>
<tr>
<td>boolean <strong>RAMJobStore.replaceTrigger(TriggerKey triggerKey, org.quartz.spi.OperableTrigger newTrigger)</strong></td>
</tr>
<tr>
<td>void <strong>RAMJobStore.storeJobAndTrigger(JobDetail newJob, org.quartz.spi.OperableTrigger newTrigger)</strong></td>
</tr>
<tr>
<td>Store the given JobDetail and Trigger.</td>
</tr>
<tr>
<td>void <strong>RAMJobStore.storeJobsAndTriggers(Map&lt;JobDetail, List&gt; replace)</strong></td>
</tr>
<tr>
<td>void <strong>RAMJobStore.storeTrigger(org.quartz.spi.OperableTrigger newTrigger, boolean replaceExisting)</strong></td>
</tr>
<tr>
<td>Store the given Trigger.</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Interface

```
org.quartz.ListenerManager
```

### Packages that use **ListenerManager**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz</code></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><code>org.quartz.core</code></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><code>org.quartz.impl</code></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of **ListenerManager** in **org.quartz**

### Methods in **org.quartz** that return **ListenerManager**

```
Scheduler.getList()  // Get a reference to the scheduler's ListenerManager, through which listeners may be registered.
```

### Uses of **ListenerManager** in **org.quartz.core**

### Classes in **org.quartz.core** that implement **ListenerManager**

```
class ListenerManagerImpl
```

### Methods in **org.quartz.core** that return **ListenerManager**

```
QuartzScheduler.getList()  // Get a reference to the QuartzScheduler's ListenerManager. 
```
## Uses of `ListenerManager` in `org.quartz.impl`

### Methods in `org.quartz.impl` that return `ListenerManager`

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoteScheduler.<code>getListenerManager()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>StdScheduler.<code>getListenerManager()</code></td>
<td></td>
</tr>
<tr>
<td>RemoteMBeanScheduler.<code>getListenerManager()</code></td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, [Terracotta, Inc.](https://www.terracotta.com/)
# Uses of Interface
**org.quartz.Matcher**

## Packages that use Matcher

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.impl.matchers</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of Matcher in org.quartz

### Methods in org.quartz that return types with arguments of type Matcher

- `List<Matcher<JobKey>> ListenerManager.getJobListenerMatchers(String listenerName)`
  - Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

- `List<Matcher<TriggerKey>> ListenerManager.getTriggerListenerMatchers(String listenerName)`
  - Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

### Methods in org.quartz with parameters of type Matcher

- `void ListenerManager.addJobListener(JobListener jobListener, Matcher<JobKey>... matchers)`
  - Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers.

- `boolean ListenerManager.addJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)`
  - Add the given Matcher to the set of matchers for which the listener receive events if ANY of the matchers match.

- `void ListenerManager.addTriggerListener(TriggerListener triggerListener, TriggerKey... triggerKeys)`
  - Add the given TriggerListener to the Scheduler, and register it to receive events if ANY of the trigger keys match.
Add the given TriggerListener to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers.

```java
ListenerManager.addTriggerListenerMatcher(String listenerName, Matcher<TriggerKey> matcher)
```

Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

```java
ListenerManager.addJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)
```

Add the given JobListener to the Scheduler, and register it to receive events for Jobs that are matched by ANY of the given Matchers.

```java
ListenerManager.addJobListener(JobListener jobListener, List<Matcher<JobKey>> matchers)
```

Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

```java
ListenerManager.setJobListenerMatchers(String listenerName, List<Matcher<JobKey>> matchers)
```

Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

```java
ListenerManager.setTriggerListenerMatchers(String listenerName, List<Matcher<TriggerKey>> matchers)
```

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

```java
ListenerManager.removeJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)
```

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

```java
ListenerManager.removeTriggerListenerMatcher(String listenerName, Matcher<TriggerKey> matcher)
```

Remove the given TriggerListener to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers.

```java
ListenerManager.addTriggerListener(TriggerListener triggerListener, List<Matcher<TriggerKey>> matchers)
```

Uses of `Matcher` in `org.quartz.core`
**Methods in** `org.quartz.core` **that return types with arguments of type** `Matcher`  

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>List&lt;Matcher&lt;JobKey&gt;&gt;</code></td>
<td><code>ListenerManagerImpl.getJobListenerMatchers(String)</code></td>
</tr>
<tr>
<td><code>List&lt;Matcher&lt;TriggerKey&gt;&gt;</code></td>
<td><code>ListenerManagerImpl.getTriggerListenerMatchers()</code></td>
</tr>
</tbody>
</table>

**Methods in** `org.quartz.core` **with parameters of type** `Matcher`  

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>ListenerManagerImpl.addJobListener(JobListener, Matcher&lt;JobKey&gt;... matchers)</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>ListenerManagerImpl.addJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>ListenerManagerImpl.addTriggerListener(TriggerListener, Matcher&lt;TriggerKey&gt;... matchers)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>ListenerManagerImpl.addTriggerListener(TriggerListener, Matcher&lt;TriggerKey&gt; matcher)</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>ListenerManagerImpl.addTriggerListenerMatcher(String listenerName, Matcher&lt;TriggerKey&gt; matcher)</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>ListenerManagerImpl.removeJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>ListenerManagerImpl.removeTriggerListenerMatcher(String listenerName, Matcher&lt;TriggerKey&gt; matcher)</code></td>
</tr>
</tbody>
</table>

**Method parameters in** `org.quartz.core` **with type arguments of type** `Matcher`  

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>ListenerManagerImpl.addJobListener(JobListener, List&lt;Matcher&lt;JobKey&gt;&gt; matchers)</code></td>
</tr>
</tbody>
</table>
```java
void ListenerManagerImpl.addTriggerListener(TriggerListener triggerListener, List<Matcher<TriggerKey>> matchers)

boolean ListenerManagerImpl.setJobListenerMatchers(String listenerName, List<Matcher<JobKey>> matchers)

boolean ListenerManagerImpl.setTriggerListenerMatchers(String listenerName, List<Matcher<TriggerKey>> matchers)
```

**Uses of **Matcher in org.quartz.impl.matchers

**Classes in org.quartz.impl.matchers that implement Matcher**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndMatcher&lt;T extends Key&gt;</td>
<td>Matches using an AND operator on two Matcher operands.</td>
</tr>
<tr>
<td>EverythingMatcher&lt;T extends Key&gt;</td>
<td>Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td>GroupMatcher&lt;T extends Key&gt;</td>
<td>Matches on group (ignores name) property of Keys.</td>
</tr>
<tr>
<td>KeyMatcher&lt;T extends Key&gt;</td>
<td>Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td>NameMatcher&lt;T extends Key&gt;</td>
<td>Matches on name (ignores group) property of Keys.</td>
</tr>
<tr>
<td>NotMatcher&lt;T extends Key&gt;</td>
<td>Matches using an NOT operator on another Matcher.</td>
</tr>
<tr>
<td>OrMatcher&lt;T extends Key&gt;</td>
<td>Matches using an OR operator on two Matcher operands.</td>
</tr>
<tr>
<td>StringMatcher&lt;T extends Key&gt;</td>
<td>An abstract base class for some types of matchers.</td>
</tr>
</tbody>
</table>

**Fields in org.quartz.impl.matchers declared as Matcher**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Matcher&lt;T&gt; AndMatcher.leftOperand</td>
<td></td>
</tr>
</tbody>
</table>
protected Matcher<T> OrMatcher.leftOperand

protected Matcher<T> NotMatcher.operand

protected Matcher<T> AndMatcher.rightOperand

protected Matcher<T> OrMatcher.rightOperand

Methods in org.quartz.impl.matchers that return Matcher

| Matcher<T> | AndMatcher.getLeftOperand() |
| Matcher<T> | OrMatcher.getLeftOperand() |
| Matcher<T> | NotMatcher.getOperand() |
| Matcher<T> | AndMatcher.getRightOperand() |
| Matcher<T> | OrMatcher.getRightOperand() |

Methods in org.quartz.impl.matchers with parameters of type Matcher

static <U extends Key> AndMatcher<T>

AndMatcher.and(Matcher<U> leftOperand, Matcher<U> rightOperand)

Create an AndMatcher that depends upon the result of both of the given matchers.
matchers.

static
<U extends Key>
NotMatcher<U>

NotMatcher.<u>not</u>(Matcher<U> operand)
Create a NotMatcher that reverses the result of the given matcher.

static
<U extends Key>
OrMatcher<U>

OrMatcher.<u>or</u>(Matcher<U> leftOperand, Matcher<U> rightOperand)
Create an OrMatcher that depends upon the result of at least one of the given matchers.

Constructors in org.quartz.impl.matchers with parameters of type Matcher

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndMatcher&lt;Matcher&lt;T&gt; leftOperand, Matcher&lt;T&gt; rightOperand)</td>
<td></td>
</tr>
<tr>
<td>AndMatcher&lt;Matcher&lt;T&gt; leftOperand, Matcher&lt;T&gt; rightOperand)</td>
<td></td>
</tr>
<tr>
<td>NotMatcher&lt;Matcher&lt;T&gt; operand)</td>
<td></td>
</tr>
<tr>
<td>OrMatcher&lt;Matcher&lt;T&gt; leftOperand, Matcher&lt;T&gt; rightOperand)</td>
<td></td>
</tr>
<tr>
<td>OrMatcher&lt;Matcher&lt;T&gt; leftOperand, Matcher&lt;T&gt; rightOperand)</td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.ObjectAlreadyExistsException**

## Packages that use **ObjectAlreadyExistsException**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of **ObjectAlreadyExistsException** in

**org.quartz.impl.jdbcjobstore**

## Methods in **org.quartz.impl.jdbcjobstore** that throw **ObjectAlreadyExistsException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void JobStoreSupport.storeCalendar(...)</td>
<td></td>
</tr>
<tr>
<td>void JobStoreSupport.storeCalendar(...)</td>
<td></td>
</tr>
<tr>
<td>protected void JobStoreSupport.storeJob(...)</td>
<td></td>
</tr>
<tr>
<td>void JobStoreSupport.storeJob(...)</td>
<td></td>
</tr>
<tr>
<td>void JobStoreSupport.storeJobAndTrigger(...)</td>
<td></td>
</tr>
<tr>
<td>void JobStoreSupport.storeJobsAndTriggers(...)</td>
<td></td>
</tr>
</tbody>
</table>
protected void JobStoreSupport.storeTrigger(Connection conn, org.quartz.spi.OperableTrigger newTrigger, JobDetail job, String state, boolean forceState, boolean recovering)
    Insert or update a trigger.

void JobStoreSupport.storeTrigger(org.quartz.spi.OperableTrigger newTrigger, boolean replaceExisting)
    Store the given Trigger.

Uses of **ObjectAlreadyExistsException** in org.quartz.simpl

### Methods in org.quartz.simpl that throw **ObjectAlreadyExistsException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>RAMJobStore.storeCalendar(String name, Calendar calendar, boolean replaceExisting, boolean updateTriggers)</td>
</tr>
<tr>
<td></td>
<td>Store the given Calendar.</td>
</tr>
<tr>
<td>void</td>
<td>RAMJobStore.storeJob(JobDetail newJob, boolean replaceExisting)</td>
</tr>
<tr>
<td></td>
<td>Store the given Job.</td>
</tr>
<tr>
<td>void</td>
<td>RAMJobStore.storeJobsAndTriggers(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; triggersAndJobs, boolean replace)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.PersistJobDataAfterExecution

Packages that use PersistJobDataAfterExecution

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.jobs</td>
<td></td>
</tr>
</tbody>
</table>

Uses of PersistJobDataAfterExecution in org.quartz

Classes in org.quartz with annotations of type PersistJobDataAfterExecution

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatefulJob</td>
<td>Deprecated. use DisallowConcurrentExecution and/or PersistJobDataAfterExecution annotations instead.</td>
</tr>
</tbody>
</table>

Uses of PersistJobDataAfterExecution in org.quartz.jobs

Classes in org.quartz.jobs with annotations of type PersistJobDataAfterExecution

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectoryScanJob</td>
<td>Inspects a directory and compares whether any files' &quot;last modified dates&quot; have changed since the last time it was inspected.</td>
</tr>
<tr>
<td>FileScanJob</td>
<td>Inspects a file and compares whether it's &quot;last modified date&quot; has changed since the last time it was inspected.</td>
</tr>
</tbody>
</table>
## Uses of Class org.quartz.ScheduleBuilder

### Packages that use ScheduleBuilder

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of ScheduleBuilder in org.quartz

### Subclasses of ScheduleBuilder in org.quartz

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalendarIntervalScheduleBuilder</td>
<td>CalendarIntervalScheduleBuilder is a ScheduleBuilder that defines calendar time (day, week, month, year) interval-based schedules for Triggers.</td>
</tr>
<tr>
<td>CronScheduleBuilder</td>
<td>CronScheduleBuilder is a ScheduleBuilder that defines CronExpression-based schedules for Triggers.</td>
</tr>
<tr>
<td>SimpleScheduleBuilder</td>
<td>SimpleScheduleBuilder is a ScheduleBuilder that defines strict/literal interval-based schedules for Triggers.</td>
</tr>
</tbody>
</table>

### Methods in org.quartz that return ScheduleBuilder

<table>
<thead>
<tr>
<th><code>ScheduleBuilder&lt;? extends Trigger&gt;</code></th>
<th><code>Trigger.getScheduleBuilder()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.</td>
</tr>
</tbody>
</table>
Methods in org.quartz with parameters of type ScheduleBuilder

TriggerBuilder.withSchedule(ScheduleBuilder<SBT> scheduleBuilder)

Set the ScheduleBuilder that will be used to define the Trigger's schedule.

Uses of ScheduleBuilder in org.quartz.impl.jdbcjobstore

Methods in org.quartz.impl.jdbcjobstore that return ScheduleBuilder

ScheduleBuilder TriggerPersistenceDelegate.TriggerPropertyBundle.getScheduleBuilder

Constructors in org.quartz.impl.jdbcjobstore with parameters of type ScheduleBuilder

TriggerPersistenceDelegate.TriggerPropertyBundle(ScheduleBuilder sb, String[] statePropertyNames, Object[] statePropertyValues)

Uses of ScheduleBuilder in org.quartz.impl.triggers

Methods in org.quartz.impl.triggers that return ScheduleBuilder

abstract ScheduleBuilder<SBT> AbstractTrigger.getScheduleBuilder()

ScheduleBuilder<CronTrigger> CronTriggerImpl.getScheduleBuilder()

Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's.

ScheduleBuilder<CalendarIntervalTrigger> CalendarIntervalTriggerImpl.getScheduleBuilder()
Get a `ScheduleBuilder` that is configured to produce a schedule identical to this trigger's schedule.

```java
SimpleTriggerImpl.getScheduleBuilder
```

Get a `ScheduleBuilder` that is configured to produce a schedule identical to this trigger's schedule.

<table>
<thead>
<tr>
<th>Uses of <strong>ScheduleBuilder</strong> in <strong>org.quartz.locality</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.locality</strong> that return <strong>ScheduleBuilder</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ScheduleBuilder&lt;Trigger&gt;</code></td>
<td>DelegatingLocalityTrigger.getScheduleBuilder()</td>
</tr>
<tr>
<td>Get a <code>ScheduleBuilder</code> that is configured to produce a schedule identical to this trigger's schedule.</td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
## Uses of Interface

**org.quartz.Scheduler**

### Packages that use **Scheduler**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.ee.jmx.jboss</td>
<td></td>
</tr>
<tr>
<td>org.quartz.ee.jta</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.plugins</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.history</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.management</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.xml</td>
<td></td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.xml</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of **Scheduler** in **org.quartz**

### Methods in **org.quartz** that return **Scheduler**

- **Scheduler**
  - `SchedulerFactory.getScheduler()`
    - Returns a client-usuable handle to a Scheduler.
- **Scheduler**
  - `JobExecutionContext.getScheduler()`
Get a handle to the Scheduler instance that fired the Job.

<table>
<thead>
<tr>
<th>SchedulerFactory</th>
<th>getScheduler(String schedName)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a handle to the Scheduler with the given name, if it exists.</td>
<td></td>
</tr>
</tbody>
</table>

Methods in `org.quartz` that return types with arguments of type `Scheduler`

<table>
<thead>
<tr>
<th>Collection&lt;Scheduler&gt;</th>
<th>SchedulerFactory. getAllSchedulers()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns handles to all known Schedulers (made by any SchedulerFactory within this jvm.).</td>
<td></td>
</tr>
</tbody>
</table>

Uses of `Scheduler` in `org.quartz.core`

Fields in `org.quartz.core` declared as `Scheduler`

<table>
<thead>
<tr>
<th>protected Scheduler</th>
<th>JobRunShell. scheduler</th>
</tr>
</thead>
</table>

Methods in `org.quartz.core` with parameters of type `Scheduler`

<table>
<thead>
<tr>
<th>void</th>
<th>JobRunShellFactory. initialize(Scheduler scheduler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContext s within it.</td>
<td></td>
</tr>
</tbody>
</table>

Constructors in `org.quartz.core` with parameters of type `Scheduler`

| JobRunShell(Scheduler scheduler, org.quartz.spi.TriggerFiredBundle bndle) |
| Create a JobRunShell instance with the given settings. |

Uses of `Scheduler` in `org.quartz.ee.jmx.jboss`
### Classes in `org.quartz.ee.jmx.jboss` that implement `Scheduler`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBoss4RMIRemoteMBeanScheduler</td>
<td>An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JBoss's JMX RMIAdaptor.</td>
</tr>
</tbody>
</table>

### Uses of `Scheduler` in `org.quartz.ee.jta`

### Methods in `org.quartz.ee.jta` with parameters of type `Scheduler`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void JTAAnnotationAwareJobRunShellFactory.initialize(Scheduler scheduler)</code></td>
<td>Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.</td>
</tr>
<tr>
<td><code>void JTAJobRunShellFactory.initialize(Scheduler scheduler)</code></td>
<td>Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.</td>
</tr>
</tbody>
</table>

### Constructors in `org.quartz.ee.jta` with parameters of type `Scheduler`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JTAJobRunShell(Scheduler scheduler, org.quartz.spi.TriggerFiredBundle bndle)</code></td>
<td>Create a JTAJobRunShell instance with the given settings.</td>
</tr>
</tbody>
</table>

### Uses of `Scheduler` in `org.quartz.impl`

### Classes in `org.quartz.impl` that implement `Scheduler`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoteMBeanScheduler</td>
<td>An implementation of the Scheduler interface that remotely</td>
</tr>
</tbody>
</table>
proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JMX.

<table>
<thead>
<tr>
<th>class</th>
<th>RemoteScheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via RMI.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
<th>StdScheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An implementation of the Scheduler interface that directly proxies all method calls to the equivalent call on a given QuartzScheduler instance.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl` that return `Scheduler`

<table>
<thead>
<tr>
<th>static <code>Scheduler</code></th>
<th><code>StdSchedulerFactory.getDefaultScheduler()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a handle to the default Scheduler, creating it if it does not yet exist.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Scheduler</code></th>
<th><code>StdSchedulerFactory.getScheduler()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a handle to the Scheduler produced by this factory.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Scheduler</code></th>
<th><code>DirectSchedulerFactory.getScheduler()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a handle to the Scheduler produced by this factory.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Scheduler</code></th>
<th><code>JobExecutionContextImpl.getScheduler()</code></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><code>Scheduler</code></th>
<th><code>StdSchedulerFactory.getScheduler(String schedName)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a handle to the Scheduler with the given name, if it has already been instantiated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>Scheduler</code></th>
<th><code>DirectSchedulerFactory.getScheduler(String schedName)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a handle to the Scheduler with the given name.</td>
</tr>
</tbody>
</table>

| protected `Scheduler` | `StdSchedulerFactory.instantiate(QuartzSchedulerResources quartzScheduler)` |

<table>
<thead>
<tr>
<th><code>Scheduler</code></th>
<th><code>SchedulerRepository.lookup(String schedName)</code></th>
</tr>
</thead>
</table>

### Methods in `org.quartz.impl` that return types with arguments of type `Scheduler`
<table>
<thead>
<tr>
<th>Collection&lt;Scheduler&gt;</th>
<th>StdSchedulerFactory.</th>
<th>getAllSchedulers()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Returns a handle to all known Schedulers (made by any StdSchedulerFactory instance.).</td>
</tr>
<tr>
<td>Collection&lt;Scheduler&gt;</td>
<td>SchedulerRepository.</td>
<td>lookupAll()</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl` with parameters of type `Scheduler`

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>SchedulerRepository.bind</td>
<td>(Scheduler sched)</td>
</tr>
<tr>
<td>void</td>
<td>StdJobRunShellFactory.initialize</td>
<td>(Scheduler scheduler)</td>
</tr>
</tbody>
</table>

Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it.

### Constructors in `org.quartz.impl` with parameters of type `Scheduler`

<table>
<thead>
<tr>
<th>Class</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobExecutionContextImpl</td>
<td>(Scheduler scheduler, org.quartz.spi.TriggerFiredBundle firedBundle, Job job)</td>
</tr>
</tbody>
</table>

Create a JobExecutionContext with the given context data.

### Uses of `Scheduler` in `org.quartz.plugins`

### Methods in `org.quartz.plugins` that return `Scheduler`

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Scheduler</td>
<td>SchedulerPluginWithUserTransactionSupport.getScheduler()</td>
<td></td>
</tr>
</tbody>
</table>

Get this plugin's Scheduler.

### Methods in `org.quartz.plugins` with parameters of type `Scheduler`

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>SchedulerPluginWithUserTransactionSupport.initialize</td>
<td>(String name, Scheduler scheduler)</td>
</tr>
</tbody>
</table>


Uses of **Scheduler** in **org.quartz.plugins.history**

Methods in **org.quartz.plugins.history** with parameters of type **Scheduler**

```java
void LoggingJobHistoryPlugin.initialize(String name, Scheduler scheduler)
   Called during creation of the Scheduler in order to give the
   SchedulerPlugin a chance to initialize.
```

```java
void LoggingTriggerHistoryPlugin.initialize(String name, Scheduler scheduler)
   Called during creation of the Scheduler in order to give the
   SchedulerPlugin a chance to initialize.
```

Uses of **Scheduler** in **org.quartz.plugins.management**

Methods in **org.quartz.plugins.management** with parameters of type **Scheduler**

```java
void ShutdownHookPlugin.initialize(String name, Scheduler scheduler)
   Called during creation of the Scheduler in order to give the
   SchedulerPlugin a chance to initialize.
```

Uses of **Scheduler** in **org.quartz.plugins.xml**

Methods in **org.quartz.plugins.xml** with parameters of type **Scheduler**

```java
void XMLSchedulingDataProcessorPlugin.initialize(String name, Scheduler scheduler)
   Called during creation of the Scheduler in order to give the
   SchedulerPlugin a chance to initialize.
```
### Uses of **Scheduler** in **org.quartz.simpl**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.simpl</strong> with parameters of type <strong>Scheduler</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Job</strong> PropertySettingJobFactory. <strong>newJob</strong>(org.quartz.spi.TriggerFiredBundle bundle, <strong>Scheduler</strong> scheduler)</td>
</tr>
<tr>
<td><strong>Job</strong> SimpleJobFactory. <strong>newJob</strong>(org.quartz.spi.TriggerFiredBundle bundle, <strong>Scheduler</strong> Scheduler)</td>
</tr>
</tbody>
</table>

### Uses of **Scheduler** in **org.quartz.xml**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.xml</strong> with parameters of type <strong>Scheduler</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protected void</strong> XMLSchedulingDataProcessor. <strong>executePreProcessCommands</strong>( <strong>Scheduler</strong> sched)</td>
</tr>
<tr>
<td><strong>void</strong> XMLSchedulingDataProcessor. <strong>processFileAndScheduleJobs</strong>( <strong>Scheduler</strong> sched, boolean overwriteExistingJobs)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>void</strong> XMLSchedulingDataProcessor. <strong>processFileAndScheduleJobs</strong>(String systemId, <strong>Scheduler</strong> sched)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>void</strong> XMLSchedulingDataProcessor. <strong>processFileAndScheduleJobs</strong>(String fileName, <strong>Scheduler</strong> sched)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>protected void</strong> XMLSchedulingDataProcessor. <strong>scheduleJobs</strong>( <strong>Scheduler</strong> sched)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Uses of SchedulerConfigException in org.quartz.core</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Methods in org.quartz.core that throw SchedulerConfigException</td>
</tr>
<tr>
<td>Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContext within it.</td>
</tr>
</tbody>
</table>

Packages that use SchedulerConfigException:

- org.quartz.core
- org.quartz.impl.jdbcjobstore
- org.quartz.simpl
- org.quartz.core
- org.quartz.ee.jta

Methods in org.quartz.core that throw SchedulerConfigException:

- void JobRunShellFactory.initialize(Scheduler scheduler)

Methods in org.quartz.core that throw SchedulerConfigException:

- void JTAAnnotationAwareJobRunShellFactory.initialize(Scheduler scheduler)
void JTAJobRunShellFactory.\texttt{initialize}(Scheduler\ scheduler)\n
Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContext within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.

\textbf{Uses of} \texttt{SchedulerConfigException} \textbf{in} \texttt{org.quartz.impl.jdbcjobstore}

\textbf{Methods in} \texttt{org.quartz.impl.jdbcjobstore} \textbf{that throw} \texttt{SchedulerConfigException}

void JobStoreSupport.\texttt{initialize}(org.quartz.spi.ClassLoadHelper\ loadHelper,
                                          org.quartz.spi.SchedulerSignaler\ signaler)\n
Called by the QuartzScheduler before the JobStore is used, in order to give it a chance to initialize.

void JobStoreTX.\texttt{initialize}(org.quartz.spi.ClassLoadHelper\ loadHelper,
                                   org.quartz.spi.SchedulerSignaler\ signaler)\n
void JobStoreCMT.\texttt{initialize}(org.quartz.spi.ClassLoadHelper\ loadHelper,
                                   org.quartz.spi.SchedulerSignaler\ signaler)\n
\textbf{Uses of} \texttt{SchedulerConfigException} \textbf{in} \texttt{org.quartz.simpl}

\textbf{Methods in} \texttt{org.quartz.simpl} \textbf{that throw} \texttt{SchedulerConfigException}

void SimpleTimeBroker.\texttt{initialize}()\n
void ZeroSizeThreadPool.\texttt{initialize}()
void SimpleThreadPool.initialize()
# Uses of Class

**org.quartz.SchedulerContext**

## Packages that use **SchedulerContext**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Uses of **SchedulerContext** in **org.quartz**

## Methods in **org.quartz** that return **SchedulerContext**

<table>
<thead>
<tr>
<th>SchedulerContext</th>
<th>Methods in <code>org.quartz</code> that return <strong>SchedulerContext</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduler.getContext()</td>
<td>Scheduler.getContext() Returns the SchedulerContext of the Scheduler.</td>
</tr>
</tbody>
</table>

## Uses of **SchedulerContext** in **org.quartz.core**

## Methods in **org.quartz.core** that return **SchedulerContext**

<table>
<thead>
<tr>
<th>SchedulerContext</th>
<th>Methods in <code>org.quartz.core</code> that return <strong>SchedulerContext</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzScheduler.getContext()</td>
<td>QuartzScheduler.getContext() Returns the SchedulerContext of the Scheduler.</td>
</tr>
<tr>
<td>SchedulerContext</td>
<td>QuartzScheduler.getContext()()</td>
</tr>
</tbody>
</table>

## Uses of **SchedulerContext** in **org.quartz.impl**

<table>
<thead>
<tr>
<th>SchedulerContext</th>
<th>Methods in <code>org.quartz.impl</code> that return <strong>SchedulerContext</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoteableQuartzScheduler</td>
<td>RemoteableQuartzScheduler.getContext()()</td>
</tr>
</tbody>
</table>
### Methods in `org.quartz.impl` that return `SchedulerContext`

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SchedulerContext</code></td>
<td><code>StdScheduler.getContext()</code></td>
<td>Returns the SchedulerContext of the Scheduler.</td>
</tr>
</tbody>
</table>
# Uses of Class

**org.quartz.SchedulerException**

<table>
<thead>
<tr>
<th>Packages that use <strong>SchedulerException</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.core.jmx</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.ee.jmx.jboss</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.ee.jta</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.triggers</strong></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td><strong>org.quartz.jobs.ee.jms</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.listeners</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.plugins</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.plugins.history</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.plugins.management</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.plugins.xml</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.simpl</strong></td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.utils</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.xml</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Uses of `SchedulerException` in `org.quartz`

### Subclasses of `SchedulerException` in `org.quartz`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobExecutionException</code></td>
<td>An exception that can be thrown by a Job to indicate to the Quartz Scheduler that an error occurred while executing, and whether or not the Job requests to be re-fired immediately (using the same <code>JobExecutionContext</code>, or whether it wants to be unscheduled.</td>
</tr>
<tr>
<td><code>JobPersistenceException</code></td>
<td>An exception that is thrown to indicate that there has been a failure in the scheduler's underlying persistence mechanism.</td>
</tr>
<tr>
<td><code>ObjectAlreadyExistsException</code></td>
<td>An exception that is thrown to indicate that an attempt to store a new object (i.e.</td>
</tr>
<tr>
<td><code>SchedulerConfigException</code></td>
<td>An exception that is thrown to indicate that there is a misconfiguration of the <code>SchedulerFactory</code>- or one of the components it configures.</td>
</tr>
<tr>
<td><code>UnableToInterruptJobException</code></td>
<td>An exception that is thrown to indicate that a call to <code>InterruptableJob.interrupt()</code> failed without interrupting the Job.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz` with parameters of type `SchedulerException`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SchedulerListener.scheduledError(String msg, SchedulerException cause)</code></td>
<td>Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the <code>JobStore</code>, or the inability to instantiate a Job instance when its <code>Trigger</code> has fired.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz` that throw `SchedulerException`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Scheduler.addCalendar(String calName, Calendar calendar)</code></td>
<td></td>
</tr>
</tbody>
</table>
boolean replace, boolean updateTriggers)

Add (register) the given calendar to the scheduler.

void Scheduler.addJob(JobDetail jobDetail, boolean replace)

Add the given job to the scheduler - with no associated trigger.

boolean Scheduler.checkExists(JobKey jobKey)

Determine whether a job with the given identifier already exists within the scheduler.

boolean Scheduler.checkExists(TriggerKey triggerKey)

Determine whether a trigger with the given identifier already exists within the scheduler.

void Scheduler.clear()

Clears (deletes!) all scheduling data - all jobs, triggers.

boolean Scheduler.deleteCalendar(String calName)

Delete the identified calendar from the scheduler.

boolean Scheduler.deleteJob(JobKey jobKey)

Delete the identified job from the scheduler - and any associated triggers.

boolean Scheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)

Delete the identified jobs from the scheduler - and any associated triggers.

Collection&lt;Scheduler&gt; SchedulerFactory.getAllSchedulers()

Returns handles to all known schedulers (made by any scheduler factory within this jvm.).

Calendar Scheduler.getCalendar(String calName)

Get the calendar instance with the given name.

List&lt;String&gt; Scheduler.getCalendarNames()

Get the names of all registered calendars.

SchedulerContext Scheduler.getContext()

Returns the scheduler context of the scheduler.

List&lt;JobExecutionContext&gt; Scheduler.getCurrentlyExecutingJobs()

Return a list of job execution contexts that are currently executing jobs in this scheduler instance.

JobDetail Scheduler.getJobDetail(JobKey jobKey)

Get the job detail for the job instance with the given key.

List&lt;String&gt; Scheduler.getJobGroupNames()
Get the names of all known JobDetail groups.

```java
Set<JobKey> Scheduler.getJobKeys(GroupMatcher<JobKey> matcher)
```
Get the keys of all the JobDetails in the matching groups.

`ListenerManager`  
Get a reference to the scheduler's ListenerManager; listeners may be registered.

```java
Scheduler.getListenerManager()
```

`SchedulerMetaData`  
Get a `SchedulerMetaData` object describing the state of the scheduler instance.

```java
Scheduler.getMetaData()
```

Get the keys of all the JobDetails in the matching groups.

```java
Set<String> Scheduler.getPausedTriggerGroups()
```
Get the names of all Trigger groups that are paused.

```java
SchedulerFactory.getScheduler()
```
Returns a client-usable handle to a Scheduler.

```java
SchedulerFactory.getScheduler(String schedName)
```
Returns a handle to the Scheduler with the given name, if it exists.

```java
String Scheduler.getSchedulerInstanceId()
```
Returns the instance Id of the Scheduler.

```java
String Scheduler.getSchedulerName()
```
Returns the name of the Scheduler.

```java
SchedulerMetaData.getSummary()
```
Returns a formatted (human readable) String describing the Scheduler's meta-data values.

```java
Trigger Scheduler.getTrigger(TriggerKey triggerKey)
```
Get the Trigger instance with the given key.

```java
List<String> Scheduler.getTriggerGroupNames()
```
Get the names of all known Trigger groups.

```java
Set<TriggerKey> Scheduler.getTriggerKeys(GroupMatcher<TriggerKey> matcher)
```
Get the names of all the Triggers in the matching groups.

```java
List<? extends Trigger> Scheduler.getTriggersOfJob(JobKey jobKey)
```
Get all Triggers that are associated with the identified Job.

```java
Trigger.TriggerState Scheduler.getTriggerState(TriggerKey triggerKey)
```
Get the current state of the identified Trigger.

```java
boolean Scheduler.isInStandbyMode()
```
Reports whether the Scheduler is in stand-by mode.
Scheduler.isShutdown()
Reports whether the Scheduler has been shutdown.

boolean Scheduler.isStarted()
Whether the scheduler has been started.

void Scheduler.pauseAll()
Pause all triggers - similar to calling pauseTriggerGroup,
however, after using this method resumeAll() must
be called to clear the scheduler's state of 'remembering' that all new triggers were paused.

void Scheduler.pauseJob(JobKey jobKey)
Pause the JobDetail with the given key - by pausing all of its Triggers.

void Scheduler.pauseJobs(GroupMatcher<JobKey> matcher)
Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.

void Scheduler.pauseTrigger(TriggerKey triggerKey)
Pause the Trigger with the given key.

void Scheduler.pauseTriggers(GroupMatcher<TriggerKey> matcher)
Pause all of the Triggers in the groups matching.

Date Scheduler.rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)
Remove (delete) the Trigger with the given key, and replace it with a new one - which must be associated with the same job (the new Trigger must specify the same job name & group) - however, the new Trigger's name need not be the same as the old trigger.

void Scheduler.resumeAll()
Resume (un-pause) all triggers - similar to calling resumeTriggerGroup(group) on every group.

void Scheduler.resumeJob(JobKey jobKey)
Resume (un-pause) the JobDetail with the given key.

void Scheduler.resumeJobs(GroupMatcher<JobKey> matcher)
Resume (un-pause) all of the JobDetails in matching groups.

void Scheduler.resumeTrigger(TriggerKey triggerKey)
Resume (un-pause) the Trigger with the given key.

void Scheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)
Resume (un-pause) all of the Triggers in matching groups.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Scheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</code></td>
<td>Add the given JobDetail to the Scheduler, and associate the given Trigger with it.</td>
</tr>
<tr>
<td><code>Scheduler.scheduleJob(Trigger trigger)</code></td>
<td>Schedule the given Trigger with the Job identified by the settings.</td>
</tr>
<tr>
<td><code>void Scheduler.scheduleJobs(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; jobs, boolean replace)</code></td>
<td>Schedule all of the given jobs with the related set of triggers.</td>
</tr>
<tr>
<td><code>void Scheduler.setJobFactory(org.quartz.spi.JobFactory factory)</code></td>
<td>Set the JobFactory that will be responsible for producing classes.</td>
</tr>
<tr>
<td><code>void Scheduler.shutdown()</code></td>
<td>Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler.</td>
</tr>
<tr>
<td><code>void Scheduler.shutdown(boolean waitForJobsToComplete)</code></td>
<td>Halts the Scheduler's firing of Triggers, and cleans up all resources associated with the Scheduler.</td>
</tr>
<tr>
<td><code>void Scheduler.standby()</code></td>
<td>Temporarily halts the Scheduler's firing of Triggers.</td>
</tr>
<tr>
<td><code>void Scheduler.start()</code></td>
<td>Starts the Scheduler's threads that fire Triggers.</td>
</tr>
<tr>
<td><code>void Scheduler.startDelayed(int seconds)</code></td>
<td>Calls <code>#start()</code> after the indicated number of seconds.</td>
</tr>
<tr>
<td><code>void Scheduler.triggerJob(JobKey jobKey)</code></td>
<td>Trigger the identified JobDetail (execute it now).</td>
</tr>
<tr>
<td><code>void Scheduler.triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td>Trigger the identified JobDetail (execute it now).</td>
</tr>
<tr>
<td><code>boolean Scheduler.unscheduleJob(TriggerKey triggerKey)</code></td>
<td>Remove the indicated Trigger from the scheduler.</td>
</tr>
<tr>
<td><code>boolean Scheduler.unscheduleJobs(List&lt;TriggerKey&gt; triggerKeys)</code></td>
<td>Remove all of the indicated Triggers from the scheduler.</td>
</tr>
</tbody>
</table>

**Uses of** **SchedulerException** **in** org.quartz.core
### Methods in `org.quartz.core` with parameters of type `SchedulerException`

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzScheduler.notifySchedulerListenersError</td>
<td><code>void QuartzScheduler.notifySchedulerListenersError(String msg, SchedulerException se)</code></td>
</tr>
<tr>
<td>QuartzSchedulerMBEanImpl.schedulerError</td>
<td><code>void QuartzSchedulerMBEanImpl.schedulerError(String msg, SchedulerException cause)</code></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.core` that throw `SchedulerException`

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzScheduler.addCalendar</td>
<td><code>void QuartzScheduler.addCalendar(String calName, Calendar cal, boolean updateTriggers)</code></td>
</tr>
<tr>
<td>QuartzSchedulerMBEanImpl.addCalendar</td>
<td><code>void QuartzSchedulerMBEanImpl.addCalendar(String calName, boolean replace, boolean updateTriggers)</code></td>
</tr>
<tr>
<td>QuartzScheduler.addJob</td>
<td><code>void QuartzScheduler.addJob(JobDetail jobDetail, boolean replace)</code></td>
</tr>
<tr>
<td>QuartzSchedulerMBEanImpl.addJob</td>
<td><code>void QuartzSchedulerMBEanImpl.addJob(JobDetail jobDetail)</code></td>
</tr>
<tr>
<td>JobRunShell.begin</td>
<td><code>protected void JobRunShell.begin()</code></td>
</tr>
<tr>
<td>QuartzScheduler.checkExists</td>
<td><code>boolean QuartzScheduler.checkExists(JobKey jobKey)</code></td>
</tr>
<tr>
<td>QuartzSchedulerMBEanImpl.checkExists</td>
<td><code>boolean QuartzSchedulerMBEanImpl.checkExists(JobKey jobKey)</code></td>
</tr>
<tr>
<td>QuartzScheduler.checkExists</td>
<td><code>boolean QuartzScheduler.checkExists(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>QuartzSchedulerMBEanImpl.checkExists</td>
<td><code>boolean QuartzSchedulerMBEanImpl.checkExists(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>QuartzScheduler.clear</td>
<td><code>void QuartzScheduler.clear()</code></td>
</tr>
<tr>
<td>QuartzSchedulerMBEanImpl.clear</td>
<td><code>void QuartzSchedulerMBEanImpl.clear()</code></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>JobRunShell.complete</code></td>
<td>Called by the QuartzSchedulerThread to obtain information about a job.</td>
</tr>
<tr>
<td><code>JobRunShellFactory.createJobRunShell</code></td>
<td></td>
</tr>
<tr>
<td><code>QuartzScheduler.deleteCalendar(String calName)</code></td>
<td>Delete the identified Calendar from the Scheduler.</td>
</tr>
<tr>
<td><code>QuartzScheduler.deleteJob(JobKey jobKey)</code></td>
<td>Delete the identified Job from the Scheduler - and any future references.</td>
</tr>
<tr>
<td><code>QuartzScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</code></td>
<td>Delete the identified Jobs from the Scheduler.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getCalendar(String calName)</code></td>
<td>Get the Calendar instance with the given name.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getCalendarNames()</code></td>
<td>Get the names of all registered Calendars.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getCalendarNames()</code></td>
<td>Get the names of all registered Calendars.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getJobDetail(JobKey jobKey)</code></td>
<td>Get the JobDetail for the Job instance with the given name and Group.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getJobDetail(JobKey jobKey)</code></td>
<td>Get the JobDetail for the Job instance with the given name and Group.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getJobGroupNames()</code></td>
<td>Get the names of all known Job groups.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getJobGroupNames()</code></td>
<td>Get the names of all known Job groups.</td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt;</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>QuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt;)</code></td>
<td>Get the names of all the Jobs in the matching group.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt;)</code></td>
<td>Get the names of all the Jobs in the matching group.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getPausedTriggerGroups()</code></td>
<td>Get the names of all the Triggers in the matching group.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getPausedTriggerGroups()</code></td>
<td>Get the names of all the Triggers in the matching group.</td>
</tr>
<tr>
<td><code>SchedulerContext QuartzScheduler.getSchedulerContext()</code></td>
<td>Returns the SchedulerContext of the Scheduler.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getSchedulerContext()</code></td>
<td>Returns the SchedulerContext of the Scheduler.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getTrigger(TriggerKey triggerKey)</code></td>
<td>Get the Trigger instance with the given name and value.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getTrigger(TriggerKey triggerKey)</code></td>
<td>Get the Trigger instance with the given name and value.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getTriggerGroupNames()</code></td>
<td>Get the names of all known Trigger groups.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getTriggerGroupNames()</code></td>
<td>Get the names of all known Trigger groups.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt;)</code></td>
<td>Get the names of all the Triggers in the matching group.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt;)</code></td>
<td>Get the names of all the Triggers in the matching group.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getTriggersOfJob(JobKey jobKey)</code></td>
<td>Get all Triggers that are associated with the identified Job.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getTriggersOfJob(JobKey jobKey)</code></td>
<td>Get all Triggers that are associated with the identified Job.</td>
</tr>
<tr>
<td><code>QuartzScheduler.getTriggerState(TriggerKey triggerKey)</code></td>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.getTriggerState(TriggerKey triggerKey)</code></td>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td><code>void QuartzScheduler.initialize()</code></td>
<td>Initialize the QuartzScheduler.</td>
</tr>
<tr>
<td><code>void</code></td>
<td>Initialize the QuartzScheduler.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>JobRunShell.initialize(QuartzScheduler qs)</code></td>
<td>Initialize QuartzScheduler</td>
</tr>
<tr>
<td><code>void QuartzScheduler.notifyJobListenersToBeExecuted()</code></td>
<td>Notify job listeners to be executed</td>
</tr>
<tr>
<td><code>void QuartzScheduler.notifyJobListenersWasExecuted(JobExecutionContext je)</code></td>
<td>Notify job listeners was executed</td>
</tr>
<tr>
<td><code>void QuartzScheduler.notifyJobListenersWasVetoed(JobExecutionContext je)</code></td>
<td>Notify job listeners was vetoed</td>
</tr>
<tr>
<td><code>void QuartzScheduler.notifyTriggerListenersComplete(JobExecutionContext je, Trigger.CompletedExecutionInstruction instCode)</code></td>
<td>Notify trigger listeners complete</td>
</tr>
<tr>
<td><code>boolean QuartzScheduler.notifyTriggerListenersFired(JobExecutionContext je)</code></td>
<td>Notify trigger listeners fired</td>
</tr>
<tr>
<td><code>void QuartzScheduler.notifyTriggerListenersMisfired(TriggerKey triggerKey)</code></td>
<td>Notify trigger listeners misfired</td>
</tr>
<tr>
<td><code>void QuartzScheduler.pauseAll()</code></td>
<td>Pause all triggers - equivalent of calling <code>pauseTriggers(GroupMatcher matching all known groups)</code></td>
</tr>
<tr>
<td><code>void QuartzScheduler.pauseAll()</code></td>
<td>Pause all triggers - equivalent of calling <code>pauseTriggers(GroupMatcher matching all known groups)</code></td>
</tr>
<tr>
<td><code>void QuartzScheduler.pauseJob(JobKey jobKey)</code></td>
<td>Pause the JobDetail with the given name - by pausing all of its current JobExecutions</td>
</tr>
<tr>
<td><code>void QuartzScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; groupMatcher)</code></td>
<td>Pause all of the JobDetails in the matching groups</td>
</tr>
<tr>
<td><code>void QuartzScheduler.pauseTrigger(TriggerKey triggerKey)</code></td>
<td>Pause the Trigger with the given name</td>
</tr>
<tr>
<td><code>void QuartzScheduler.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; groupMatcher)</code></td>
<td>Pause all of the Triggers in the matching groups</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Pauses all triggers in all matching groups.</td>
</tr>
<tr>
<td><code>Date QuartzScheduler.rescheduleJob(TriggerKey trigger)</code></td>
<td>Reschedules a trigger.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.rescheduleJob(TriggerKey trigger)</code></td>
<td>Reschedules a trigger.</td>
</tr>
<tr>
<td><code>void QuartzScheduler.resumeAll()</code></td>
<td>Resume all triggers - equivalent to calling <code>resumeJobs</code> on every group.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.resumeAll()</code></td>
<td>Resume all triggers - equivalent to calling <code>resumeJobs</code> on every group.</td>
</tr>
<tr>
<td><code>void QuartzScheduler.resumeJob(JobKey jobKey)</code></td>
<td>Resume the job identified by the given JobKey.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.resumeJob(JobKey jobKey)</code></td>
<td>Resume the job identified by the given JobKey.</td>
</tr>
<tr>
<td><code>void QuartzScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Resume all the JobDetails in the matching groups.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Resume all the JobDetails in the matching groups.</td>
</tr>
<tr>
<td><code>void QuartzScheduler.resumeTrigger(TriggerKey trigger)</code></td>
<td>Resume the trigger with the given name.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.resumeTrigger(TriggerKey trigger)</code></td>
<td>Resume the trigger with the given name.</td>
</tr>
<tr>
<td><code>void QuartzScheduler.resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Resume all the Triggers in the matching groups.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Resume all the Triggers in the matching groups.</td>
</tr>
<tr>
<td><code>Date QuartzScheduler.scheduleJob(JobDetail jobDetail)</code></td>
<td>Adds the Job identified by the given JobDetail to the Scheduler.</td>
</tr>
<tr>
<td><code>Date RemotableQuartzScheduler.scheduleJob(JobDetail jobDetail)</code></td>
<td>Adds the Job identified by the given JobDetail to the Scheduler.</td>
</tr>
<tr>
<td><code>Date QuartzScheduler.scheduleJob(Trigger trigger)</code></td>
<td>Schedule the given Trigger with the Job identified by the given Trigger.</td>
</tr>
<tr>
<td><code>Date RemotableQuartzScheduler.scheduleJob(Trigger trigger)</code></td>
<td>Schedule the given Trigger with the Job identified by the given Trigger.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>RemotableQuartzScheduler.scheduleJob(Trigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.scheduleJobs(Map&lt;JobDetail, List&gt; jobs, boolean replace)</code></td>
<td></td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.scheduleJobs(Map&lt;JobDetail, List&gt; jobs, boolean replace)</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.setJobFactory(org.quartz.spi.JobFactory factory)</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.start()</code></td>
<td>Starts the QuartzScheduler's threads that fire Triggers.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.start()</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.startDelayed(int seconds)</code></td>
<td></td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.startDelayed(int seconds)</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.triggerJob(JobKey jobKey, JobDataMap jobData)</code></td>
<td>Trigger the identified Job (execute it now) - with a non-volatile Trigger.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.triggerJob(JobKey jobKey, JobDataMap jobData)</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.triggerJob(org.quartz.spi.OperableTrigger trigger)</code></td>
<td>Store and schedule the identified OperableTrigger.</td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.triggerJob(org.quartz.spi.OperableTrigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean QuartzScheduler.unscheduleJob(TriggerKey triggerKey)</code></td>
<td>Remove the indicated Trigger from the scheduler.</td>
</tr>
<tr>
<td><code>boolean RemotableQuartzScheduler.unscheduleJob(TriggerKey triggerKey)</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean QuartzScheduler.unscheduleJobs(List&lt;TriggerKey&gt; triggerKeys)</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean RemotableQuartzScheduler.unscheduleJobs(List&lt;TriggerKey&gt; triggerKeys)</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.validateState()</code></td>
<td></td>
</tr>
</tbody>
</table>
Constructors in **org.quartz.core** that throw **SchedulerException**

```java
QuartzScheduler(QuartzSchedulerResources resources,
long idleWaitTime, long dbRetryInterval)
```
Create a QuartzScheduler with the given configuration properties.

Uses of **SchedulerException** in **org.quartz.core.jmx**

Methods in **org.quartz.core.jmx** that throw **SchedulerException**

```java
static CompositeData JobExecutionContextSupport.toCompositeData(JobExecutionContext context)
static TabularData JobExecutionContextSupport.toTabularData(List<JobExecutionContext> contexts)
```

Uses of **SchedulerException** in **org.quartz.ee.jmx.jboss**

Methods in **org.quartz.ee.jmx.jboss** that throw **SchedulerException**

```java
protected Object JBoss4RMIRemoteMBeanScheduler.getAttribute(String attribute)
protected AttributeList JBoss4RMIRemoteMBeanScheduler.getAttributes(String[] attributes)
void JBoss4RMIRemoteMBeanScheduler.initialize()
    Initialize this remote MBean scheduler, getting the JBoss RMIAdaptor for communication.
protected Object JBoss4RMIRemoteMBeanScheduler.invoke(String operationName, Object[] params, String[] signature)
```
### Constructors in `org.quartz.ee.jmx.jboss` that throw `SchedulerException`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBoss4RMIRemoteMBeanScheduler()</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of `SchedulerException` in `org.quartz.ee.jta`

### Methods in `org.quartz.ee.jta` that throw `SchedulerException`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void JTAJobRunShell.begin()</td>
<td></td>
</tr>
<tr>
<td>protected void JTAJobRunShell.complete(boolean successfulExecution)</td>
<td></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.ee.jta` with parameters of type `SchedulerException`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void QuartzServer.schedulerError(String msg, SchedulerException cause)</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of `SchedulerException` in `org.quartz.impl`

### Methods in `org.quartz.impl` that return `SchedulerException`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected SchedulerException RemoteScheduler.invalidateHandleCreateException(String cause)</td>
<td></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl` with parameters of type `SchedulerException`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void QuartzServer.schedulerError(String msg, SchedulerException cause)</td>
<td></td>
</tr>
</tbody>
</table>
Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

### Methods in `org.quartz.impl` that throw `SchedulerException`

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>RemoteScheduler.addCalendar(String calName, Calendar boolean updateTriggers)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><code>StdScheduler.addCalendar(String calName, Calendar boolean updateTriggers)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><code>RemoteMBeanScheduler.addCalendar(String calName, boolean replace, boolean updateTriggers)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><code>RemoteScheduler.addJob(JobDetail jobDetail, boolean replace)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><code>StdScheduler.addJob(JobDetail jobDetail, boolean replace)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><code>RemoteMBeanScheduler.addJob(JobDetail jobDetail)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><code>SchedulerRepository.bind(Scheduler sched)</code></td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td><code>RemoteScheduler.checkExists(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>StdScheduler.checkExists(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>RemoteMBeanScheduler.checkExists(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>RemoteScheduler.checkExists(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>StdScheduler.checkExists(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RemoteMBeanScheduler.checkExists(TriggerKey triggerKey)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteScheduler.clear()
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void StdScheduler.clear()
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteMBeanScheduler.clear()

JobRunShell StdJobRunShellFactory.createJobRunShell(org.quartz.spi.TriggerFiredBundle bundle)
Called by the QuartzSchedulerThread to obtain instances of

void DirectSchedulerFactory.createRemoteScheduler(String schedulerInstanceId, String rmiHost, int rmiPort)
Same as DirectSchedulerFactory.createRemoteScheduler(String rmiHost, int rmiPort), with the addition of specifying the scheduler name.

void DirectSchedulerFactory.createRemoteScheduler(String schedulerInstanceId, String rmiBindName, String rmiHost, int rmiPort)
Same as DirectSchedulerFactory.createRemoteScheduler(String rmiHost, int rmiPort), with the addition of specifying the scheduler name and instance ID.

void DirectSchedulerFactory.createScheduler(String schedulerName, String schedulerInstanceId, org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore)
Same as DirectSchedulerFactory.createScheduler(ThreadPool threadPool, JobStore jobStore), with the addition of specifying the scheduler name and instance ID.

void DirectSchedulerFactory.createScheduler(String schedulerName, String schedulerInstanceId, org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore, Map schedulerPluginMap, String rmiRegistryHost, int rmiRegistryPort, long idleWaitTime, long dbFailureRetryInterval, boolean jmxExport)
Creates a scheduler using the specified thread pool, job store, and RMI.

void DirectSchedulerFactory.createScheduler(String schedulerName, String schedulerInstanceId, org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore, String rmiRegistryHost, int rmiRegistryPort, long idleWaitTime, long dbFailureRetryInterval)
Creates a scheduler using the specified thread pool, and RMI.

void DirectSchedulerFactory.createScheduler(String schedulerName, String schedulerInstanceId, org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore, String rmiRegistryHost, int rmiRegistryPort, long idleWaitTime, long dbFailureRetryInterval, boolean jmxExport)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`org.quartz.spi.JobStore jobStore)</td>
<td>Creates a scheduler using the specified thread pool and job store.</td>
</tr>
<tr>
<td>`DirectSchedulerFactory.createVolatileScheduler(int maxThreads)</td>
<td>Creates an in memory job store (RAMJobStore) Thread priority is set to Thread.NORM_PRIORITY, and correctly spelled method.</td>
</tr>
<tr>
<td>`RemoteScheduler.deleteCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`StdScheduler.deleteCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`RemoteMBeanScheduler.deleteCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`RemoteScheduler.deleteJob(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`StdScheduler.deleteJob(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`RemoteMBeanScheduler.deleteJob(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`RemoteScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`StdScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`RemoteMBeanScheduler.deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>`StdSchedulerFactory.getAllSchedulers()</td>
<td>Returns a handle to all known Schedulers (made by any Quartz implementation).</td>
</tr>
<tr>
<td>`DirectSchedulerFactory.getAllSchedulers()</td>
<td>Returns a handle to all known Schedulers (made by any Quartz implementation).</td>
</tr>
<tr>
<td>`RemoteMBeanScheduler.getAttribute(String attribute)</td>
<td>Get the given attribute of the remote Scheduler MBean.</td>
</tr>
<tr>
<td>`RemoteMBeanScheduler.getAttributes(String[] attributes)</td>
<td>Get the given attributes of the remote Scheduler MBean.</td>
</tr>
<tr>
<td>Method</td>
<td>RemoteScheduler Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td><code>getCalendar(String calName)</code></td>
<td>Calls the equivalent method on the 'proxied' Quartz</td>
</tr>
<tr>
<td><code>getCalendarNames()</code></td>
<td><code>getCalendarNames()</code></td>
</tr>
<tr>
<td><code>getContext()</code></td>
<td><code>getContext()</code></td>
</tr>
<tr>
<td><code>getCurrentlyExecutingJobs()</code></td>
<td>Calls the equivalent method on the 'proxied' Quartz</td>
</tr>
<tr>
<td><code>getJobGroupNames()</code></td>
<td></td>
</tr>
<tr>
<td><code>getJobDetail(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' Quartz</td>
</tr>
<tr>
<td><code>getJobDetail(JobKey jobKey)</code></td>
<td></td>
</tr>
<tr>
<td><code>getJobDetail(JobKey jobKey)</code></td>
<td></td>
</tr>
<tr>
<td><code>getJobGroupNames()</code></td>
<td><code>getJobGroupNames()</code></td>
</tr>
<tr>
<td>Method Call</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code> StdScheduler. <strong>getJobGroupNames</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code> RemoteMBeanScheduler. <strong>getJobGroupNames</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code> StdScheduler. <strong>getJobGroupNames</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code> RemoteMBeanScheduler. <strong>getJobGroupNames</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> RemoteScheduler. <strong>getJobKeys</strong>([GroupMatcher&lt;JobKey&gt;] matcher)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> StdScheduler. <strong>getJobKeys</strong>([GroupMatcher&lt;JobKey&gt;] matcher)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> RemoteMBeanScheduler. <strong>getJobKeys</strong>([GroupMatcher&lt;JobKey&gt;] matcher)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> StdScheduler. <strong>getJobKeys</strong>([GroupMatcher&lt;JobKey&gt;] matcher)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> RemoteMBeanScheduler. <strong>getJobKeys</strong>([GroupMatcher&lt;JobKey&gt;] matcher)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>ListenerManager</code> RemoteScheduler. <strong>getListenerManager</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>ListenerManager</code> StdScheduler. <strong>getListenerManager</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>ListenerManager</code> RemoteMBeanScheduler. <strong>getListenerManager</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>SchedulerMetaData</code> RemoteScheduler. <strong>getMetaData</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>SchedulerMetaData</code> RemoteMBeanScheduler. <strong>getMetaData</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set</code> RemoteScheduler. <strong>getPausedTriggerGroups</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set</code> StdScheduler. <strong>getPausedTriggerGroups</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> RemoteMBeanScheduler. <strong>getPausedTriggerGroups</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Scheduler</code> StdSchedulerFactory. <strong>getScheduler</strong>()</td>
<td>Returns a handle to the Scheduler produced by this factory.</td>
</tr>
<tr>
<td><code>Scheduler</code> DirectSchedulerFactory. <strong>getScheduler</strong>()</td>
<td>Returns a handle to the Scheduler produced by this factory.</td>
</tr>
<tr>
<td><code>Scheduler</code> RemoteScheduler. <strong>getRemoteScheduler</strong>()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>Scheduler</code> StdSchedulerFactory. <strong>getScheduler</strong>()</td>
<td>Returns a handle to the Scheduler produced by this factory.</td>
</tr>
<tr>
<td><code>Scheduler</code> DirectSchedulerFactory. <strong>getScheduler</strong>()</td>
<td>Returns a handle to the Scheduler produced by this factory.</td>
</tr>
<tr>
<td>Class</td>
<td>Method</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StdSchedulerFactory</td>
<td><code>getScheduler(String schedName)</code></td>
</tr>
<tr>
<td>DirectSchedulerFactory</td>
<td><code>getScheduler(String schedName)</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getSchedulerInstanceId()</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getSchedulerName()</code></td>
</tr>
<tr>
<td>RemoteScheduler</td>
<td><code>getSchedulerInstanceId()</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getSchedulerName()</code></td>
</tr>
<tr>
<td>RemoteMBeanScheduler</td>
<td><code>getTrigger(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>Trigger</td>
<td><code>getTrigger(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>RemoteMBeanScheduler</td>
<td><code>getTrigger(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td><code>getTriggerGroupNames()</code></td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td><code>getTriggerGroupNames()</code></td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td><code>getTriggerGroupNames()</code></td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td><code>getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td><code>getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td><code>getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
</tr>
<tr>
<td>List&lt;? extends Trigger&gt;</td>
<td><code>getTriggersOfJob(JobKey jobKey)</code></td>
</tr>
<tr>
<td>List? extends Trigger&gt; &amp; StdScheduler.getTriggersOfJob(JobKey jobKey)</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>List&lt;Trigger&gt; &amp; RemoteMBeanScheduler.getTriggersOfJob(JobKey jobKey)</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>SchedulingContext associated with this instance.</td>
<td></td>
</tr>
<tr>
<td>Trigger.TriggerState &amp; RemoteScheduler.getTriggerState(TriggerKey triggerKey)</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>SchedulingContext associated with this instance.</td>
<td></td>
</tr>
<tr>
<td>Trigger.TriggerState &amp; StdScheduler.getTriggerState(TriggerKey triggerKey)</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>Trigger.TriggerState &amp; RemoteMBeanScheduler.getTriggerState(TriggerKey triggerKey)</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>SchedulingContext associated with this instance.</td>
<td></td>
</tr>
<tr>
<td>void &amp; StdSchedulerFactory.initialize()</td>
<td></td>
</tr>
<tr>
<td>Initialize the SchedulerFactory with the contents of System properties.</td>
<td></td>
</tr>
<tr>
<td>abstract void &amp; RemoteMBeanScheduler.initialize()</td>
<td></td>
</tr>
<tr>
<td>Initialize this RemoteMBeanScheduler instance, connecting to the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>void &amp; StdSchedulerFactory.initialize(InputStream propertiesStream)</td>
<td></td>
</tr>
<tr>
<td>Initialize the SchedulerFactory with the contents of the given InputStream.</td>
<td></td>
</tr>
<tr>
<td>void &amp; StdSchedulerFactory.initialize(Properties props)</td>
<td></td>
</tr>
<tr>
<td>Initialize the SchedulerFactory with the contents of the given Properties.</td>
<td></td>
</tr>
<tr>
<td>void &amp; StdSchedulerFactory.initialize(String filename)</td>
<td></td>
</tr>
<tr>
<td>Initialize the SchedulerFactory with the contents of the given filename.</td>
<td></td>
</tr>
<tr>
<td>protected abstract Object &amp; RemoteMBeanScheduler.invoke(String operationName, String[] signature)</td>
<td></td>
</tr>
<tr>
<td>Invoke the given operation on the remote Scheduler.</td>
<td></td>
</tr>
<tr>
<td>boolean &amp; RemoteScheduler.isInStandbyMode()</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>boolean &amp; RemoteMBeanScheduler.isInStandbyMode()</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>boolean &amp; RemoteScheduler.isShutdown()</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>boolean &amp; RemoteMBeanScheduler.isShutdown()</td>
<td></td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>RemoteScheduler.isStarted()</code></td>
<td>Whether the scheduler has been started.</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler.isStarted()</code></td>
<td>Whether the scheduler has been started.</td>
</tr>
<tr>
<td><code>RemoteScheduler.pauseAll()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>StdScheduler.pauseAll()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler.pauseAll()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteScheduler.pauseJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>StdScheduler.pauseJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler.pauseJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>StdScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteScheduler.pauseTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>StdScheduler.pauseTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler.pauseTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteScheduler.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
</tbody>
</table>

Calls the equivalent method on the 'proxied' QuartzScheduler.

SchedulingContext associated with this instance.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StdScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void pauseTriggers(GroupMatcher&lt;TriggerKey&gt; triggers)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteMBeanScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void pauseTriggers(GroupMatcher&lt;TriggerKey&gt; triggers)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>Date rescheduleJob(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>StdScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>Date rescheduleJob(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteMBeanScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>Date rescheduleJob(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeAll()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>StdScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeAll()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteMBeanScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeAll()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>StdScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteMBeanScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>StdScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteMBeanScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>RemoteScheduler.</strong></td>
<td></td>
</tr>
<tr>
<td><code>void resumeTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void StdScheduler.resumeTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteMBeanScheduler.resumeTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteScheduler.resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void StdScheduler.resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteMBeanScheduler.resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteScheduler.scheduleJob(JobDetail jobDetail, boolean replace)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void StdScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteMBeanScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteScheduler.scheduleJobs(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; jobDetails, boolean replace)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void StdScheduler.scheduleJobs(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; jobDetails, boolean replace)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteMBeanScheduler.scheduleJobs(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; jobDetails, boolean replace)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td><code>void RemoteScheduler.setJobFactory(org.quartz.spi.JobFactory factory)</code></td>
<td>Sets the job factory for the 'proxied' QuartzScheduler associated with this instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>void StdScheduler._setJobFactory(JobFactory)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._setJobFactory(JobFactory)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._setSchedulerObjectName(ObjectName)</td>
<td>Set the name under which the Scheduler MBean is registered on the server.</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._setSchedulerObjectName(String)</td>
<td>Set the name under which the Scheduler MBean is registered on the server.</td>
</tr>
<tr>
<td>void RemoteScheduler._shutdown()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._shutdown()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteScheduler._shutdown(boolean)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._shutdown(boolean)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteScheduler._standby()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._standby()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteScheduler._start()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void StdScheduler._start()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._start()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteScheduler._startDelayed(int)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void StdScheduler._startDelayed(int)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void RemoteMBeanScheduler._startDelayed(int)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
</tbody>
</table>
Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteScheduler.\texttt{triggerJob}(JobKey jobKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void StdScheduler.\texttt{triggerJob}(JobKey jobKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteMBEanScheduler.\texttt{triggerJob}(JobKey jobKey)
Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.

void RemoteScheduler.\texttt{triggerJob}(JobKey jobKey, JobDataMap)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void StdScheduler.\texttt{triggerJob}(JobKey jobKey, JobDataMap)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteMBEanScheduler.\texttt{triggerJob}(JobKey jobKey, JobDataMap)
Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.

boolean RemoteScheduler.\texttt{unscheduleJob}(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

boolean StdScheduler.\texttt{unscheduleJob}(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

boolean RemoteMBEanScheduler.\texttt{unscheduleJob}(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler associated with this instance.

boolean RemoteScheduler.\texttt{unscheduleJobs}(List<TriggerKey> triggerKeys)

boolean StdScheduler.\texttt{unscheduleJobs}(List<TriggerKey> triggerKeys)

boolean RemoteMBEanScheduler.\texttt{unscheduleJobs}(List<TriggerKey> triggerKeys)

Constructors in \texttt{org.quartz.impl} that throw \texttt{SchedulerException}

\texttt{StdSchedulerFactory(Properties props)}
Create a StdSchedulerFactory that has been initialized via
\texttt{StdSchedulerFactory.initialize(Properties)}.

\texttt{StdSchedulerFactory(String fileName)}
Create a StdSchedulerFactory that has been initialized via StdSchedulerFactory.initialize(String).

Uses of **SchedulerException** in **org.quartz.impl.jdbcjobstore**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockException</td>
<td>Exception class for when there is a failure obtaining or releasing a resource lock.</td>
</tr>
<tr>
<td>NoSuchDelegateException</td>
<td>Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.</td>
</tr>
</tbody>
</table>

Methods in **org.quartz.impl.jdbcjobstore** that throw **SchedulerException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void JobStoreSupport.&lt;code&gt;schedulerStarted&lt;/code&gt;()</td>
<td></td>
</tr>
</tbody>
</table>

Uses of **SchedulerException** in **org.quartz.impl.triggers**

Methods in **org.quartz.impl.triggers** that throw **SchedulerException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void AbstractTrigger.&lt;code&gt;validate&lt;/code&gt;()</td>
<td>Validates whether the properties of the JobDetail are valid for submission into a Scheduler.</td>
</tr>
<tr>
<td>void CalendarIntervalTriggerImpl.&lt;code&gt;validate&lt;/code&gt;()</td>
<td>Validates whether the properties of the JobDetail are valid for submission into a Scheduler.</td>
</tr>
<tr>
<td>void SimpleTriggerImpl.&lt;code&gt;validate&lt;/code&gt;()</td>
<td></td>
</tr>
</tbody>
</table>
Validates whether the properties of the JobDetail are valid for submission into a Scheduler.

**Uses of SchedulerException in org.quartz.jobs.ee.jms**

<table>
<thead>
<tr>
<th>Subclasses of SchedulerException in org.quartz.jobs.ee.jms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class JmsJobException</strong> in org.quartz.jobs.ee.jms</td>
</tr>
</tbody>
</table>
| The JmsJobException is used to indicate an error during sending of a javax.jms.Message.

**Uses of SchedulerException in org.quartz.listeners**

<table>
<thead>
<tr>
<th>Methods in org.quartz.listeners with parameters of type SchedulerException</th>
</tr>
</thead>
<tbody>
<tr>
<td>void SchedulerListenerSupport.scheduledError(String msg, SchedulerException cause)</td>
</tr>
<tr>
<td>void BroadcastSchedulerListener.scheduledError(String msg, SchedulerException cause)</td>
</tr>
</tbody>
</table>

**Uses of SchedulerException in org.quartz.locality**

<table>
<thead>
<tr>
<th>Subclasses of SchedulerException in org.quartz.locality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class LocalityException</strong> in org.quartz.locality</td>
</tr>
<tr>
<td>Exception thrown when some Constraint cannot be handled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.locality that throw SchedulerException</th>
</tr>
</thead>
</table>
void DelegatingLocalityTrigger.validate()

Utilities of `SchedulerException` in `org.quartz.plugins`

Methods in `org.quartz.plugins` that throw `SchedulerException`

```java
void SchedulerPluginWithUserTransactionSupport.initialize(String name, Scheduler scheduler)
```

Utilities of `SchedulerException` in `org.quartz.plugins.history`

Methods in `org.quartz.plugins.history` that throw `SchedulerException`

```java
void LoggingJobHistoryPlugin.initialize(String name, Scheduler scheduler)
```

```java
void LoggingTriggerHistoryPlugin.initialize(String name, Scheduler scheduler)
```

Utilities of `SchedulerException` in `org.quartz.plugins.management`

Methods in `org.quartz.plugins.management` that throw `SchedulerException`
void `ShutdownHookPlugin.initialize(String name, Scheduler scheduler)`

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

### Uses of `SchedulerException` in `org.quartz.plugins.xml`

### Methods in `org.quartz.plugins.xml` that throw `SchedulerException`

void `XMLSchedulingDataProcessorPlugin.initialize(String name, Scheduler scheduler)`

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

### Uses of `SchedulerException` in `org.quartz.simpl`

### Methods in `org.quartz.simpl` that throw `SchedulerException`

**String**

`SystemPropertyInstanceIdGenerator.generateInstanceId()`

Returns the cluster wide value for this scheduler instance's id, based on a system property

**String**

`HostnameInstanceIdGenerator.generateInstanceId()`

**String**

`SimpleInstanceIdGenerator.generateInstanceId()`

**Job**

`PropertySettingJobFactory.newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler scheduler)`

**Job**

`SimpleJobFactory.newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler scheduler)`

**void**

`RAMJobStore.schedulerStarted()`
### Uses of `SchedulerException` in `org.quartz.utils`

### Constructors in `org.quartz.utils` that throw `SchedulerException`

<table>
<thead>
<tr>
<th>Class</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PoolingConnectionProvider</code></td>
<td><code>Properties config</code></td>
<td>Create a connection pool using the given properties.</td>
</tr>
<tr>
<td><code>PoolingConnectionProvider</code></td>
<td><code>String dbDriver, String dbURL, String dbUser, String dbPassword, int maxConnections, String dbValidationQuery</code></td>
<td>Uses of <code>SchedulerException</code> in <code>org.quartz.xml</code></td>
</tr>
</tbody>
</table>

### Uses of `SchedulerException` in `org.quartz.xml`

### Methods in `org.quartz.xml` that throw `SchedulerException`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>executePreProcessCommands</code></td>
<td><code>Scheduler sched</code></td>
<td>Schedules the given sets of jobs and triggers.</td>
</tr>
<tr>
<td><code>processFile</code></td>
<td><code>String fileName, String systemId, Scheduler sched</code></td>
<td>Process the xml file named fileName with the given system ID.</td>
</tr>
<tr>
<td><code>processFileAndScheduleJobs</code></td>
<td><code>InputStream input, String systemId, Scheduler sched</code></td>
<td>Process the xml file in the default location, and schedule all of the jobs within it.</td>
</tr>
<tr>
<td><code>processStreamAndScheduleJobs</code></td>
<td><code>InputStream input, String systemId, Scheduler sched</code></td>
<td>Process the xml file named fileName with the given system ID.</td>
</tr>
<tr>
<td><code>scheduleJobs</code></td>
<td><code>Scheduler sched</code></td>
<td>Schedules the given sets of jobs and triggers.</td>
</tr>
</tbody>
</table>
## Uses of Interface org.quartz.SchedulerFactory

### Packages that use **SchedulerFactory**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of **SchedulerFactory** in org.quartz.impl

### Classes in org.quartz.impl that implement **SchedulerFactory**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectSchedulerFactory</td>
<td>A singleton implementation of SchedulerFactory.</td>
</tr>
<tr>
<td>StdSchedulerFactory</td>
<td>An implementation of SchedulerFactory that does all of its work of creating a QuartzScheduler instance based on the contents of a Properties file.</td>
</tr>
</tbody>
</table>

### Methods in org.quartz.impl with parameters of type **SchedulerFactory**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void QuartzServer.serve</td>
<td>(SchedulerFactory schedFact, boolean console)</td>
</tr>
</tbody>
</table>
Uses of Interface org.quartz.SchedulerListener

<table>
<thead>
<tr>
<th>Packages that use SchedulerListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
</tr>
<tr>
<td>org.quartz.core</td>
</tr>
<tr>
<td>org.quartz.ee.jta</td>
</tr>
<tr>
<td>org.quartz.impl</td>
</tr>
<tr>
<td>org.quartz.listeners</td>
</tr>
</tbody>
</table>

Uses of SchedulerListener in org.quartz

Methods in org.quartz that return types with arguments of type SchedulerListener

<table>
<thead>
<tr>
<th>Methods in org.quartz with parameters of type SchedulerListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;SchedulerListener&gt;</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz with parameters of type SchedulerListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
# Uses of `SchedulerListener` in `org.quartz.core`

## Classes in `org.quartz.core` that implement `SchedulerListener`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobRunShell</code></td>
<td>JobRunShell instances are responsible for providing the 'safe' environment for Jobs to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the Trigger with the Job's completion code, etc.</td>
</tr>
<tr>
<td><code>QuartzSchedulerMBeanImpl</code></td>
<td></td>
</tr>
<tr>
<td><code>SampledStatisticsImpl</code></td>
<td></td>
</tr>
</tbody>
</table>

## Methods in `org.quartz.core` that return types with arguments of type `SchedulerListener`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>List&lt;SchedulerListener&gt;</code></td>
<td><code>QuartzScheduler.getInternalSchedulerListeners()</code> Get a List containing all of the internal SchedulerListeners registered with the Scheduler.</td>
</tr>
<tr>
<td><code>List&lt;SchedulerListener&gt;</code></td>
<td><code>ListenerManagerImpl.getSchedulerListeners()</code></td>
</tr>
</tbody>
</table>

## Methods in `org.quartz.core` with parameters of type `SchedulerListener`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>QuartzScheduler.addInternalSchedulerListener(SchedulerListener)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>ListenerManagerImpl.addSchedulerListener(SchedulerListener)</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>QuartzScheduler.removeInternalSchedulerListener(SchedulerListener)</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>ListenerManagerImpl.removeSchedulerListener(SchedulerListener)</code></td>
</tr>
</tbody>
</table>
## Uses of `SchedulerListener` in `org.quartz.ee.jta`

### Classes in `org.quartz.ee.jta` that implement `SchedulerListener`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JTAJobRunShell</code></td>
<td>An extension of <code>JobRunShell</code> that begins an XA transaction before executing the Job, and commits (or rolls-back) the transaction after execution completes.</td>
</tr>
</tbody>
</table>

## Uses of `SchedulerListener` in `org.quartz.impl`

### Classes in `org.quartz.impl` that implement `SchedulerListener`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzServer</code></td>
<td>Instantiates an instance of Quartz Scheduler as a stand-alone program, if the scheduler is configured for RMI it will be made available.</td>
</tr>
</tbody>
</table>

## Uses of `SchedulerListener` in `org.quartz.listeners`

### Classes in `org.quartz.listeners` that implement `SchedulerListener`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BroadcastSchedulerListener</code></td>
<td>Holds a List of references to SchedulerListener instances and broadcasts all events to them (in order).</td>
</tr>
<tr>
<td><code>SchedulerListenerSupport</code></td>
<td>A helpful abstract base class for implementors of SchedulerListener.</td>
</tr>
</tbody>
</table>

## Methods in `org.quartz.listeners` that return types with arguments of type `SchedulerListener`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BroadcastSchedulerListener.getListeners()</code></td>
<td>Returns a List of SchedulerListener instances.</td>
</tr>
</tbody>
</table>
Methods in `org.quartz.listeners` with parameters of type `SchedulerListener`

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>BroadcastSchedulerListener.addListener</code></td>
<td><code>SchedulerListener listener</code></td>
</tr>
<tr>
<td>boolean</td>
<td><code>BroadcastSchedulerListener.removeListener</code></td>
<td><code>SchedulerListener listener</code></td>
</tr>
</tbody>
</table>
## Uses of Class
### org.quartz.SchedulerMetaData

<table>
<thead>
<tr>
<th>Packages that use <strong>SchedulerMetaData</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of **SchedulerMetaData** in **org.quartz**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz</strong> that return <strong>SchedulerMetaData</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Scheduler.getMetaData()</code></td>
<td>Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.</td>
</tr>
</tbody>
</table>

### Uses of **SchedulerMetaData** in **org.quartz.impl**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.impl</strong> that return <strong>SchedulerMetaData</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RemoteScheduler.getMetaData()</code></td>
<td></td>
</tr>
<tr>
<td><code>StdScheduler.getMetaData()</code></td>
<td></td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler.getMetaData()</code></td>
<td></td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class

org.quartz.SimpleScheduleBuilder

Packages that use SimpleScheduleBuilder

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
</tbody>
</table>

Uses of SimpleScheduleBuilder in org.quartz

Methods in org.quartz that return SimpleScheduleBuilder

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleScheduleBuilder.repeatForever()</td>
<td>Specify that the trigger will repeat indefinitely.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatHourlyForever()</td>
<td>Create a SimpleScheduleBuilder set to repeat forever with a 1 hour interval.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatHourlyForever(int hours)</td>
<td>Create a SimpleScheduleBuilder set to repeat forever with an interval number of hours.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatHourlyForTotalCount()</td>
<td>Create a SimpleScheduleBuilder set to repeat the given number of hour interval.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatHourlyForTotalCount(int hours)</td>
<td>Create a SimpleScheduleBuilder set to repeat the given number of hour interval of the given number of hours.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatMinutelyForever()</td>
<td>Create a SimpleScheduleBuilder set to repeat forever with a 1 minute interval.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatMinutelyForever(int minutes)</td>
<td>Create a SimpleScheduleBuilder set to repeat forever with an interval number of minutes.</td>
</tr>
<tr>
<td>static SimpleScheduleBuilder.repeatMinutelyForTotalCount()</td>
<td>Create a SimpleScheduleBuilder set to repeat the given number of minute interval.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.repeatMinutelyForTotalCount(int total)</code></td>
<td>Create a <code>SimpleScheduleBuilder</code> set to repeat the given number of minutes.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.repeatSecondlyForever()</code></td>
<td>Create a <code>SimpleScheduleBuilder</code> set to repeat forever with a 1 second interval.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.repeatSecondlyForever(int seconds)</code></td>
<td>Create a <code>SimpleScheduleBuilder</code> set to repeat forever with an interval number of seconds.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.repeatSecondlyForTotalCount(int total)</code></td>
<td>Create a <code>SimpleScheduleBuilder</code> set to repeat the given number of second interval.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.simpleSchedule()</code></td>
<td>Create a <code>SimpleScheduleBuilder</code>.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withIntervalInHours(int intervalInHours)</code></td>
<td>Specify a repeat interval in minutes - which will then be multiplied by 1000 to produce milliseconds.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withIntervalInMilliseconds()</code></td>
<td>Specify a repeat interval in milliseconds.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withIntervalInMinutes(int intervalInMinutes)</code></td>
<td>Specify a repeat interval in minutes - which will produce milliseconds.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withIntervalInSeconds(int intervalInSeconds)</code></td>
<td>Specify a repeat interval in seconds - which will produce milliseconds.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withMisfireHandlingInstructionFireNow()</code></td>
<td>If the Trigger misfires, use the <code>SimpleTrigger.MISFIRE_INSTRUCTION_FIRE_NOW</code> instruction.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withMisfireHandlingInstructionIgnoreMisfires()</code></td>
<td>If the Trigger misfires, use the <code>Trigger.MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY</code> instruction.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withMisfireHandlingInstructionNextWithExistingCount()</code></td>
<td>If the Trigger misfires, use the <code>SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT</code> instruction.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withMisfireHandlingInstruction()</code></td>
<td>If the Trigger misfires, use the <code>SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE</code> instruction.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withMisfireHandlingInstruction()</code></td>
<td>If the Trigger misfires, use the <code>SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE</code> instruction.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withMisfireHandlingInstruction()</code></td>
<td>If the Trigger misfires, use the <code>SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE</code> instruction.</td>
</tr>
<tr>
<td><code>SimpleScheduleBuilder.withRepeatCount(int repeatCount)</code></td>
<td>Specify a the number of time the trigger will repeat this number + 1.</td>
</tr>
</tbody>
</table>
### Uses of Interface
**org.quartz.SimpleTrigger**

#### Packages that use **SimpleTrigger**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz</code></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><code>org.quartz.core.jmx</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.impl.triggers</code></td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
</tbody>
</table>

#### Uses of **SimpleTrigger** in **org.quartz**

#### Methods in **org.quartz** that return types with arguments of type **SimpleTrigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TriggerBuilder&lt;SimpleTrigger&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

#### Uses of **SimpleTrigger** in **org.quartz.core.jmx**

#### Methods in **org.quartz.core.jmx** with parameters of type **SimpleTrigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>CompositeData</code></td>
<td>SimpleTriggerSupport.<strong>toCompositeData</strong>(SimpleTrigger trigger)</td>
</tr>
</tbody>
</table>

#### Method parameters in **org.quartz.core.jmx** with type arguments of type **SimpleTrigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>TabularData</code></td>
<td>SimpleTriggerSupport.<strong>toTabularData</strong>(List&lt;? extends SimpleTrigger&gt; triggers)</td>
</tr>
</tbody>
</table>
## Uses of SimpleTrigger in org.quartz.impl.triggers

<table>
<thead>
<tr>
<th>Class</th>
<th>SimpleTriggerImpl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A concrete Trigger that is used to fire a JobDetail at a given moment in time, and optionally repeated at a specified interval.</td>
</tr>
</tbody>
</table>

## Methods in org.quartz.impl.triggers that return types with arguments of type SimpleTrigger

<table>
<thead>
<tr>
<th>Method</th>
<th>SimpleTriggerImpl.getScheduleBuilder()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.</td>
</tr>
</tbody>
</table>
Uses of Interface
org.quartz.StatefulJob

No usage of org.quartz.StatefulJob

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

## org.quartz.Trigger.CompletedExecutionInstruction

### Packages that use

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td>org.quartz.listeners</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
<tr>
<td>org.quartz.plugins.history</td>
<td></td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of `Trigger.CompletedExecutionInstruction` in org.quartz

### Methods in org.quartz that return `Trigger.CompletedExecutionInstruction`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Trigger.CompletedExecutionInstruction</code></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static <code>Trigger.CompletedExecutionInstruction[]</code></td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>
Methods in `org.quartz` with parameters of type `Trigger.CompletedExecutionInstruction`:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void TriggerListener.triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</code></td>
<td>Called by the Scheduler when a Trigger has fired, its associated JobDetail has been executed, and its <code>triggered(xx)</code> method has been called.</td>
</tr>
</tbody>
</table>

Uses of `Trigger.CompletedExecutionInstruction` in `org.quartz.core`:

Methods in `org.quartz.core` with parameters of type `Trigger.CompletedExecutionInstruction`:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean JobRunShell.completeTriggerRetryLoop(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected void QuartzScheduler.notifyJobStoreJobComplete(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected void QuartzScheduler.notifyJobStoreJobVetoed(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)</code></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.notifyTriggerListenersComplete(JobExecutionContext context, Trigger.CompletedExecutionInstruction instCode)</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean JobRunShell.vetoedJobRetryLoop(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)</code></td>
<td></td>
</tr>
</tbody>
</table>

Uses of `Trigger.CompletedExecutionInstruction` in `org.quartz.impl.jdbcjobstore`:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Methods in `org.quartz.impl.jdbcjobstore` with parameters of type `Trigger.CompletedExecutionInstruction`

- **protected void** `JobStoreSupport.triggeredJobComplete`(
  `Connection conn`,
  `org.quartz.spi.OperableTrigger trigger`,
  `JobDetail jobDetail`,
  `trigger.CompletedExecutionInstruction triggerInstCode`
)

- **void** `JobStoreSupport.triggeredJobComplete`(
  `org.quartz.spi.OperableTrigger trigger`,
  `JobDetail jobDetail`,
  `Trigger.CompletedExecutionInstruction triggerInstCode`
)

Inform the JobStore that the scheduler has completed the firing of the Trigger (and the execution of its associated Job), and that the JobDataMap in the JobDetail should be updated if the Job is stateful.

Uses of `Trigger.CompletedExecutionInstruction` in `org.quartz.impl.triggers`

Methods in `org.quartz.impl.triggers` that return `Trigger.CompletedExecutionInstruction`

<table>
<thead>
<tr>
<th><code>Trigger.CompletedExecutionInstruction</code></th>
<th><code>AbstractTrigger.executionComplete(JobExecutionContext context, JobExecutionException result)</code></th>
</tr>
</thead>
</table>

This method should not be used by the Quartz client.

Uses of `Trigger.CompletedExecutionInstruction` in `org.quartz.listeners`

Methods in `org.quartz.listeners` with parameters of type `Trigger.CompletedExecutionInstruction`

| void `TriggerListenerSupport.triggerComplete`(`Trigger trigger`,
  `JobExecutionContext context`,
  `Trigger.CompletedExecutionInstruction triggerInstructionCode`)
| void |
BroadcastTriggerListener. **triggerComplete**(Trigger trigger, 
JobExecutionContext context, 
Trigger.CompletedExecutionInstruction triggerInstructionCode)

### Uses of **Trigger.CompletedExecutionInstruction** in **org.quartz.locality**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.locality</strong> that return <strong>Trigger.CompletedExecutionInstruction</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>DelegatingLocalityTrigger. <strong>executionComplete</strong>(JobExecutionContext context, JobExecutionException result)</td>
</tr>
</tbody>
</table>

### Uses of **Trigger.CompletedExecutionInstruction** in **org.quartz.plugins.history**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.plugins.history</strong> with parameters of type <strong>Trigger.CompletedExecutionInstruction</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>void LoggingTriggerHistoryPlugin. <strong>triggerComplete</strong>(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</td>
</tr>
</tbody>
</table>

### Uses of **Trigger.CompletedExecutionInstruction** in **org.quartz.simpl**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.simpl</strong> with parameters of type <strong>Trigger.CompletedExecutionInstruction</strong></th>
</tr>
</thead>
</table>

void RAMJobStore.<a href="triggeredJobComplete">triggeredJobComplete</a>(org.quartz.spi.OperableTrigger JobDetail jobDetail,
-trigger.CompletedExecutionInstruction triggerInstCode)
  Inform the JobStore that the scheduler has completed the firing of the Trigger (and the execution its associated Job), and that the JobDataMap in the JobDetail should be updated if the Job is stateful.
Uses of Interface
org.quartz.Trigger

<table>
<thead>
<tr>
<th>Packages that use Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
</tr>
<tr>
<td><strong>org.quartz.core.jmx</strong></td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
</tr>
<tr>
<td><strong>org.quartz.impl.triggers</strong></td>
</tr>
<tr>
<td><strong>org.quartz.listeners</strong></td>
</tr>
<tr>
<td><strong>org.quartz.locality</strong></td>
</tr>
<tr>
<td><strong>org.quartz.plugins.history</strong></td>
</tr>
<tr>
<td><strong>org.quartz.simpl</strong></td>
</tr>
<tr>
<td><strong>org.quartz.xml</strong></td>
</tr>
</tbody>
</table>

Uses of Trigger in org.quartz

Classes in org.quartz with type parameters of type Trigger

| Class | ScheduleBuilder<T extends Trigger> |
class `TriggerBuilder<T extends Trigger>`

`TriggerBuilder` is used to instantiate `Triggers`.

### Subinterfaces of `Trigger` in `org.quartz`

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalendarIntervalTrigger</code></td>
<td>A concrete <code>Trigger</code> that is used to fire a <code>JobDetail</code> based upon repeating calendar time intervals.</td>
</tr>
<tr>
<td><code>CronTrigger</code></td>
<td>The public interface for inspecting settings specific to a <code>CronTrigger</code>.</td>
</tr>
<tr>
<td><code>SimpleTrigger</code></td>
<td>A <code>Trigger</code> that is used to fire a <code>Job</code> at a given moment in time, and optionally repeated at a specified interval.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz` that return `Trigger`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getTrigger()</code></td>
<td>Get a handle to the <code>Trigger</code> instance that fired the <code>Job</code>.</td>
</tr>
<tr>
<td><code>getTrigger(TriggerKey triggerKey)</code></td>
<td>Get the <code>Trigger</code> instance with the given key.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz` that return types with arguments of type `Trigger`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getScheduleBuilder()</code></td>
<td>Get a <code>ScheduleBuilder</code> that is configured to produce a schedule identical to this trigger's schedule.</td>
</tr>
<tr>
<td><code>getTriggerBuilder()</code></td>
<td>Get a <code>TriggerBuilder</code> that is configured to produce a <code>Trigger</code> identical to this one.</td>
</tr>
<tr>
<td><code>getTriggersOfJob(JobKey jobKey)</code></td>
<td>Get all <code>Triggers</code> that are associated with the identified <code>JobDetail</code>.</td>
</tr>
<tr>
<td><code>newTrigger()</code></td>
<td></td>
</tr>
</tbody>
</table>
Create a new TriggerBuilder with which to define a specification for a Trigger.

## Methods in `org.quartz` with parameters of type `Trigger`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int Trigger.TriggerTimeComparator.compare(Trigger trig1, Trigger trig2)</code></td>
<td>Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural order.</td>
</tr>
<tr>
<td><code>int Trigger.compareTo(Trigger other)</code></td>
<td>Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural order.</td>
</tr>
<tr>
<td><code>void SchedulerListener.jobScheduled(Trigger trigger)</code></td>
<td>Called by the Scheduler when a JobDetail is scheduled.</td>
</tr>
<tr>
<td><code>Date Scheduler.rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)</code></td>
<td>Remove (delete) the Trigger with the given key, and store the new given one - which must be associated with the same job (the new trigger must have the job name &amp; group specified) - however, the new trigger need not have the same name as the old trigger.</td>
</tr>
<tr>
<td><code>Date Scheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)</code></td>
<td>Add the given JobDetail to the Scheduler, and associate the given Trigger with it.</td>
</tr>
<tr>
<td><code>Date Scheduler.scheduleJob(Trigger trigger)</code></td>
<td>Schedule the given Trigger with the Job identified by the trigger's settings.</td>
</tr>
<tr>
<td><code>void TriggerListener.triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</code></td>
<td>Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered(xx) method has been called.</td>
</tr>
<tr>
<td><code>void SchedulerListener.triggerFinalized(Trigger trigger)</code></td>
<td>Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.</td>
</tr>
<tr>
<td><code>void TriggerListener.triggerFired(Trigger trigger, JobExecutionContext context)</code></td>
<td>Called by the Scheduler when a Trigger has fired.</td>
</tr>
</tbody>
</table>
Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.

```java
void TriggerListener.triggerMisfired(Trigger trigger)
```

Called by the Scheduler when a Trigger has misfired.

```java
boolean TriggerListener.vetoJobExecution(Trigger trigger, JobExecutionContext context)
```

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.

Method parameters in **org.quartz** with type arguments of type **Trigger**

```java
void Scheduler.scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace)
```

Schedule all of the given jobs with the related set of triggers.

Constructors in **org.quartz** with parameters of type **Trigger**

```java
ObjectAlreadyExistsException(Trigger offendingTrigger)
```

Create an ObjectAlreadyExistsException and auto-generate a message using the name/group from the given Trigger.

Uses of **Trigger** in **org.quartz.core**

Methods in **org.quartz.core** that return **Trigger**

```java
Trigger QuartzScheduler.getTrigger(TriggerKey triggerKey)
```

Get the Trigger instance with the given name and group.

```java
Trigger RemotableQuartzScheduler.getTrigger(TriggerKey triggerKey)
```

Methods in **org.quartz.core** that return types with arguments of type **Trigger**

```java
List<? extends QuartzScheduler.getTriggersOfJob(JobKey jobKey)
```
Get all Trigger s that are associated with the identified JobDetail.

```java
List<? extends Trigger> RemotableQuartzScheduler.getTriggersOfJob(JobKey jobKey)
```

### Methods in `org.quartz.core` with parameters of type `Trigger`

- **void** `QuartzSchedulerMBeanImpl.jobScheduled(Trigger trigger)`
- **void** `SampledStatisticsImpl.jobScheduled(Trigger trigger)`
- **void** `QuartzScheduler.notifySchedulerListenersFinalized(Trigger trigger)`
- **void** `SchedulerSignalerImpl.notifySchedulerListenersFinalized(Trigger trigger)`
- **void** `QuartzScheduler.notifyTriggerListenersMisfired(Trigger trigger)`
- **void** `SchedulerSignalerImpl.notifyTriggerListenersMisfired(Trigger trigger)`
- **Date** `QuartzScheduler.rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)`
  - Remove (delete) the Trigger with the given name, and store the new one - which must be associated with the same job.
- **Date** `RemotableQuartzScheduler.rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)`
- **Date** `QuartzScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)`
  - Add the Job identified by the given JobDetail to the Scheduler, and the given Trigger with it.
- **Date** `RemotableQuartzScheduler.scheduleJob(JobDetail jobDetail, Trigger trigger)`
- **Date** `QuartzScheduler.scheduleJob(Trigger trigger)`
  - Schedule the given Trigger with the Job identified by the Trigger's
RemotableQuartzScheduler.

```
scheduleJob(Trigger trigger)
```

```java
void QuartzSchedulerMBeanImpl.triggerFinalized(Trigger trigger)
```

### Method parameters in org.quartz.core with type arguments of type Trigger

```java
void QuartzScheduler.scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace)
```

```java
void RemotableQuartzScheduler.scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace)
```

### Uses of Trigger in org.quartz.core.jmx

### Methods in org.quartz.core.jmx with parameters of type Trigger

```java
static CompositeData TriggerSupport.toCompositeData(Trigger trigger)
```

### Method parameters in org.quartz.core.jmx with type arguments of type Trigger

```java
static List<CompositeData> TriggerSupport.toCompositeList(List<? extends Trigger> triggers)
```

```java
static TabularData TriggerSupport.toTabularData(List<? extends Trigger> triggers)
```

### Uses of Trigger in org.quartz.impl
### Methods in `org.quartz.impl` that return `Trigger`

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobExecutionContextImpl</code></td>
<td><code>getTrigger()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteScheduler</code></td>
<td><code>getTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>StdScheduler</code></td>
<td><code>getTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler</code></td>
<td><code>getTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl` that return types with arguments of type `Trigger`

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>List&lt;Trigger&gt;</code></td>
<td><code>RemoteScheduler.getTriggersOfJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>List&lt;Trigger&gt;</code></td>
<td><code>StdScheduler.getTriggersOfJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>List&lt;Trigger&gt;</code></td>
<td><code>RemoteMBeanScheduler.getTriggersOfJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.impl` with parameters of type `Trigger`

<table>
<thead>
<tr>
<th>Date</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RemoteScheduler</code></td>
<td><code>rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>StdScheduler</code></td>
<td><code>rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>RemoteMBeanScheduler</code></td>
<td><code>rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteScheduler.\texttt{scheduleJob}(JobDetail jobDetail, Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Date</td>
<td>StdScheduler.\texttt{scheduleJob}(JobDetail jobDetail, Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteMBeanScheduler.\texttt{scheduleJob}(JobDetail jobDetail, Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteScheduler.\texttt{scheduleJob}(Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>Date</td>
<td>StdScheduler.\texttt{scheduleJob}(Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteMBeanScheduler.\texttt{scheduleJob}(Trigger trigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
</tbody>
</table>

**Method parameters in \texttt{org.quartz.impl} with type arguments of type Trigger**

| void | RemoteScheduler.\texttt{scheduleJobs}(Map\texttt{<JobDetail,List<Trigger>>} triggersAndJobs, boolean replace) |
|------|----------------------------------------------------------------|--------------------------------------------------|
| void | StdScheduler.\texttt{scheduleJobs}(Map\texttt{<JobDetail,List<Trigger>>} triggersAndJobs, boolean replace) |
| void | RemoteMBeanScheduler.\texttt{scheduleJobs}(Map\texttt{<JobDetail,List<Trigger>>} triggersAndJobs, boolean replace) |

**Uses of \texttt{Trigger} in \texttt{org.quartz.impl.jdbcjobstore}**

**Method parameters in \texttt{org.quartz.impl.jdbcjobstore} with type arguments of**

| void | JobStoreSupport.\texttt{storeJobsAndTriggers}(Map\texttt{<JobDetail,List<Trigger>>} triggersAndJobs, boolean replace) |
Uses of **Trigger** in org.quartz.impl.triggers

### Classes in org.quartz.impl.triggers with type parameters of type Trigger

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AbstractTrigger&lt;T extends Trigger&gt;</strong></td>
<td>The base abstract class to be extended by all Triggers.</td>
</tr>
</tbody>
</table>

### Subinterfaces of **Trigger** in org.quartz.impl.triggers

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CoreTrigger</strong></td>
<td>internal interface preserved for backward compatibility</td>
</tr>
</tbody>
</table>

### Classes in org.quartz.impl.triggers that implement **Trigger**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AbstractTrigger&lt;T extends Trigger&gt;</strong></td>
<td>The base abstract class to be extended by all Triggers.</td>
</tr>
<tr>
<td><strong>CalendarIntervalTriggerImpl</strong></td>
<td>A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.</td>
</tr>
<tr>
<td><strong>CronTriggerImpl</strong></td>
<td>A concrete Trigger that is used to fire a JobDetail at given moments in time, defined with Unix 'cron-like' definitions.</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl</strong></td>
<td>A concrete Trigger that is used to fire a JobDetail at a given moment in time, and optionally repeated at a specified interval.</td>
</tr>
</tbody>
</table>

### Methods in org.quartz.impl.triggers with parameters of type **Trigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| **AbstractTrigger.compareTo(Trigger other)** | Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e.
# Uses of `Trigger` in `org.quartz.listeners`

## Methods in `org.quartz.listeners` with parameters of type `Trigger`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>SchedulerListenerSupport.jobScheduled(Trigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>BroadcastSchedulerListener.jobScheduled(Trigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>TriggerListenerSupport.triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>BroadcastTriggerListener.triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>SchedulerListenerSupport.triggerFinalized(Trigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>BroadcastSchedulerListener.triggerFinalized(Trigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>TriggerListenerSupport.triggerFired(Trigger trigger, JobExecutionContext context)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>BroadcastTriggerListener.triggerFired(Trigger trigger, JobExecutionContext context)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>TriggerListenerSupport.triggerMisfired(Trigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>BroadcastTriggerListener.triggerMisfired(Trigger trigger)</code></td>
<td></td>
</tr>
<tr>
<td>boolean <code>TriggerListenerSupport.vetoJobExecution(Trigger trigger, JobExecutionContext context)</code></td>
<td></td>
</tr>
<tr>
<td>boolean <code>BroadcastTriggerListener.vetoJobExecution(Trigger trigger, JobExecutionContext context)</code></td>
<td></td>
</tr>
</tbody>
</table>
### Uses of **Trigger** in **org.quartz.locality**

### Subinterfaces of **Trigger** in **org.quartz.locality**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LocalityTrigger</code></td>
<td>A specialized Trigger that contains Quartz Where information</td>
</tr>
</tbody>
</table>

### Classes in **org.quartz.locality** that implement **Trigger**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DelegatingLocalityTrigger</code></td>
<td>Wrapping a Trigger instance while adding the LocalityAware contract. All Trigger method calls will be delegated to the wrapped Trigger instance</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz.locality** that return types with arguments of type **Trigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ScheduleBuilder&lt;Trigger&gt;</code></td>
<td><strong>DelegatingLocalityTrigger.getScheduleBuilder()</strong></td>
</tr>
<tr>
<td>-</td>
<td>Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.</td>
</tr>
<tr>
<td><code>TriggerBuilder&lt;Trigger&gt;</code></td>
<td><strong>DelegatingLocalityTrigger.getTriggerBuilder()</strong></td>
</tr>
<tr>
<td>-</td>
<td>Get a TriggerBuilder that is configured to produce a Trigger identical to this one.</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz.locality** with parameters of type **Trigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td><strong>DelegatingLocalityTrigger.compareTo(Trigger other)</strong></td>
</tr>
<tr>
<td>-</td>
<td>Compare the next fire time of this Trigger to another by comparing their keys, or in other words, sort them according to the natural (i.e., static) order.</td>
</tr>
<tr>
<td><code>static</code></td>
<td><strong>LocalityTriggerBuilder.localTrigger(Trigger trigger)</strong></td>
</tr>
</tbody>
</table>
Creates a delegating LocalityTriggerBuilder based on the Trigger

**Constructors in org.quartz.locality with parameters of type Trigger**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelegatingLocalityTrigger(Trigger trigger, NodeSpec nodeSpec)</td>
<td>Constructs a LocalityAware Trigger, wrapping an existing Trigger instance, with additional NodeSpec</td>
</tr>
<tr>
<td>LocalityTriggerBuilder(Trigger trigger)</td>
<td></td>
</tr>
</tbody>
</table>

**Uses of Trigger in org.quartz.plugins.history**

**Methods in org.quartz.plugins.history with parameters of type Trigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoggingTriggerHistoryPlugin.triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</td>
<td></td>
</tr>
<tr>
<td>LoggingTriggerHistoryPlugin.triggerFired(Trigger trigger, JobExecutionContext context)</td>
<td></td>
</tr>
<tr>
<td>LoggingTriggerHistoryPlugin.triggerMisfired(Trigger trigger)</td>
<td></td>
</tr>
<tr>
<td>LoggingTriggerHistoryPlugin.vetoJobExecution(Trigger trigger, JobExecutionContext context)</td>
<td></td>
</tr>
</tbody>
</table>

**Uses of Trigger in org.quartz.simpl**

**Method parameters in org.quartz.simpl with type arguments of type Trigger**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMJobStore.storeJobsAndTriggers(Map&lt;JobDetail, List&lt;Trigger&gt;&gt; triggers)</td>
<td></td>
</tr>
</tbody>
</table>


Uses of **Trigger** in **org.quartz.xml**

### Fields in **org.quartz.xml** with type parameters of type **Trigger**

<table>
<thead>
<tr>
<th>protected List&lt;Trigger&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLSchedulingDataProcessor.\loadedTriggers\</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz.xml** that return types with arguments of type **Trigger**

<table>
<thead>
<tr>
<th>protected List&lt;Trigger&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLSchedulingDataProcessor.\getLoadedTriggers()</td>
</tr>
<tr>
<td>Returns a List of triggers loaded from the xml file.</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz.xml** with parameters of type **Trigger**

<table>
<thead>
<tr>
<th>protected void</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLSchedulingDataProcessor.\addTriggerToSchedule(Trigger trig)</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

## org.quartz.Trigger.TriggerState

### Packages that use **org.quartz.Trigger.TriggerState**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
<td></td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of **org.quartz.Trigger.TriggerState**

#### Methods in **org.quartz** that return **org.quartz.Trigger.TriggerState**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger.TriggerState getTriggerState(TriggerKey triggerKey)</td>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td>static Trigger.TriggerState.valueOf(String name)</td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static Trigger.TriggerState[] values()</td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>
### Uses of `Trigger.TriggerState` in `org.quartz.core`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.core</code> that return <code>Trigger.TriggerState</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Trigger.TriggerState</code></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><code>Trigger.TriggerState</code></td>
</tr>
</tbody>
</table>

### Uses of `Trigger.TriggerState` in `org.quartz.impl`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.impl</code> that return <code>Trigger.TriggerState</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Trigger.TriggerState</code></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><code>Trigger.TriggerState</code></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><code>Trigger.TriggerState</code></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

### Uses of `Trigger.TriggerState` in `org.quartz.impl.jdbcjobstore`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.impl.jdbcjobstore</code> that return <code>Trigger.TriggerState</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Trigger.TriggerState</code></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Uses of `Trigger.TriggerState` in `org.quartz.simpl`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.simpl</code> that return <code>Trigger.TriggerState</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RAMJobStore.getTriggerState(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>Get the current state of the identified <code>Trigger</code>.</td>
</tr>
</tbody>
</table>

Uses of Class
org.quartz.Trigger.TriggerTimeComparator

No usage of org.quartz.Trigger.TriggerTimeComparator

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.TriggerBuilder

### Packages that use TriggerBuilder

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of TriggerBuilder in org.quartz

<table>
<thead>
<tr>
<th>Methods in org.quartz that return TriggerBuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerBuilder.&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder.&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder.&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder.&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder.&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder.&lt;CronTrigger&gt;</td>
</tr>
<tr>
<td>TriggerBuilder&lt;CalendarIntervalTrigger&gt;</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>TriggerBuilder&lt;SimpleTrigger&gt;</td>
</tr>
<tr>
<td>TriggerBuilder&lt;? extends Trigger&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>static TriggerBuilder&lt;Trigger&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TriggerBuilder&lt;T&gt;</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
TriggerBuilder\textless T\textgreater

- **usingJobData**(String key)
  - Add the given key-value pair to the Trigger.

- **withDescription**(String)
  - Set the given (human-meaningful) description of the Trigger.

- **withIdentity**(String name)
  - Use a TriggerKey with the given name and default group to identify the Trigger.

- **withIdentity**(String name, TriggerKey)
  - Use the given TriggerKey to identify the Trigger.

- **withPriority**(int priority)
  - Set the Trigger's priority.

- **withSchedule**(ScheduleBuilder scheduleBuilder)
  - Set the ScheduleBuilder that will be used to define the Trigger's schedule.

**Uses of** TriggerBuilder **in** org.quartz.impl.triggers

**Methods in** org.quartz.impl.triggers **that return** TriggerBuilder

- TriggerBuilder\textless T\textgreater
  - AbstractTrigger.\textit{getTriggerBuilder}()

**Uses of** TriggerBuilder **in** org.quartz.locality

**Methods in** org.quartz.locality **that return** TriggerBuilder

- TriggerBuilder\textless Trigger\textgreater
  - DelegatingLocalityTrigger.\textit{getTriggerBuilder}()
    - Get a TriggerBuilder that is configured to
produce a `Trigger` identical to this one.

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.locality</code> with parameters of type <code>TriggerBuilder</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>LocalityTriggerBuilder</code> <code>localTrigger</code> (<code>TriggerBuilder</code>)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructors in <code>org.quartz.locality</code> with parameters of type <code>TriggerBuilder</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LocalityTriggerBuilder</code> (<code>TriggerBuilder</code> triggerBuilder)</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.TriggerKey

<table>
<thead>
<tr>
<th>Packages that use TriggerKey</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl.matchers</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
<tr>
<td>org.quartz.listeners</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
<tr>
<td>org.quartz.simpl</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.xml</td>
<td></td>
</tr>
</tbody>
</table>

Uses of TriggerKey in org.quartz

<table>
<thead>
<tr>
<th>Methods in org.quartz that return TriggerKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerKey</td>
</tr>
<tr>
<td>Trigger.getKey()</td>
</tr>
<tr>
<td>static TriggerKey</td>
</tr>
<tr>
<td>TriggerKey.triggerKey(String name)</td>
</tr>
<tr>
<td>Method Description</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Get the names of all the Triggers in the given group.</td>
</tr>
<tr>
<td>Get the set of Matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td>Determine whether a Trigger with the given identifier already exists within the scheduler.</td>
</tr>
<tr>
<td>Get the Trigger instance with the given key.</td>
</tr>
<tr>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td>Called by the Scheduler when a JobDetail is unscheduled.</td>
</tr>
<tr>
<td>Pause the Trigger with the given key.</td>
</tr>
<tr>
<td>Remove (delete) the Trigger with the given key, and store the new given one - which must be associated with the same job (the new trigger must have the job name &amp; group specified) - however, the new trigger need not have the same name as the trigger.</td>
</tr>
<tr>
<td>Resume (un-pause) the Trigger with the given key.</td>
</tr>
<tr>
<td>Called by the Scheduler when a JobDetail is unscheduled.</td>
</tr>
</tbody>
</table>
Called by the Scheduler when a Trigger has been paused.

```java
void SchedulerListener.triggerResumed(TriggerKey triggerKey)
```

Called by the Scheduler when a Trigger has been un-paused.

```java
boolean Scheduler.unscheduleJob(TriggerKey triggerKey)
```

Remove the indicated Trigger from the scheduler.

```java
TriggerBuilder<T>.withIdentity(TriggerKey key)
```

Use the given TriggerKey to identify the Trigger.

---

### Method parameters in `org.quartz` with type arguments of type `TriggerKey`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void ListenerManager.addTriggerListener(TriggerListener triggerListener, List&lt;Matcher&lt;TriggerKey&gt;&gt; matchers)</code></td>
<td>Add the given TriggerListener to the Scheduler, and register to receive events for Triggers that are matched by ANY of the given Matchers.</td>
</tr>
<tr>
<td><code>boolean ListenerManager.addTriggerListenerMatcher(String listenerName, Matcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt; Scheduler.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Get the names of all the Triggers in the given group.</td>
</tr>
<tr>
<td><code>void Scheduler.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Pause all of the Triggers in the groups matching.</td>
</tr>
<tr>
<td><code>boolean ListenerManager.removeTriggerListenerMatcher(String listenerName, Matcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td><code>void Scheduler.resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Resume (un-pause) all of the Triggers in matching groups.</td>
</tr>
<tr>
<td><code>boolean ListenerManager.setTriggerListenerMatchers(String listenerName, List&lt;Matcher&lt;TriggerKey&gt;&gt; matchers)</code></td>
<td>Set the set of Matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td><code>boolean Scheduler.unscheduleJobs(List&lt;TriggerKey&gt; triggerKeys)</code></td>
<td>Remove all of the indicated Triggers from the scheduler.</td>
</tr>
</tbody>
</table>
# Uses of `TriggerKey` in `org.quartz.core`

## Methods in `org.quartz.core` that return types with arguments of type `TriggerKey`

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Type</th>
<th>Method Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Set&lt;TriggerKey&gt;</code></td>
<td>QuartzScheduler. <code>getTriggerKeys</code> (<code>GroupMatcher&lt;TriggerKey&gt;</code> triggerKeys)</td>
<td>Get the names of all the Triggers in the matching groups.</td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt;</code></td>
<td>RemotableQuartzScheduler. <code>getTriggerKeys</code> (<code>GroupMatcher&lt;TriggerKey&gt;</code> triggerKeys)</td>
<td></td>
</tr>
<tr>
<td><code>List&lt;Matcher&lt;TriggerKey&gt;&gt;</code></td>
<td>ListenerManagerImpl. <code>getTriggerListenerMatchers</code> ()</td>
<td></td>
</tr>
</tbody>
</table>

## Methods in `org.quartz.core` with parameters of type `TriggerKey`

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Type</th>
<th>Method Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean</code></td>
<td>QuartzScheduler. <code>checkExists</code> (<code>TriggerKey</code> triggerKey)</td>
<td>Determine whether a Trigger with the given identifier already exists in the scheduler.</td>
</tr>
<tr>
<td><code>boolean</code></td>
<td>RemotableQuartzScheduler. <code>checkExists</code> (<code>TriggerKey</code> triggerKey)</td>
<td></td>
</tr>
<tr>
<td><code>Trigger</code></td>
<td>QuartzScheduler. <code>getTrigger</code> (<code>TriggerKey</code> triggerKey)</td>
<td>Get the Trigger instance with the given name and group.</td>
</tr>
<tr>
<td><code>Trigger</code></td>
<td>RemotableQuartzScheduler. <code>getTrigger</code> (<code>TriggerKey</code> triggerKey)</td>
<td></td>
</tr>
<tr>
<td><code>Trigger.TriggerState</code></td>
<td>QuartzScheduler. <code>getTriggerState</code> (<code>TriggerKey</code> triggerKey)</td>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td><code>Trigger.TriggerState</code></td>
<td>RemotableQuartzScheduler. <code>getTriggerState</code> (<code>TriggerKey</code> triggerKey)</td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td>QuartzSchedulerMBeanImpl. <code>jobUnscheduled</code> (<code>TriggerKey</code> triggerKey)</td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td>QuartzScheduler. <code>notifySchedulerListenersPausedTrigger</code> ()</td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td>QuartzScheduler. <code>notifySchedulerListenersResumedTrigger</code> ()</td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td>QuartzScheduler. <code>notifySchedulerListenersUnscheduled</code> ()</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.pauseTrigger(TriggerKey triggerKey)</code></td>
<td>Pause the Trigger with the given name.</td>
<td></td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.pauseTrigger(TriggerKey triggerKey)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Date QuartzScheduler.rescheduleJob(TriggerKey triggerKey)</code></td>
<td>Remove (delete) the Trigger with the given name, and which must be associated with the same job.</td>
<td></td>
</tr>
<tr>
<td><code>Date RemotableQuartzScheduler.rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzScheduler.resumeTrigger(TriggerKey triggerKey)</code></td>
<td>Resume (un-pause) the Trigger with the given name.</td>
<td></td>
</tr>
<tr>
<td><code>void RemotableQuartzScheduler.resumeTrigger(TriggerKey triggerKey)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzSchedulerMBeanImpl.triggerPaused(TriggerKey triggerKey)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>void QuartzSchedulerMBeanImpl.triggerResumed(TriggerKey triggerKey)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>boolean QuartzScheduler.unscheduleJob(TriggerKey triggerKey)</code></td>
<td>Remove the indicated Trigger from the scheduler.</td>
<td></td>
</tr>
<tr>
<td><code>boolean RemotableQuartzScheduler.unscheduleJob(TriggerKey triggerKey)</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Method parameters in** org.quartz.core **with type arguments of type TriggerKey**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void ListenerManagerImpl.addTriggerListener(TriggerListener triggerListener, List&lt;Matcher&lt;TriggerKey&gt;&gt; matchers)</code></td>
<td></td>
</tr>
<tr>
<td><code>void ListenerManagerImpl.addTriggerListener(TriggerListener triggerListener, Matcher&lt;TriggerKey&gt; matcher)</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean ListenerManagerImpl.addTriggerListenerMatcher(String listenerName, Matcher&lt;TriggerKey&gt; matcher)</code></td>
<td></td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt; QuartzScheduler.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Get the names of all the Triggers in the matching groups.</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td>RemoteQuartzScheduler. getTriggerKeys( GroupMatcher&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>void</td>
<td>QuartzScheduler. pauseTriggers( GroupMatcher&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>Pause all of the Triggers in the matching groups.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>RemotableQuartzScheduler. pauseTriggers( GroupMatcher&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>void</td>
<td>ListenerManagerImpl. removeTriggerListenerMatcher( Matcher&lt;TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>void</td>
<td>QuartzScheduler. resumeTriggers( GroupMatcher&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>Resume (un-pause) all of the Triggers in the matching groups.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>RemotableQuartzScheduler. resumeTriggers( GroupMatcher&lt;TriggerKey&gt;)</td>
</tr>
<tr>
<td>boolean</td>
<td>ListenerManagerImpl. setTriggerListenerMatchers( String listenerName, List&lt;Matcher&lt;TriggerKey&gt;&gt; matchers)</td>
</tr>
<tr>
<td>boolean</td>
<td>QuartzScheduler. unscheduleJobs( List&lt;TriggerKey&gt; triggerKeys)</td>
</tr>
<tr>
<td>boolean</td>
<td>RemotableQuartzScheduler. unscheduleJobs( List&lt;TriggerKey&gt; triggerKeys)</td>
</tr>
</tbody>
</table>

**Uses of TriggerKey in org.quartz.impl**

<table>
<thead>
<tr>
<th>Uses of TriggerKey in org.quartz.impl</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl that return types with arguments of type TriggerKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
</tr>
<tr>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
</tbody>
</table>

SchedulingContext associated with this instance.
<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>RemoteScheduler.checkExists(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>boolean</td>
<td>StdScheduler.checkExists(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>boolean</td>
<td>RemoteMBeanScheduler.checkExists(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Trigger</td>
<td>RemoteScheduler.getTrigger(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Trigger</td>
<td>StdScheduler.getTrigger(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Trigger</td>
<td>RemoteMBeanScheduler.getTrigger(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Trigger.TriggerState</td>
<td>RemoteScheduler.getTriggerState(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Trigger.TriggerState</td>
<td>StdScheduler.getTriggerState(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Trigger.TriggerState</td>
<td>RemoteMBeanScheduler.getTriggerState(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Trigger.TriggerState</td>
<td>RemoteMBeanScheduler.getTriggerState(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void</td>
<td>RemoteScheduler.pauseTrigger(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void</td>
<td>StdScheduler.pauseTrigger(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>void</td>
<td>RemoteMBeanScheduler.pauseTrigger(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteScheduler.rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Date</td>
<td>StdScheduler.rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
<tr>
<td>Date</td>
<td>RemoteMBeanScheduler.rescheduleJob(TriggerKey trigger newTrigger)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler</td>
</tr>
</tbody>
</table>
Calls the equivalent method on the 'proxied' QuartzScheduler passing the SchedulingContext associated with this instance.

void RemoteScheduler.resumeTrigger(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void StdScheduler.resumeTrigger(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteMBeanScheduler.resumeTrigger(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler passing the SchedulingContext associated with this instance.

boolean RemoteScheduler.unscheduleJob(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

boolean StdScheduler.unscheduleJob(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler.

boolean RemoteMBeanScheduler.unscheduleJob(TriggerKey triggerKey)
Calls the equivalent method on the 'proxied' QuartzScheduler passing the SchedulingContext associated with this instance.

Method parameters in org.quartz.impl with type arguments of type TriggerKey:

Set<TriggerKey> RemoteScheduler.getTriggerKeys(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler.

Set<TriggerKey> StdScheduler.getTriggerKeys(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler.

Set<TriggerKey> RemoteMBeanScheduler.getTriggerKeys(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler passing the SchedulingContext associated with this instance.

void RemoteScheduler.pauseTriggers(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void StdScheduler.pauseTriggers(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteMBeanScheduler.pauseTriggers(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler passing the SchedulingContext associated with this instance.

void RemoteScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler.
void StdScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler.

void RemoteMBeanScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)
Calls the equivalent method on the 'proxied' QuartzScheduler.

SchedulingContext associated with this instance.

boolean RemoteScheduler.unscheduleJobs(List<TriggerKey> triggerKeys)

boolean StdScheduler.unscheduleJobs(List<TriggerKey> triggerKeys)

boolean RemoteMBeanScheduler.unscheduleJobs(List<TriggerKey> triggerKeys)

---

**Uses of TriggerKey in org.quartz.impl.jdbcjobstore**

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl.jdbcjobstore that return TriggerKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerKey</td>
</tr>
<tr>
<td>TriggerKey</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl.jdbcjobstore that return types with arguments or return values of TriggerKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
</tr>
<tr>
<td>protected Set&lt;TriggerKey&gt;</td>
</tr>
<tr>
<td>List&lt;TriggerKey&gt;</td>
</tr>
<tr>
<td>List&lt;TriggerKey&gt;</td>
</tr>
</tbody>
</table>

Get the names of all of the Triggers that match the given group Matcher.

Get the names of all of the Triggers that have misfired.

Get the names of all of the Triggers that have misfired according to given timestamp.
List\<TriggerKey\>

**StdJDBCDelegate**\.`selectMisfiredTriggersInGroupInState`(Connection\String\groupName, String\ state, long\ ts)

Get the names of all of the triggers in the given group and state, that misfired.

List\<TriggerKey\>

**DriverDelegate**\.`selectMisfiredTriggersInGroupInState`(Connection\String\ groupName, String\ state, long\ ts)

Get the names of all of the triggers in the given group and state, that misfired - according to the given timestamp.

List\<TriggerKey\>

**StdJDBCDelegate**\.`selectMisfiredTriggersInState`(ConnectionString\ state, long\ ts)

List\<TriggerKey\>

**DriverDelegate**\.`selectMisfiredTriggersInState`(ConnectionString\ state, long\ ts)

Get the names of all of the triggers in the given state that have misfired, according to the given timestamp.

List\<TriggerKey\>

**StdJDBCDelegate**\.`selectTriggerKeysForJob`(Connection conn, JobKey jobKey)

Get the names of all of the triggers associated with the given job.

List\<TriggerKey\>

**DriverDelegate**\.`selectTriggerKeysForJob`(Connection conn, JobKey jobKey)

Get the names of all of the triggers associated with the given job.

List\<TriggerKey\>

**StdJDBCDelegate**\.`selectTriggersInGroup`(Connection conn, GroupMatcher\<TriggerKey\>\ matcher)

Select all of the triggers contained in a given group.

List\<TriggerKey\>

**DriverDelegate**\.`selectTriggersInGroup`(Connection conn, GroupMatcher\<TriggerKey\>\ matcher)

Select all of the triggers contained in a given group.

List\<TriggerKey\>

**StdJDBCDelegate**\.`selectTriggersInState`(Connection conn, String\ state)

Select all of the triggers in a given state.

List\<TriggerKey\>

**DriverDelegate**\.`selectTriggersInState`(Connection conn, String\ state)

Select all of the triggers in a given state.

List\<TriggerKey\>

**StdJDBCDelegate**\.`selectTriggerToAcquire`(Connection conn, long\ noLaterThan, long\ noEarlierThan)

Select the next trigger which will fire to fire between the two given timestamps in ascending order of fire time, and then descending by priority.

List\<TriggerKey\>

**DriverDelegate**\.`selectTriggerToAcquire`(Connection conn, long\ noLaterThan, long\ noEarlierThan)

Select the next trigger which will fire to fire between the two given timestamps in ascending order of fire time, and then descending by priority.
timestamps in ascending order of fire time, and then descending

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl.jdbcjobstore with parameters of type TriggerKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected boolean JobStoreSupport.checkExists(Connection)</td>
</tr>
<tr>
<td>boolean JobStoreSupport.checkExists(Trigger)</td>
</tr>
<tr>
<td>Determine whether a Trigger exists</td>
</tr>
<tr>
<td>int StdJDBCDelegate.deleteBlobTrigger(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>Delete the cron trigger data</td>
</tr>
<tr>
<td>int SimpleTriggerPersistenceDelegate(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>int SimplePropertiesTriggerPersistenceDelegate(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>int CronTriggerPersistenceDelegate(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>int TriggerPersistenceDelegate.delete(Trigger)</td>
</tr>
<tr>
<td>Delete the base trigger data</td>
</tr>
<tr>
<td>int StdJDBCDelegate.deleteTrigger(Trigger)</td>
</tr>
<tr>
<td>Delete the base trigger data</td>
</tr>
<tr>
<td>int DriverDelegate.deleteTrigger(Connection)</td>
</tr>
<tr>
<td>Delete the base trigger data</td>
</tr>
<tr>
<td>protected void StdJDBCDelegate.deleteTriggerExtension(Trigger.TriggerState)</td>
</tr>
<tr>
<td>JobStoreSupport.getTriggerState(Trigger)</td>
</tr>
<tr>
<td>Get the current state of the identified trigger</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
</tr>
<tr>
<td>SimpleTriggerPersistenceDelegate(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>SimplePropertiesTriggerPersistenceDelegate(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
</tr>
</tbody>
</table>
TriggerPersistenceDelegate.TriggerPropertyBundle

CronTriggerPersistenceDelegate.TriggerKey triggerKey)

TriggerPersistenceDelegate.TriggerPropertyBundle

TriggerPersistenceDelegate. Load

void JobStoreSupport. pauseTrigger(C
Pause the Trigger with the given name.

void JobStoreSupport. pauseTrigger(T
Pause the Trigger with the given name.

protected boolean JobStoreSupport. removeTrigger()

boolean JobStoreSupport. removeTrigger()
Remove (delete) the Trigger.

protected boolean JobStoreSupport. replaceTrigger

boolean JobStoreSupport. replaceTrigger

void JobStoreSupport. resumeTrigger()
Resume (un-pause) the Trigger.

void JobStoreSupport. resumeTrigger()
Resume (un-pause) the Trigger.

protected org.quartz.spi.OperableTrigger JobStoreSupport. retrieveTrigger

org.quartz.spi.OperableTrigger JobStoreSupport. retrieveTrigger
Retrieve the given Trigger.

StdJDBCDelegate. selectJobForTrigger

JobDetail StdJDBCDelegate. selectJobForTrigger

DriverDelegate. selectJobForTrigger

JobDetail DriverDelegate. selectJobForTrigger

Select the job to which the trigger is associated.

Select the job to which the trigger is associated.

Select a trigger.
DriverDelegate.selectTrigger(Connection)
Select a trigger.

String StdJDBCDelegate.selectTriggerState()
Select a trigger's state value.

String DriverDelegate.selectTriggerState()
Select a trigger's state value.

TriggerStatus StdJDBCDelegate.selectTriggerStatus()
Select a trigger's status (state & next fire time).

TriggerStatus DriverDelegate.selectTriggerStatus()
Select a trigger's status (state & next fire time).

void TriggerStatus.setKey(TriggerKey)

void FiredTriggerRecord.setTriggerKey()

boolean StdJDBCDelegate.triggerExists()
Check whether or not a trigger exists.

boolean DriverDelegate.triggerExists(Connection)
Check whether or not a trigger exists.

protected boolean JobStoreSupport.triggerExists()
Check existence of a given trigger.

protected boolean JobStoreSupport.updateMisfired(String newStateIfNotComplete,

int StdJDBCDelegate.updateTriggerState()
Update the state for a given trigger.

int DriverDelegate.updateTriggerState()
Update the state for a given trigger.

int StdJDBCDelegate.updateTriggerStateFromOtherState(String newState, String oldState)
Update the given trigger to the given new state, if it is in the given old state.

int DriverDelegate.updateTriggerStateFromOtherState(String newState, String oldState)
Update the given trigger to the given new state, if it is in the given old state.

int StdJDBCDelegate.updateTriggerStateFromOtherStates(String newState, String oldState1,
Update the given trigger to the given new state, if it is one of the Int DriverDelegate.updateTriggerStateFromOtherStates(String newState, String oldState)

Method parameters in org.quartz.impl.jdbcjobstore with type arguments of:

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>StdJDBCDelegate.deletePausedTriggerGroup(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Delete paused trigger group from database.</td>
</tr>
<tr>
<td>int</td>
<td>DriverDelegate.deletePausedTriggerGroup(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Delete paused trigger group from database.</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td>JobStoreSupport.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Get the names of all of the Triggers that match the given GroupMatcher&lt;TriggerKey&gt;.</td>
</tr>
<tr>
<td>protected Set&lt;TriggerKey&gt;</td>
<td>JobStoreSupport.getTriggerNames(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Get the names of all of the Triggers matching the given GroupMatcher&lt;TriggerKey&gt;.</td>
</tr>
<tr>
<td>boolean</td>
<td>StdJDBCDelegate.hasMisfiredTriggersInState(Connection conn, String state, long ts, int count, List&lt;TriggerKey&gt; result)</td>
<td>Get the names of all of the Triggers in the given state that have misfired according to the given timestamp.</td>
</tr>
<tr>
<td>boolean</td>
<td>DriverDelegate.hasMisfiredTriggersInState(Connection conn, String state, long ts, int count, List&lt;TriggerKey&gt; result)</td>
<td>Get the names of all of the Triggers in the given states that have misfired according to the given timestamp.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.pauseTriggerGroup(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Pause all of the Triggers matching the given GroupMatcher&lt;TriggerKey&gt;.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Pause all of the Triggers matching the given GroupMatcher&lt;TriggerKey&gt;.</td>
</tr>
<tr>
<td>boolean</td>
<td>JobStoreSupport.removeTriggers(List&lt;TriggerKey&gt; triggerKeys)</td>
<td>Remove Triggers from the database.</td>
</tr>
<tr>
<td>boolean</td>
<td>JobStoreSupport.resumeTriggers(List&lt;TriggerKey&gt; triggerKeys)</td>
<td>Resume (un-pause) all of the Triggers matching the given GroupMatcher&lt;TriggerKey&gt;.</td>
</tr>
<tr>
<td>boolean</td>
<td>JobStoreSupport.resumeTriggerGroup(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Resume (un-pause) all of the Triggers matching the given GroupMatcher&lt;TriggerKey&gt;.</td>
</tr>
</tbody>
</table>
Resume (un-pause) all of the Triggers matching the given groupMatcher.

| List<String> | StdJDBCDelegate.selectTriggerGroups(Connection conn, GroupMatcher<TriggerKey> matcher) |
| List<String> | DriverDelegate.selectTriggerGroups(Connection conn, GroupMatcher<TriggerKey> matcher) |
| Set<TriggerKey> | StdJDBCDelegate.selectTriggersInGroup(Connection conn, GroupMatcher<TriggerKey> matcher) |
| Set<TriggerKey> | DriverDelegate.selectTriggersInGroup(Connection conn, GroupMatcher<TriggerKey> matcher) |
| int | StdJDBCDelegate.updateTriggerGroupStateFromOtherState(Connection conn, GroupMatcher<TriggerKey> matcher, String newState, String oldState) |
| int | DriverDelegate.updateTriggerGroupStateFromOtherState(Connection conn, GroupMatcher<TriggerKey> matcher, String newState, String oldState) |
| int | StdJDBCDelegate.updateTriggerGroupStateFromOtherStates(Connection conn, GroupMatcher<TriggerKey> matcher, String newState, String oldState2, String oldState3) |
| int | DriverDelegate.updateTriggerGroupStateFromOtherStates(Connection conn, GroupMatcher<TriggerKey> matcher, String newState, String oldState2, String oldState3) |

**Uses of TriggerKey in org.quartz.impl.matchers**

**Methods in org.quartz.impl.matchers that return types with arguments of**
### type `TriggerKey`

<table>
<thead>
<tr>
<th>static <code>EverythingMatcher&lt;TriggerKey&gt;</code></th>
<th><code>EverythingMatcher.allTriggers()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create an EverythingMatcher that matches all triggers.</td>
</tr>
</tbody>
</table>

### Uses of `TriggerKey` in `org.quartz.impl.triggers`

### Methods in `org.quartz.impl.triggers` that return `TriggerKey`

<table>
<thead>
<tr>
<th><code>TriggerKey</code></th>
<th><code>AbstractTrigger.getKey()</code></th>
</tr>
</thead>
</table>

### Methods in `org.quartz.impl.triggers` with parameters of type `TriggerKey`

| `void` | `AbstractTrigger.setKey(TriggerKey key)` |

### Uses of `TriggerKey` in `org.quartz.listeners`

### Methods in `org.quartz.listeners` with parameters of type `TriggerKey`

<table>
<thead>
<tr>
<th><code>void</code></th>
<th><code>SchedulerListenerSupport.jobUnscheduled(TriggerKey triggerKey)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>BroadcastSchedulerListener.jobUnscheduled(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>SchedulerListenerSupport.triggerPaused(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BroadcastSchedulerListener.triggerPaused(TriggerKey key)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>SchedulerListenerSupport.triggerResumed(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BroadcastSchedulerListener.triggerResumed(TriggerKey key)</code></td>
</tr>
</tbody>
</table>
# Uses of **TriggerKey** in **org.quartz.locality**

## Methods in **org.quartz.locality** that return **TriggerKey**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerKey</td>
<td>DelegatingLocalityTrigger.getKey()</td>
<td>org.quartz.locality</td>
</tr>
</tbody>
</table>

## Methods in **org.quartz.locality** with parameters of type **TriggerKey**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>DelegatingLocalityTrigger.setKey(TriggerKey key)</td>
<td>org.quartz.locality</td>
</tr>
</tbody>
</table>

# Uses of **TriggerKey** in **org.quartz.simpl**

## Fields in **org.quartz.simpl** with type parameters of type **TriggerKey**

<table>
<thead>
<tr>
<th>Type</th>
<th>Field</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>HashMap&lt;String, HashMap&lt;TriggerKey, org.quartz.simpl.TriggerWrapper&gt;&gt;</td>
<td>protected RAMJobStore.triggersByGroup</td>
<td>org.quartz.simpl</td>
</tr>
<tr>
<td>HashMap&lt;TriggerKey, org.quartz.simpl.TriggerWrapper&gt;</td>
<td>protected RAMJobStore.triggersByKey</td>
<td>org.quartz.simpl</td>
</tr>
</tbody>
</table>

## Methods in **org.quartz.simpl** that return types with arguments of type **TriggerKey**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td>RAMJobStore.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>org.quartz.simpl</td>
</tr>
</tbody>
</table>

Get the names of all of the Triggers that match the given groupMatcher.

## Methods in **org.quartz.simpl** with parameters of type **TriggerKey**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Method parameters in org.quartz.simpl with type arguments of type TriggerKey

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td>RAMJobStore.getTriggerKeys(groupMatcher&lt;TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>RAMJobStore.pauseTriggers(groupMatcher&lt;TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>boolean</td>
<td>RAMJobStore.removeTriggers(list&lt;TriggerKey&gt; triggerKeys)</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>RAMJobStore.resumeTriggers(groupMatcher&lt;TriggerKey&gt; matcher)</td>
</tr>
</tbody>
</table>

### Uses of TriggerKey in org.quartz.xml

<table>
<thead>
<tr>
<th>Type</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected List&lt;TriggerKey&gt;</td>
<td>XMLSchedulingDataProcessor.triggersToDelete</td>
</tr>
</tbody>
</table>

### Uses of TriggerKey in org.quartz.xml

- Determine whether a Trigger with the given identifier already exists within the scheduler.
- Get the current state of the identified Trigger.
- Pause the Trigger with the given name.
- Remove (delete) the Trigger with the given name.
- Remove (delete) the Trigger with the given name.
- Resume (un-pause) the Trigger with the given name.
- Retrieve the given Trigger.

- Get the names of all of the Triggers that match the given groupMatcher.
- Pause all of the known Triggers matching.
- Resume (un-pause) all of the Triggers in the given group.
# Uses of Interface

## org.quartz.TriggerListener

<table>
<thead>
<tr>
<th>Packages that use <strong>TriggerListener</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
</tr>
<tr>
<td><strong>org.quartz.listeners</strong></td>
</tr>
<tr>
<td><strong>org.quartz.plugins.history</strong></td>
</tr>
</tbody>
</table>

## Uses of **TriggerListener** in **org.quartz**

### Methods in **org.quartz** that return **TriggerListener**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ListenerManager.getTriggerListener(String name)</strong></td>
<td>Get the <strong>TriggerListener</strong> that has the given name.</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz** that return types with arguments of type **TriggerListener**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ListenerManager.getTriggerListeners()</strong></td>
<td>Get a List containing all of the <strong>TriggerListeners</strong> in the Scheduler.</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz** with parameters of type **TriggerListener**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ListenerManager.addTriggerListener(TriggerListener triggerListe List&lt;Matcher&lt;TriggerKey&gt;&gt; matchers)</strong></td>
<td>Add the given <strong>TriggerListener</strong> to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers.</td>
</tr>
<tr>
<td><strong>ListenerManager.addTriggerListener(TriggerListener triggerListe</strong></td>
<td></td>
</tr>
</tbody>
</table>
Add the given `TriggerListener` to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers.

## Uses of `TriggerListener` in `org.quartz.core`

### Methods in `org.quartz.core` that return `TriggerListener`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzScheduler.getInternalTriggerListener(String name)</code></td>
<td>Get the <em>internal</em> <code>TriggerListener</code> that has the given name.</td>
</tr>
<tr>
<td><code>ListenerManagerImpl.getTriggerListener(String name)</code></td>
<td></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.core` that return types with arguments of type `TriggerListener`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzScheduler.getInternalTriggerListeners()</code></td>
<td>Get a list containing all of the <code>TriggerListeners</code> in the Scheduler's <em>internal</em> list.</td>
</tr>
<tr>
<td><code>ListenerManagerImpl.getTriggerListeners()</code></td>
<td></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.core` with parameters of type `TriggerListener`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzScheduler.addInternalTriggerListener(TriggerListener triggerListener)</code></td>
<td>Add the given <code>TriggerListener</code> to the Scheduler's <em>internal</em> list.</td>
</tr>
<tr>
<td><code>ListenerManagerImpl.addTriggerListener(TriggerListener triggerListener, List&lt;Matcher&lt;TriggerKey&gt;&gt; matchers)</code></td>
<td></td>
</tr>
<tr>
<td><code>ListenerManagerImpl.addTriggerListener(TriggerListener triggerListener, Matcher&lt;TriggerKey&gt;... matchers)</code></td>
<td></td>
</tr>
<tr>
<td><code>ListenerManagerImpl.addTriggerListener(TriggerListener triggerListener, Matcher&lt;TriggerKey&gt; matcher)</code></td>
<td></td>
</tr>
</tbody>
</table>
Uses of **TriggerListener** in **org.quartz.listeners**

<table>
<thead>
<tr>
<th>Classes in <strong>org.quartz.listeners</strong> that implement <strong>TriggerListener</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Methods in **org.quartz.listeners** that return types with arguments of type **TriggerListener**

| List<TriggerListener> | BroadcastTriggerListener. **getListeners**() |

Methods in **org.quartz.listeners** with parameters of type **TriggerListener**

| void | BroadcastTriggerListener. **addListener**(TriggerListener listener) |
| boolean | BroadcastTriggerListener. **removeListener**(TriggerListener listener) |

Uses of **TriggerListener** in **org.quartz.plugins.history**

<table>
<thead>
<tr>
<th>Classes in <strong>org.quartz.plugins.history</strong> that implement <strong>TriggerListener</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.TriggerUtils

No usage of org.quartz.TriggerUtils

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.UnableToInterruptJobException**

## Packages that use `UnableToInterruptJobException`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Uses of `UnableToInterruptJobException` in `org.quartz`

## Methods in `org.quartz` that throw `UnableToInterruptJobException`

<table>
<thead>
<tr>
<th>void</th>
<th><code>InterruptableJob.interrupt()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Called by the Scheduler when a user interrupts the Job.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th><code>Scheduler.interrupt(JobKey jobKey)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which must be an implementor of the InterruptableJob interface.</td>
</tr>
</tbody>
</table>

## Uses of `UnableToInterruptJobException` in `org.quartz.core`

## Methods in `org.quartz.core` that throw ` UnableToInterruptJobException`

<table>
<thead>
<tr>
<th>boolean</th>
<th><code>QuartzScheduler.interrupt(JobKey jobKey)</code></th>
</tr>
</thead>
</table>
Interrupt all instances of the identified InterruptableJob executing in this Scheduler instance.

```java
boolean RemotableQuartzScheduler.interrupt(JobKey jobKey)
```

## Uses of `UnableToInterruptJobException` in `org.quartz.impl`

### Methods in `org.quartz.impl` that throw `UnableToInterruptJobException`

```java
boolean RemoteScheduler.interrupt(JobKey jobKey)
```

```java
boolean StdScheduler.interrupt(JobKey jobKey)
```

```java
boolean RemoteMBeanScheduler.interrupt(JobKey jobKey)
```
Class JobRunShell

java.lang.Object
   └ org.quartz.listeners.SchedulerListenerSupport
       └ org.quartz.core.JobRunShell

All Implemented Interfaces:
   Runnable, SchedulerListener

Direct Known Subclasses:
   JTAJobRunShell

public class JobRunShell
   extends SchedulerListenerSupport
   implements Runnable

JobRunShell instances are responsible for providing the 'safe' environment for Jobs to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the Trigger with the Job's completion code, etc.

A JobRunShell instance is created by a JobRunShellFactory on behalf of the QuartzSchedulerThread which then runs the shell in a thread from the configured ThreadPool when the scheduler determines that a Job has been triggered.

Author:
   James House

See Also:
   JobRunShellFactory, QuartzSchedulerThread, Job, Trigger

Field Summary

| protected org.quartz.spi.TriggerFiredBundle | firedTriggerBundle |
| protected JobExecutionContextImpl | jec |
protected QuartzScheduler **qs**

protected Scheduler **scheduler**

protected boolean **shutdownRequested**

### Constructor Summary

**JobRunShell** *(Scheduler scheduler, org.quartz.spi.TriggerFiredBundle bndle)*

Create a JobRunShell instance with the given settings.

### Method Summary

protected void **begin()**

protected void **complete**(boolean successfulExecution)

boolean **completeTriggerRetryLoop**(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)

protected org.slf4j.Logger **getLog()**

Get the Logger for this class's category.

void **initialize**(QuartzScheduler qs)

void **passivate()**

void **requestShutdown()**

void **run()**

void **schedulerShuttingdown()**

Called by the Scheduler to inform the listener that it has l
shutdown sequence.

```java
boolean vetoedJobRetryLoop(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instCode)
```

### Methods inherited from class `org.quartz.listeners.SchedulerListenerSupport`

- `jobAdded`, `jobDeleted`, `jobPaused`, `jobResumed`, `jobScheduled`, `jobsPaused`, `jobsResumed`, `jobUnscheduled`, `schedulerError`, `schedulerInStandbyMode`, `schedulerShutdown`, `schedulerStarted`, `schedulingDataCleared`, `triggerFinalized`, `triggerPaused`, `triggerResumed`, `triggersPaused`, `triggersResumed`

### Methods inherited from class `java.lang.Object`

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Field Detail

**jec**

protected `JobExecutionContextImpl` jec

**qs**

protected `QuartzScheduler` qs

**firedTriggerBundle**

protected `org.quartz.spi.TriggerFiredBundle` firedTriggerBundle

**scheduler**
protected `Scheduler` scheduler

shutdownRequested

protected volatile boolean shutdownRequested

Constructor Detail

JobRunShell

public `JobRunShell`(`Scheduler` scheduler, org.quartz.spi.TriggerFiredBundle bndle)

Create a JobRunShell instance with the given settings.

Parameters:

jobRunShellFactory - A handle to the JobRunShellFactory that produced this JobRunShell.
scheduler - The Scheduler instance that should be made available within the JobExecutionContext.
schdCtxt - the SchedulingContext that should be used by the JobRunShell when making updates to the JobStore.

Method Detail

schedulerShuttingdown

public void `schedulerShuttingdown`()  

Description copied from interface: `SchedulerListener`  

Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

Specified by:

`schedulerShuttingdown` in interface `SchedulerListener`  

Overrides:
schedulerShuttingdown in class SchedulerListenerSupport

---

**getLog**

protected org.slf4j.Logger **getLog()**

*Description copied from class: SchedulerListenerSupport*
Get the Logger for this class's category. This should be used by subclasses for logging.

**Overrides:**

getLog in class SchedulerListenerSupport

---

**initialize**

public void **initialize(QuartzScheduler qs)**

*Throws:*

SchedulerException

---

**requestShutdown**

public void **requestShutdown()**

---

**run**

public void **run()**

*Specified by:*

run in interface Runnable

---

**begin**

protected void **begin()**
complete

protected void complete(boolean successfulExecution)
   throws SchedulerException

   Throws:
   SchedulerException

passivate

public void passivate()

completeTriggerRetryLoop

public boolean completeTriggerRetryLoop(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction vetoedJobRetryLoop

public boolean vetoedJobRetryLoop(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction vetoedJobRetryLoop

public boolean vetoedJobRetryLoop(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction vetoedJobRetryLoop

Copyright 2001-2011, Terracotta, Inc.
**Interface JobRunShellFactory**

All Known Implementing Classes:
- `JTAAnnotationAwareJobRunShellFactory`, `JTAJobRunShellFactory`, `StdJobRunShellFactory`

```java
public interface JobRunShellFactory
```

Responsible for creating the instances of `JobRunShell` to be used within the `QuartzScheduler` instance.

**Author:**
James House

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobRunShell</td>
<td><code>createJobRunShell(org.quartz.spi.TriggerFiredBundle bundle)</code></td>
<td>Called by the QuartzSchedulerThread to obtain instances of JobRunShell.</td>
</tr>
<tr>
<td><code>initialize</code></td>
<td><code>void initialize(Scheduler scheduler)</code></td>
<td>Initialize the factory, providing a handle to the Scheduler that should be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>made available within the JobRunShell and the JobExecutionContexts within it.</td>
</tr>
</tbody>
</table>

### Method Detail

**initialize**

```java
void initialize(Scheduler scheduler)
```

Initializes the factory, providing a handle to the `Scheduler` that should be made available within the `JobRunShell` and the `JobExecutionContext` within it.
Throws:
SchedulerConfigException

createJobRunShell

JobRunShell createJobRunShell(org.quartz.spi.TriggerFiredBundle bund
throws SchedulerException

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

Throws:
SchedulerException

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public class **ListenerManagerImpl**

extends **Object**

implements **ListenerManager**

---

### Constructor Summary

**ListenerManagerImpl()**

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>addJobListener(JobListener jobListener, List&lt;Matcher&lt;JobKey&gt;&gt; matchers)</strong></td>
<td>Add the given JobListener to the Scheduler, and receive events for Jobs that are matched by ANY of the given Matchers.</td>
</tr>
<tr>
<td>void</td>
<td><strong>addJobListener(JobListener jobListener, Matcher&lt;JobKey&gt;... matchers)</strong></td>
<td>Add the given JobListener to the Scheduler, and receive events for Jobs that are matched by ANY of the given Matchers.</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>addJobListenerMatcher(String listenerName, Matcher&lt;JobKey&gt; matcher)</strong></td>
<td>Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.</td>
</tr>
<tr>
<td>void</td>
<td><strong>addSchedulerListener(SchedulerListener scheduler)</strong></td>
<td>Register the given SchedulerListener with the Scheduler.</td>
</tr>
</tbody>
</table>
void addTriggerListener(TriggerListener triggerListener, List<Matcher<TriggerKey>> matchers)

    Add the given TriggerListener to the Scheduler to receive events for Triggers that are matched by ANY of the Matchers.

void addTriggerListener(TriggerListener triggerListener, Matcher<TriggerKey>... matchers)

    Add the given TriggerListener to the Scheduler to receive events for Triggers that are matched by ANY of the Matchers.

void addTriggerListener(TriggerListener triggerListener, Matcher<TriggerKey> matcher)

boolean addTriggerListenerMatcher(String listenerName, Matcher<TriggerKey> matcher)

    Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

JobListener getJobListener(String name)

    Get the JobListener that has the given name.

List<Matcher<JobKey>> getJobListenerMatchers(String listenerName)

    Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

List<JobListener> getJobListeners()

    Get a List containing all of the JobListeners in the Scheduler.

List<SchedulerListener> getScheduledListeners()

    Get a List containing all of the SchedulerListeners with the Scheduler.

TriggerListener getTriggerListener(String name)

    Get the TriggerListener that has the given name.

List<Matcher<TriggerKey>> getTriggerListenerMatchers(String listenerName)

    Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

List<TriggerListener> getTriggerListeners()

    Get a List containing all of the TriggerListeners in the Scheduler.

boolean removeJobListener(String name)
Remove the identified JobListener from the Scheduler.

```java
boolean removeJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)
```

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

```java
boolean removeSchedulerListener(SchedulerListener schedulerListener)
```

Remove the given SchedulerListener from the Scheduler.

```java
boolean removeTriggerListener(String name)
```

Remove the identified TriggerListener from the Scheduler.

```java
boolean removeTriggerListenerMatcher(String listenerName, Matcher<TriggerKey> matcher)
```

Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

```java
boolean setJobListenerMatchers(String listenerName, List<Matcher<JobKey>> matchers)
```

Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

```java
boolean setTriggerListenerMatchers(String listenerName, List<Matcher<TriggerKey>> matchers)
```

Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

---

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

---

**Constructor Detail**

**ListenerManagerImpl**

```java
public ListenerManagerImpl()
```

---

**Method Detail**

**addJobListener**
public void addJobListener(JobListener jobListener,
        Matcher<JobKey>... matchers)

Description copied from interface: ListenerManager
Add the given JobListener to the Scheduler, and register it to receive
events for Jobs that are matched by ANY of the given Matchers. If no
matchers are provided, the EverythingMatcher will be used.

Specified by:
    addJobListener in interface ListenerManager
See Also:
    Matcher, EverythingMatcher

addJobListener

public void addJobListener(JobListener jobListener,
        List<Matcher<JobKey>> matchers)

Description copied from interface: ListenerManager
Add the given JobListener to the Scheduler, and register it to receive
events for Jobs that are matched by ANY of the given Matchers. If no
matchers are provided, the EverythingMatcher will be used.

Specified by:
    addJobListener in interface ListenerManager
See Also:
    Matcher, EverythingMatcher

addJobListenerMatcher

public boolean addJobListenerMatcher(String listenerName,
        Matcher<JobKey> matcher)

Description copied from interface: ListenerManager
Add the given Matcher to the set of matchers for which the listener will
receive events if ANY of the matchers match.

Specified by:
addJobListenerMatcher in interface ListenerManager

**Parameters:**
- listenerName - the name of the listener to add the matcher to
- matcher - the additional matcher to apply for selecting events

**Returns:**
true if the identified listener was found and updated

---

removeJobListenerMatcher

```java
public boolean removeJobListenerMatcher(String listenerName, Matcher<JobKey> matcher)
```

**Description copied from interface:** ListenerManager
Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

**Specified by:**
removeJobListenerMatcher in interface ListenerManager

**Parameters:**
- listenerName - the name of the listener to add the matcher to
- matcher - the additional matcher to apply for selecting events

**Returns:**
true if the given matcher was found and removed from the listener's list of matchers

---

getJobListenerMatchers

```java
public List<Matcher<JobKey>> getJobListenerMatchers(String listenerName)
```

**Description copied from interface:** ListenerManager
Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

**Specified by:**
getJobListenerMatchers in interface ListenerManager

**Parameters:**
- listenerName - the name of the listener to add the matcher to

---
Returns:
the matchers registered for selecting events for the identified listener

setJobListenerMatchers

public boolean setJobListenerMatchers(String listenerName,
        List<Matcher<JobKey>> matchers)

Description copied from interface: ListenerManager
Set the set of Matchers for which the listener will receive events if ANY of the matchers match.

Removes any existing matchers for the identified listener!

Specified by:
setJobListenerMatchers in interface ListenerManager

Parameters:
listenerName - the name of the listener to add the matcher to
matchers - the matchers to apply for selecting events

Returns:
true if the given matcher was found and removed from the listener's list of matchers

removeJobListener

public boolean removeJobListener(String name)

Description copied from interface: ListenerManager
Remove the identified JobListener from the Scheduler.

Specified by:
removeJobListener in interface ListenerManager

Returns:
true if the identified listener was found in the list, and removed.

gettoJobListeners
public List<JobListener> getJobListeners()

Description copied from interface: ListenerManager
Get a List containing all of the JobListeners in the Scheduler.

Specified by:
getJobListeners in interface ListenerManager

getJobListener

public JobListener getJobListener(String name)

Description copied from interface: ListenerManager
Get the JobListener that has the given name.

Specified by:
getJobListener in interface ListenerManager

addTriggerListener

public void addTriggerListener(TriggerListener triggerListener,
Matcher<TriggerKey>... matchers)

Description copied from interface: ListenerManager
Add the given TriggerListener to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers. If no matcher is provided, the EverythingMatcher will be used.

Specified by:
addTriggerListener in interface ListenerManager
See Also:
Matcher, EverythingMatcher

addTriggerListener

public void addTriggerListener(TriggerListener triggerListener,
List<Matcher<TriggerKey>>... matchers)
Description copied from interface: **ListenerManager**
Add the given TriggerListener to the Scheduler, and register it to receive events for Triggers that are matched by ANY of the given Matchers. If no matcher is provided, the EverythingMatcher will be used.

**Specified by:**
- `addTriggerListener` in interface **ListenerManager**

**See Also:**
- **Matcher**, **EverythingMatcher**

---

**addTriggerListener**

**public void addTriggerListener** *(TriggerListener triggerListener, Matcher<TriggerKey> matcher)*

---

**addTriggerListenerMatcher**

**public boolean addTriggerListenerMatcher** *(String listenerName, Matcher<TriggerKey> matcher)*

Description copied from interface: **ListenerManager**
Add the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

**Specified by:**
- `addTriggerListenerMatcher` in interface **ListenerManager**

**Parameters:**
- `listenerName` - the name of the listener to add the matcher to
- `matcher` - the additional matcher to apply for selecting events

**Returns:**
- true if the identified listener was found and updated

---

**removeTriggerListenerMatcher**

**public boolean removeTriggerListenerMatcher** *(String listenerName, Matcher<TriggerKey> matcher)*
**Description copied from interface:** ListenerManager
Remove the given Matcher to the set of matchers for which the listener will receive events if ANY of the matchers match.

**Specified by:**
removeTriggerListenerMatcher in interface ListenerManager

**Parameters:**
listenerName - the name of the listener to add the matcher to
matcher - the additional matcher to apply for selecting events

**Returns:**
true if the given matcher was found and removed from the listener's list of matchers

---

**getTriggerListenerMatchers**

```java
public List<Matcher<TriggerKey>> getTriggerListenerMatchers(String listenerName)
```

**Description copied from interface:** ListenerManager
Get the set of Matchers for which the listener will receive events if ANY of the matchers match.

**Specified by:**
getTriggerListenerMatchers in interface ListenerManager

**Parameters:**
listenerName - the name of the listener to add the matcher to

**Returns:**
the matchers registered for selecting events for the identified listener

---

**setTriggerListenerMatchers**

```java
public boolean setTriggerListenerMatchers(String listenerName,
                                          List<Matcher<TriggerKey>> matchers)
```

**Description copied from interface:** ListenerManager
Set the set of Matchers for which the listener will receive events if ANY of the matchers match.
Removes any existing matchers for the identified listener!

Specified by:
\[\text{setTriggerListenerMatchers}\] in interface \text{ListenerManager}

Parameters:
- \text{listenerName} - the name of the listener to add the matcher to
- \text{matchers} - the matchers to apply for selecting events

Returns:
true if the given matcher was found and removed from the listener's list of matchers

---

\text{removeTriggerListener}

public boolean \text{removeTriggerListener}(\text{String name})

Description copied from interface: \text{ListenerManager}
Remove the identified TriggerListener from the Scheduler.

Specified by:
\[\text{removeTriggerListener}\] in interface \text{ListenerManager}

Returns:
true if the identified listener was found in the list, and removed.

---

\text{getTriggerListeners}

public \text{List<TriggerListener>} \text{getTriggerListeners}()

Description copied from interface: \text{ListenerManager}
Get a List containing all of the TriggerListeners in the Scheduler.

Specified by:
\[\text{getTriggerListeners}\] in interface \text{ListenerManager}

---

\text{getTriggerListener}

public \text{TriggerListener} \text{getTriggerListener}(\text{String name})
Description copied from interface: ListenerManager
Get the TriggerListener that has the given name.

Specified by:
    getTriggerListener in interface ListenerManager

addSchedulerListener

public void addSchedulerListener(SchedulerListener schedulerListener)

Description copied from interface: ListenerManager
Register the given SchedulerListener with the Scheduler.

Specified by:
    addSchedulerListener in interface ListenerManager

removeSchedulerListener

public boolean removeSchedulerListener(SchedulerListener schedulerListener)

Description copied from interface: ListenerManager
Remove the given SchedulerListener from the Scheduler.

Specified by:
    removeSchedulerListener in interface ListenerManager
Returns:
    true if the identified listener was found in the list, and removed.

getSchedulerListeners

public List<SchedulerListener> getSchedulerListeners()

Description copied from interface: ListenerManager
Get a List containing all of the SchedulerListeners registered with the Scheduler.

Specified by:
getSchedulerListeners in interface ListenerManager
**org.quartz.core Class NullSampledStatisticsImpl**

*java.lang.Object*

Inherits interface from: *SampledStatistics*

---

**All Implemented Interfaces:**

- SampledStatistics

---

```java
public class NullSampledStatisticsImpl extends Object implements SampledStatistics
```

---

### Constructor Summary

**NullSampledStatisticsImpl()**

---

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>getJobsCompletedMostRecentSample()</td>
</tr>
<tr>
<td>long</td>
<td>getJobsExecutingMostRecentSample()</td>
</tr>
<tr>
<td>long</td>
<td>getJobsScheduledMostRecentSample()</td>
</tr>
<tr>
<td>void</td>
<td>shutdown()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
NullSampledStatisticsImpl

public NullSampledStatisticsImpl()

Method Detail

getJobsCompletedMostRecentSample

public long getJobsCompletedMostRecentSample()

Specified by:
getJobsCompletedMostRecentSample in interface SampledStatistics

getJobsExecutingMostRecentSample

public long getJobsExecutingMostRecentSample()

Specified by:
getJobsExecutingMostRecentSample in interface SampledStatistics

getJobsScheduledMostRecentSample

public long getJobsScheduledMostRecentSample()

Specified by:
getJobsScheduledMostRecentSample in interface SampledStatistics

shutdown

public void shutdown()
Specified by:

shutdown in interface SampledStatistics
org.quartz.core

Interfaces
JobRunShellFactory
RemotableQuartzScheduler
SampledStatistics

Classes
JobRunShell
ListenerManagerImpl
NullSampledStatisticsImpl
QuartzScheduler
QuartzSchedulerMBeanImpl
QuartzSchedulerResources
QuartzSchedulerThread
SampledStatisticsImpl
SchedulerSignalerImpl
Package org.quartz.core

Contains the core classes and interfaces for the Quartz job scheduler.

See: Description

### Interface Summary

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobRunShellFactory</strong></td>
<td>Responsible for creating the instances of JobRunShell to be used within the QuartzScheduler instance.</td>
</tr>
<tr>
<td><strong>RemovableQuartzScheduler</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SampledStatistics</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobRunShell</strong></td>
<td>JobRunShell instances are responsible for providing the 'safe' environment for Job s to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the Trigger with the Job's completion code, etc.</td>
</tr>
<tr>
<td><strong>ListenerManagerImpl</strong></td>
<td></td>
</tr>
<tr>
<td><strong>NullSampledStatisticsImpl</strong></td>
<td></td>
</tr>
<tr>
<td><strong>QuartzScheduler</strong></td>
<td>This is the heart of Quartz, an indirect implementation of the Scheduler interface, containing methods to schedule Jobs, register JobListener instances, etc.</td>
</tr>
<tr>
<td><strong>QuartzSchedulerMBeanImpl</strong></td>
<td></td>
</tr>
<tr>
<td><strong>QuartzSchedulerResources</strong></td>
<td>Contains all of the resources (JobStore, ThreadPool, etc.) necessary to create a QuartzScheduler instance.</td>
</tr>
<tr>
<td><strong>QuartzSchedulerThread</strong></td>
<td>The thread responsible for performing the work of firing Trigger s that are registered with the</td>
</tr>
<tr>
<td>SampledStatisticsImpl</td>
<td>QuartzScheduler.</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>SchedulerSignalerImpl</td>
<td>An interface to be used by JobStore instances in order to communicate signals back to the QuartzScheduler.</td>
</tr>
</tbody>
</table>
Contains the core classes and interfaces for the Quartz job scheduler.

See the Quartz project for more information.
Hierarchy For Package org.quartz.core

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.core.**ListenerManagerImpl** (implements org.quartz.**ListenerManager**)
  - org.quartz.core.**NullSampledStatisticsImpl** (implements org.quartz.core.**SampledStatistics**)
  - org.quartz.core.**QuartzScheduler** (implements org.quartz.core.**RemotableQuartzScheduler**)
  - org.quartz.core.**QuartzSchedulerResources**
  - org.quartz.listeners.**SchedulerListenerSupport** (implements org.quartz.**SchedulerListener**)
    - org.quartz.core.**JobRunShell** (implements java.lang.**Runnable**)
    - org.quartz.core.**SampledStatisticsImpl** (implements org.quartz.**JobListener**, org.quartz.core.**SampledStatistics**, org.quartz.**SchedulerListener**)
  - org.quartz.core.**SchedulerSignalerImpl** (implements org.quartz.spi.**SchedulerSignaler**)
  - javax.management.**StandardMBean** (implements javax.management.**DynamicMBean**, javax.management.**MBeanRegistration**)
    - org.quartz.core.**QuartzSchedulerMBeanImpl** (implements org.quartz.**JobListener**, javax.management.**NotificationEmitter**, org.quartz.core.jmx.**QuartzSchedulerMBean**, org.quartz.**SchedulerListener**)
  - java.lang.**Thread** (implements java.lang.**Runnable**)
    - org.quartz.core.**QuartzSchedulerThread**
Interface Hierarchy

- org.quartz.core.JobRunShellFactory
- java.rmi.Remote
  - org.quartz.core.RemotableQuartzScheduler
- org.quartz.core.SampledStatistics

Overview  Package  Class  Use  Deprecated  Index  Help
# Uses of Package

**org.quartz.core**

## Packages that use org.quartz.core

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.ee.jta</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Classes in org.quartz.core used by org.quartz.core

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobRunShell</td>
<td>JobRunShell instances are responsible for providing the 'safe' environment for Job s to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the Trigger with the Job's completion code, etc.</td>
</tr>
<tr>
<td>JobRunShellFactory</td>
<td>Responsible for creating the instances of JobRunShell to be used within the QuartzScheduler instance.</td>
</tr>
<tr>
<td>QuartzScheduler</td>
<td>This is the heart of Quartz, an indirect implementation of the Scheduler interface, containing methods to schedule Jobs, register JobListener instances, etc.</td>
</tr>
<tr>
<td>QuartzSchedulerResources</td>
<td>Contains all of the resources (JobStore,ThreadPool, etc.) necessary to create a QuartzScheduler instance.</td>
</tr>
<tr>
<td>QuartzSchedulerThread</td>
<td>The thread responsible for performing the work of firing Trigger s that are registered with the QuartzScheduler.</td>
</tr>
<tr>
<td>RemotableQuartzScheduler</td>
<td></td>
</tr>
</tbody>
</table>
### Classes in org.quartz.core used by org.quartz.ee.jta

**JobRunShell**  
JobRunShell instances are responsible for providing the 'safe' environment for Jobs to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the Trigger with the Job's completion code, etc.

**JobRunShellFactory**  
Responsible for creating the instances of JobRunShell to be used within the QuartzScheduler instance.

---

### Classes in org.quartz.core used by org.quartz.impl

**JobRunShell**  
JobRunShell instances are responsible for providing the 'safe' environment for Jobs to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the Trigger with the Job's completion code, etc.

**JobRunShellFactory**  
Responsible for creating the instances of JobRunShell to be used within the QuartzScheduler instance.

**QuartzScheduler**  
This is the heart of Quartz, an indirect implementation of the Scheduler interface, containing methods to schedule Jobs, register JobListener instances, etc.

**QuartzSchedulerResources**  
Contains all of the resources (JobStore, ThreadPool, etc.) necessary to create a QuartzScheduler instance.

**RemotableQuartzScheduler**
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public class QuartzScheduler

extends Object

implements RemotableQuartzScheduler

This is the heart of Quartz, an indirect implementation of the Scheduler interface, containing methods to schedule Jobs, register JobListener instances, etc.

Author:
James House

See Also:
Scheduler, QuartzSchedulerThread, JobStore, ThreadPool

Constructor Summary

QuartzScheduler(QuartzSchedulerResources resources, long idleWaitTime, long dbRetryInterval)

Create a QuartzScheduler with the given configuration properties.

Method Summary

void addCalendar(String calName, Calendar calendar, boolean updateTriggers)
Add (register) the given calendar to the Scheduler.

void addInternalJobListener(JobListener jobListener)
Add the given JobListener to the Scheduler.

void addInternalSchedulerListener(SchedulerListener schedulerListener)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addInternalTriggerListener(TriggerListener triggerListener)</code></td>
<td>Add the given TriggerListener to the Scheduler's internal listeners.</td>
</tr>
<tr>
<td><code>addJob(JobDetail jobDetail, boolean replace)</code></td>
<td>Add the given Job to the Scheduler - with no associated Trigger.</td>
</tr>
<tr>
<td><code>addNoGCObject(Object obj)</code></td>
<td>Add the given Object to the Scheduler.</td>
</tr>
<tr>
<td><code>checkExists(JobKey jobKey)</code></td>
<td>Determine whether a Job with the given identifier already exists within the scheduler.</td>
</tr>
<tr>
<td><code>checkExists(TriggerKey triggerKey)</code></td>
<td>Determine whether a Trigger with the given identifier already exists within the scheduler.</td>
</tr>
<tr>
<td><code>clear()</code></td>
<td>Clears (deletes!) all scheduling data - all Jobs.</td>
</tr>
<tr>
<td><code>deleteCalendar(String calName)</code></td>
<td>Delete the identified Calendar from the Scheduler.</td>
</tr>
<tr>
<td><code>deleteJob(JobKey jobKey)</code></td>
<td>Delete the identified Job from the Scheduler - along with any associated Triggers.</td>
</tr>
<tr>
<td><code>deleteJobs(List&lt;JobKey&gt; jobKeys)</code></td>
<td>Delete the identified Jobs from the Scheduler - along with any associated Triggers.</td>
</tr>
<tr>
<td><code>getCalendar(String calName)</code></td>
<td>Get the Calendar instance with the given name.</td>
</tr>
<tr>
<td><code>getCalendarNames()</code></td>
<td>Get the names of all registered Calendars.</td>
</tr>
<tr>
<td><code>getCurrentlyExecutingJobs()</code></td>
<td>Return a list of JobExecutionContext objects for Jobs executing in this Scheduler instance.</td>
</tr>
<tr>
<td><code>getInternalJobListener(String name)</code></td>
<td>Get the internal JobListener that has the given name.</td>
</tr>
<tr>
<td><code>getInternalJobListeners()</code></td>
<td>Get a List containing all of the JobListener list.</td>
</tr>
<tr>
<td>Class/Interface</td>
<td>Method</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>List&lt;SchedulerListener&gt;</td>
<td>getInternalSchedulerListeners()</td>
</tr>
<tr>
<td>TriggerListener</td>
<td>getInternalTriggerListener(String name)</td>
</tr>
<tr>
<td>List&lt;TriggerListener&gt;</td>
<td>getInternalTriggerListeners()</td>
</tr>
<tr>
<td>JobDetail</td>
<td>getJobDetail(JobKey jobKey)</td>
</tr>
<tr>
<td>org.quartz.spi.JobFactory</td>
<td>getJobFactory()</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>getJobGroupNames()</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt;</td>
<td>getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>Class</td>
<td>getJobStoreClass()</td>
</tr>
<tr>
<td>ListenerManager</td>
<td>getListenerManager()</td>
</tr>
<tr>
<td>org.slf4j.Logger</td>
<td>getLog()</td>
</tr>
<tr>
<td>Set</td>
<td>getPausedTriggerGroups()</td>
</tr>
<tr>
<td>SchedulerContext</td>
<td>getSchedulerContext()</td>
</tr>
<tr>
<td>String</td>
<td>getSchedulerInstanceId()</td>
</tr>
<tr>
<td>String</td>
<td>getSchedulerName()</td>
</tr>
<tr>
<td>org.quartz.spi.SchedulerSignaler</td>
<td>getSchedulerSignaler()</td>
</tr>
<tr>
<td>ThreadGroup</td>
<td>getSchedulerThreadGroup()</td>
</tr>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>getThreadPoolClass()</code></td>
<td></td>
</tr>
<tr>
<td><code>getThreadPoolSize()</code></td>
<td></td>
</tr>
<tr>
<td><code>Trigger getTrigger(TriggerKey triggerKey)</code></td>
<td>Get the Trigger instance with the given name.</td>
</tr>
<tr>
<td><code>List&lt;String&gt; getTriggerGroupNames()</code></td>
<td>Get the names of all known Trigger groups.</td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt; getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Get the names of all the Triggers in the matching groups.</td>
</tr>
<tr>
<td><code>List&lt;? extends Trigger&gt; getTriggersOfJob(JobKey jobKey)</code></td>
<td>Get all Triggers that are associated with the job.</td>
</tr>
<tr>
<td><code>Trigger.TriggerState getTriggerState(TriggerKey triggerKey)</code></td>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td><code>String getVersion()</code></td>
<td></td>
</tr>
<tr>
<td><code>static String getVersionIteration()</code></td>
<td></td>
</tr>
<tr>
<td><code>static String getVersionMajor()</code></td>
<td></td>
</tr>
<tr>
<td><code>static String getVersionMinor()</code></td>
<td></td>
</tr>
<tr>
<td><code>void initialize()</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean interrupt(JobKey jobKey)</code></td>
<td>Interrupt all instances of the identified InteruptableJob executing in Scheduler instance.</td>
</tr>
<tr>
<td><code>boolean isClustered()</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean isInStandbyMode()</code></td>
<td>Reports whether the Scheduler is paused.</td>
</tr>
<tr>
<td><code>boolean isShutdown()</code></td>
<td>Reports whether the Scheduler has been shut down.</td>
</tr>
<tr>
<td><code>boolean isShuttingDown()</code></td>
<td></td>
</tr>
</tbody>
</table>
boolean isSignalOnSchedulingChange()

boolean isStarted()

void notifyJobListenersToBeExecuted(JobExecutionContext)

void notifyJobListenersWasExecuted(JobExecutionException je)

void notifyJobListenersWasVetoed(JobExecutionCo)

protected void notifyJobStoreJobComplete(org.quartz.spi.O

protected void notifyJobStoreJobVetoed(org.quartz.spi.Ope

void notifySchedulerListenersError(String msg,

void notifySchedulerListenersFinalized(Trigg

void notifySchedulerListenersInStandbyMode()

void notifySchedulerListenersJobAdded(JobDetail)

void notifySchedulerListenersJobDeleted(JobKey)

void notifySchedulerListenersPausedJob(JobKey k

void notifySchedulerListenersPausedJobs(String

void notifySchedulerListenersPausedTrigger(Trig

void notifySchedulerListenersPausedTriggers(Str
void notifySchedulerListenersResumedJob(JobKey key)

void notifySchedulerListenersResumedJobs(String String)

void notifySchedulerListenersResumedTrigger(TriggerKey triggerKey)

void notifySchedulerListenersResumedTriggers(String String)

void notifySchedulerListenersScheduled(Trigger trigger)

void notifySchedulerListenersShutdown()

void notifySchedulerListenersShuttingdown()

void notifySchedulerListenersStarted()

void notifySchedulerListenersUnscheduled(TriggerKey triggerKey)

protected void notifySchedulerThread(long candidateNewNextFireTime)

void notifyTriggerListenersComplete(JobExecutionContext jobExecutionContext, Trigger.CompletedExecutionInstruction instCode)

boolean notifyTriggerListenersFired(JobExecutionContext jobExecutionContext, Trigger trigger)

void notifyTriggerListenersMisfired(Trigger trigger)

int numJobsExecuted()

void pauseAll()
          Pause all triggers - equivalent of calling pauseTriggers with a matcher matching all known groups.

void pauseJob(JobKey jobKey)
              Pause the JobDetail with the given name -
void pauseJobs(GroupMatcher<JobKey> groupMatcher)
    Pause all of the JobDetails in the matching groups and their Triggers.

void pauseTrigger(TriggerKey triggerKey)
    Pause the Trigger with the given name.

void pauseTriggers(GroupMatcher<TriggerKey> matcher)
    Pause all of the Triggers in the matching groups.

boolean removeInternalJobListener(String name)
    Remove the identified JobListener from the listeners.

boolean removeInternalSchedulerListener(SchedulerListener listener)
    Remove the given SchedulerListener from the internal listeners.

boolean removeInternalTriggerListener(String name)
    Remove the identified TriggerListener from the internal listeners.

boolean removeNoGCObject(Object obj)

Date rescheduleJob(TriggerKey triggerKey, Trigger trigger)
    Remove (delete) the Trigger with the given name and replace it with the given one - which must be associated with the same Job.

void resumeAll()
    Resume (un-pause) all triggers - equivalent to calling resumeTriggerGroup(group) on every group.

void resumeJob(JobKey jobKey)
    Resume (un-pause) the JobDetail with the given name.

void resumeJobs(GroupMatcher<JobKey> matcher)
    Resume (un-pause) all of the JobDetails in the matching groups.

void resumeTrigger(TriggerKey triggerKey)
    Resume (un-pause) the Trigger with the given name.

void resumeTriggers(GroupMatcher<TriggerKey> matcher)
    Resume (un-pause) all of the Triggers in the matching groups.

Date runningSince()
scheduleJob(JobDetail jobDetail, Trigger trigger)
   Add the Job identified by the given JobDetail to associate the given Trigger with it.

scheduleJob(Trigger trigger)
   Schedule the given Trigger with the Job identified by the settings.

void scheduleJobs(Map<JobDetail,List<Trigger>> triggersAndJobs, boolean replace)

void setJobFactory(org.quartz.spi.JobFactory factory)

void setSignalOnSchedulingChange(boolean signalOnSchedulingChange)

void shutdown()
   Halts the QuartzScheduler's firing of Trigger resources associated with the QuartzScheduler.

void shutdown(boolean waitForJobsToComplete)
   Halts the QuartzScheduler's firing of Trigger resources associated with the QuartzScheduler.

void standby()
   Temporarily halts the QuartzScheduler's firing of Trigger resources associated with the QuartzScheduler.

void start()
   Starts the QuartzScheduler's threads that fire the scheduled resources.

void startDelayed(int seconds)

boolean supportsPersistence()

void triggerJob(JobKey jobKey, JobDataMap data)
   Trigger the identified Job (execute it now).

void triggerJob(org.quartz.spi.OperableTrigger trigger)
   Store and schedule the identified OperableTrigger.

boolean unscheduleJob(TriggerKey triggerKey)
   Remove the indicated Trigger from the scheduler.

boolean unscheduleJobs(List<TriggerKey> triggerKeys)
void validateState()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

QuartzScheduler

public QuartzScheduler(QuartzSchedulerResources resources, long idleWaitTime, long dbRetryInterval) throws SchedulerException

Create a QuartzScheduler with the given configuration properties.

Throws:
SchedulerException

See Also:
QuartzSchedulerResources

Method Detail

initialize

public void initialize() throws SchedulerException

Throws:
SchedulerException

getVersion

public String getVersion()
getVersion

public static String getVersionMajor()

getVersionMinor

public static String getVersionMinor()

getVersionIteration

public static String getVersionIteration()

getSchedulerSignaler

public org.quartz.spi.SchedulerSignaler getSchedulerSignaler()

getLog

public org.slf4j.Logger getLog()

getSchedulerName

public String getSchedulerName()

Returns the name of the QuartzScheduler.

Specified by:

getSchedulerName in interface RemotableQuartzScheduler
getSchedulerInstanceId

```java
public String getSchedulerInstanceId()
```

Returns the instance Id of the QuartzScheduler.

**Specified by:**
```
ger得不到SchedulerInstanceId in interface RemotableQuartzScheduler
```

getSchedulerThreadGroup

```java
public ThreadGroup getSchedulerThreadGroup()
```

Returns the name of the thread group for Quartz's main threads.

addNoGCObject

```java
public void addNoGCObject(Object obj)
```

removeNoGCObject

```java
public boolean removeNoGCObject(Object obj)
```

getSchedulerContext

```java
public SchedulerContext getSchedulerContext() throws SchedulerException
```

Returns the SchedulerContext of the Scheduler.

**Specified by:**
```
ger得不到SchedulerContext in interface RemotableQuartzScheduler
```

**Throws:**
```
SchedulerException
```
isSignalOnSchedulingChange

public boolean isSignalOnSchedulingChange()

setSignalOnSchedulingChange

public void setSignalOnSchedulingChange(boolean signalOnSchedulingChange)

start

public void start()
throws SchedulerException

Starts the QuartzScheduler's threads that fire Triggers.

All Triggers that have misfired will be passed to the appropriate TriggerListener(s).

Specified by:
start in interface RemotableQuartzScheduler

Throws:
SchedulerException

startDelayed

public void startDelayed(int seconds)
throws SchedulerException

Specified by:
startDelayed in interface RemotableQuartzScheduler

Throws:
SchedulerException

standby

public void standby()
Temporarily halts the QuartzScheduler's firing of Triggers.

The scheduler is not destroyed, and can be re-started at any time.

**Specified by:**

standby in interface RemotableQuartzScheduler

---

**isInStandbyMode**

class public boolean isInStandbyMode()

Reports whether the Scheduler is paused.

**Specified by:**

isInStandbyMode in interface RemotableQuartzScheduler

---

**runningSince**

class public Date runningSince()

**Specified by:**

runningSince in interface RemotableQuartzScheduler

---

**numJobsExecuted**

class public int numJobsExecuted()

**Specified by:**

numJobsExecuted in interface RemotableQuartzScheduler

---

**getJobStoreClass**

class public Class getJobStoreClass()

**Specified by:**

getJobStoreClass in interface RemotableQuartzScheduler
supportsPersistence

public boolean supportsPersistence()

Specified by:

supportsPersistence in interface RemotableQuartzScheduler

isClustered

public boolean isClustered()

Specified by:

isClustered in interface RemotableQuartzScheduler

getThreadPoolClass

public Class getThreadPoolClass()

Specified by:

getThreadPoolClass in interface RemotableQuartzScheduler

getThreadPoolSize

public int getThreadPoolSize()

Specified by:

getThreadPoolSize in interface RemotableQuartzScheduler

shutdown

public void shutdown()

Halts the QuartzScheduler's firing of Triggers, and cleans up all resources associated with the QuartzScheduler. Equivalent to shutdown(false).
The scheduler cannot be re-started.

**Specified by:**

`shutdown` in interface `RemotableQuartzScheduler`

---

**shutdown**

```java
public void shutdown(boolean waitForJobsToComplete)
```

Halts the QuartzScheduler's firing of Triggers, and cleans up all resources associated with the QuartzScheduler.

The scheduler cannot be re-started.

**Specified by:**

`shutdown` in interface `RemotableQuartzScheduler`

**Parameters:**

`waitForJobsToComplete` - if true the scheduler will not allow this method to return until all currently executing jobs have completed.

---

**isShutdown**

```java
public boolean isShutdown()
```

Reports whether the Scheduler has been shutdown.

**Specified by:**

`isShutdown` in interface `RemotableQuartzScheduler`

---

**isShuttingDown**

```java
public boolean isShuttingDown()
```

---

**isStarted**
public boolean isStarted()

validateState

public void validateState() throws SchedulerException

Throws:
   SchedulerException

getCurrentlyExecutingJobs

public List&lt;JobExecutionContext&gt; getCurrentlyExecutingJobs()

Return a list of JobExecutionContext objects that represent all currently executing Jobs in this Scheduler instance.

This method is not cluster aware. That is, it will only return Jobs currently executing in this Scheduler instance, not across the entire cluster.

Note that the list returned is an 'instantaneous' snap-shot, and that as soon as it's returned, the true list of executing jobs may be different.

Specified by:
   getCurrentlyExecutingJobs in interface RemotableQuartzScheduler

scheduleJob

public Date scheduleJob(JobDetail jobDetail, Trigger trigger) throws SchedulerException

Add the Job identified by the given JobDetail to the Scheduler, and associate the given Trigger with it.

If the given Trigger does not reference any Job, then it will be set to reference the Job passed with it into this method.
Specified by:
   scheduleJob in interface RemotableQuartzScheduler
Throws:
   SchedulerException - if the Job or Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

---

**scheduleJob**

public Date scheduleJob(Trigger trigger)
   throws SchedulerException

Schedule the given Trigger with the Job identified by the Trigger's settings.

Specified by:
   scheduleJob in interface RemotableQuartzScheduler
Throws:
   SchedulerException - if the indicated Job does not exist, or the Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

---

**addJob**

public void addJob(JobDetail jobDetail, boolean replace)
   throws SchedulerException

Add the given Job to the Scheduler - with no associated Trigger. The Job will be 'dormant' until it is scheduled with a Trigger, or Scheduler.triggerJob() is called for it.

The Job must by definition be 'durable', if it is not, SchedulerException will be thrown.

Specified by:
   addJob in interface RemotableQuartzScheduler
Throws:
   SchedulerException - if there is an internal Scheduler error, or if the
Job is not durable, or a Job with the same name already exists, and replace is false.

---

**deleteJob**

```java
public boolean deleteJob(JobKey jobKey) throws SchedulerException
```

Delete the identified Job from the Scheduler - and any associated Triggers.

**Specified by:**
- `deleteJob` in interface `RemotableQuartzScheduler`

**Returns:**
- true if the Job was found and deleted.

**Throws:**
- `SchedulerException` - if there is an internal Scheduler error.

---

**deleteJobs**

```java
public boolean deleteJobs(List<JobKey> jobKeys) throws SchedulerException
```

**Specified by:**
- `deleteJobs` in interface `RemotableQuartzScheduler`

**Throws:**
- `SchedulerException`  

---

**scheduleJobs**

```java
public void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace) throws SchedulerException
```

**Specified by:**
- `scheduleJobs` in interface `RemotableQuartzScheduler`

**Throws:**
- `SchedulerException`
unscheduleJobs

public boolean unscheduleJobs(List<TriggerKey> triggerKeys)
    throws SchedulerException

Specified by:
    unscheduleJobs in interface RemotableQuartzScheduler

Throws:
    SchedulerException

unscheduleJob

public boolean unscheduleJob(TriggerKey triggerKey)
    throws SchedulerException

Remove the indicated Trigger from the scheduler.

Specified by:
    unscheduleJob in interface RemotableQuartzScheduler

Throws:
    SchedulerException

rescheduleJob

public Date rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)
    throws SchedulerException

Remove (delete) the Trigger with the given name, and store the new given one - which must be associated with the same job.

Specified by:
    rescheduleJob in interface RemotableQuartzScheduler

Parameters:
    newTrigger - The new Trigger to be stored.

Returns:
    null if a Trigger with the given name & group was not found and
removed from the store, otherwise the first fire time of the newly scheduled trigger.

Throws:

SchedulerException

triggerJob

public void triggerJob(JobKey jobKey,
                   JobDataMap data)
   throws SchedulerException

  Trigger the identified Job (execute it now) - with a non-volatile trigger.

Specified by:

triggerJob in interface RemotableQuartzScheduler

Throws:

SchedulerException

triggerJob

public void triggerJob(org.quartz.spi.OperableTrigger trig)
   throws SchedulerException

  Store and schedule the identified OperableTrigger

Specified by:

triggerJob in interface RemotableQuartzScheduler

Throws:

SchedulerException

pauseTrigger

public void pauseTrigger(TriggerKey triggerKey)
   throws SchedulerException

  Pause the Trigger with the given name.

Specified by:
pauseTriggers

```java
public void pauseTriggers(
        GroupMatcher<TriggerKey> matcher)
        throws SchedulerException
```

Pause all of the Triggers in the matching groups.

Specified by:
`pauseTriggers` in interface `RemotableQuartzScheduler`

Throws:
`SchedulerException`

---

pauseJob

```java
public void pauseJob(
        JobKey jobKey)
        throws SchedulerException
```

Pause the JobDetail with the given name - by pausing all of its current Triggers.

Specified by:
`pauseJob` in interface `RemotableQuartzScheduler`

Throws:
`SchedulerException`

---

pauseJobs

```java
public void pauseJobs(
        GroupMatcher<JobKey> groupMatcher)
        throws SchedulerException
```

Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.
Specified by:
  `pauseJobs` in interface `RemotableQuartzScheduler`

Throws:
  `SchedulerException`

---

**resumeTrigger**

```java
public void resumeTrigger(TriggerKey triggerKey)
    throws SchedulerException
```

Resume (un-pause) the Trigger with the given name.

If the Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
  `resumeTrigger` in interface `RemotableQuartzScheduler`

Throws:
  `SchedulerException`

---

**resumeTriggers**

```java
public void resumeTriggers(GroupMatcher<TriggerKey> matcher)
    throws SchedulerException
```

Resume (un-pause) all of the Triggers in the matching groups.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
  `resumeTriggers` in interface `RemotableQuartzScheduler`

Throws:
  `SchedulerException`

---

**getPausedTriggerGroups**
public Set getPausedTriggerGroups()
    throws SchedulerException

Specified by:
    getPausedTriggerGroups in interface RemotableQuartzScheduler

Throws:
    SchedulerException

resumeJob

public void resumeJob(JobKey jobKey)
    throws SchedulerException

Resume (un-pause) the JobDetail with the given name.

If any of the Job's Trigger s missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
    resumeJob in interface RemotableQuartzScheduler

Throws:
    SchedulerException

resumeJobs

public void resumeJobs(GroupMatcher<JobKey> matcher)
    throws SchedulerException

Resume (un-pause) all of the JobDetails in the matching groups.

If any of the Job s had Trigger s that missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
    resumeJobs in interface RemotableQuartzScheduler

Throws:
    SchedulerException
pauseAll

public void pauseAll()
    throws SchedulerException

Pause all triggers - equivalent of calling pauseTriggers(GroupMatcher)
with a matcher matching all known groups.

When resumeAll() is called (to un-pause), trigger misfire instructions
WILL be applied.

 Specified by:
    pauseAll in interface RemotableQuartzScheduler

Throws:
    SchedulerException

See Also:
    resumeAll(),
    pauseTriggers(org.quartz.impl.matchers.GroupMatcher),
    standby()

------

resumeAll

public void resumeAll()
    throws SchedulerException

Resume (un-pause) all triggers - equivalent of calling
resumeTriggerGroup(group) on every group.

If any Trigger missed one or more fire-times, then the Trigger's misfire
instruction will be applied.

 Specified by:
    resumeAll in interface RemotableQuartzScheduler

Throws:
    SchedulerException

See Also:
    pauseAll()
getJobGroupNames

public List<String> getJobGroupNames()
    throws SchedulerException

Get the names of all known Job groups.

Specified by: 
    getJobGroupNames in interface RemotableQuartzScheduler

Throws: 
    SchedulerException

------------------------------

genJobKeys

public Set<JobKey> getJobKeys(GroupMatcher<JobKey> matcher)
    throws SchedulerException

Get the names of all the Jobs in the matching groups.

Specified by: 
    getJobKeys in interface RemotableQuartzScheduler

Throws: 
    SchedulerException

------------------------------

genTriggersOfJob

public List<? extends Trigger> getTriggersOfJob(JobKey jobKey)
    throws SchedulerException

Get all Trigger s that are associated with the identified JobDetail.

Specified by: 
    getTriggersOfJob in interface RemotableQuartzScheduler

Throws: 
    SchedulerException

------------------------------

genTriggerGroupNames
public List<String> getTriggerGroupNames()
  throws SchedulerException

Get the names of all known Trigger groups.

Specified by:
  getTriggerGroupNames in interface RemotableQuartzScheduler
Throws:
  SchedulerException

getTriggerKeys

public Set<TriggerKey> getTriggerKeys(GroupMatcher<TriggerKey> matcher)
  throws SchedulerException

Get the names of all the Triggers in the matching groups.

Specified by:
  getTriggerKeys in interface RemotableQuartzScheduler
Throws:
  SchedulerException

getJobDetail

public JobDetail getJobDetail(JobKey jobKey)
  throws SchedulerException

Get the JobDetail for the Job instance with the given name and group.

Specified by:
  getJobDetail in interface RemotableQuartzScheduler
Throws:
  SchedulerException

getTrigger

public Trigger getTrigger(TriggerKey triggerKey)
  throws SchedulerException
Get the `Trigger` instance with the given name and group.

**Specified by:**
getTrigger in interface `RemotableQuartzScheduler`

**Throws:**
`SchedulerException`

---

**checkExists**

public boolean `checkExists(JobKey jobKey)`

Determine whether a `Job` with the given identifier already exists within the scheduler.

**Specified by:**
`checkExists` in interface `RemotableQuartzScheduler`

**Parameters:**
jobKey - the identifier to check for

**Returns:**
true if a Job exists with the given identifier

**Throws:**
`SchedulerException`

---

**checkExists**

public boolean `checkExists(TriggerKey triggerKey)`

Determine whether a `Trigger` with the given identifier already exists within the scheduler.

**Specified by:**
`checkExists` in interface `RemotableQuartzScheduler`

**Parameters:**
triggerKey - the identifier to check for

**Returns:**
true if a Trigger exists with the given identifier

**Throws:**
`SchedulerException`
Throws:

SchedulerException

---

**clear**

```java
public void clear()
    throws SchedulerException
```

Clears (deletes!) all scheduling data - all Jobs, Triggers Calendars.

**Specified by:**

clear in interface RemotableQuartzScheduler

**Throws:**

SchedulerException

---

**getTriggerState**

```java
public Trigger.TriggerState getTriggerState(TriggerKey triggerKey)
    throws SchedulerException
```

Get the current state of the identified Trigger.

**Specified by:**

getTriggerState in interface RemotableQuartzScheduler

**Throws:**

SchedulerException

**See Also:**

Trigger.TriggerState

---

**addCalendar**

```java
public void addCalendar(String calName,
    Calendar calendar,
    boolean replace,
    boolean updateTriggers)
    throws SchedulerException
```

Add (register) the given Calendar to the Scheduler.
addCalendar in interface RemotableQuartzScheduler

Throws:
SchedulerException - if there is an internal Scheduler error, or a Calendar with the same name already exists, and replace is false.

deleteCalendar

public boolean deleteCalendar(String calName) throws SchedulerException

Delete the identified calendar from the Scheduler.

Specified by:
deleteCalendar in interface RemotableQuartzScheduler

Returns:
true if the Calendar was found and deleted.

Throws:
SchedulerException - if there is an internal Scheduler error.

getCalendar

public Calendar getCalendar(String calName) throws SchedulerException

Get the Calendar instance with the given name.

Specified by:
getCalendar in interface RemotableQuartzScheduler

Throws:
SchedulerException

getCalendarNames

public List<String> getCalendarNames() throws SchedulerException
Get the names of all registered calendars.

**Specified by:**
getCalendarNames in interface RemotableQuartzScheduler

**Throws:**
SchedulerException

---

**getListenerManager**

public ListenerManager getListenerManager()

---

**addInternalJobListener**

public void addInternalJobListener(JobListener jobListener)

Add the given JobListener to the Scheduler's internal list.

---

**removeInternalJobListener**

public boolean removeInternalJobListener(String name)

Remove the identified JobListener from the Scheduler's list of internal listeners.

**Returns:**
true if the identified listener was found in the list, and removed.

---

**getInternalJobListeners**

public List&lt;JobListener&gt; getInternalJobListeners()

Get a List containing all of the JobListeners in the Scheduler's internal list.
getInternalJobListener

public JobListener getInternalJobListener(String name)

Get the internal JobListener that has the given name.

addInternalTriggerListener

public void addInternalTriggerListener(TriggerListener triggerListener)

Add the given TriggerListener to the Scheduler's internal list.

removeInternalTriggerListener

public boolean removeInternalTriggerListener(String name)

Remove the identified TriggerListener from the Scheduler's list of internal listeners.

Returns:
true if the identified listener was found in the list, and removed.

getInternalTriggerListeners

public List<TriggerListener> getInternalTriggerListeners()

Get a list containing all of the TriggerListeners in the Scheduler's internal list.

getInternalTriggerListener

public TriggerListener getInternalTriggerListener(String name)

Get the internal TriggerListener that has the given name.
addInternalSchedulerListener

public void addInternalSchedulerListener(SchedulerListener schedulerListener)

Register the given SchedulerListener with the Scheduler's list of internal listeners.

removeInternalSchedulerListener

public boolean removeInternalSchedulerListener(SchedulerListener schedulerListener)

Remove the given SchedulerListener from the Scheduler's list of internal listeners.

Returns:
true if the identified listener was found in the list, and removed.

getInternalSchedulerListeners

public List<SchedulerListener> getInternalSchedulerListeners()

Get a List containing all of the internal SchedulerListeners registered with the Scheduler.

notifyJobStoreJobComplete

protected void notifyJobStoreJobComplete(org.quartz.spi.OperableTrigger trigger, JobDetail detail, Trigger.CompletedExecutionInstruction instruction) throws JobPersistenceException

Throws:
JobPersistenceException

notifyJobStoreJobVetoed
protected void notifyJobStoreJobVetoed(org.quartz.spi.OperableTrigger jobDetail, Trigger.CompletedExecutionInstruction detail) throws JobPersistenceException

Throws:
   JobPersistenceException

notifySchedulerThread

protected void notifySchedulerThread(long candidateNewNextFireTime)

notifyTriggerListenersFired

public boolean notifyTriggerListenersFired(JobExecutionContext jec) throws SchedulerException

Throws:
   SchedulerException

notifyTriggerListenersMisfired

public void notifyTriggerListenersMisfired(Trigger trigger) throws SchedulerException

Throws:
   SchedulerException

notifyTriggerListenersComplete

public void notifyTriggerListenersComplete(JobExecutionContext jec, Trigger.CompletedExecutionInstruction detail) throws SchedulerException

Throws:
   SchedulerException
**notifyJobListenersToBeExecuted**

```java
public void notifyJobListenersToBeExecuted(JobExecutionContext jec)
    throws SchedulerException
```

Throws:  
SchedulerException

---

**notifyJobListenersWasVetoed**

```java
public void notifyJobListenersWasVetoed(JobExecutionContext jec)
    throws SchedulerException
```

Throws:  
SchedulerException

---

**notifyJobListenersWasExecuted**

```java
public void notifyJobListenersWasExecuted(JobExecutionContext jec, 
    JobExecutionException je)
    throws SchedulerException
```

Throws:  
SchedulerException

---

**notifySchedulerListenersError**

```java
public void notifySchedulerListenersError(String msg, 
    SchedulerException se)
```

---

**notifySchedulerListenersScheduled**

```java
public void notifySchedulerListenersScheduled(Trigger trigger)
```

---

**notifySchedulerListeners Unscheduled**
public void notifySchedulerListenersUnscheduled(TriggerKey triggerKey)

notifySchedulerListenersFinalized

public void notifySchedulerListenersFinalized(Trigger trigger)

notifySchedulerListenersPausedTrigger

public void notifySchedulerListenersPausedTrigger(TriggerKey triggerKey)

notifySchedulerListenersPausedTriggers

public void notifySchedulerListenersPausedTriggers(String group)

notifySchedulerListenersResumedTrigger

public void notifySchedulerListenersResumedTrigger(TriggerKey key)

notifySchedulerListenersResumedTriggers

public void notifySchedulerListenersResumedTriggers(String group)

notifySchedulerListenersPausedJob

public void notifySchedulerListenersPausedJob(JobKey key)

notifySchedulerListenersPausedJobs

public void notifySchedulerListenersPausedJobs(String group)

notifySchedulerListenersResumedJob
public void notifySchedulerListenersResumedJob(JobKey key)

notifySchedulerListenersResumedJobs

public void notifySchedulerListenersResumedJobs(String group)

notifySchedulerListenersInStandbyMode

public void notifySchedulerListenersInStandbyMode()

notifySchedulerListenersStarted

public void notifySchedulerListenersStarted()

notifySchedulerListenersShutdown

public void notifySchedulerListenersShutdown()

notifySchedulerListenersShuttingdown

public void notifySchedulerListenersShuttingdown()

notifySchedulerListenersJobAdded

public void notifySchedulerListenersJobAdded(JobDetail jobDetail)

notifySchedulerListenersJobDeleted

public void notifySchedulerListenersJobDeleted(JobKey jobKey)

setJobFactory
public void setJobFactory(org.quartz.spi.JobFactory factory)

    Throws:
    SchedulerException

getJobFactory

public org.quartz.spi.JobFactory getJobFactory()

interrupt

public boolean interrupt(JobKey jobKey)

    throws UnableToInterruptJobException

    Interrupt all instances of the identified InterruptableJob executing in this Scheduler instance.

    This method is not cluster aware. That is, it will only interrupt instances of the identified InterruptableJob currently executing in this Scheduler instance, not across the entire cluster.

    Specified by:
    interrupt in interface RemotableQuartzScheduler

    Throws:
    UnableToInterruptJobException

    See Also:
    RemotableQuartzScheduler.interrupt(JobKey)
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
Class QuartzSchedulerMBeanImpl

public class QuartzSchedulerMBeanImpl

extends StandardMBean
implements NotificationEmitter, QuartzSchedulerMBean, JobListener, SchedulerListener

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>emitter</td>
<td>emitter</td>
</tr>
<tr>
<td>sequenceNumber</td>
<td>sequenceNumber</td>
</tr>
</tbody>
</table>

Fields inherited from interface org.quartz.core.jmx.QuartzSchedulerMBean

<table>
<thead>
<tr>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB_ADDED, JOB_DELETED, JOB_EXECUTION_VETOED, JOB_SCHEDULED, JOB_TO_BE_EXECUTED, JOB_UNSCHEDULED, JOB_WAS_EXECUTED, JOBS_PAUSED, JOBS_RESUMED, SAMPLED_STATISTICS_ENABLED, SAMPLED_STATISTICS_RESET, SCHEDULER_ERROR, SCHEDULER_PAUSED, SCHEDULER_SHUTDOWN, SCHEDULER_STARTED, SCHEDULING_DATA_CLEARED, TRIGGER_FINALIZED, TRIGGERS_PAUSED, TRIGGERS_RESUMED</td>
</tr>
</tbody>
</table>

Constructor Summary

protected QuartzSchedulerMBeanImpl(QuartzScheduler scheduler)
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>addJob(CompositeData jobDetail, boolean replace)</td>
<td>Adds a durable job described by abstractJobInfo, which must contain the fully-qualified JobDetail class name under the &quot;jobDetailClass.&quot; That JobDetail type must contain a no-arg constructor and have public access.</td>
</tr>
<tr>
<td>void</td>
<td>addJob(Map&lt;String, Object&gt; abstractJobInfo, boolean replace)</td>
<td>Adds a durable job described by abstractJobInfo, which must contain the fully-qualified JobDetail class name under the &quot;jobDetailClass.&quot; That JobDetail type must contain a no-arg constructor and have public access.</td>
</tr>
<tr>
<td>void</td>
<td>addNotificationListener(NotificationListener notif, NotificationFilter filter, Object callBack)</td>
<td>Adds a notification listener to the QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td>clear()</td>
<td>Clears all jobs and calendars.</td>
</tr>
<tr>
<td>void</td>
<td>deleteCalendar(String calendarName)</td>
<td>Deletes a calendar.</td>
</tr>
<tr>
<td>boolean</td>
<td>deleteJob(String jobName, String jobGroupName)</td>
<td>Deletes a job.</td>
</tr>
<tr>
<td>TabularData</td>
<td>getAllJobDetails()</td>
<td>Returns all job details.</td>
</tr>
<tr>
<td>List&lt;CompositeData&gt;</td>
<td>getAllTriggers()</td>
<td>Returns all triggers.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>getCalendarNames()</td>
<td>Returns all calendar names.</td>
</tr>
<tr>
<td>TabularData</td>
<td>getCurrentlyExecutingJobs()</td>
<td>Returns all currently executing jobs.</td>
</tr>
<tr>
<td>CompositeData</td>
<td>getJobDetail(String jobName, String jobGroupName)</td>
<td>Returns the composite data for a job.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>getJobGroupNames()</td>
<td>Returns all job group names.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>getJobNames(String groupName)</td>
<td>Returns all job names in a given group.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>getJobsCompletedMostRecentSample()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getJobsExecutedMostRecentSample()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getJobsScheduledMostRecentSample()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getJobStoreClassName()</code></td>
<td>Get the name of the JobListener.</td>
<td></td>
</tr>
<tr>
<td><code>getName()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getNotificationInfo()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getPausedTriggerGroups()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getPerformanceMetrics()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getSchedulerInstanceId()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getSchedulerName()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getThreadPoolClassName()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getThreadPoolSize()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getTrigger(String name, String groupName)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getTriggerGroupNames()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getTriggerNames(String groupName)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getTriggersOfJob(String jobName, String jobGroupName)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getTriggerState(String triggerName, String triggerGroupName)</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
String getVersion()

boolean interruptJob(String jobName, String jobGroupName)

boolean isSampledStatisticsEnabled()

boolean isShutdown()

boolean isStandbyMode()

boolean isStarted()

void jobAdded(JobDetail jobDetail)
    Called by the Scheduler when a JobDetail has been added.

void jobDeleted(JobKey jobKey)
    Called by the Scheduler when a JobDetail has been deleted.

void jobExecutionVetoed(JobExecutionContext context)
    Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

void jobPaused(JobKey jobKey)
    Called by the Scheduler when a JobDetail has been paused.

void jobResumed(JobKey jobKey)
    Called by the Scheduler when a JobDetail has been un-paused.

void jobScheduled(Trigger trigger)
    Called by the Scheduler when a JobDetail is scheduled.

void jobsPaused(String jobGroup)
    Called by the Scheduler when a group of JobDetails have been paused.

void jobsResumed(String jobGroup)
    Called by the Scheduler when a group of JobDetails have been un-paused.

void jobToBeExecuted(JobExecutionContext context)
Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

```java
void jobUnscheduled(TriggerKey triggerKey)
```

Called by the Scheduler when a JobDetail is unscheduled.

```java
void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)
```

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered() method has been called.

```java
void pauseAllTriggers()
void pauseJob(String jobName, String jobGroup)
void pauseJobGroup(String jobGroup)
void pauseJobs(GroupMatcher matcher)
void pauseJobsContaining(String jobGroupToken)
    Pause all jobs whose group contains jobGroupToken
void pauseJobsEndingWith(String jobGroupSuffix)
    Pause all jobs whose group ends with jobGroupSuffix
void pauseJobsStartingWith(String jobGroupPrefix)
    Pause all jobs whose group starts with jobGroupPrefix
void pauseTrigger(String triggerName, String triggerGroup)
void pauseTriggerGroup(String triggerGroup)
void pauseTriggersContaining(String triggerGroupToken)
    Pause all triggers whose group contains triggerGroupToken
void pauseTriggersEndingWith(String triggerGroupSuffix)
    Pause all triggers whose group ends with triggerGroupSuffix
void pauseTriggersStartingWith(String triggerGroupPrefix)
    Pause all triggers whose group starts with triggerGroupPrefix
void removeNotificationListener(NotificationListener listener)
```
void removeNotificationListener(NotificationListener listener, NotificationFilter filter, Object callBack)

void resumeAllTriggers()

void resumeJob(String jobName, String jobGroup)

void resumeJobGroup(String jobGroup)

void resumeJobs(GroupMatcher matcher)

void resumeJobsContaining(String jobGroupToken)
    Resume all jobs whose group contains jobGroupToken

void resumeJobsEndingWith(String jobGroupSuffix)
    Resume all jobs whose group ends with jobGroupSuffix

void resumeJobsStartingWith(String jobGroupPrefix)
    Resume all jobs whose group starts with jobGroupPrefix

void resumeTrigger(String triggerName, String triggerGroup)

void resumeTriggerGroup(String triggerGroup)

void resumeTriggersContaining(String triggerGroupToken)
    Resume all triggers whose group contains triggerGroupToken

void resumeTriggersEndingWith(String triggerGroupSuffix)
    Resume all triggers whose group ends with triggerGroupSuffix

void resumeTriggersStartingWith(String triggerGroupPrefix)
    Resume all triggers whose group starts with triggerGroupPrefix

void scheduleBasicJob(Map<String, Object> jobDetailInfo, Map<String, Object> triggerInfo)
    Schedules a job using the given Cron/Simple trigger

void scheduleJob(Map<String, Object> abstractJobInfo, Map<String, Object> abstractTriggerInfo)
    Schedules an arbitrary job described by abstractJobInfo and a trigger specified by abstractTriggerInfo.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void scheduleJob(String jobName, String jobGroup, Map&lt;String, Object&gt; abstractTriggerInfo)</code></td>
<td>Schedules the specified job using a trigger described by <code>abstractTriggerInfo</code>, which must contain the fully-qualified trigger class name under the key &quot;triggerClass.&quot; That trigger type must contain a no-arg constructor and have public access.</td>
</tr>
<tr>
<td><code>Date scheduleJob(String jobName, String jobGroup, String triggerName, String triggerGroup)</code></td>
<td>Schedule an existing job with an existing trigger.</td>
</tr>
<tr>
<td><code>void schedulerError(String msg, SchedulerException cause)</code></td>
<td>Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the Job Store, the inability to instantiate a Job instance when its TriggerExpired, or a trigger not matching the specified key.</td>
</tr>
<tr>
<td><code>void schedulerInStandbyMode()</code></td>
<td>Called by the Scheduler to inform the listener that it has moved to standby mode.</td>
</tr>
<tr>
<td><code>void schedulerShutdown()</code></td>
<td>Called by the Scheduler to inform the listener that it has started.</td>
</tr>
<tr>
<td><code>void schedulerShuttingdown()</code></td>
<td>Called by the Scheduler to inform the listener that it has begun the shutdown sequence.</td>
</tr>
<tr>
<td><code>void schedulerStarted()</code></td>
<td>Called by the Scheduler to inform the listener that it has started.</td>
</tr>
<tr>
<td><code>void schedulingDataCleared()</code></td>
<td>Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.</td>
</tr>
<tr>
<td><code>void sendNotification(String eventType)</code></td>
<td></td>
</tr>
<tr>
<td><code>void sendNotification(String eventType, Object data)</code></td>
<td></td>
</tr>
<tr>
<td><code>void sendNotification(String eventType, Object data, String msg)</code></td>
<td></td>
</tr>
<tr>
<td><code>void setSampledStatisticsEnabled(boolean enabled)</code></td>
<td></td>
</tr>
</tbody>
</table>
void shutdown()

void standby()

void start()

void triggerFinalized(Trigger trigger)
    Called by the Scheduler when a Trigger has reached condition in which it will never fire again.

void triggerJob(CompositeData trigger)

void triggerJob(String jobName, String jobGroup, Map<String, String> jobDataMap)

void triggerPaused(TriggerKey triggerKey)
    Called by the Scheduler when a Trigger has been paused.

void triggerResumed(TriggerKey triggerKey)
    Called by the Scheduler when a Trigger has been unpaused.

void triggersPaused(String triggerGroup)
    Called by the Scheduler when a group of Triggers has been paused.

void triggersResumed(String triggerGroup)
    Called by the Scheduler when a group of Triggers has been un-paused.

boolean unscheduleJob(String triggerName, String triggerGroup)

Methods inherited from class javax.management.StandardMBean
    cacheMBeanInfo, getAttribute, getAttributes, getCachedMBeanInfo, getClassName, getConstructors, getDescription, getDescription, getDescription, getDescription, getDescription, getDescription, getDescription, getDescription, getDescription, getDescription, getImpact, getImplementation, getImplementationClass, getMBeanInfo, getMBeanInterface, getParameterName, getParameterName, invoke, postDeregister,
postRegister, preDeregister, preRegister, setAttribute, setAttributes, setImplementation

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

emitter
protected final org.quartz.core.QuartzSchedulerMBeanImpl.Emitter emitter

sequenceNumber
protected final AtomicLong sequenceNumber

Constructor Detail

QuartzSchedulerMBeanImpl
protected QuartzSchedulerMBeanImpl(QuartzScheduler scheduler) throws NotCompliantMBeanException

Throws:
    NotCompliantMBeanException

Method Detail
**getCurrentlyExecutingJobs**

```java
public TabularData getCurrentlyExecutingJobs() throws Exception
```

**Specified by:**
- `getCurrentlyExecutingJobs` in interface `QuartzSchedulerMBean`

**Returns:**
TabularData of CompositeData:JobExecutionContext

**Throws:**
- `Exception`

---

**getAllJobDetails**

```java
public TabularData getAllJobDetails() throws Exception
```

**Specified by:**
- `getAllJobDetails` in interface `QuartzSchedulerMBean`

**Returns:**
TabularData of CompositeData:JobDetail

**Throws:**
- `Exception`

**See Also:**
- `JobDetailSupport`

---

**getAllTriggers**

```java
public List<CompositeData> getAllTriggers() throws Exception
```

**Specified by:**
- `getAllTriggers` in interface `QuartzSchedulerMBean`

**Returns:**
List of CompositeData:[CronTrigger|SimpleTrigger]

**Throws:**
- `Exception`

**See Also:**
addJob

```java
generic void addJob(CompositeData jobDetail,
                    boolean replace)
throws Exception
```

Specified by:
  addJob in interface QuartzSchedulerMBean

Throws:
  Exception

scheduleBasicJob

```java
generic void scheduleBasicJob(Map<String, Object> jobDetailInfo,
                                Map<String, Object> triggerInfo)
throws Exception
```

Description copied from interface: QuartzSchedulerMBean
Schedules a job using the given Cron/Simple triggerInfo. The triggerInfo and jobDetailInfo must contain well-known attribute values. TriggerInfo attributes: name, group, description, calendarName, priority, CronExpression | (startTime, endTime, repeatCount, repeatInterval)
JobDetailInfo attributes: name, group, description, jobClass, jobDataMap, durability, shouldRecover

Specified by:
  scheduleBasicJob in interface QuartzSchedulerMBean

Throws:
  Exception

scheduleJob

```java
generic void scheduleJob(Map<String, Object> abstractJobInfo,
                         Map<String, Object> abstractTriggerInfo)
throws Exception
```
Description copied from interface: QuartzSchedulerMBean
Schedules an arbitrary job described by abstractJobInfo using a trigger specified by abstractTriggerInfo. AbstractTriggerInfo and AbstractJobInfo must contain the following String attributes. AbstractTriggerInfo: triggerClass, the fully-qualified class name of a concrete Trigger type AbstractJobInfo: jobDetailClass, the fully-qualified class name of a concrete JobDetail type If the Trigger and JobDetail can be successfully instantiated, the remaining attributes will be reflectively applied to those instances. The remaining attributes are limited to the types: Integer, Double, Float, String, Boolean, Date, Character, Map. Maps are further limited to containing values from the same set of types, less Map itself.

Specified by:
   scheduleJob in interface QuartzSchedulerMBean
Throws:
   Exception

scheduleJob

public void scheduleJob(String jobName,
                        String jobGroup,
                        Map<String, Object> abstractTriggerInfo)
   throws Exception

Description copied from interface: QuartzSchedulerMBean
Schedules the specified job using a trigger described by abstractTriggerInfo, which must contain the fully-qualified trigger class name under the key "triggerClass." That trigger type must contain a no-arg constructor and have public access. Other attributes are applied reflectively and are limited to the types: Integer, Double, Float, String, Boolean, Date, Character, Map. Maps are limited to containing values from the same set of types, less Map itself.

Specified by:
   scheduleJob in interface QuartzSchedulerMBean
Throws:
   Exception

addJob
public void addJob(Map<String, Object> abstractJobInfo, boolean replace)
throws Exception

Description copied from interface: QuartzSchedulerMBean
Adds a durable job described by abstractJobInfo, which must contain the fully-qualified JobDetail class name under the key "jobDetailClass." That JobDetail type must contain a no-arg constructor and have public access. Other attributes are applied reflectively and are limited to the types: Integer, Double, Float, String, Boolean, Date, Character, Map. Maps are limited to containing values from the same set of types, less Map itself.

Specified by: addJob in interface QuartzSchedulerMBean
Throws: Exception

deleteCalendar

public void deleteCalendar(String calendarName)
throws Exception

Specified by: deleteCalendar in interface QuartzSchedulerMBean
Throws: Exception

deleteJob

public boolean deleteJob(String jobName, String jobGroupName)
throws Exception

Specified by: deleteJob in interface QuartzSchedulerMBean
Throws: Exception
**getCalendarNames**

```java
public List<String> getCalendarNames()
    throws Exception
```

Specified by:
- `getCalendarNames` in interface `QuartzSchedulerMBean`

Throws:
- `Exception`

---

**getJobDetail**

```java
public CompositeData getJobDetail(String jobName,
       String jobGroupName)
    throws Exception
```

Specified by:
- `getJobDetail` in interface `QuartzSchedulerMBean`

Returns:
- `CompositeData:JobDetail`

Throws:
- `Exception`

See Also:
- `JobDetailSupport`

---

**getJobGroupNames**

```java
public List<String> getJobGroupNames()
    throws Exception
```

Specified by:
- `getJobGroupNames` in interface `QuartzSchedulerMBean`

Throws:
- `Exception`

---

**getJobNames**
public `List<String>` `getJobNames(String groupName)`

Throws: `Exception`

Specified by: `getJobNames` in interface `QuartzSchedulerMBean`

---

**getJobStoreClassName**

public `String` `getJobStoreClassName()`

Specified by: `getJobStoreClassName` in interface `QuartzSchedulerMBean`

---

**getPausedTriggerGroups**

public `Set<String>` `getPausedTriggerGroups()`

Throws: `Exception`

Specified by: `getPausedTriggerGroups` in interface `QuartzSchedulerMBean`

---

**getTrigger**

public `CompositeData` `getTrigger(String name, String groupName)`

Throws: `Exception`

Specified by: `getTrigger` in interface `QuartzSchedulerMBean`

Throws: `Exception`
getTriggerGroupNames

public List<String> getTriggerGroupNames()
    throws Exception

Specified by:
    getTriggerGroupNames in interface QuartzSchedulerMBean

Throws:
    Exception

getTriggerNames

public List<String> getTriggerNames(String groupName)
    throws Exception

Specified by:
    getTriggerNames in interface QuartzSchedulerMBean

Throws:
    Exception

getTriggerState

public String getTriggerState(String triggerName,
    String triggerGroupName)
    throws Exception

Specified by:
    getTriggerState in interface QuartzSchedulerMBean

throws:
    Exception

getTriggersOfJob

public List<CompositeData> getTriggersOfJob(String jobName,
    String jobGroupName)
    throws Exception

Specified by:

throws:
    Exception
getTriggersOfJob in interface QuartzSchedulerMBean

Returns:
List of CompositeData:[CronTrigger|SimpleTrigger] for the specified job.

Throws:
Exception

See Also:
TriggerSupport

interruptJob

public boolean interruptJob(String jobName,
                            String jobGroupName)
throws Exception

Specified by:
interruptJob in interface QuartzSchedulerMBean

Throws:
Exception

scheduleJob

public Date scheduleJob(String jobName,
                        String jobGroup,
                        String triggerName,
                        String triggerGroup)
throws Exception

Description copied from interface: QuartzSchedulerMBean
Schedule an existing job with an existing trigger.

Specified by:
scheduleJob in interface QuartzSchedulerMBean

Returns:
date of nextFireTime

Throws:
Exception

unscheduleJob

public boolean unscheduleJob(String triggerName, String triggerGroup)
throws Exception

Specified by:
unscheduleJob in interface QuartzSchedulerMBean

Throws:
Exception

clear

public void clear()
throws Exception

Specified by:
clear in interface QuartzSchedulerMBean

Throws:
Exception

getVersion

public String getVersion()

Specified by:
getVersion in interface QuartzSchedulerMBean

isShutdown

public boolean isShutdown()

Specified by:
isShutdown in interface QuartzSchedulerMBean

isStarted
public boolean isStarted()

**Specified by:**

`isStarted` in interface `QuartzSchedulerMBean`

---

**start**

public void start()
    throws Exception

**Specified by:**

`start` in interface `QuartzSchedulerMBean`

**Throws:**

`Exception`

---

**shutdown**

public void shutdown()

**Specified by:**

`shutdown` in interface `QuartzSchedulerMBean`

---

**standby**

public void standby()

**Specified by:**

`standby` in interface `QuartzSchedulerMBean`

---

**isStandbyMode**

public boolean isStandbyMode()

**Specified by:**

`isStandbyMode` in interface `QuartzSchedulerMBean`
**getSchedulerName**

public String getSchedulerName()

Specified by:
  getSchedulerName in interface QuartzSchedulerMBean

**getSchedulerInstanceId**

public String getSchedulerInstanceId()

Specified by:
  getSchedulerInstanceId in interface QuartzSchedulerMBean

**getThreadPoolClassName**

public String getThreadPoolClassName()

Specified by:
  getThreadPoolClassName in interface QuartzSchedulerMBean

**getThreadPoolSize**

public int getThreadPoolSize()

Specified by:
  getThreadPoolSize in interface QuartzSchedulerMBean

**pauseJob**

public void pauseJob(String jobName, String jobGroup) throws Exception

Specified by:
  pauseJob in interface QuartzSchedulerMBean
Throws:

Exception

pauseJobs

public void pauseJobs(GroupMatcher matcher)
    throws Exception

Throws:

Exception

pauseJobGroup

public void pauseJobGroup(String jobGroup)
    throws Exception

Specified by:

pauseJobGroup in interface QuartzSchedulerMBean

Throws:

Exception

pauseJobsStartingWith

public void pauseJobsStartingWith(String jobGroupPrefix)
    throws Exception

Description copied from interface: QuartzSchedulerMBean
Pause all jobs whose group starts with jobGroupPrefix

Specified by:

pauseJobsStartingWith in interface QuartzSchedulerMBean

Throws:

Exception

pauseJobsEndingWith
public void pauseJobsEndingWith(String jobGroupSuffix) 
throws Exception

Description copied from interface: QuartzSchedulerMBean
Pause all jobs whose group ends with jobGroupSuffix

Specified by:
    pauseJobsEndingWith in interface QuartzSchedulerMBean
Throws:
    Exception

-----------------------------

pauseJobsContaining

public void pauseJobsContaining(String jobGroupToken) 
throws Exception

Description copied from interface: QuartzSchedulerMBean
Pause all jobs whose group contains jobGroupToken

Specified by:
    pauseJobsContaining in interface QuartzSchedulerMBean
Throws:
    Exception

-----------------------------

pauseAllTriggers

public void pauseAllTriggers() 
throws Exception

Specified by:
    pauseAllTriggers in interface QuartzSchedulerMBean
Throws:
    Exception

-----------------------------

pauseTriggerGroup

public void pauseTriggerGroup(String triggerGroup) 
throws Exception
Specified by:
   `pauseTriggerGroup` in interface `QuartzSchedulerMBean`

Throws:
   `Exception`

---

`pauseTriggersStartingWith`

```java
public void pauseTriggersStartingWith(String triggerGroupPrefix)
    throws Exception
```

**Description copied from interface: `QuartzSchedulerMBean`**
Pause all triggers whose group starts with triggerGroupPrefix

Specified by:
   `pauseTriggersStartingWith` in interface `QuartzSchedulerMBean`

Throws:
   `Exception`

---

`pauseTriggersEndingWith`

```java
public void pauseTriggersEndingWith(String triggerGroupSuffix)
    throws Exception
```

**Description copied from interface: `QuartzSchedulerMBean`**
Pause all triggers whose group ends with triggerGroupSuffix

Specified by:
   `pauseTriggersEndingWith` in interface `QuartzSchedulerMBean`

Throws:
   `Exception`

---

`pauseTriggersContaining`

```java
public void pauseTriggersContaining(String triggerGroupToken)
    throws Exception
```

**Description copied from interface: `QuartzSchedulerMBean`**

---
Pause all triggers whose group contains triggerGroupToken

**Specified by:**

`pauseTriggersContaining` in interface `QuartzSchedulerMBean`

**Throws:**

`Exception`

---

**pauseTrigger**

```java
public void pauseTrigger(String triggerName, String triggerGroup)
throws Exception
```

**Specified by:**

`pauseTrigger` in interface `QuartzSchedulerMBean`

**Throws:**

`Exception`

---

**resumeAllTriggers**

```java
public void resumeAllTriggers()
throws Exception
```

**Specified by:**

`resumeAllTriggers` in interface `QuartzSchedulerMBean`

**Throws:**

`Exception`

---

**resumeJob**

```java
public void resumeJob(String jobName, String jobGroup)
throws Exception
```

**Specified by:**

`resumeJob` in interface `QuartzSchedulerMBean`

**Throws:**

`Exception`
resumeJobs

public void resumeJobs(GroupMatcher matcher)
    throws Exception

    Throws:
    Exception

resumeJobGroup

public void resumeJobGroup(String jobGroup)
    throws Exception

    Specified by:
    resumeJobGroup in interface QuartzSchedulerMBean
    Throws:
    Exception

description copied from interface: QuartzSchedulerMBean
Resume all jobs whose group starts with jobGroupPrefix

Specified by:
    resumeJobsStartingWith in interface QuartzSchedulerMBean
    Throws:
    Exception

resumeJobsEndingWith

public void resumeJobsEndingWith(String jobGroupSuffix)
    throws Exception
Description copied from interface: QuartzSchedulerMBean
Resume all jobs whose group ends with jobGroupSuffix

Specified by:  
resumeJobsEndingWith in interface QuartzSchedulerMBean

Throws:  
Exception

resumeJobsContaining

public void resumeJobsContaining(String jobGroupToken)  
throws Exception

Description copied from interface: QuartzSchedulerMBean
Resume all jobs whose group contains jobGroupToken

Specified by:  
resumeJobsContaining in interface QuartzSchedulerMBean

Throws:  
Exception

resumeTrigger

public void resumeTrigger(String triggerName,  
String triggerGroup)  
throws Exception

Specified by:  
resumeTrigger in interface QuartzSchedulerMBean

Throws:  
Exception

resumeTriggerGroup

public void resumeTriggerGroup(String triggerGroup)  
throws Exception

Specified by:  

resumeTriggerGroup in interface QuartzSchedulerMBean

Throws:
Exception

-------------------

resumeTriggersStartingWith

public void resumeTriggersStartingWith(String triggerGroupPrefix)
throws Exception

Description copied from interface: QuartzSchedulerMBean
Resume all triggers whose group starts with triggerGroupPrefix

Specified by:
resumeTriggersStartingWith in interface QuartzSchedulerMBean

Throws:
Exception

-------------------

resumeTriggersEndingWith

public void resumeTriggersEndingWith(String triggerGroupSuffix)
throws Exception

Description copied from interface: QuartzSchedulerMBean
Resume all triggers whose group ends with triggerGroupSuffix

Specified by:
resumeTriggersEndingWith in interface QuartzSchedulerMBean

Throws:
Exception

-------------------

resumeTriggersContaining

public void resumeTriggersContaining(String triggerGroupToken)
throws Exception

Description copied from interface: QuartzSchedulerMBean
Resume all triggers whose group contains triggerGroupToken

Specified by:
resumeTriggersContaining in interface QuartzSchedulerMBean

Throws:
Exception
Specified by: resumeTriggersContaining in interface QuartzSchedulerMBean
Throws: Exception

triggerJob

public void triggerJob(String jobName, String jobGroup, Map<String, String> jobDataMap) throws Exception

Specified by: triggerJob in interface QuartzSchedulerMBean
Throws: Exception

triggerJob

public void triggerJob(CompositeData trigger) throws Exception

Throws: Exception

jobAdded

public void jobAdded(JobDetail jobDetail)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been added.

Specified by: jobAdded in interface SchedulerListener
jobDeleted

public void jobDeleted(JobKey jobKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been deleted.

Specified by:
jobDeleted in interface SchedulerListener

jobScheduled

public void jobScheduled(Trigger trigger)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail is scheduled.

Specified by:
jobScheduled in interface SchedulerListener

jobUnscheduled

public void jobUnscheduled(TriggerKey triggerKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail is unscheduled.

Specified by:
jobUnscheduled in interface SchedulerListener

See Also:
SchedulerListener.schedulingDataCleared()

schedulingDataCleared
public void schedulingDataCleared()

   **Description copied from interface:** SchedulerListener
   Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.

   **Specified by:**
   schedulingDataCleared in interface SchedulerListener

---

**jobPaused**

public void jobPaused(JobKey jobKey)

   **Description copied from interface:** SchedulerListener
   Called by the Scheduler when a JobDetail has been paused.

   **Specified by:**
   jobPaused in interface SchedulerListener

---

**jobsPaused**

public void jobsPaused(String jobGroup)

   **Description copied from interface:** SchedulerListener
   Called by the Scheduler when a group of JobDetails has been paused.

   **Specified by:**
   jobsPaused in interface SchedulerListener

   **Parameters:**
   jobGroup - the paused group, or null if all were paused

---

**jobsResumed**

public void jobsResumed(String jobGroup)
Description copied from interface: `SchedulerListener`

Called by the Scheduler when a group of JobDetails has been un-paused.

**Specified by:**

`jobsResumed` in interface `SchedulerListener`

---

**jobResumed**

public void `jobResumed(JobKey jobKey)`

Description copied from interface: `SchedulerListener`

Called by the Scheduler when a JobDetail has been un-paused.

**Specified by:**

`jobResumed` in interface `SchedulerListener`

---

**schedulerError**

public void `schedulerError(String msg, SchedulerException cause)`

Description copied from interface: `SchedulerListener`

Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

The `getErrorCode()` method of the given SchedulerException can be used to determine more specific information about the type of error that was encountered.

**Specified by:**

`schedulerError` in interface `SchedulerListener`

---

**schedulerStarted**
public void schedulerStarted()

Description copied from interface: SchedulerListener

Called by the Scheduler to inform the listener that it has started.

Specified by:
 schedulerStarted in interface SchedulerListener

--------------------------------------

schedulerInStandbyMode

public void schedulerInStandbyMode()

Description copied from interface: SchedulerListener

Called by the Scheduler to inform the listener that it has move to standby mode.

Specified by:
 schedulerInStandbyMode in interface SchedulerListener

--------------------------------------

schedulerShutdown

public void schedulerShutdown()

Description copied from interface: SchedulerListener

Called by the Scheduler to inform the listener that it has shutdown.

Specified by:
 schedulerShutdown in interface SchedulerListener

--------------------------------------

schedulerShuttingdown

public void schedulerShuttingdown()

Description copied from interface: SchedulerListener
Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

Specified by:

schedulerShuttingdown in interface SchedulerListener

---

triggerFinalized

```java
public void triggerFinalized(Trigger trigger)
```

Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.

Specified by:

triggerFinalized in interface SchedulerListener

---

triggersPaused

```java
public void triggersPaused(String triggerGroup)
```

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of Triggers has been paused.

If all groups were paused then triggerGroup will be null

Specified by:

triggersPaused in interface SchedulerListener

Parameters:

triggerGroup - the paused group, or null if all were paused

---

triggerPaused

```java
public void triggerPaused(TriggerKey triggerKey)
```
Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has been paused.

Specified by:
  triggerPaused in interface SchedulerListener

---

triggersResumed

```java
public void triggersResumed(String triggerGroup)
```

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of Triggers has been un-paused.

Specified by:
  triggersResumed in interface SchedulerListener

---

triggerResumed

```java
public void triggerResumed(TriggerKey triggerKey)
```

Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has been un-paused.

Specified by:
  triggerResumed in interface SchedulerListener

---

getName

```java
public String getName()
```

Description copied from interface: JobListener

Get the name of the JobListener.
Specified by:
  getName in interface JobListener

---

**jobExecutionVetoed**

```java
public void jobExecutionVetoed(JobExecutionContext context)
```

**Description copied from interface: jobExecutionVetoed in interface JobListener**

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

**Specified by:**
  jobExecutionVetoed in interface JobListener

**See Also:**
  JobListener.jobToBeExecuted(JobExecutionContext)

---

**jobToBeExecuted**

```java
public void jobToBeExecuted(JobExecutionContext context)
```

**Description copied from interface: jobToBeExecuted in interface JobListener**

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

This method will not be invoked if the execution of the Job was vetoed by a TriggerListener.

**Specified by:**
  jobToBeExecuted in interface JobListener

**See Also:**
  JobListener.jobExecutionVetoed(JobExecutionContext)

---

**jobWasExecuted**
public void jobWasExecuted(JobExecutionContext context,
JobExecutionException jobException)

Description copied from interface: JobListener

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

Specified by:

jobWasExecuted in interface JobListener

sendNotification

public void sendNotification(String eventType)

sendNotification

Parameters:

eventType -

sendNotification

public void sendNotification(String eventType,
Object data)

sendNotification

Parameters:

eventType -

data -

sendNotification

public void sendNotification(String eventType,
Object data,
String msg)

sendNotification
Parameters:
   eventType -
   data -
   msg -

addNotificationListener

public void \texttt{addNotificationListener}(\texttt{NotificationListener} notif, 
\texttt{NotificationFilter} filter, 
\texttt{Object} callBack)

Specified by:
   \texttt{addNotificationListener} in interface \texttt{NotificationBroadcaster}

See Also:
   \texttt{NotificationBroadcaster.addNotificationListener(javax.management.NotificationListener,
   javax.management.NotificationFilter, java.lang.Object)}

getNotificationInfo

public \texttt{MBeanNotificationInfo[]} \texttt{getNotificationInfo}()

Specified by:
   \texttt{getNotificationInfo} in interface \texttt{NotificationBroadcaster}

See Also:
   \texttt{NotificationBroadcaster.getNotificationInfo()}

removeNotificationListener

public void \texttt{removeNotificationListener}(\texttt{NotificationListener} listener 
throws \texttt{ListenerNotFoundException})

Specified by:
   \texttt{removeNotificationListener} in interface \texttt{NotificationBroadcaster}

Throws:
   \texttt{ListenerNotFoundException}

See Also:
   \texttt{NotificationBroadcaster.removeNotificationListener(javax.management.NotificationListener)}
removeNotificationListener

public void removeNotificationListener(NotificationListener notif, NotificationFilter filter, Object callBack)
throws ListenerNotFoundException

Specified by:
removeNotificationListener in interface NotificationEmitter

Throws:
ListenerNotFoundException

See Also:

isSampledStatisticsEnabled

public boolean isSampledStatisticsEnabled()

Specified by:
isSampledStatisticsEnabled in interface QuartzSchedulerMBean

setSampledStatisticsEnabled

public void setSampledStatisticsEnabled(boolean enabled)

Specified by:
setSampledStatisticsEnabled in interface QuartzSchedulerMBean

getJobsCompletedMostRecentSample

public long getJobsCompletedMostRecentSample()

Specified by:
getJobsCompletedMostRecentSample in interface QuartzSchedulerMBean
getJobsExecutedMostRecentSample

public long getJobsExecutedMostRecentSample()

Specified by:

getJobsExecutedMostRecentSample in interface QuartzSchedulerMBean

getJobsScheduledMostRecentSample

public long getJobsScheduledMostRecentSample()

Specified by:

getJobsScheduledMostRecentSample in interface QuartzSchedulerMBean

getPerformanceMetrics

public Map<String, Long> getPerformanceMetrics()

Specified by:

getPerformanceMetrics in interface QuartzSchedulerMBean
public class QuartzSchedulerResources
extends Object

Contains all of the resources (JobStore, ThreadPool, etc.) necessary to create a QuartzScheduler instance.

Author:
James House

See Also:
QuartzScheduler

Field Summary

| static String CREATE_REGISTRY_ALWAYS |
| static String CREATE_REGISTRY_AS_NEEDED |
| static String CREATE_REGISTRY_NEVER |

Constructor Summary

QuartzSchedulerResources()
Create an instance with no properties initialized.

Method Summary

void addSchedulerPlugin(org.quartz.spi.SchedulerPlugin plugin)
Add the given SchedulerPlugin for the

static String generateJMXObjectName(String schedName,
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>long getBatchTimeWindow()</code></td>
<td>Create the name under which this scheduler should be registered in JMX.</td>
</tr>
<tr>
<td><code>String getInstanceId()</code></td>
<td>Get the instance Id for the QuartzScheduler.</td>
</tr>
<tr>
<td><code>boolean getJMXExport()</code></td>
<td>Get whether the QuartzScheduler should be registered with the local JMX.</td>
</tr>
<tr>
<td><code>String getJMXObjectName()</code></td>
<td>Get the name under which the QuartzScheduler should be registered with JMX.</td>
</tr>
<tr>
<td><code>JobRunShellFactory getJobRunShellFactory()</code></td>
<td>Get the JobRunShellFactory for the QuartzScheduler.</td>
</tr>
<tr>
<td><code>String getRMIBindName()</code></td>
<td>Get the name under which to bind the QuartzScheduler in RMI.</td>
</tr>
<tr>
<td><code>String getRMICreateRegistryStrategy()</code></td>
<td>Get the setting of whether or not Quartz should create an RMI Registry.</td>
</tr>
<tr>
<td><code>String getRMIRegistryHost()</code></td>
<td>Get the host name of the RMI Registry.</td>
</tr>
<tr>
<td><code>int getRMIRegistryPort()</code></td>
<td>Get the port number of the RMI Registry.</td>
</tr>
<tr>
<td><code>int getRMIServerPort()</code></td>
<td>Get the port number the scheduler server will be bound to.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.SchedulerPlugins&gt; getSchedulerPlugins()</code></td>
<td>Get the List of all SchedulerPlugins for the QuartzScheduler.</td>
</tr>
<tr>
<td><code>String getThreadName()</code></td>
<td>Get the name for the QuartzSchedulerThread.</td>
</tr>
<tr>
<td><code>org.quartz.spi.ThreadPool getThreadPool()</code></td>
<td>Get the ThreadPool for the QuartzScheduler.</td>
</tr>
</tbody>
</table>
Get the ThreadPool for the QuartzScheduler

```
String getUniqueIdentifier()

static String getUniqueIdentifier(String schedName, S

boolean isInterruptJobsOnShutdown()

boolean isInterruptJobsOnShutdownWithWait()

boolean isRunUpdateCheck()

boolean isThreadsInheritInitializersClassLoader()

void setBatchTimeWindow(long batchTimeWindow)

void setInstanceId(String instanceId)

void setInterruptJobsOnShutdown(boolean interruptJobsOnShutdown)

void setInterruptJobsOnShutdownWithWait(boolean interruptJobsOnShutdownWithWait)

void setJMXExport(boolean jmxExport)

void setJMXObjectName(String jmxObjectName)

void setJobRunShellFactory(JobRunShellFactory)

void setJobStore(org.quartz.spi.JobStore jobStore)

void setMakeSchedulerThreadDaemon(boolean makeSchedulerThreadDaemon)

void setMaxBatchSize(int maxBatchSize)

void setName(String name)
```
void setRMIBindName(String rmiBindName)
Set the name under which to bind the QuartzScheduler in RMI.

void setRMICreateRegistryStrategy(String rmiCreateRegistryStrategy)
Set whether or not Quartz should create an RMI Registry, and if so, how.

void setRMIREgistryHost(String hostName)
Set the host name of the RMI Registry that the scheduler should export.

void setRMIREgistryPort(int port)
Set the port number of the RMI Registry that the scheduler should export.

void setRMIServerPort(int port)
Set the port number the scheduler server will be bound to.

void setRunUpdateCheck(boolean runUpdateCheck)

void setThreadName(String threadName)
Set the name for the QuartzSchedulerThread.

void setThreadPool(org.quartz.spi.ThreadPool threadPool)
Set the ThreadPool for the QuartzScheduler.

void setThreadsInheritInitializersClassLoaderContext(boolean inheritInitializersClassLoaderContext)
Set whether to set the class load context for spawned threads.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

CREATE_REGISTRY_NEVER
public static final String CREATE_REGISTRY_NEVER

See Also:
Constant Field Values
CREATE_REGISTRY_ALWAYS

public static final String CREATE_REGISTRY_ALWAYS

See Also:
Constant Field Values

CREATE_REGISTRY_AS_NEEDED

public static final String CREATE_REGISTRY_AS_NEEDED

See Also:
Constant Field Values

Constructor Detail

QuartzSchedulerResources

public QuartzSchedulerResources()

Create an instance with no properties initialized.

Method Detail

getName

public String getName()

Get the name for the QuartzScheduler.

setName

public void setName(String name)

Set the name for the QuartzScheduler.
Throws:  
**IllegalArgumentException** - if name is null or empty.

---

**getinstanceId**

```java
public String getInstanceId()
```

Get the instance Id for the QuartzScheduler.

---

**setInstanceId**

```java
public void setInstanceId(String instanceId)
```

Set the name for the QuartzScheduler.

**Throws:**  
**IllegalArgumentException** - if name is null or empty.

---

**getUniqueIdentifier**

```java
public static String getUniqueIdentifier(String schedName, String schedInstId)
```

---

**getUniqueIdentifier**

```java
public String getUniqueIdentifier()
```

---

**getRMIRegistryHost**

```java
public String getRMIRegistryHost()
```

Get the host name of the RMI Registry that the scheduler should export itself to.
setRMIRegistryHost

public void setRMIRegistryHost(String hostName)

    Set the host name of the RMI Registry that the scheduler should export itself to.

---------------------

getRMIRegistryPort

public int getRMIRegistryPort()

    Get the port number of the RMI Registry that the scheduler should export itself to.

---------------------

setRMIRegistryPort

public void setRMIRegistryPort(int port)

    Set the port number of the RMI Registry that the scheduler should export itself to.

---------------------

getRMIServerPort

public int getRMIServerPort()

    Get the port number the scheduler server will be bound to.

---------------------

setRMIServerPort

public void setRMIServerPort(int port)

    Set the port number the scheduler server will be bound to.

---------------------

getRMICreateRegistryStrategy
public String getRMICreateRegistryStrategy()

    Get the setting of whether or not Quartz should create an RMI Registry, and if so, how.

getThreadName

public String getThreadName()

    Get the name for the QuartzSchedulerThread.

setThreadName

public void setThreadName(String threadName)

    Set the name for the QuartzSchedulerThread.

    Throws: 
        IllegalArgumentException - if name is null or empty.

setRMICreateRegistryStrategy

public void setRMICreateRegistryStrategy(String rmiCreateRegistryStrategy)

    Set whether or not Quartz should create an RMI Registry, and if so, how.

    See Also: 
        CREATE_REGISTRY_ALWAYS, CREATE_REGISTRY_AS_NEEDED, CREATE_REGISTRY_NEVER

g ThreadPool

public org.quartz.spi.ThreadPool getThreadPool()

    Get the ThreadPool for the QuartzScheduler to use.
setThreadPool

public void setThreadPool(org.quartz.spi.ThreadPool threadPool)

Set the ThreadPool for the QuartzScheduler to use.

Throws:

IllegalArgumentException - if threadPool is null.

getJobStore

public org.quartz.spi.JobStore getJobStore()

Get the JobStore for the QuartzScheduler to use.

setJobStore

public void setJobStore(org.quartz.spi.JobStore jobStore)

Set the JobStore for the QuartzScheduler to use.

Throws:

IllegalArgumentException - if jobStore is null.

getJobRunShellFactory

public JobRunShellFactory getJobRunShellFactory()

Get the JobRunShellFactory for the QuartzScheduler to use.

setJobRunShellFactory

public void setJobRunShellFactory(JobRunShellFactory jobRunShellFact)

Set the JobRunShellFactory for the QuartzScheduler to use.
Throws:

IllegalArgumentException - if jobRunShellFactory is null.

---

**addSchedulerPlugin**

public void addSchedulerPlugin(org.quartz.spi.SchedulerPlugin plugin)

Add the given SchedulerPlugin for the QuartzScheduler to use. This method expects the plugin's "initialize" method to be invoked externally (either before or after this method is called).

---

**getSchedulerPlugins**

public List&lt;org.quartz.spi.SchedulerPlugin&gt; getSchedulerPlugins()

Get the List of all SchedulerPlugins for the QuartzScheduler to use.

---

**getMakeSchedulerThreadDaemon**

public boolean getMakeSchedulerThreadDaemon()

Get whether to mark the Quartz scheduling thread as daemon.

**See Also:**

Thread.setDaemon(boolean)

---

**setMakeSchedulerThreadDaemon**

public void setMakeSchedulerThreadDaemon(boolean makeSchedulerThread)

Set whether to mark the Quartz scheduling thread as daemon.

**See Also:**

Thread.setDaemon(boolean)
isThreadsInheritInitializersClassLoadContext

public boolean isThreadsInheritInitializersClassLoadContext()

Get whether to set the class load context of spawned threads to that of the initializing thread.

setThreadsInheritInitializersClassLoadContext

public void setThreadsInheritInitializersClassLoadContext(boolean threadsInheritInitializersClassLoadContext)

Set whether to set the class load context of spawned threads to that of the initializing thread.

getRMIBindName

public String getRMIBindName()

Get the name under which to bind the QuartzScheduler in RMI. Will return the value of the uniqueIdentifier property if explicit RMI bind name was never set.

See Also:
getUniqueIdentifier()

setRMIBindName

public void setRMIBindName(String rmiBindName)

Set the name under which to bind the QuartzScheduler in RMI. If unset, defaults to the value of the uniqueIdentifier property.

See Also:
getUniqueIdentifier()

getJMXExport
public boolean getJMXExport()

Get whether the QuartzScheduler should be registered with the local MBeanServer.

setJMXExport

public void setJMXExport(boolean jmxExport)

Set whether the QuartzScheduler should be registered with the local MBeanServer.

getJMXObjectName

public String getJMXObjectName()

Get the name under which the QuartzScheduler should be registered with the local MBeanServer. If unset, defaults to the value calculated by generateJMXObjectName.

See Also:
generateJMXObjectName(String, String)

setJMXObjectName

public void setJMXObjectName(String jmxObjectName)

Set the name under which the QuartzScheduler should be registered with the local MBeanServer. If unset, defaults to the value calculated by generateJMXObjectName.

See Also:
generateJMXObjectName(String, String)

generateJMXObjectName
public static String generateJMXObjectName(String schedName, String schedInstId)

Create the name under which this scheduler should be registered in JMX.

The name is composed as: quartz:type=QuartzScheduler,name=[schedName],instance=[schedInstId]

---

isRunUpdateCheck

public boolean isRunUpdateCheck()

---

setRunUpdateCheck

public void setRunUpdateCheck(boolean runUpdateCheck)

---

getBatchTimeWindow

public long getBatchTimeWindow()

---

setBatchTimeWindow

public void setBatchTimeWindow(long batchTimeWindow)

---

getMaxBatchSize

public int getMaxBatchSize()

---

setMaxBatchSize

public void setMaxBatchSize(int maxBatchSize)
isInterruptJobsOnShutdown

public boolean isInterruptJobsOnShutdown()

setInterruptJobsOnShutdown

public void setInterruptJobsOnShutdown(boolean interruptJobsOnShutdo

isInterruptJobsOnShutdownWithWait

public boolean isInterruptJobsOnShutdownWithWait()

setInterruptJobsOnShutdownWithWait

public void setInterruptJobsOnShutdownWithWait(boolean interruptJobs

Copyright 2001-2011, Terracotta, Inc.
public class QuartzSchedulerThread
extends Thread

The thread responsible for performing the work of firing Triggers that are registered with the QuartzScheduler.

Author:
James House

See Also:
QuartzScheduler, Job, Trigger

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class java.lang.Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread.State, Thread.UncaughtExceptionHandler</td>
</tr>
</tbody>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from class java.lang.Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>void clearSignaledSchedulingChange()</th>
</tr>
</thead>
</table>
void errorTriggerRetryLoop(org.quartz.spi.TriggerFiredBundle bndle)

org.slf4j.Logger getLog()

long getSignaledNextFireTime()

boolean isScheduleChanged()

void releaseTriggerRetryLoop(org.quartz.spi.OperableTrigger trigger)

void run()

The main processing loop of the QuartzSchedulerThread

void setDbFailureRetryInterval(long dbFailureRetryInterval)

void signalSchedulingChange(long candidateNewNextFireTime)

Signals the main processing loop that a change in scheduling has been made - in order to interrupt any sleeping that may be occurring waiting for the fire time to arrive.

<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>activeCount, checkAccess, clone, countStackFrames, currentThread, destroy, dumpStack, enumerate, getAllStackTraces, getExecutionContext, getDefaultValue, getPriority, getStackTrace, getState, getThreadGroup, getUncaughtExceptionHandler, holdsLock, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setDefaultUncaughtExceptionHandler, setName, setPriority, setUncaughtExceptionHandler, sleep, sleep, start, stop, stop, suspend, toString, yield</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait</td>
</tr>
</tbody>
</table>
### Method Detail

#### setDbFailureRetryInterval

```java
class Method
{
    public void setDbFailureRetryInterval(long dbFailureRetryInterval)
}
```

#### signalSchedulingChange

```java
class Method
{
    public void signalSchedulingChange(long candidateNewNextFireTime)
    {
        Signals the main processing loop that a change in scheduling has been made - in order to interrupt any sleeping that may be occurring while waiting for the fire time to arrive.

        **Parameters:**
        - `candidateNewNextFireTime`: the time (in millis) when the newly scheduled trigger will fire. If this method is being called do to some other event (rather than scheduling a trigger), the caller should pass zero (0).
    }
}
```

#### clearSignaledSchedulingChange

```java
class Method
{
    public void clearSignaledSchedulingChange()
}
```

#### isScheduleChanged

```java
class Method
{
    public boolean isScheduleChanged()
}
```

#### getSignaledNextFireTime

```java
class Method
{
    public long getSignaledNextFireTime()
}
```

#### run

```java
class Method
{
    run()
}
```
public void run()

    The main processing loop of the QuartzSchedulerThread.

    Specified by:  
        run in interface Runnable

    Overrides:  
        run in class Thread

errorTriggerRetryLoop

public void errorTriggerRetryLoop(org.quartz.spi.TriggerFiredBundle bundle)

releaseTriggerRetryLoop

public void releaseTriggerRetryLoop(org.quartz.spi.OperableTrigger trigger)

goingLog

public org.slf4j.Logger getLog()
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Summary</th>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
<th>Constructor</th>
<th>Field</th>
</tr>
</thead>
</table>
org.quartz.core Interface RemotableQuartzScheduler

All Superinterfaces:
    Remote

All Known Implementing Classes:
    QuartzScheduler

public interface RemotableQuartzScheduler
extends Remote

Author:
    James House

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</td>
</tr>
<tr>
<td>void addJob(JobDetail jobDetail, boolean replace)</td>
</tr>
<tr>
<td>boolean checkExists(JobKey jobKey)</td>
</tr>
<tr>
<td>boolean checkExists(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>void clear()</td>
</tr>
<tr>
<td>boolean deleteCalendar(String calName)</td>
</tr>
<tr>
<td>boolean deleteJob(JobKey jobKey)</td>
</tr>
<tr>
<td>boolean deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td><code>getCalendar(String calName)</code></td>
</tr>
<tr>
<td><code>List&lt;String&gt; getCalendarNames()</code></td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt; getCurrentlyExecutingJobs()</code></td>
</tr>
<tr>
<td><code>JobDetail getJobDetail(JobKey jobKey)</code></td>
</tr>
<tr>
<td><code>List&lt;String&gt; getJobGroupNames()</code></td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt; getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
</tr>
<tr>
<td><code>Class getJobStoreClass()</code></td>
</tr>
<tr>
<td><code>Set&lt;String&gt; getPausedTriggerGroups()</code></td>
</tr>
<tr>
<td><code>SchedulerContext getSchedulerContext()</code></td>
</tr>
<tr>
<td><code>String getSchedulerInstanceId()</code></td>
</tr>
<tr>
<td><code>String getSchedulerName()</code></td>
</tr>
<tr>
<td><code>Class getThreadPoolClass()</code></td>
</tr>
<tr>
<td><code>int getThreadPoolSize()</code></td>
</tr>
<tr>
<td><code>Trigger getTrigger(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td><code>List&lt;String&gt; getTriggerGroupNames()</code></td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt; getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
</tr>
<tr>
<td><code>List&lt;? extends Trigger&gt; getTriggersOfJob(JobKey jobKey)</code></td>
</tr>
<tr>
<td><code>Trigger.TriggerState</code></td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td><code>getTriggerState</code></td>
</tr>
<tr>
<td><code>getVersion</code></td>
</tr>
<tr>
<td><code>interrupt</code></td>
</tr>
<tr>
<td><code>isClustered</code></td>
</tr>
<tr>
<td><code>isInStandbyMode</code></td>
</tr>
<tr>
<td><code>isShutdown</code></td>
</tr>
<tr>
<td><code>numJobsExecuted</code></td>
</tr>
<tr>
<td><code>pauseAll</code></td>
</tr>
<tr>
<td><code>pauseJob</code></td>
</tr>
<tr>
<td><code>pauseJobs</code></td>
</tr>
<tr>
<td><code>pauseTrigger</code></td>
</tr>
<tr>
<td><code>pauseTriggers</code></td>
</tr>
<tr>
<td><code>rescheduleJob</code></td>
</tr>
<tr>
<td><code>resumeAll</code></td>
</tr>
<tr>
<td><code>resumeJob</code></td>
</tr>
<tr>
<td><code>resumeJobs</code></td>
</tr>
<tr>
<td><code>resumeTrigger</code></td>
</tr>
<tr>
<td>Method Name</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td><code>resumeTriggers</code></td>
</tr>
<tr>
<td><code>runningSince</code></td>
</tr>
<tr>
<td><code>scheduleJob</code></td>
</tr>
<tr>
<td><code>scheduleJob</code></td>
</tr>
<tr>
<td><code>scheduleJobs</code></td>
</tr>
<tr>
<td><code>shutdown</code></td>
</tr>
<tr>
<td><code>shutdown</code></td>
</tr>
<tr>
<td><code>standby</code></td>
</tr>
<tr>
<td><code>start</code></td>
</tr>
<tr>
<td><code>startDelayed</code></td>
</tr>
<tr>
<td><code>supportsPersistence</code></td>
</tr>
<tr>
<td><code>triggerJob</code></td>
</tr>
<tr>
<td><code>triggerJob</code></td>
</tr>
<tr>
<td><code>unscheduleJob</code></td>
</tr>
<tr>
<td><code>unscheduleJobs</code></td>
</tr>
</tbody>
</table>
getSchedulerName

String getSchedulerName() throws RemoteException

Throws:
   RemoteException

getSchedulerInstanceId

String getSchedulerInstanceId() throws RemoteException

Throws:
   RemoteException

getSchedulerContext

SchedulerContext getSchedulerContext() throws SchedulerException, RemoteException

Throws:
   SchedulerException
   RemoteException

start

void start() throws SchedulerException, RemoteException

Throws:
   SchedulerException
   RemoteException

startDelayed
void **startDelayed**(int seconds)  
throws SchedulerException, RemoteException

Throws:
- SchedulerException
- RemoteException

---

**standby**

void **standby**()  
throws RemoteException

Throws:
- RemoteException

---

**isInStandbyMode**

boolean **isInStandbyMode**()  
throws RemoteException

Throws:
- RemoteException

---

**shutdown**

void **shutdown**()  
throws RemoteException

Throws:
- RemoteException

---

**shutdown**

void **shutdown**(boolean waitForJobsToComplete)  
throws RemoteException
Throws: `RemoteException`

**isShutdown**

```java
boolean is_shutdown() throws RemoteException
```

Throws: `RemoteException`

**runningSince**

```java
Date running_since() throws RemoteException
```

Throws: `RemoteException`

**getVersion**

```java
String get_version() throws RemoteException
```

Throws: `RemoteException`

**numJobsExecuted**

```java
int num_jobs_executed() throws RemoteException
```

Throws: `RemoteException`

**getJobStoreClass**
Class getJobStoreClass() throws RemoteException

Throws:
    RemoteException

supportsPersistence

boolean supportsPersistence() throws RemoteException

Throws:
    RemoteException

isClustered

boolean isClustered() throws RemoteException

Throws:
    RemoteException

getThreadPoolClass

Class getThreadPoolClass() throws RemoteException

Throws:
    RemoteException

getThreadPoolSize

int getThreadPoolSize() throws RemoteException

Throws:
    RemoteException
void clear()
    throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

getCurrentlyExecutingJobs

List&lt;JobExecutionContext&gt; getCurrentlyExecutingJobs()
    throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

scheduleJob

Date scheduleJob(JobDetail jobDetail, Trigger trigger)
    throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

scheduleJob

Date scheduleJob(Trigger trigger)
    throws SchedulerException, RemoteException

Throws:
addJob

```java
void addJob(JobDetail jobDetail,
            boolean replace)
    throws SchedulerException,
            RemoteException
```

Throws:
- `SchedulerException`
- `RemoteException`

deleteJob

```java
boolean deleteJob(JobKey jobKey)
    throws SchedulerException,
            RemoteException
```

Throws:
- `SchedulerException`
- `RemoteException`

unscheduleJob

```java
boolean unscheduleJob(TriggerKey triggerKey)
    throws SchedulerException,
            RemoteException
```

Throws:
- `SchedulerException`
- `RemoteException`

rescheduleJob

```java
Date rescheduleJob(TriggerKey triggerKey,
                    Trigger newTrigger)
```
throws SchedulerException, RemoteException

Throws:
SchedulerException
RemoteException

triggerJob

void triggerJob(JobKey jobKey,
JobDataMap data)
throws SchedulerException,
RemoteException

Throws:
SchedulerException
RemoteException

triggerJob

void triggerJob(org.quartz.spi.OperableTrigger trig)
throws SchedulerException,
RemoteException

Throws:
SchedulerException
RemoteException

pauseTrigger

void pauseTrigger(TriggerKey triggerKey)
throws SchedulerException,
RemoteException

Throws:
SchedulerException
RemoteException
pauseTriggers

void pauseTriggers(GroupMatcher<TriggerKey> matcher)
     throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

pauseJob

void pauseJob(JobKey jobKey)
     throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

pauseJobs

void pauseJobs(GroupMatcher<JobKey> matcher)
     throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

resumeTrigger

void resumeTrigger(TriggerKey triggerKey)
     throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException
resumeTriggers

void resumeTriggers(GroupMatcher<TriggerKey> matcher)
  throws SchedulerException, RemoteException

  Throws:
  SchedulerException
  RemoteException

getPausedTriggerGroups

Set<String> getPausedTriggerGroups()
  throws SchedulerException, RemoteException

  Throws:
  SchedulerException
  RemoteException

resumeJob

void resumeJob(JobKey jobKey)
  throws SchedulerException, RemoteException

  Throws:
  SchedulerException
  RemoteException

resumeJobs

void resumeJobs(GroupMatcher<JobKey> matcher)
  throws SchedulerException, RemoteException

  Throws:
  SchedulerException
PauseAll

void pauseAll()
throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

ResumeAll

void resumeAll()
throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

GetJobGroupNames

List<String> getJobGroupNames()
throws SchedulerException, RemoteException

Throws:
    SchedulerException
    RemoteException

GetJobKeys

Set<JobKey> getJobKeys(GroupMatcher<JobKey> matcher)
throws SchedulerException, RemoteException
Throws:  
SchedulerException  
RemoteException

getTriggersOfJob

List<? extends Trigger> getTriggersOfJob(JobKey jobKey) throws SchedulerException, RemoteException

Throws:  
SchedulerException  
RemoteException

getTriggerGroupNames

List<String> getTriggerGroupNames() throws SchedulerException, RemoteException

Throws:  
SchedulerException  
RemoteException

getTriggerKeys

Set<TriggerKey> getTriggerKeys(GroupMatcher<TriggerKey> matcher) throws SchedulerException, RemoteException

Throws:  
SchedulerException  
RemoteException

getJobDetail

JobDetail getJobDetail(JobKey jobKey) throws SchedulerException,
getTrigger

```
Trigger getTrigger(TriggerKey triggerKey)
throws SchedulerException, RemoteException
```

Throws:

```
SchedulerException
RemoteException
```

getTriggerState

```
Trigger.TriggerState getTriggerState(TriggerKey triggerKey)
throws SchedulerException, RemoteException
```

Throws:

```
SchedulerException
RemoteException
```

addCalendar

```
void addCalendar(String calName, Calendar calendar,
boolean replace,
boolean updateTriggers)
throws SchedulerException, RemoteException
```

Throws:

```
SchedulerException
RemoteException
```
deleteCalendar

boolean deleteCalendar(String calName)
throws SchedulerException, RemoteException

Throws:
SchedulerException
RemoteException

getCalendar

Calendar getCalendar(String calName)
throws SchedulerException, RemoteException

Throws:
SchedulerException
RemoteException

getCalendarNames

List<String> getCalendarNames()
throws SchedulerException, RemoteException

Throws:
SchedulerException
RemoteException

interrupt

boolean interrupt(JobKey jobKey)
throws UnableToInterruptJobException, RemoteException

Throws:
UnableToInterruptJobException
RemoteException
checkExists

boolean checkExists(JobKey jobKey)
throws SchedulerException,
RemoteException

Throws:
SchedulerException
RemoteException

deleJobs

boolean deleteJobs(List<JobKey> jobKeys)
throws SchedulerException,
RemoteException

Throws:
SchedulerException
RemoteException

disScheduleJobs

void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs,
boolean replace)
throws SchedulerException,
RemoteException

Throws:
unscheduleJobs

boolean unscheduleJobs(List<TriggerKey> triggerKeys)
throws SchedulerException, RemoteException

Throws:
SchedulerException
RemoteException
org.quartz.core Interface SampledStatistics

All Known Implementing Classes:
   NullSampledStatisticsImpl, SampledStatisticsImpl

public interface SampledStatistics

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long getJobsCompletedMostRecentSample()</td>
<td></td>
</tr>
<tr>
<td>long getJobsExecutingMostRecentSample()</td>
<td></td>
</tr>
<tr>
<td>long getJobsScheduledMostRecentSample()</td>
<td></td>
</tr>
<tr>
<td>void shutdown()</td>
<td></td>
</tr>
</tbody>
</table>

Method Detail

getJobsScheduledMostRecentSample

long getJobsScheduledMostRecentSample()  

getJobsExecutingMostRecentSample

long getJobsExecutingMostRecentSample()  

getJobsCompletedMostRecentSample

long getJobsCompletedMostRecentSample()  

void shutdown()
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clearStatistics()</td>
<td>Clears the collected statistics.</td>
</tr>
<tr>
<td>long getJobsCompletedMostRecentSample()</td>
<td></td>
</tr>
<tr>
<td>long getJobsExecutingMostRecentSample()</td>
<td></td>
</tr>
<tr>
<td>long getJobsScheduledMostRecentSample()</td>
<td></td>
</tr>
<tr>
<td>String getName()</td>
<td>Get the name of the JobListener.</td>
</tr>
<tr>
<td>void jobAdded(JobDetail jobDetail)</td>
<td>Called by the Scheduler when a JobDetail has been added.</td>
</tr>
<tr>
<td>void jobDeleted(String jobName, String groupName)</td>
<td></td>
</tr>
<tr>
<td>void jobExecutionVetoed(JobExecutionContext context)</td>
<td>Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.</td>
</tr>
</tbody>
</table>
jobScheduled(Trigger trigger)
Called by the Scheduler when a JobDetail is scheduled.

void jobToBeExecuted(JobExecutionContext context)
Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)
Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

void shutdown()
public void shutdown()

    Specified by:
    shutdown in interface SampledStatistics

clearStatistics
public void clearStatistics()

    Clears the collected statistics. Resets all counters to zero

getJobsCompletedMostRecentSample
public long getJobsCompletedMostRecentSample()

    Specified by:
    getJobsCompletedMostRecentSample in interface SampledStatistics

getJobsExecutingMostRecentSample
public long getJobsExecutingMostRecentSample()

    Specified by:
    getJobsExecutingMostRecentSample in interface SampledStatistics

getJobsScheduledMostRecentSample
public long getJobsScheduledMostRecentSample()

    Specified by:
    getJobsScheduledMostRecentSample in interface SampledStatistics
**getName**

```java
public String getName()
```

**Description copied from interface: JobListener**

Get the name of the JobListener.

**Specified by:**

`getName` in interface `JobListener`

---

**jobScheduled**

```java
public void jobScheduled(Trigger trigger)
```

**Description copied from interface: SchedulerListener**

Called by the Scheduler when a JobDetail is scheduled.

**Specified by:**

`jobScheduled` in interface `SchedulerListener`

**Overrides:**

`jobScheduled` in class `SchedulerListenerSupport`

---

**jobExecutionVetoed**

```java
public void jobExecutionVetoed(JobExecutionContext context)
```

**Description copied from interface: JobListener**

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

**Specified by:**

`jobExecutionVetoed` in interface `JobListener`

**See Also:**

`JobListener.jobToBeExecuted(JobExecutionContext)`
**jobToBeExecuted**

```java
public void jobToBeExecuted(JobExecutionContext context)
```

**Description copied from interface: JobListener**

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

This method will not be invoked if the execution of the Job was vetoed by a TriggerListener.

**Specified by:**

`jobToBeExecuted` in interface `JobListener`

**See Also:**

`JobListener.jobExecutionVetoed(JobExecutionContext)`

---

**jobWasExecuted**

```java
public void jobWasExecuted(JobExecutionContext context,
                            JobExecutionException jobException)
```

**Description copied from interface: JobListener**

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

**Specified by:**

`jobWasExecuted` in interface `JobListener`

---

**jobAdded**

```java
public void jobAdded(JobDetail jobDetail)
```

**Description copied from interface: SchedulerListener**

Called by the Scheduler when a JobDetail has been added.
Specified by:

jobAdded in interface SchedulerListener

Overrides:

jobAdded in class SchedulerListenerSupport

---

jobDeleted

public void jobDeleted(String jobName, String groupName)
public class SchedulerSignalerImpl
extends Object
implements org.quartz.spi.SchedulerSignaler

An interface to be used by JobStore instances in order to communicate signals back to the QuartzScheduler.

Author:
  jhouse

Field Summary

| protected QuartzScheduler sched |
| protected QuartzSchedulerThread schedThread |

Constructor Summary

SchedulerSignalerImpl(QuartzScheduler sched, QuartzSchedulerThread schedThread)

Method Summary

void notifySchedulerListenersFinalized(Trigger trigger)
void notifySchedulerListenersJobDeleted(JobKey jobKey)

void notifyTriggerListenersMisfired(Trigger trigger)

void signalSchedulingChange(long candidateNewNextFireTime)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail
sched
protected QuartzScheduler sched

schedThread
protected QuartzSchedulerThread schedThread

Constructor Detail
SchedulerSignalerImpl
public SchedulerSignalerImpl(QuartzScheduler sched, QuartzSchedulerThread schedThread)

Method Detail
notifyTriggerListenersMisfired
public void notifyTriggerListenersMisfired(Trigger trigger)
Specified by:

notifyTriggerListenersMisfired in interface org.quartz.spi.SchedulerSignaler

### notifySchedulerListenersFinalized

```java
public void notifySchedulerListenersFinalized(Trigger trigger)
```

Specified by:

notifySchedulerListenersFinalized in interface org.quartz.spi.SchedulerSignaler

### signalSchedulingChange

```java
public void signalSchedulingChange(long candidateNewNextFireTime)
```

Specified by:

signalSchedulingChange in interface org.quartz.spi.SchedulerSignaler

### notifySchedulerListenersJobDeleted

```java
public void notifySchedulerListenersJobDeleted(JobKey jobKey)
```

Specified by:

notifySchedulerListenersJobDeleted in interface org.quartz.spi.SchedulerSignaler

Overview Package Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES NO FRAMES DETAIL: FIELD | CONSTR | METHOD

Copyright 2001-2011, Terracotta, Inc.
## Uses of Class org.quartz.core.JobRunShell

### Packages that use JobRunShell

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.ee.jta</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and</td>
</tr>
<tr>
<td></td>
<td>other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of JobRunShell in org.quartz.core

### Methods in org.quartz.core that return JobRunShell

- `JobRunShellFactory.createJobRunShell(org.quartz.spi.TriggerFiredBundle bundle)`
  Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

### Uses of JobRunShell in org.quartz.ee.jta

### Subclasses of JobRunShell in org.quartz.ee.jta

- `JTAJobRunShell`
  An extension of JobRunShell that begins an XA transaction before executing the Job, and commits (or rolls-back) the transaction after execution completes.

### Methods in org.quartz.ee.jta that return JobRunShell

- `JTAAnnotationAwareJobRunShellFactory.createJobRunShell(TriggerFiredBundle bundle)`
Called by the QuartzSchedulerThread to obtain instances of JTAJobRunShellFactory.\texttt{createJobRunShell}(org.quartz.spi.TriggerFiredBundle \texttt{bundle})

### Uses of JobRunShell in org.quartz.impl

### Methods in org.quartz.impl that return JobRunShell

\texttt{JobRunShell}

\texttt{StdJobRunShellFactory.\texttt{createJobRunShell}}(org.quartz.spi.TriggerFiredBundle \texttt{bundle})

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.
### Uses of Interface

**org.quartz.core.JobRunShellFactory**

<table>
<thead>
<tr>
<th>Packages that use <code>JobRunShellFactory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz.core</strong></td>
</tr>
<tr>
<td><strong>org.quartz.ee.jta</strong></td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
</tr>
</tbody>
</table>

### Uses of `JobRunShellFactory` in `org.quartz.core`

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.core</code> that return <code>JobRunShellFactory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobRunShellFactory</code></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.core</code> with parameters of type <code>JobRunShellFactory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

### Uses of `JobRunShellFactory` in `org.quartz.ee.jta`

<table>
<thead>
<tr>
<th>Classes in <code>org.quartz.ee.jta</code> that implement <code>JobRunShellFactory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JTAAnnotationAwareJobRunShellFactory</code></td>
</tr>
</tbody>
</table>
used within the QuartzScheduler instance.

<table>
<thead>
<tr>
<th>class</th>
<th>JTAJobRunShellFactory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Responsible for creating the instances of JTAJobRunShell to be used within the QuartzScheduler instance.</td>
</tr>
</tbody>
</table>

Uses of JobRunShellFactory in org.quartz.impl

Classes in org.quartz.impl that implement JobRunShellFactory

<table>
<thead>
<tr>
<th>class</th>
<th>StdJobRunShellFactory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Responsible for creating the instances of JobRunShell to be used within the QuartzScheduler instance.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class

org.quartz.core.ListenerManagerImpl

No usage of org.quartz.core.ListenerManagerImpl

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.core_NullSampledStatisticsImpl

No usage of org.quartz.core_NullSampledStatisticsImpl

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.core.QuartzScheduler**

## Packages that use `QuartzScheduler`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.core</code></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><code>org.quartz.impl</code></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Uses of `QuartzScheduler` in `org.quartz.core`

### Fields in `org.quartz.core` declared as `QuartzScheduler`

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected QuartzScheduler JobRunShell.qs</code></td>
<td></td>
</tr>
<tr>
<td><code>protected QuartzScheduler SchedulerSignalerImpl.sched</code></td>
<td></td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.core` with parameters of type `QuartzScheduler`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobRunShell.initialize</code></td>
<td><code>(QuartzScheduler qs)</code></td>
</tr>
</tbody>
</table>

### Constructors in `org.quartz.core` with parameters of type `QuartzScheduler`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QuartzSchedulerMBeanImpl</code></td>
<td><code>(QuartzScheduler scheduler)</code></td>
</tr>
<tr>
<td><code>SchedulerSignalerImpl</code></td>
<td><code>(QuartzScheduler sched, QuartzSchedulerThread schedThread)</code></td>
</tr>
</tbody>
</table>
# Uses of QuartzScheduler in org.quartz.impl

## Methods in org.quartz.impl with parameters of type QuartzScheduler

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| `StdSchedulerFactory.instantiate(QuartzSchedulerResources rs, QuartzScheduler qs)` | protected Scheduler StdSchedulerFactory

## Constructors in org.quartz.impl with parameters of type QuartzScheduler

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StdScheduler(QuartzScheduler sched)</code></td>
<td>Construct a StdScheduler instance to proxy the given QuartzScheduler instance, and with the given SchedulingContext.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.core.QuartzSchedulerMBeanImpl

No usage of org.quartz.core.QuartzSchedulerMBeanImpl

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.core.QuartzSchedulerResources**

<table>
<thead>
<tr>
<th>Packages that use QuartzSchedulerResources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz.core</strong></td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
</tr>
</tbody>
</table>

## Uses of QuartzSchedulerResources in org.quartz.core

### Constructors in org.quartz.core with parameters of type QuartzSchedulerResources

<table>
<thead>
<tr>
<th>QuartzScheduler</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>(QuartzSchedulerResources resources, long idleWaitTime, long dbRetryInterval)</td>
<td>Create a QuartzScheduler with the given configuration properties.</td>
</tr>
</tbody>
</table>

## Uses of QuartzSchedulerResources in org.quartz.impl

### Methods in org.quartz.impl with parameters of type QuartzSchedulerResources

<table>
<thead>
<tr>
<th>StdSchedulerFactory</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>instantiate</td>
<td>QuartzSchedulerResources rs, QuartzScheduler qs</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.core.QuartzSchedulerThread

 Packages that use QuartzSchedulerThread

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
</tbody>
</table>

Uses of QuartzSchedulerThread in org.quartz.core

 Fields in org.quartz.core declared as QuartzSchedulerThread

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedThread</td>
<td>QuartzSchedulerThread</td>
</tr>
</tbody>
</table>

Constructors in org.quartz.core with parameters of type QuartzSchedulerThread

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerSignalerImpl(QQuartzScheduler sched, QuartzSchedulerThread schedThread)</td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.core.RemotableQuartzScheduler

<table>
<thead>
<tr>
<th>Packages that use RemotableQuartzScheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
</tr>
<tr>
<td>org.quartz.impl</td>
</tr>
</tbody>
</table>

Uses of RemotableQuartzScheduler in org.quartz.core

<table>
<thead>
<tr>
<th>Classes in org.quartz.core that implement RemotableQuartzScheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>class QuartzScheduler</td>
</tr>
</tbody>
</table>

Uses of RemotableQuartzScheduler in org.quartz.impl

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl that return RemotableQuartzScheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected RemotableQuartzScheduler RemoteScheduler. getRemoteScheduler()</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

**org.quartz.core.SampledStatistics**

<table>
<thead>
<tr>
<th>Packages that use SampledStatistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
</tr>
<tr>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
</tbody>
</table>

## Uses of SampledStatistics in org.quartz.core

<table>
<thead>
<tr>
<th>Classes in org.quartz.core that implement SampledStatistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>class NullSampledStatisticsImpl</td>
</tr>
<tr>
<td>class SampledStatisticsImpl</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.core.SampledStatisticsImpl

No usage of org.quartz.core.SampledStatisticsImpl

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.core.SchedulerSignalerImpl

No usage of org.quartz.core.SchedulerSignalerImpl

Copyright 2001-2011, Terracotta, Inc.
public class CronTriggerSupport
extends Object

Constructor Summary

CronTriggerSupport()

Method Summary

static String[] getItemDescriptions()
static String[] getItemNames()
static OpenType[] getItemTypes()
static org.quartz.spi.OperableTrigger newTrigger(CompositeData cData)
static org.quartz.spi.OperableTrigger newTrigger(Map<String, Object> attrMap)
static CompositeData toCompositeData(CronTrigger trigger)
static TabularData toTabularData(List<? extends CronTrigger> triggers)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
## Constructor Detail

**CronTriggerSupport**

```java
public CronTriggerSupport()
```

## Method Detail

### getItemNames

```java
public static String[] getItemNames()
```

### getItemDescriptions

```java
public static String[] getItemDescriptions()
```

### getItemTypes

```java
public static OpenType[] getItemTypes()
```

### toCompositeData

```java
public static CompositeData toCompositeData(CronTrigger trigger)
```

### toTabularData

```java
public static TabularData toTabularData(List<? extends CronTrigger> triggers)
```

### newTrigger

```java
```
public static org.quartz.spi.OperableTrigger newTrigger(CompositeData throws ParseException

Throws:
ParseException

newTrigger

public static org.quartz.spi.OperableTrigger newTrigger(Map<String, Object> throws ParseException

Throws:
ParseException

Overview Package Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

Copyright 2001-2011, Terracotta, Inc.
**Class JobDataMapSupport**

*java.lang.Object*

`org.quartz.core.jmx.JobDataMapSupport` extends `Object`

### Field Summary

<table>
<thead>
<tr>
<th>static <code>TabularType</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>TABULAR_TYPE</td>
</tr>
</tbody>
</table>

### Constructor Summary

- `JobDataMapSupport()`

### Method Summary

<table>
<thead>
<tr>
<th>static <code>JobDataMap</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>newJobDataMap(Map&lt;String, Object&gt; map)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <code>JobDataMap</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>newJobDataMap(TabularData tabularData)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <code>CompositeData</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>toCompositeData(String key, String value)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <code>TabularData</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>toTabularData(JobDataMap jobDataMap)</code></td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
Field Detail

TABULAR_TYPE

public static final TabularType TABULAR_TYPE

Constructor Detail

JobDataMapSupport

public JobDataMapSupport()

Method Detail

newJobDataMap

public static JobDataMap newJobDataMap(TabularData tabularData)

toCompositeData

public static CompositeData toCompositeData(String key, String value)

    Returns:
    composite data

toTabularData

public static TabularData toTabularData(JobDataMap jobDataMap)
Parameters:
  jobDataMap -

Returns:
  TabularData
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td>FRAME</td>
<td>NO FRAMES</td>
<td></td>
<td>DETAIL: FIELD</td>
</tr>
</tbody>
</table>
public class JobDetailSupport
extends Object

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDetailSupport()</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>static JobDetail newJobDetail(CompositeData cData)</td>
<td></td>
</tr>
<tr>
<td>static JobDetail newJobDetail(Map&lt;String, Object&gt; attrMap)</td>
<td></td>
</tr>
<tr>
<td>static CompositeData toCompositeData(JobDetail jobDetail)</td>
<td></td>
</tr>
<tr>
<td>static TabularData toTabularData(JobDetail[] jobDetails)</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

JobDetailSupport
public JobDetailSupport()

| Method Detail |

**newJobDetail**

```java
public static JobDetail newJobDetail(CompositeData cData)
throws ClassNotFoundException
```

*Parameters:*
- cData -

*Returns:*
- JobDetail

*Throws:*
- ClassNotFoundException

```java
public static JobDetail newJobDetail(Map<String, Object> attrMap)
throws ClassNotFoundException
```

*Parameters:*
- Map - Object>

*Returns:*
- JobDetail

*Throws:*
- ClassNotFoundException

**toCompositeData**

```java
public static CompositeData toCompositeData(JobDetail jobDetail)
```

*Parameters:*
- jobDetail -

*Returns:*
- CompositeData
public static TabularData toTabularData(JobDetail[] jobDetails)
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
public class JobExecutionContextSupport
extends Object

Constructor Summary

| JobExecutionContextSupport() |

Method Summary

| toCompositeData(JobExecutionContext jec) |
| toTabularData(List<JobExecutionContext> executingJobs) |

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

public JobExecutionContextSupport()
toCompositeData

public static CompositeData toCompositeData(JobExecutionContext jec) throws SchedulerException

Returns:
    composite data

Throws:
    SchedulerException

toTabularData

public static TabularData toTabularData(List<JobExecutionContext> ex) throws SchedulerException

Parameters:
    tabularData -

Returns:
    array of region statistics

Throws:
    SchedulerException
org.quartz.core.jmx

Interfaces  QuartzSchedulerMBean

Classes  CronTriggerSupport  JobDataMapSupport  JobDetailSupport  JobExecutionContextSupport  SimpleTriggerSupport  TriggerSupport
## Interface Summary

| QuartzSchedulerMBean |

## Class Summary

<table>
<thead>
<tr>
<th>CronTriggerSupport</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDataMapSupport</td>
</tr>
<tr>
<td>JobDetailSupport</td>
</tr>
<tr>
<td>JobExecutionContextSupport</td>
</tr>
<tr>
<td>SimpleTriggerSupport</td>
</tr>
<tr>
<td>TriggerSupport</td>
</tr>
</tbody>
</table>
Hierarchy For Package org.quartz.core.jmx

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.core.jmx.**CronTriggerSupport**
  - org.quartz.core.jmx.**JobDataMapSupport**
  - org.quartz.core.jmx.**JobDetailSupport**
  - org.quartz.core.jmx.**JobExecutionContextSupport**
  - org.quartz.core.jmx.**SimpleTriggerSupport**
  - org.quartz.core.jmx.**TriggerSupport**
Interface Hierarchy

- org.quartz.core.jmx.QQuartzSchedulerMBean

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.core.jmx

<table>
<thead>
<tr>
<th>Packages that use <strong>org.quartz.core.jmx</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in <strong>org.quartz.core.jmx</strong> used by <strong>org.quartz.core</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzSchedulerMBean</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
**Interface QuartzSchedulerMBean**

All Known Implementing Classes:  
*QuartzSchedulerMBeanImpl*

```java
public interface QuartzSchedulerMBean

Field Summary
```

<table>
<thead>
<tr>
<th>static String</th>
<th>JOB_ADDED</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>JOB_DELETED</td>
</tr>
<tr>
<td>static String</td>
<td>JOB_EXECUTION_VETOED</td>
</tr>
<tr>
<td>static String</td>
<td>JOB_SCHEDULED</td>
</tr>
<tr>
<td>static String</td>
<td>JOB_TO_BE_EXECUTED</td>
</tr>
<tr>
<td>static String</td>
<td>JOB_UNSCHEDULED</td>
</tr>
<tr>
<td>static String</td>
<td>JOB_WAS_EXECUTED</td>
</tr>
<tr>
<td>static String</td>
<td>JOBS_PAUSED</td>
</tr>
<tr>
<td>static String</td>
<td>JOBS_RESUMED</td>
</tr>
<tr>
<td>static String</td>
<td>SAMPLED_STATISTICS_ENABLED</td>
</tr>
<tr>
<td>static String</td>
<td>SAMPLED_STATISTICS_RESET</td>
</tr>
<tr>
<td>static String</td>
<td>SCHEDULER_ERROR</td>
</tr>
<tr>
<td>Method Summary</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>void addJob(CompositeData jobDetail, boolean replace)</td>
<td></td>
</tr>
<tr>
<td>void addJob(Map&lt;String, Object&gt; abstractJobInfo, boolean replace)</td>
<td></td>
</tr>
<tr>
<td>Adds a durable job described by abstractJobInfo, which must contain the fully-qualified JobDetail class name under the key &quot;jobDetailClass.&quot; That JobDetail type must contain a no-arg constructor and have public access.</td>
<td></td>
</tr>
<tr>
<td>void clear()</td>
<td></td>
</tr>
<tr>
<td>void deleteCalendar(String name)</td>
<td></td>
</tr>
<tr>
<td>boolean deleteJob(String jobName, String jobGroupName)</td>
<td></td>
</tr>
<tr>
<td>TabularData getAllJobDetails()</td>
<td></td>
</tr>
<tr>
<td>List&lt;CompositeData&gt; getAllTriggers()</td>
<td></td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td></td>
</tr>
</tbody>
</table>
- `getCalendarNames()`
- `TabularData getCurrentlyExecutingJobs()`
- `CompositeData getJobDetail(String jobName, String jobGroupName)`
- `List<String> getJobGroupNames()`
- `List<String> getJobNames(String groupName)`
- `long getJobsCompletedMostRecentSample()`
- `long getJobsExecutedMostRecentSample()`
- `long getJobsScheduledMostRecentSample()`
- `String getJobStoreClassName()`
- `Set<String> getPausedTriggerGroups()`
- `Map<String, Long> getPerformanceMetrics()`
- `String getSchedulerInstanceId()`
- `String getSchedulerName()`
- `String getThreadPoolClassName()`
- `int getThreadPoolSize()`
- `CompositeData getTrigger(String triggerName, String triggerGroup)`
- `List<String> getTriggerGroupNames()`
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;String&gt; getTriggerNames(String triggerGroupName)</td>
<td>Get the names of all triggers in a specified trigger group.</td>
</tr>
<tr>
<td>List&lt;CompositeData&gt; getTriggersOfJob(String jobName, String jobGroupName)</td>
<td>Get triggers associated with a specific job and job group.</td>
</tr>
<tr>
<td>String getTriggerState(String triggerName, String triggerGroupName)</td>
<td>Get the state of a trigger in a specified trigger group.</td>
</tr>
<tr>
<td>String getVersion()</td>
<td>Get the version of the system.</td>
</tr>
<tr>
<td>boolean interruptJob(String jobName, String jobGroupName)</td>
<td>Interrupt a job in a specified job group.</td>
</tr>
<tr>
<td>boolean isSampledStatisticsEnabled()</td>
<td>Check if sampled statistics are enabled.</td>
</tr>
<tr>
<td>boolean isShutdown()</td>
<td>Check if the system is in shutdown mode.</td>
</tr>
<tr>
<td>boolean isStandbyMode()</td>
<td>Check if the system is in standby mode.</td>
</tr>
<tr>
<td>boolean isStarted()</td>
<td>Check if the system is started.</td>
</tr>
<tr>
<td>void pauseAllTriggers()</td>
<td>Pause all triggers in the system.</td>
</tr>
<tr>
<td>void pauseJob(String jobName, String groupName)</td>
<td>Pause a specific job in a specified job group.</td>
</tr>
<tr>
<td>void pauseJobGroup(String jobGroup)</td>
<td>Pause all jobs in a specified job group.</td>
</tr>
<tr>
<td>void pauseJobsContaining(String jobGroupToken)</td>
<td>Pause all jobs whose group contains a specified token.</td>
</tr>
<tr>
<td>void pauseJobsEndingWith(String jobGroupSuffix)</td>
<td>Pause all jobs whose group ends with a specified suffix.</td>
</tr>
<tr>
<td>void pauseJobsStartingWith(String jobGroupPrefix)</td>
<td>Pause all jobs whose group starts with a specified prefix.</td>
</tr>
<tr>
<td>void pauseTrigger(String triggerName, String triggerGroupName)</td>
<td>Pause a specific trigger in a specified trigger group.</td>
</tr>
<tr>
<td>void pauseTriggerGroup(String triggerGroup)</td>
<td>Pause all triggers in a specified trigger group.</td>
</tr>
</tbody>
</table>
void pauseTriggersContaining(String triggerGroupToken)
    Pause all triggers whose group contains triggerGroupToken

void pauseTriggersEndingWith(String suffix)
    Pause all triggers whose group ends with triggerGroupSuffix

void pauseTriggersStartingWith(String triggerGroupPrefix)
    Pause all triggers whose group starts with triggerGroupPrefix

void resumeAllTriggers()

void resumeJob(String jobName, String jobGroupName)

void resumeJobGroup(String jobGroup)

void resumeJobsContaining(String jobGroupToken)
    Resume all jobs whose group contains jobGroupToken

void resumeJobsEndingWith(String jobGroupSuffix)
    Resume all jobs whose group ends with jobGroupSuffix

void resumeJobsStartingWith(String jobGroupPrefix)
    Resume all jobs whose group starts with jobGroupPrefix

void resumeTrigger(String triggerName, String triggerGroupName)

void resumeTriggerGroup(String triggerGroup)

void resumeTriggersContaining(String triggerGroupToken)
    Resume all triggers whose group contains triggerGroupToken

void resumeTriggersEndingWith(String triggerGroupSuffix)
    Resume all triggers whose group ends with triggerGroupSuffix

void resumeTriggersStartingWith(String triggerGroupPrefix)
    Resume all triggers whose group starts with
void scheduleBasicJob(Map<String, Object> jobDetailInfo, Map<String, Object> triggerInfo)
    Schedules a job using the given Cron/Simple triggerInfo.

void scheduleJob(Map<String, Object> abstractJobInfo, Map<String, Object> abstractTriggerInfo)
    Schedules an arbitrary job described by abstractJobInfo using a trigger specified by abstractTriggerInfo.

void scheduleJob(String jobName, String jobGroup, Map<String, Object> abstractTriggerInfo)
    Schedules the specified job using a trigger described by abstractTriggerInfo, which must contain the fully-qualified trigger class name under the key "triggerClass." That trigger type must contain a no-arg constructor and have public access.

Data scheduleJob(String jobName, String jobGroup, String triggerName, String triggerGroup)
    Schedule an existing job with an existing trigger.

void setSampledStatisticsEnabled(boolean enabled)

void shutdown()

void standby()

void start()

void triggerJob(String jobName, String jobGroupName, Map<String, String> jobDataMap)

boolean unscheduleJob(String triggerName, String triggerGroup)

Field Detail

SCHEDULER_STARTED
static final String SCHEDULER_STARTED

See Also:
Constant Field Values

SCHEDULER_PAUSED

static final String SCHEDULER_PAUSED

See Also:
Constant Field Values

SCHEDULER_SHUTDOWN

static final String SCHEDULER_SHUTDOWN

See Also:
Constant Field Values

SCHEDULER_ERROR

static final String SCHEDULER_ERROR

See Also:
Constant Field Values

JOB_ADDED

static final String JOB_ADDED

See Also:
Constant Field Values

JOB_DELETED
static final String JOB_DELETED

See Also:
Constant Field Values

JOB_SCHEDULED

static final String JOB_SCHEDULED

See Also:
Constant Field Values

JOB_UNSCHEDULED

static final String JOB_UNSCHEDULED

See Also:
Constant Field Values

JOBS_PAUSED

static final String JOBS_PAUSED

See Also:
Constant Field Values

JOBS_RESUMED

static final String JOBS_RESUMED

See Also:
Constant Field Values

JOB_EXECUTION_VETOED
static final String JOB_EXECUTION_VETOED

See Also:
   Constant Field Values

---

JOB_TO_BE_EXECUTED

static final String JOB_TO_BE_EXECUTED

See Also:
   Constant Field Values

---

JOB_WAS_EXECUTED

static final String JOB_WAS_EXECUTED

See Also:
   Constant Field Values

---

TRIGGER_FINALIZED

static final String TRIGGER_FINALIZED

See Also:
   Constant Field Values

---

TRIGGERS_PAUSED

static final String TRIGGERS_PAUSED

See Also:
   Constant Field Values

---

TRIGGERS_RESUMED

---
static final String TRIGGERS_RESUMED

See Also:
Constant Field Values

SCHEDULING_DATA_CLEARED
static final String SCHEDULING_DATA_CLEARED

See Also:
Constant Field Values

SAMPLED_STATISTICS_ENABLED
static final String SAMPLED_STATISTICS_ENABLED

See Also:
Constant Field Values

SAMPLED_STATISTICS_RESET
static final String SAMPLED_STATISTICS_RESET

See Also:
Constant Field Values

Method Detail

getSchedulerName
String getSchedulerName()

getSchedulerInstanceId
String getSchedulerInstanceId()
isStandbyMode

boolean isStandbyMode()

isShutdown

boolean isShutdown()

getVersion

String getVersion()

getJobStoreClassName

String getJobStoreClassName()

getThreadPoolClassName

String getThreadPoolClassName()

getThreadPoolSize

int getThreadPoolSize()

getJobsScheduledMostRecentSample

long getJobsScheduledMostRecentSample()

getJobsExecutedMostRecentSample

long getJobsExecutedMostRecentSample()
getJobsCompletedMostRecentSample

long getJobsCompletedMostRecentSample()

getPerformanceMetrics

Map<String, Long> getPerformanceMetrics()

getCurrentlyExecutingJobs

TabularData getCurrentlyExecutingJobs() throws Exception

Returns:
TabularData of CompositeData:JobExecutionContext

Throws:
Exception

getAllJobDetails

TabularData getAllJobDetails() throws Exception

Returns:
TabularData of CompositeData:JobDetail

Throws:
Exception

See Also:
JobDetailSupport

getAllTriggers

List<CompositeData> getAllTriggers() throws Exception
Returns:
List of CompositeData:[CronTrigger|SimpleTrigger]

Throws:
Exception

See Also:
TriggerSupport

---

**getJobGroupNames**

```
List<String> getJobGroupNames()
```

Throws:
Exception

---

**getJobNames**

```
List<String> getJobNames(String groupName)
```

Throws:
Exception

---

**getJobDetail**

```
CompositeData getJobDetail(String jobName,
String jobGroupName)
```

Returns:
CompositeData:JobDetail

Throws:
Exception

See Also:
JobDetailSupport

---

**isStarted**
boolean isStarted()

start

void start() throws Exception

   Throws: Exception

shutdown

void shutdown()

standby

void standby()

clear

void clear() throws Exception

   Throws: Exception

scheduleJob

Date scheduleJob(String jobName, 
                String jobGroup, 
                String triggerName, 
                String triggerGroup) 
throws Exception

   Schedule an existing job with an existing trigger.
Parameters:
jobName -
jobGroup -
triggerName -
triggerGroup -

Returns:
date of nextFireTime

Throws:
Exception

scheduleBasicJob

void scheduleBasicJob(Map<String, Object> jobDetailInfo,
                        Map<String, Object> triggerInfo)
                        throws Exception

Schedules a job using the given Cron/Simple triggerInfo. The triggerInfo and jobDetailInfo must contain well-known attribute values. TriggerInfo attributes: name, group, description, calendarName, priority, CronExpression | (startTime, endTime, repeatCount, repeatInterval) JobDetailInfo attributes: name, group, description, jobClass, jobDataMap, durability, shouldRecover

Throws:
Exception

scheduleJob

void scheduleJob(Map<String, Object> abstractJobInfo,
                  Map<String, Object> abstractTriggerInfo)
                  throws Exception

Schedules an arbitrary job described by abstractJobInfo using a trigger specified by abstractTriggerInfo. AbstractTriggerInfo and AbstractJobInfo must contain the following String attributes. AbstractTriggerInfo: triggerClass, the fully-qualified class name of a concrete Trigger type AbstractJobInfo: jobDetailClass, the fully-qualified class name of a concrete JobDetail type If the Trigger and JobDetail can be successfully
instantiated, the remaining attributes will be reflectively applied to those instances. The remaining attributes are limited to the types: Integer, Double, Float, String, Boolean, Date, Character, Map. Maps are further limited to containing values from the same set of types, less Map itself.

**Throws:**

Exception

---

**scheduleJob**

```java
void scheduleJob(String jobName,
                 String jobGroup,
                 Map<String, Object> abstractTriggerInfo)
throws Exception
```

Schedules the specified job using a trigger described by abstractTriggerInfo, which must contain the fully-qualified trigger class name under the key "triggerClass." That trigger type must contain a no-arg constructor and have public access. Other attributes are applied reflectively and are limited to the types: Integer, Double, Float, String, Boolean, Date, Character, Map. Maps are limited to containing values from the same set of types, less Map itself.

**Parameters:**

- jobName
- jobGroup
- abstractTriggerInfo

**Throws:**

Exception

---

**unscheduleJob**

```java
boolean unscheduleJob(String triggerName,
                      String triggerGroup)
throws Exception
```

**Throws:**

Exception
interruptJob

boolean interruptJob(String jobName,
                      String jobGroupName)
throws Exception

Throws:
   Exception

triggerJob

void triggerJob(String jobName,
                String jobGroupName,
                Map<String, String> jobDataMap)
throws Exception

Throws:
   Exception

deleteJob

boolean deleteJob(String jobName,
                   String jobGroupName)
throws Exception

Throws:
   Exception

addJob

void addJob(CompositeData jobDetail,
            boolean replace)
throws Exception

Throws:
   Exception
addJob

```java
void addJob(Map<String, Object> abstractJobInfo,
            boolean replace)
            throws Exception
```

Adds a durable job described by abstractJobInfo, which must contain the fully-qualified JobDetail class name under the key "jobDetailClass." That JobDetail type must contain a no-arg constructor and have public access. Other attributes are applied reflectively and are limited to the types: Integer, Double, Float, String, Boolean, Date, Character, Map. Maps are limited to containing values from the same set of types, less Map itself.

**Parameters:**
- `jobDetailMap`
- `replace`

**Throws:**
- `Exception`

---

pauseJobGroup

```java
void pauseJobGroup(String jobGroup)
                throws Exception
```

**Throws:**
- `Exception`

---

pauseJobsStartingWith

```java
void pauseJobsStartingWith(String jobGroupPrefix)
                throws Exception
```

Pause all jobs whose group starts with jobGroupPrefix

**Parameters:**
- `jobGroupPrefix`

**Throws:**
- `Exception`
pauseJobsEndingWith

void pauseJobsEndingWith(String jobGroupSuffix)
throws Exception

Pause all jobs whose group ends with jobGroupSuffix

Parameters:
  jobGroupSuffix -

Throws:
  Exception

pauseJobsContaining

void pauseJobsContaining(String jobGroupToken)
throws Exception

Pause all jobs whose group contains jobGroupToken

Parameters:
  jobGroupToken -

Throws:
  Exception

resumeJobGroup

void resumeJobGroup(String jobGroup)
throws Exception

Throws:
  Exception

resumeJobsStartingWith

void resumeJobsStartingWith(String jobGroupPrefix)
throws Exception
Resume all jobs whose group starts with jobGroupPrefix

**Parameters:**
- jobGroupPrefix

**Throws:**
- Exception

`resumeJobsEndingWith`  

```java
void resumeJobsEndingWith(String jobGroupSuffix)
throws Exception
```

Resume all jobs whose group ends with jobGroupSuffix

**Parameters:**
- jobGroupSuffix

**Throws:**
- Exception

`resumeJobsContaining`  

```java
void resumeJobsContaining(String jobGroupToken)
throws Exception
```

Resume all jobs whose group contains jobGroupToken

**Parameters:**
- jobGroupToken

**Throws:**
- Exception

`pauseJob`  

```java
void pauseJob(String jobName, String groupName)
throws Exception
```

**Throws:**
resumeJob

```java
void resumeJob(String jobName, String jobGroupName) throws Exception
```

Throws:
```java
Exception
```

getTriggerGroupNames

```java
List<String> getTriggerGroupNames() throws Exception
```

Throws:
```java
Exception
```

getTriggerNames

```java
List<String> getTriggerNames(String triggerGroupName) throws Exception
```

Throws:
```java
Exception
```

getTrigger

```java
CompositeData getTrigger(String triggerName, String triggerGroupName) throws Exception
```

Throws:
```java
Exception
```
getTriggerState

```java
String getTriggerState(String triggerName,
                        String triggerGroupName)
throws Exception
```

Throws:

```
Exception
```

getTriggersOfJob

```java
List<CompositeData> getTriggersOfJob(String jobName,
                                      String jobGroupName)
throws Exception
```

Returns:

List of CompositeData:[CronTrigger|SimpleTrigger] for the specified job.

Throws:

```
Exception
```

See Also:

```
TriggerSupport
```

getPausedTriggerGroups

```java
Set<String> getPausedTriggerGroups()
throws Exception
```

Throws:

```
Exception
```

pauseAllTriggers

```java
void pauseAllTriggers()
throws Exception
```

Throws:

```
Exception
```
resumeAllTriggers

```java
void resumeAllTriggers() throws Exception
```

**Throws:**

`Exception`

pauseTriggerGroup

```java
void pauseTriggerGroup(String triggerGroup) throws Exception
```

**Throws:**

`Exception`

pauseTriggersStartingWith

```java
void pauseTriggersStartingWith(String triggerGroupPrefix) throws Exception
```

Pause all triggers whose group starts with `triggerGroupPrefix`

**Parameters:**

`triggerGroupPrefix`

**Throws:**

`Exception`

pauseTriggersEndingWith

```java
void pauseTriggersEndingWith(String suffix) throws Exception
```

Pause all triggers whose group ends with `triggerGroupSuffix`

**Parameters:**

`suffix`
pauseTriggersContaining

```java
void pauseTriggersContaining(String triggerGroupToken)
throws Exception
```

Pause all triggers whose group contains triggerGroupToken

**Parameters:**

- `triggerGroupToken`

**Throws:**

`Exception`

-------------------

resumeTriggerGroup

```java
void resumeTriggerGroup(String triggerGroup)
throws Exception
```

**Throws:**

`Exception`

-------------------

resumeTriggersStartingWith

```java
void resumeTriggersStartingWith(String triggerGroupPrefix)
throws Exception
```

Resume all triggers whose group starts with triggerGroupPrefix

**Parameters:**

- `triggerGroupPrefix`

**Throws:**

`Exception`

-------------------

resumeTriggersEndingWith
void `resumeTriggersEndingWith(String triggerGroupSuffix)` throws `Exception`

Resume all triggers whose group ends with triggerGroupSuffix

**Parameters:**
- triggerGroupSuffix -

**Throws:**
- `Exception`

---

void `resumeTriggersContaining(String triggerGroupToken)` throws `Exception`

Resume all triggers whose group contains triggerGroupToken

**Parameters:**
- triggerGroupToken -

**Throws:**
- `Exception`

---

void `pauseTrigger(String triggerName, String triggerGroupName)` throws `Exception`

**Throws:**
- `Exception`

---

void `resumeTrigger(String triggerName, String triggerGroupName)` throws `Exception`

**Throws:**
Exception

getCalendarNames

List<String> getCalendarNames() throws Exception

Throws: Exception

deleteCalendar

void deleteCalendar(String name) throws Exception

Throws: Exception

setSampledStatisticsEnabled

void setSampledStatisticsEnabled(boolean enabled)

isEnabled

boolean isEnabled()
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAME</td>
</tr>
</tbody>
</table>
public class SimpleTriggerSupport
extends Object

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleTriggerSupport()</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String[] getItemDescriptions()</td>
</tr>
<tr>
<td>static String[] getItemNames()</td>
</tr>
<tr>
<td>static OpenType[] getItemTypes()</td>
</tr>
<tr>
<td>static org.quartz.spi.OperableTrigger newTrigger(CompositeData cData)</td>
</tr>
<tr>
<td>static org.quartz.spi.OperableTrigger newTrigger(Map&lt;String, Object&gt; attrMap)</td>
</tr>
<tr>
<td>static CompositeData toCompositeData(SimpleTrigger trigger)</td>
</tr>
<tr>
<td>static TabularData toTabularData(List&lt;? extends SimpleTrigger&gt; triggers)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Constructor Detail

SimpleTriggerSupport

public SimpleTriggerSupport()

Method Detail

getItemNames

public static String[] getItemNames()

getItemDescriptions

public static String[] getItemDescriptions()

g.getItemTypes

public static OpenType[] getItemTypes()

toCompositeData

public static CompositeData toCompositeData(SimpleTrigger trigger)

toTabularData

public static TabularData toTabularData(List<? extends SimpleTrigger> triggers)

newTrigger
public static org.quartz.spi.OperableTrigger newTrigger(CompositeData) throws ParseException

Throws:
   ParseException

newTrigger

public static org.quartz.spi.OperableTrigger newTrigger(Map<String, Object>) throws ParseException

Throws:
   ParseException

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public class TriggerSupport
extends Object

Constructor Summary

| TriggerSupport() |

Method Summary

| String[] getIndexNames() |
| static String[] getItemDescriptions() |
| static String[] getItemNames() |
| static OpenType[] getItemTypes() |
| static void initializeTrigger(org.quartz.spi.MutableCompositeData cData) |
| static void initializeTrigger(org.quartz.spi.MutableMap<String, Object> attrMap) |
| static org.quartz.spi.OperableTrigger newTrigger(CompositeData cData) |
| static org.quartz.spi.OperableTrigger newTrigger(Map<String, Object> attrMap) |
| static CompositeData toCompositeData(Trigger trigger) |
static List<CompositeData> toCompositeList(List<? extends Trigger> triggers)

static TabularData toTabularData(List<? extends Trigger> triggers)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

TriggerSupport
public TriggerSupport()

Method Detail

getItemNames
public static String[] getItemNames()

getItemDescriptions
public static String[] getItemDescriptions()

getItemTypes
public static OpenType[] getItemTypes()

getIndexNames
public String[] getIndexNames()

---

toCompositeData

public static CompositeData toCompositeData(Trigger trigger)

---

toTabularData

public static TabularData toTabularData(List<? extends Trigger> triggers)

---

toCompositeList

public static List<CompositeData> toCompositeList(List<? extends Trigger> triggers)

---

initializeTrigger

public static void initializeTrigger(org.quartz.spi.MutableTrigger trigger, CompositeData cData)

---

initializeTrigger

public static void initializeTrigger(org.quartz.spi.MutableTrigger trigger, Map<String, Object> attrMap)

---

newTrigger

public static org.quartz.spi.OperableTrigger newTrigger(CompositeData cData) throws ParseException

Throws:

ParseException
newTrigger

public static org.quartz.spi.OperableTrigger newTrigger(Map<String, Object> params) throws ParseException

Throws:
ParseException
Uses of Class
org.quartz.core.jmx.CronTriggerSupport

No usage of org.quartz.core.jmx.CronTriggerSupport

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.core.jmx.JobDataMapSupport

No usage of org.quartz.core.jmx.JobDataMapSupport

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.core.jmx.JobDetailSupport

No usage of org.quartz.core.jmx.JobDetailSupport

Copyright 2001-2011, Terracotta, Inc.
No usage of org.quartz.core.jmx.JobExecutionContextSupport

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.core.jmx.QuartzSchedulerMBean

<table>
<thead>
<tr>
<th>Packages that use</th>
<th>QuartzSchedulerMBean</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
</tbody>
</table>

Uses of QuartzSchedulerMBean in org.quartz.core

<table>
<thead>
<tr>
<th>Classes in org.quartz.core that implement QuartzSchedulerMBean</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
</tr>
</tbody>
</table>

Overview Package Class Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
Uses of Class org.quartz.core.jmx.SimpleTriggerSupport

No usage of org.quartz.core.jmx.SimpleTriggerSupport

Copyright 2001-2011, Terracotta, Inc.
Uses of Class

org.quartz.core.jmx.TriggerSupport

No usage of org.quartz.core.jmx.TriggerSupport

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS | NEXT CLASS |
SUMMARY: NESTED | FIELD | CONSTR | METHOD | CONSTR | METHOD | NO FRAMES |
DETAIL: FIELD | CONSTR | METHOD | FRAMES |
Class
JBoss4RMIRemoteMBeanScheduler

All Implemented Interfaces:
Scheduler

public class JBoss4RMIRemoteMBeanScheduler
extends RemoteMBeanScheduler

An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JBoss's JMX RMIAdaptor.

Set the providerURL property to your MBean server URL. This defaults to: jnp://localhost:1099

See Also:
Scheduler, QuartzScheduler, org.quartz.core.SchedulingContext

Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from interface org.quartz.Scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_FAIL_OVER_GROUP, DEFAULT_GROUP, DEFAULT_RECOVERY_GROUP, FAILED_JOBORIGINAL_TRIGGER_FIREFTIME_IN_MILLISECONDS, FAILED_JOBORIGINAL_TRIGGER_GROUP, FAILED_JOBORIGINAL_TRIGGER_NAME</td>
</tr>
</tbody>
</table>

Constructor Summary

| JBoss4RMIRemoteMBeanScheduler() |
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Object</td>
<td><code>getAttribute(String attribute)</code></td>
<td>Get the given attribute of the remote Scheduler MBean.</td>
</tr>
<tr>
<td>protected AttributeList</td>
<td><code>getAttributes(String[] attributes)</code></td>
<td>Get the given attributes of the remote Scheduler MBean.</td>
</tr>
<tr>
<td>protected Properties</td>
<td><code>getContextProperties()</code></td>
<td>Get the properties to use when creating a JNDI InitialContext.</td>
</tr>
<tr>
<td>void</td>
<td><code>initialize()</code></td>
<td>Initialize this remote MBean scheduler, getting the JBoss RMIAdaptor for communication.</td>
</tr>
<tr>
<td>protected Object</td>
<td><code>invoke(String operationName, Object[] params, String[] signature)</code></td>
<td>Invoke the given operation on the remote Scheduler MBean.</td>
</tr>
<tr>
<td>void</td>
<td><code>setProviderURL(String providerURL)</code></td>
<td>Set the remote MBean server URL.</td>
</tr>
</tbody>
</table>

### Methods inherited from class org.quartz.impl.RemoteMBeanScheduler

- `addCalendar`, `addJob`, `checkExists`, `checkExists`, `clear`, `deleteCalendar`, `deleteJob`, `deleteJobs`, `getCalendar`, `getCalendarNames`, `getContext`, `getCurrentlyExecutingJobs`, `getJobDetail`, `getJobGroupNames`, `getJobKeys`, `getListenerManager`, `getMetaData`, `getPausedTriggerGroups`, `getSchedulerInstanceId`, `getSchedulerName`, `getSchedulerObjectName`, `getTrigger`, `getTriggerGroupNames`, `getTriggerKeys`, `getTriggersOfJob`, `getTriggerState`, `interrupt`, `isInStandbyMode`, `isShutdown`, `isStarted`, `pauseAll`, `pauseJob`, `pauseJobs`, `pauseTrigger`, `pauseTriggers`, `rescheduleJob`, `resumeAll`, `resumeJob`, `resumeJobs`, `resumeTrigger`, `resumeTriggers`, `scheduleJob`, `scheduleJobs`, `setJobFactory`, `setSchedulerObjectName`, `setSchedulerObjectName`, `shutdown`, `shutdown`, `standby`, `start`, `startDelayed`, `toBoolean`, `triggerJob`, `triggerJob`, `unscheduleJob`, `unscheduleJobs` |

### Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`,
Constructor Detail

`JBoss4RMIRemoteMBeanScheduler`  

```java
public JBoss4RMIRemoteMBeanScheduler()
  throws SchedulerException
```

Throws:  
`SchedulerException`

Method Detail

`setProviderURL`  

```java
public void setProviderURL(String providerURL)
```

Set the remote MBean server URL. Defaults to: `jnp://localhost:1099`

`initialize`  

```java
public void initialize()
  throws SchedulerException
```

Initialize this remote MBean scheduler, getting the JBoss RMIAdaptor for communication.

Specified by:  
`initialize` in class `RemoteMBeanScheduler`

Throws:  
`SchedulerException`

`getContextProperties`
protected Properties getContextProperties()

Get the properties to use when creating a JNDI InitialContext.
This method is broken out so it can be extended to pass credentials or other properties not currently supported.

**getAttribute**

protected Object getAttribute(String attribute)
throws SchedulerException

Description copied from class: RemoteMBeanScheduler
Get the given attribute of the remote Scheduler MBean.

Specified by:
getAttribute in class RemoteMBeanScheduler

Throws:
SchedulerException

**getAttributes**

protected AttributeList getAttributes(String[] attributes)
throws SchedulerException

Description copied from class: RemoteMBeanScheduler
Get the given attributes of the remote Scheduler MBean.

Specified by:
getAttributes in class RemoteMBeanScheduler

Throws:
SchedulerException

**invoke**

protected Object invoke(String operationName,
Object[] params,
String[] signature)
throws SchedulerException

**Description copied from class:** RemoteMBeanScheduler

Invoke the given operation on the remote Scheduler MBean.

**Specified by:**

*invoke* in class RemoteMBeanScheduler

**Throws:**

SchedulerException
org.quartz.ee.jmx.jboss

Interfaces  QuartzServiceMBean

Classes  JBoss4RMIRemoteMBeanScheduler  QuartzService
## Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzServiceMBean</td>
<td>Interface exposed via JMX for MBean for configuring, starting, and binding to JNDI a Quartz Scheduler instance.</td>
</tr>
</tbody>
</table>

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBoss4RMIRemoteMBeanScheduler</td>
<td>An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JBoss's JMX RMIAAdaptor.</td>
</tr>
<tr>
<td>QuartzService</td>
<td>JBoss specific MBean implementation for configuring, starting, and binding to JNDI a Quartz Scheduler instance.</td>
</tr>
</tbody>
</table>
Hierarchy For Package org.quartz.ee.jmx.jboss

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.jboss.system.ServiceMBeanSupport (implements javax.management.MBeanRegistration, org.jboss.system.ServiceMBean)
    - org.quartz.ee.jmx.jboss.**QuartzService** (implements org.quartz.ee.jmx.jboss.**QuartzServiceMBean**)
  - org.quartz.impl.**RemoteMBeanScheduler** (implements org.quartz.**Scheduler**)
    - org.quartz.ee.jmx.jboss.**JBoss4RMIRemoteMBeanScheduler**
Interface Hierarchy

- org.jboss.system.Service
  - org.jboss.system.ServiceMBean
  - org.quartz.ee.jmx.jboss.QuartzServiceMBean

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.ee.jmx.jboss

Packages that use org.quartz.ee.jmx.jboss

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.ee.jmx.jboss</td>
</tr>
</tbody>
</table>

Classes in org.quartz.ee.jmx.jboss used by org.quartz.ee.jmx.jboss

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzServiceMBean</td>
</tr>
</tbody>
</table>

QuartzServiceMBean
Interface exposed via JMX for MBean for configuring, starting, and binding to JNDI a Quartz Scheduler instance.
Class QuartzService

java.lang.Object
    org.jboss.mx.util.JBossNotificationBroadcasterSupport
    org.jboss.system.ServiceMBeanSupport
    org.quartz.ee.jmx.jboss.QuartzService

All Implemented Interfaces:
    MBeanRegistration, NotificationBroadcaster, NotificationEmitter,
    org.jboss.system.Service, org.jboss.system.ServiceMBean,
    QuartzServiceMBean

public class QuartzService

does not implement
extends org.jboss.system.ServiceMBeanSupport
implements QuartzServiceMBean

JBoss specific MBean implementation for configuring, starting, and binding to
JNDI a Quartz Scheduler instance.

Sample MBean deployment descriptor: quartz-service.xml

Note: The Scheduler instance bound to JNDI is not Serializable, so you will get
a null reference back if you try to retrieve it from outside the JBoss server in
which it was bound. If you have a need for remote access to a Scheduler instance
you may want to consider using Quartz's RMI support instead.

Author:
    Andrew Collins

See Also:
    QuartzServiceMBean

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Fields inherited from class org.jboss.system.ServiceMBeanSupport</th>
</tr>
</thead>
<tbody>
<tr>
<td>log, server, SERVICE_CONTROLLER_SIG, serviceName</td>
</tr>
</tbody>
</table>
### Fields inherited from interface org.jboss.system.ServiceMBean

<table>
<thead>
<tr>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE_EVENT, CREATED, DESTROY_EVENT, DESTROYED, FAILED, REGISTERED, START_EVENT, STARTED, STARTING, states, STOP_EVENT, STOPPED, STOPPING, UNREGISTERED</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzService()</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void createService()</td>
<td></td>
</tr>
<tr>
<td>void destroyService()</td>
<td></td>
</tr>
<tr>
<td>String getJndiName()</td>
<td></td>
</tr>
<tr>
<td>String getName()</td>
<td></td>
</tr>
<tr>
<td>String getProperties()</td>
<td></td>
</tr>
<tr>
<td>String getPropertiesFile()</td>
<td></td>
</tr>
<tr>
<td>boolean getStartScheduler()</td>
<td></td>
</tr>
<tr>
<td>void setJndiName(String jndiName)</td>
<td></td>
</tr>
<tr>
<td>void setProperties(String properties)</td>
<td></td>
</tr>
<tr>
<td>void setPropertiesFile(String propertiesFile)</td>
<td></td>
</tr>
<tr>
<td>void setStartScheduler(boolean startScheduler)</td>
<td></td>
</tr>
<tr>
<td>void startService()</td>
<td></td>
</tr>
</tbody>
</table>
```java
void stopService()
```

### Methods inherited from class `org.jboss.system.ServiceMBeanSupport`

create, destroy, getLog, getNextNotificationSequenceNumber, getObjectName, getServer, getServiceName, getState, getStateString, jbossInternalCreate, jbossInternalDescription, jbossInternalDestroy, jbossInternalLifecycle, jbossInternalStart, jbossInternalStop, postDeregister, postRegister, preDeregister, preRegister, start, stop

### Methods inherited from class `org.jboss.mx.util.JBossNotificationBroadcasterSupport`

addNotificationListener, getNotificationInfo, handleNotification, removeNotificationListener, removeNotificationListener, sendNotification

### Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Methods inherited from interface `org.jboss.system.ServiceMBean`

getState, getStateString, jbossInternalLifecycle

### Methods inherited from interface `org.jboss.system.Service`

create, destroy, start, stop

### Constructor Detail

**QuartzService**

public QuartzService()
setJndiName

public void setJndiName(String jndiName) throws Exception

Specified by:
   setJndiName in interface QuartzServiceMBean

Throws:
   Exception

getJndiName

public String getJndiName()

Specified by:
   getJndiName in interface QuartzServiceMBean

getName

public String getName()

Specified by:
   getName in interface org.jboss.system.ServiceMBean

Overrides:
   getName in class org.jboss.system.ServiceMBeanSupport

setProperties

public void setProperties(String properties)

Specified by:
   setProperties in interface QuartzServiceMBean

getProperties
public String getProperties()

setPropertiesFile

public void setPropertiesFile(String propertiesFile)

   Specified by:
       setPropertiesFile in interface QuartzServiceMBean

getPropertiesFile

public String getPropertiesFile()

setStartScheduler

public void setStartScheduler(boolean startScheduler)

   Specified by:
       setStartScheduler in interface QuartzServiceMBean

getStartScheduler

public boolean getStartScheduler()

createService

public void createService()
   throws Exception

   Overrides:
       createService in class org.jboss.system.ServiceMBeanSupport

   Throws:
       Exception
destroyService

public void destroyService()
    throws Exception

   Overrides:
       destroyService in class org.jboss.system.ServiceMBeanSupport

   Throws:
       Exception

startService

public void startService()
    throws Exception

   Overrides:
       startService in class org.jboss.system.ServiceMBeanSupport

   Throws:
       Exception

stopService

public void stopService()
    throws Exception

   Overrides:
       stopService in class org.jboss.system.ServiceMBeanSupport

   Throws:
       Exception
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
org.quartz.ee.jmx.jboss Interface QuartzServiceMBean

All Superinterfaces: org.jboss.system.Service, org.jboss.system.ServiceMBean

All Known Implementing Classes: QuartzService

public interface QuartzServiceMBean
extends org.jboss.system.ServiceMBean

Interface exposed via JMX for MBean for configuring, starting, and binding to JNDI a Quartz Scheduler instance.

Sample MBean deployment descriptor: quartz-service.xml

Note: The Scheduler instance bound to JNDI is not Serializable, so you will get a null reference back if you try to retrieve it from outside the JBoss server in which it was bound. If you have a need for remote access to a Scheduler instance you may want to consider using Quartz's RMI support instead.

Author: Andrew Collins

See Also: QuartzService

Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from interface org.jboss.system.ServiceMBean</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE_EVENT, CREATED, DESTROY_EVENT, DESTROYED, FAILED,</td>
</tr>
<tr>
<td>REGISTERED, START_EVENT, STARTED, STARTING, states, STOP_EVENT,</td>
</tr>
<tr>
<td>STOPPED, STOPPING, UNREGISTERED</td>
</tr>
</tbody>
</table>

Method Summary
### String getJndiName()

```java
void setJndiName(String jndiName)
```

```java
void setProperties(String properties)
```

```java
void setPropertiesFile(String propertiesFile)
```

```java
void setStartScheduler(boolean startScheduler)
```

<table>
<thead>
<tr>
<th>Methods inherited from interface org.jboss.system.ServiceMBean</th>
</tr>
</thead>
<tbody>
<tr>
<td>getName, getState, getStateString, jbossInternalLifecycle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from interface org.jboss.system.Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>create, destroy, start, stop</td>
</tr>
</tbody>
</table>

### Method Detail

**setJndiName**

```java
void setJndiName(String jndiName) throws Exception
```

**Throws:**

```java
Exception
```

**getJndiName**

```java
String getJndiName()
```

**setProperties**
void setProperties(String properties)

setPropertiesFile

void setPropertiesFile(String propertiesFile)

setStartScheduler

void setStartScheduler(boolean startScheduler)
Uses of Class
org.quartz.ee.jmx.jboss.JBoss4RMIRemoteMBeanScheduler

No usage of org.quartz.ee.jmx.jboss.JBoss4RMIRemoteMBeanScheduler

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.ee.jmx.jboss.QuartzService

No usage of org.quartz.ee.jmx.jboss.QuartzService

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.ee.jmx.jboss.QuartzServiceMBean

<table>
<thead>
<tr>
<th>Packages that use QuartzServiceMBean</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.ee.jmx.jboss</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of QuartzServiceMBean in org.quartz.ee.jmx.jboss</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Classes in org.quartz.ee.jmx.jboss that implement QuartzServiceMBean</th>
</tr>
</thead>
<tbody>
<tr>
<td>class QuartzService</td>
</tr>
<tr>
<td>JBoss specific MBean implementation for configuring, starting, and binding to JNDI a Quartz Scheduler instance.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Class

`JTAAnnotationAwareJobRunShellFactory`

```java
org.quartz.ee.jta

java.lang.Object

org.quartz.ee.jta.JTAAnnotationAwareJobRunShellFactory

All Implemented Interfaces:

JobRunShellFactory
```

```java
public class JTAAnnotationAwareJobRunShellFactory

develops Object

implements JobRunShellFactory
```

Responsible for creating the instances of a `JobRunShell` to be used within the `QuartzScheduler` instance. It will create a standard `JobRunShell` unless the job class has the `ExecuteInJTATransaction` annotation in which case it will create a `JTAJobRunShell`.

This implementation does not re-use any objects, it simply makes a new `JTAJobRunShell` each time `borrowJobRunShell()` is called.

**Author:**

James House

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JTAAnnotationAwareJobRunShellFactory()</code></td>
</tr>
</tbody>
</table>

---

**Method Summary**

```java
void createJobRunShell(org.quartz.spi.TriggerFiredBundle bundle)

Called by the `QuartzSchedulerThread` to obtain instances of `JobRunShell`.

void initialize(Scheduler scheduler)
```
Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JTAAnnotationAwareJobRunShellFactory

public JTAAnnotationAwareJobRunShellFactory()

Method Detail

initialize

public void initialize(Scheduler scheduler)
    throws SchedulerConfigException

    Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.

    Specified by:
        initialize in interface JobRunShellFactory

    Throws:
        SchedulerConfigException

createJobRunShell
public JobRunShell createJobRunShell(org.quartz.spi.TriggerFiredBundle bundle) throws SchedulerException

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

Specified by: createJobRunShell in interface JobRunShellFactory

Throws: SchedulerException

Copyright 2001-2011, Terracotta, Inc.
Class JTAJobRunShell

java.lang.Object  
  └ org.quartz.listeners.SchedulerListenerSupport  
      └ org.quartz.core.JobRunShell  
          └ org.quartz.ee.jta.JTAJobRunShell

All Implemented Interfaces:
  Runnable, SchedulerListener

public class JTAJobRunShell
extends JobRunShell

An extension of JobRunShell that begins an XA transaction before executing the Job, and commits (or rolls-back) the transaction after execution completes.

Author:
  James House
See Also:
  JobRunShell

Field Summary

Fields inherited from class org.quartz.core.JobRunShell
firedTriggerBundle, jec, qs, scheduler, shutdownRequested

Constructor Summary

JTAJobRunShell(Scheduler scheduler,
  org.quartz.spi.TriggerFiredBundle bndle)
  Create a JTAJobRunShell instance with the given settings.

Method Summary

protected void begin()
protected void complete(boolean successfulExecution)

void passivate()  

Override passivate() to ensure we always cleanup the UserTransaction.

Methods inherited from class org.quartz.core.JobRunShell

completeTriggerRetryLoop, getLog, initialize, requestShutdown, run, schedulerShuttingdown, vetoedJobRetryLoop

Methods inherited from class org.quartz.listeners.SchedulerListenerSupport

jobAdded, jobDeleted, jobPaused, jobResumed, jobScheduled, jobsPaused, jobsResumed, jobUnscheduled, schedulerError, schedulerInStandbyMode, schedulerShutdown, schedulerStarted, schedulingDataCleared, triggerFinalized, triggerPaused, triggerResumed, triggersPaused, triggersResumed

Methods inherited from class java.lang.Object

close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JTAJobRunShell

public JTAJobRunShell(Scheduler scheduler, org.quartz.spi.TriggerFiredBundle bndle)

Create a JTAJobRunShell instance with the given settings.

Method Detail

begin
protected void begin()
    throws SchedulerException

    Overrides:
      begin in class JobRunShell

    Throws:
      SchedulerException

complete

protected void complete(boolean successfulExecution)
    throws SchedulerException

    Overrides:
      complete in class JobRunShell

    Throws:
      SchedulerException

passivate

public void passivate()

    Override passivate() to ensure we always cleanup the UserTransaction.

    Overrides:
      passivate in class JobRunShell
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SUMMARY:
NESTED | FIELD | CONSTR | METHOD

DETAIL:
FIELD | CONSTR | METHOD
org.quartz.ee.jta  Class JTAJobRunShellFactory

java.lang.Object
   └ org.quartz.ee.jta.JTAJobRunShellFactory

All Implemented Interfaces:
   JobRunShellFactory

public class JTAJobRunShellFactory
    extends Object
    implements JobRunShellFactory

Responsible for creating the instances of JTAJobRunShell to be used within the QuartzScheduler instance.

This implementation does not re-use any objects, it simply makes a new JTAJobRunShell each time borrowJobRunShell() is called.

Author:
   James House

---

**Constructor Summary**

| JTAJobRunShellFactory() |

**Method Summary**

<table>
<thead>
<tr>
<th>JobRunShell</th>
<th>createJobRunShell(TriggerFiredBundle bundle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>initialize(Scheduler scheduler)</td>
</tr>
</tbody>
</table>

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with
Methods inherited from class java.lang.**Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

**JTAJobRunShellFactory**

public **JTAJobRunShellFactory**()

Method Detail

**initialize**

public void **initialize**(Scheduler scheduler)

throws SchedulerConfigException

Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it, and a handle to the SchedulingContext that the shell will use in its own operations with the JobStore.

**Specified by:**

*initialize* in interface **JobRunShellFactory**

**Throws:**

SchedulerConfigException

**createJobRunShell**

public **JobRunShell** **createJobRunShell**(org.quartz.spi.TriggerFiredBundle)

throws SchedulerException

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.
Specified by:
createJobRunShell in interface JobRunShellFactory

Throws:
SchedulerException

Copyright 2001-2011, Terracotta, Inc.
## Package org.quartz.ee.jta

### Class Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTAAnnotationAwareJobRunShellFactory</td>
<td>Responsible for creating the instances of a JobRunShell11 to be used within the QuartzScheduler instance.</td>
</tr>
<tr>
<td>JTAJobRunShell</td>
<td>An extension of JobRunShell11 that begins an XA transaction before executing the Job, and commits (or rolls-back) the transaction after execution completes.</td>
</tr>
<tr>
<td>JTAJobRunShellFactory</td>
<td>Responsible for creating the instances of JTAJobRunShell11 to be used within the QuartzScheduler instance.</td>
</tr>
<tr>
<td>UserTransactionHelper</td>
<td>A helper for obtaining a handle to a UserTransaction...</td>
</tr>
</tbody>
</table>
Hierarchy For Package org.quartz.ee.jta

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.ee.jta.**JTAAnnotationAwareJobRunShellFactory** (implements org.quartz.core.**JobRunShellFactory**)
  - org.quartz.ee.jta.**JTAJobRunShellFactory** (implements org.quartz.core.**JobRunShellFactory**)
  - org.quartz.listeners.**SchedulerListenerSupport** (implements org.quartz.**SchedulerListener**)
    - org.quartz.core.**JobRunShell** (implements java.lang.**Runnable**)
    - org.quartz.ee.jta.**JTAJobRunShell**
  - org.quartz.ee.jta.**UserTransactionHelper**
No usage of org.quartz.ee.jta
public class UserTransactionHelper
extends Object

A helper for obtaining a handle to a UserTransaction...

To ensure proper cleanup of the InitialContext used to create/lookup the UserTransaction, be sure to always call returnUserTransaction() when you are done with the UserTransaction.

Author:
James House

Field Summary

| static String | DEFAULT_USER_TX_LOCATION |

Method Summary

| static String | getUserTxLocation() |
| static UserTransaction | lookupUserTransaction() |
| Create/Lookup a UserTransaction in the InitialContext the name set in setUserTxLocation(). |
| static void | returnUserTransaction(UserTransaction) |
| Return a UserTransaction that was retrieved via getUserTransaction(). |
| static void | setUserTxLocation(String userTxURL) |
| Set the JNDI URL at which the Application Server's UserTransaction can be found. |
Methods inherited from class java.lang/Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Field Detail

DEFAULT_USER_TX_LOCATION

public static final String DEFAULT_USER_TX_LOCATION

See Also:
Constant Field Values

Method Detail

getUserTxLocation

public static String getUserTxLocation()

setUserTxLocation

public static void setUserTxLocation(String userTxURL)

    Set the JNDI URL at which the Application Server's UserTransaction can be found. If not set, the default value is "java:comp/UserTransaction" - which works for nearly all application servers.

lookupUserTransaction

public static UserTransaction lookupUserTransaction() throws SchedulerException

    Create/Lookup a UserTransaction in the InitialContext via the name set in
Throws:

SchedulerException

returnUserTransaction

public static void returnUserTransaction(UserTransaction userTransaction)

Return a UserTransaction that was retrieved via getUserTransaction(). This will make sure that the InitialContext used to lookup/create the UserTransaction is properly cleaned up.
Uses of Class
org.quartz.ee.jta.JTAAnnotationAwareJobRunShellFactory

No usage of org.quartz.ee.jta.JTAAnnotationAwareJobRunShellFactory

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.ee.jta.JTAJobRunShell

No usage of org.quartz.ee.jta.JTAJobRunShell

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.ee.jta.JTAJobRunShellFactory

No usage of org.quartz.ee.jta.JTAJobRunShellFactory

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.ee.jta.UserTransactionHelper

No usage of org.quartz.ee.jta.UserTransactionHelper

Copyright 2001-2011, Terracotta, Inc.
org.quartz.ee.servlet Classes  QuartzInitializerListener
QuartzInitializerServlet
Package org.quartz.ee.servlet

Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuartzInitializerListener</td>
<td>A ServletContextListener that can be used to initialize Quartz.</td>
</tr>
<tr>
<td>QuartzInitializerServlet</td>
<td>A Servlet that can be used to initialize Quartz, if configured as a load-on-startup servlet in a web application.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.ee.servlet

Package Hierarchies:
  All Packages
Class Hierarchy

- java.lang.**Object**
    - javax.servlet.http.HttpServlet (implements java.io.**Serializable**)
      - org.quartz.ee.servlet.**QuartzInitializerServlet**
    - org.quartz.ee.servlet.**QuartzInitializerListener** (implements javax.servlet.ServletContextListener)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.ee.servlet

No usage of org.quartz.ee.servlet

Copyright 2001-2011, Terracotta, Inc.
public class QuartzInitializerListener

extends Object

implements javax.servlet.ServletContextListener

A ServletContextListner that can be used to initialize Quartz.

You'll want to add something like this to your WEB-INF/web.xml file:

```xml
<listener>
  <listener-class>
    org.quartz.ee.servlet.QuartzInitializerListener
  </listener-class>
</listener>

<context-param>
  <param-name>quartz:config-file</param-name>
  <param-value>/some/path/my_quartz.properties</param-value>
</context-param>
<context-param>
  <param-name>quartz:shutdown-on-unload</param-name>
  <param-value>true</param-value>
</context-param>
<context-param>
  <param-name>quartz:wait-on-shutdown</param-name>
  <param-value>true</param-value>
</context-param>
<context-param>
  <param-name>quartz:start-on-load</param-name>
  <param-value>true</param-value>
</context-param>

The init parameter 'quartz:config-file' can be used to specify the path (and
filename) of your Quartz properties file. If you leave out this parameter, the default ("quartz.properties") will be used.

The init parameter 'quartz:shutdown-on-unload' can be used to specify whether you want scheduler.shutdown() called when the listener is unloaded (usually when the application server is being shutdown). Possible values are "true" or "false". The default is "true".

The init parameter 'quartz:wait-on-shutdown' has effect when 'quartz:shutdown-on-unload' is specified "true", and indicates whether you want scheduler.shutdown(true) called when the listener is unloaded (usually when the application server is being shutdown). Passing "true" to the shutdown() call causes the scheduler to wait for existing jobs to complete. Possible values are "true" or "false". The default is "false".

The init parameter 'quartz:start-on-load' can be used to specify whether you want the scheduler.start() method called when the listener is first loaded. If set to false, your application will need to call the start() method before the scheduler begins to run and process jobs. Possible values are "true" or "false". The default is "true", which means the scheduler is started.

A StdSchedulerFactory instance is stored into the ServletContext. You can gain access to the factory from a ServletContext instance like this:

```java
StdSchedulerFactory factory = (StdSchedulerFactory) ctx
    .getAttribute(QuartzInitializerListener.QUARTZ_FACTORY_KEY);
```

The init parameter 'quartz:servlet-context-factory-key' can be used to override the name under which the StdSchedulerFactory is stored into the ServletContext, in which case you will want to use this name rather than QuartzInitializerListener.QUARTZ_FACTORY_KEY in the above example.

The init parameter 'quartz:scheduler-context-servlet-context-key' if set, the ServletContext will be stored in the SchedulerContext under the given key name (and will therefore be available to jobs during execution).

The init parameter 'quartz:start-delay-seconds' can be used to specify the amount of time to wait after initializing the scheduler before scheduler.start() is called.

Once you have the factory instance, you can retrieve the Scheduler instance by
calling getScheduler() on the factory.

Author:
James House, Chuck Cavaness, John Petrocik

---

### Field Summary

<table>
<thead>
<tr>
<th>static String</th>
<th>QUARTZ_FACTORY_KEY</th>
</tr>
</thead>
</table>

### Constructor Summary

**QuartzInitializerListener()**

### Method Summary

| void contextDestroyed(javax.servlet.ServletContextEvent sce) |
| void contextInitialized(javax.servlet.ServletContextEvent sce) |

Methods inherited from class java.lang.**Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Field Detail

**QUARTZ_FACTORY_KEY**

public static final String QUARTZ_FACTORY_KEY

See Also:

Constant Field Values
### Constructor Detail

**QuartzInitializerListener**

```java
public QuartzInitializerListener()
```

### Method Detail

**contextInitialized**

```java
public void contextInitialized(javax.servlet.ServletContextEvent sce)
```

**Specified by:**
- `contextInitialized` in interface `javax.servlet.ServletContextListener`

**contextDestroyed**

```java
public void contextDestroyed(javax.servlet.ServletContextEvent sce)
```

**Specified by:**
- `contextDestroyed` in interface `javax.servlet.ServletContextListener`

---

Copyright 2001-2011, Terracotta, Inc.
**Class** QuartzInitializerServlet

<table>
<thead>
<tr>
<th>java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.servlet.GenericServlet</td>
</tr>
<tr>
<td>javax.servlet.http.HttpServlet</td>
</tr>
<tr>
<td>org.quartz.ee.servlet.QuartzInitializerServlet</td>
</tr>
</tbody>
</table>

**All Implemented Interfaces:**

Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

```java
public class QuartzInitializerServlet extends javax.servlet.http.HttpServlet

A Servlet that can be used to initialize Quartz, if configured as a load-on-startup servlet in a web application.

Using this start-up servlet may be preferred to using the QuartzInitializerListener in some situations - namely when you want to initialize more than one scheduler in the same application.

You'll want to add something like this to your WEB-INF/web.xml file:

```xml
<servlet>
  <servlet-name>QuartzInitializer</servlet-name>
  <display-name>Quartz Initializer Servlet</display-name>
  <servlet-class>org.quartz.ee.servlet.QuartzInitializerServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param>
    <param-name>config-file</param-name>
    <param-value>/some/path/my_quartz.properties</param-value>
  </init-param>
</servlet>
```
The init parameter 'config-file' can be used to specify the path (and filename) of your Quartz properties file. If you leave out this parameter, the default ("quartz.properties") will be used.

The init parameter 'shutdown-on-unload' can be used to specify whether you want scheduler.shutdown() called when the servlet is unloaded (usually when the application server is being shutdown). Possible values are "true" or "false". The default is "true".

The init parameter 'wait-on-shutdown' has effect when 'shutdown-on-unload' is specified "true", and indicates whether you want scheduler.shutdown(true) called when the listener is unloaded (usually when the application server is being shutdown). Passing "true" to the shutdown() call causes the scheduler to wait for existing jobs to complete. Possible values are "true" or "false". The default is "false".

The init parameter 'start-scheduler-on-load' can be used to specify whether you want the scheduler.start() method called when the servlet is first loaded. If set to false, your application will need to call the start() method before the scheduler begins to run and process jobs. Possible values are "true" or "false". The default is "true", which means the scheduler is started.

A StdSchedulerFactory instance is stored into the ServletContext. You can gain access to the factory from a ServletContext instance like this:

```java
StdSchedulerFactory factory = (StdSchedulerFactory) ctx.getAttribute(QuartzFactoryServlet.QUARTZ_FACTORY_KEY);
```
The init parameter 'servlet-context-factory-key' can be used to override the name under which the StdSchedulerFactory is stored into the ServletContext, in which case you will want to use this name rather than QuartzFactoryServlet.QUARTZ_FACTORY_KEY in the above example.

The init parameter 'scheduler-context-servlet-context-key' if set, the ServletContext will be stored in the SchedulerContext under the given key name (and will therefore be available to jobs during execution).

The init parameter 'start-delay-seconds' can be used to specify the amount of time to wait after initializing the scheduler before scheduler.start() is called.

Once you have the factory instance, you can retrieve the Scheduler instance by calling getScheduler() on the factory.

Author:
James House, Chuck Cavaness

See Also:
Serialized Form

---

### Field Summary

| static String QUARTZ_FACTORY_KEY |

### Constructor Summary

QuartzInitializerServlet()

### Method Summary

void destroy()


void doPost(javax.servlet.http.HttpServletRequest request,
<table>
<thead>
<tr>
<th>javax.servlet.http.HttpServletResponse</th>
<th>response</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>init</td>
</tr>
</tbody>
</table>

**Methods inherited from class javax.servlet.http.HttpServlet**
- doDelete
- doHead
- doOptions
- doGet
- doTrace
- getLastModified
- service
- service

**Methods inherited from class javax.servlet.GenericServlet**
- getInitParameter
- getInitParameterNames
- getServletConfig
- getServletContext
- getServletInfo
- getServletName
- init
- log
- log

**Methods inherited from class java.lang.Object**
- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

**Field Detail**

**QUARTZ_FACTORY_KEY**

<table>
<thead>
<tr>
<th>public static final String</th>
<th>QUARTZ_FACTORY_KEY</th>
</tr>
</thead>
</table>

**See Also:**
- Constant Field Values

**Constructor Detail**

**QuartzInitializerServlet**

| public | QuartzInitializerServlet() |

**Method Detail**
public void init(javax.servlet.ServletConfig cfg)
    throws javax.servlet.ServletException

Specified by:
    init in interface javax.servlet.Servlet
Overrides:
    init in class javax.servlet.GenericServlet
Throws:
    javax.servlet.ServletException

public void destroy()

Specified by:
    destroy in interface javax.servlet.Servlet
Overrides:
    destroy in class javax.servlet.GenericServlet

    throws javax.servlet.ServletException, IOException,
    IOException

Overrides:
    doPost in class javax.servlet.http.HttpServlet
Throws:
    javax.servlet.ServletException
    IOException

public void doGet(javax.servlet.http.HttpServletRequest request,

Overrides:

doGet in class javax.servlet.http.HttpServlet

Throws:

javax.servlet.ServletException
IOException

IOException
No usage of org.quartz.ee.servlet.QuartzInitializerListener
Uses of Class
org.quartz.ee.servlet.QuartzInitializerServlet

No usage of org.quartz.ee.servlet.QuartzInitializerServlet

Copyright 2001-2011, Terracotta, Inc.
org.quartz.helpers

Classes

VersionPrinter
Package org.quartz.helpers

Contains helper classes to make working with Quartz easier.

See: Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VersionPrinter</strong></td>
</tr>
</tbody>
</table>
Package org.quartz.helpers Description

Contains helper classes to make working with Quartz easier.

See the Quartz project for more information.
Hierarchy For Package org.quartz.helpers

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.helpers.**VersionPrinter**
Uses of Package org.quartz.helpers

No usage of org.quartz.helpers

Copyright 2001-2011, Terracotta, Inc.
public class VersionPrinter

extends Object

Prints the version of Quartz on stdout.

Author:
James House

Method Summary

| static void | main(String[] args) |

Methods inherited from class java.lang.Object

cloned, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

main

public static void main(String[] args)
Uses of Class
org.quartz.helpers.VersionPrinter

No usage of org.quartz.helpers.VersionPrinter

Copyright 2001-2011, Terracotta, Inc.
Class DirectSchedulerFactory

All Implemented Interfaces:

   SchedulerFactory

-----------------------------

public class DirectSchedulerFactory

extends Object

implements SchedulerFactory

A singleton implementation of SchedulerFactory.

Here are some examples of using this class:

To create a scheduler that does not write anything to the database (is not persistent), you can call createVolatileScheduler:

    DirectSchedulerFactory.getInstance().createVolatileScheduler(10);

    // don't forget to start the scheduler:
    DirectSchedulerFactory.getInstance().getScheduler().start();

Several create methods are provided for convenience. All create methods eventually end up calling the create method with all the parameters:

    public void createScheduler(String schedulerName, String schedulerInstanceId, ThreadPool threadPool, JobStore jobStore, String rmiRegistryHost, int rmiRegistryPort)

Here is an example of using this method:

* *

    // create the thread pool SimpleThreadPool threadPool = new SimpleTh

    DirectSchedulerFactory.getInstance().createScheduler("My Quartz Sc

You can also use a JDBCJobStore instead of the RAMJobStore:
DBConnectionManager.getInstance().addConnectionProvider("someDatasource");
JobStoreTX jdbcJobStore = new JobStoreTX(); jdbcJobStore.setDataSource("someDatasource");
jdbcJobStore.setPostgresStyleBlobs(true);
jdbcJobStore.setTablePrefix("QRTZ_");
jdbcJobStore.setInstanceId("My Instance");

Author:
Mohammad Rezaei, James House

See Also:
JobStore, ThreadPool

---

**Field Summary**

| static String | DEFAULT_INSTANCE_ID |
| static String | DEFAULT_SCHEDULER_NAME |

**Constructor Summary**

| protected DirectSchedulerFactory() |
| Constructor |

**Method Summary**

| void createRemoteScheduler(String rmiHost, int rmiPort) |
| Creates a proxy to a remote scheduler. |
| void createRemoteScheduler(String schedulerName, String schedulerInstanceId, String rmiHost, int rmiPort) |
| Same as createRemoteScheduler(String rmiHost, int rmiPort), with the addition of specifying the scheduler name and instance ID. |
| void createRemoteScheduler(String schedulerName, String schedulerInstanceId, String rmiBindName, String rmiHost, int rmiPort) |
| Same as createRemoteScheduler(String rmiHost, int rmiPort), with the addition of specifying the scheduler name, instance ID, and rmi bind name. |
void createScheduler(String schedulerName, String schedulerInstanceId, org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore)

    Same as createScheduler(ThreadPool threadPool, JobStore jobStore), with the addition of specifying scheduler name and instance ID.

void createScheduler(String schedulerName, String schedulerInstanceId, org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore, Map schedulerPluginMap, String rmiRegistryHost, int rmiRegistryPort, long idleWaitTime, long dbFailureRetryInterval, boolean jmxExport, String jmxObjectName)

    Creates a scheduler using the specified thread pool, job store, and plugins, and binds it to RMI.

void createScheduler(String schedulerName, String schedulerInstanceId, org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore, String rmiRegistryHost, int rmiRegistryPort, long idleWaitTime, long dbFailureRetryInterval)

    Creates a scheduler using the specified thread pool and job store and binds it to RMI.

void createScheduler(org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore)

    Creates a scheduler using the specified thread pool and job store.

void createVolatileScheduler(int maxThreads)

    Deprecated. see correctly spelled method.

void createVolatileScheduler(int maxThreads)

    Creates an in memory job store (RAMJobStore)'s thread priority is set to Thread.NORM_PRIORITY

Collection getAllSchedulers()

    Returns a handle to all known Schedulers (made StdSchedulerFactory instance.).

static DirectSchedulerFactory getInstance()

protected org.slf4j.Logger
**getLog()**

**Scheduler getScheduler()**

Returns a handle to the Scheduler produced by factory.

**Scheduler getScheduler(String schedName)**

Returns a handle to the Scheduler with the given name, if it exists.

### Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Field Detail

**DEFAULT_INSTANCE_ID**

public static final String DEFAULT_INSTANCE_ID

See Also:
- Constant Field Values

**DEFAULT_SCHEDULER_NAME**

public static final String DEFAULT_SCHEDULER_NAME

See Also:
- Constant Field Values

### Constructor Detail

**DirectSchedulerFactory**

protected DirectSchedulerFactory()
Constructor

### Method Detail

**getLog**

```java
protected org.slf4j.Logger getLog()
```

**getInstance**

```java
public static DirectSchedulerFactory getInstance()
```

**createVolatileScheduler**

```java
public void createVolatileScheduler(int maxThreads)
```

Throws: `SchedulerException`

Creates an in memory job store (RAMJobStore) The thread priority is set to Thread.NORM_PRIORITY

**Parameters:**

- `maxThreads` - The number of threads in the thread pool

**Throws:**

- `SchedulerException` - if initialization failed.

**createVolatileScheduler**

```java
public void createVolatileScheduler(int maxThreads)
```

**Deprecated.** See correctly spelled method.

**Throws:**

- `SchedulerException`

**See Also:**

- `createVolatileScheduler(int)`
createRemoteScheduler

public void createRemoteScheduler(String rmiHost,
                                  int rmiPort)
          throws SchedulerException

  Creates a proxy to a remote scheduler. This scheduler can be retrieved via
  getScheduler()

  **Parameters:**
  
  rmiHost - The hostname for remote scheduler
  rmiPort - Port for the remote scheduler. The default RMI port is 1099.

  **Throws:**
  
  SchedulerException - if the remote scheduler could not be reached.

createRemoteScheduler

public void createRemoteScheduler(String schedulerName,
                                   String schedulerInstanceId,
                                   String rmiHost,
                                   int rmiPort)
          throws SchedulerException

  Same as createRemoteScheduler(String rmiHost, int rmiPort), with
  the addition of specifying the scheduler name and instance ID. This
  scheduler can only be retrieved via getScheduler(String)

  **Parameters:**
  
  schedulerName - The name for the scheduler.
  schedulerInstanceId - The instance ID for the scheduler.
  rmiHost - The hostname for remote scheduler
  rmiPort - Port for the remote scheduler. The default RMI port is 1099.

  **Throws:**
  
  SchedulerException - if the remote scheduler could not be reached.
public void createRemoteScheduler(String schedulerName, String schedulerInstanceId, String rmiBindName, String rmiHost, int rmiPort) throws SchedulerException

Same as createRemoteScheduler(String rmiHost, int rmiPort), with the addition of specifying the scheduler name, instance ID, and rmi bind name. This scheduler can only be retrieved via getScheduler(String)

Parameters:
- schedulerName - The name for the scheduler.
- schedulerInstanceId - The instance ID for the scheduler.
- rmiBindName - The name of the remote scheduler in the RMI repository. If null defaults to the generated unique identifier.
- rmiHost - The hostname for remote scheduler
- rmiPort - Port for the remote scheduler. The default RMI port is 1099.

Throws:
- SchedulerException - if the remote scheduler could not be reached.

createScheduler

public void createScheduler(org.quartz.spi.ThreadPool threadPool, org.quartz.spi.JobStore jobStore) throws SchedulerException

Creates a scheduler using the specified thread pool and job store. This scheduler can be retrieved via getScheduler()

Parameters:
- threadPool - The thread pool for executing jobs
- jobStore - The type of job store

Throws:
- SchedulerException - if initialization failed

createScheduler

public void createScheduler(String schedulerName,
Same as `createScheduler(ThreadPool threadPool, JobStore jobStore)`, with the addition of specifying the scheduler name and instance ID. This scheduler can only be retrieved via `getScheduler(String)`.

**Parameters:**
- `schedulerName` - The name for the scheduler.
- `schedulerInstanceId` - The instance ID for the scheduler.
- `threadPool` - The thread pool for executing jobs
- `jobStore` - The type of job store

**Throws:**
- `SchedulerException` - if initialization failed

### createScheduler

```java
public void createScheduler(String schedulerName, String schedulerInstanceId, ThreadPool threadPool, JobStore jobStore, String rmiRegistryHost, int rmiRegistryPort, long idleWaitTime, long dbFailureRetryInterval)
throws SchedulerException
```

Creates a scheduler using the specified thread pool and job store and binds it to RMI.

**Parameters:**
- `schedulerName` - The name for the scheduler.
- `schedulerInstanceId` - The instance ID for the scheduler.
- `threadPool` - The thread pool for executing jobs
- `jobStore` - The type of job store
- `rmiRegistryHost` - The hostname to register this scheduler with for RMI. Can use "null" if no RMI is required.
- `rmiRegistryPort` - The port for RMI. Typically 1099.
idleWaitTime - The idle wait time in milliseconds. You can specify "-1" for the default value, which is currently 30000 ms.

**Throws:**

*ScheduledException* - if initialization failed

---

**createScheduler**

```java
public void createScheduler(String schedulerName,
                           String schedulerInstanceId,
                           org.quartz.spi.ThreadPool threadPool,
                           org.quartz.spi.JobStore jobStore,
                           Map schedulerPluginMap,
                           String rmiRegistryHost,
                           int rmiRegistryPort,
                           long idleWaitTime,
                           long dbFailureRetryInterval,
                           boolean jmxExport,
                           String jmxObjectName)
throws SchedulerException
```

Creates a scheduler using the specified thread pool, job store, and plugins, and binds it to RMI.

**Parameters:**

- `schedulerName` - The name for the scheduler.
- `schedulerInstanceId` - The instance ID for the scheduler.
- `threadPool` - The thread pool for executing jobs
- `jobStore` - The type of job store
- `schedulerPluginMap` - Map from a String plugin names to SchedulerPlugins. Can use "null" if no plugins are required.
- `rmiRegistryHost` - The hostname to register this scheduler with for RMI. Can use "null" if no RMI is required.
- `rmiRegistryPort` - The port for RMI. Typically 1099.
- `idleWaitTime` - The idle wait time in milliseconds. You can specify "-1" for the default value, which is currently 30000 ms.

**Throws:**

*ScheduledException* - if initialization failed

---

**getScheduler**
public Scheduler getScheduler() throws SchedulerException

Returns a handle to the Scheduler produced by this factory.

you must call createRemoteScheduler or createScheduler methods before calling getScheduler()

Specified by:
getScheduler in interface SchedulerFactory

Throws:
SchedulerException - if there is a problem with the underlying Scheduler.

getScheduler

public Scheduler getScheduler(String schedName) throws SchedulerException

Returns a handle to the Scheduler with the given name, if it exists.

Specified by:
getScheduler in interface SchedulerFactory

Throws:
SchedulerException

getAllSchedulers

public Collection getAllSchedulers() throws SchedulerException

Returns a handle to all known Schedulers (made by any StdSchedulerFactory instance.).

Specified by:
getAllSchedulers in interface SchedulerFactory

Throws:
SchedulerException
**org.quartz.impl Class JobDetailImpl**

`java.lang.Object
   └ org.quartz.impl.JobDetailImpl`

**All Implemented Interfaces:**
 Serializable, Cloneable, JobDetail

```
public class JobDetailImpl
    extends Object
    implements Cloneable, Serializable, JobDetail
```

Conveys the detail properties of a given Job instance.

Quartz does not store an actual instance of a Job class, but instead allows you to define an instance of one, through the use of a JobDetail.

Jobs have a name and group associated with them, which should uniquely identify them within a single Scheduler.

Triggers are the 'mechanism' by which Jobs are scheduled. Many Triggers can point to the same Job, but a single Trigger can only point to one Job.

**Author:**
James House, Sharada Jambula

**See Also:**
Job, StatefulJob, JobDataMap, Trigger, Serialized Form

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobDetailImpl()</code></td>
<td>Create a JobDetail with no specified name or group, and the default settings of all the other properties.</td>
</tr>
<tr>
<td><code>JobDetailImpl(String name, Class&lt;? extends Job&gt; jobClass)</code></td>
<td>Deprecated. use JobBuilder</td>
</tr>
<tr>
<td><code>JobDetailImpl(String name, String group, Class&lt;? extends</code></td>
<td></td>
</tr>
</tbody>
</table>
**Job```> jobClass)```**

*Deprecated.* use `JobBuilder`

```java
JobDetailImpl(String name, String group, Class<?> extends Job> jobClass, boolean durability, boolean recover)
```

*Deprecated.* use `JobBuilder`

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Object clone()</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean equals(Object obj)</code></td>
<td></td>
</tr>
<tr>
<td><code>String getDescription()</code></td>
<td>Return the description given to the Job instance by its creator (if any).</td>
</tr>
<tr>
<td><code>String getFullName()</code></td>
<td>Returns the 'full name' of the JobDetail in the format &quot;group.name&quot;.</td>
</tr>
<tr>
<td><code>String getGroup()</code></td>
<td>Get the group of this Job.</td>
</tr>
<tr>
<td><code>JobBuilder getJobBuilder()</code></td>
<td>Get a <code>JobBuilder</code> that is configured to produce a JobDetail identical to this one.</td>
</tr>
<tr>
<td><code>Class&lt;?&gt; extends Job&gt; getJobClass()</code></td>
<td>Get the instance of Job that will be executed.</td>
</tr>
<tr>
<td><code>JobDataMap getJobDataMap()</code></td>
<td>Get the JobDataMap that is associated with the Job.</td>
</tr>
<tr>
<td><code>JobKey getKey()</code></td>
<td></td>
</tr>
<tr>
<td><code>String getName()</code></td>
<td>Get the name of this Job.</td>
</tr>
<tr>
<td><code>int hashCode()</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean isConcurrentExecutionDisallowed()</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>isDurable()</code></td>
<td>Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).</td>
</tr>
<tr>
<td><code>isPersistJobDataAfterExecution()</code></td>
<td></td>
</tr>
<tr>
<td><code>requestsRecovery()</code></td>
<td>Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.</td>
</tr>
<tr>
<td><code>setDescription(String description)</code></td>
<td>Set a description for the Job instance - may be useful for remembering/displaying the purpose of the job, though the description has no meaning to Quartz.</td>
</tr>
<tr>
<td><code>setDurability(boolean durability)</code></td>
<td>Set whether or not the Job should remain stored after it is orphaned (no Triggers point to it).</td>
</tr>
<tr>
<td><code>setGroup(String group)</code></td>
<td>Set the group of this Job.</td>
</tr>
<tr>
<td><code>setJobClass(Class&lt;? extends Job&gt; jobClass)</code></td>
<td>Set the instance of Job that will be executed.</td>
</tr>
<tr>
<td><code>setJobDataMap(JobDataMap jobDataMap)</code></td>
<td>Set the JobDataMap to be associated with the Job.</td>
</tr>
<tr>
<td><code>setKey(JobKey key)</code></td>
<td></td>
</tr>
<tr>
<td><code>setName(String name)</code></td>
<td>Set the name of this Job.</td>
</tr>
<tr>
<td><code>setRequestsRecovery(boolean shouldRecover)</code></td>
<td>Set whether or not the the Scheduler should re-execute the Job if a 'recovery' or 'fail-over' situation is encountered.</td>
</tr>
<tr>
<td><code>toString()</code></td>
<td>Return a simple string representation of this object.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**
- `finalize`, `getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`
Constructor Detail

JobDetailImpl

public JobDetailImpl()

Create a JobDetail with no specified name or group, and the default settings of all the other properties.

Note that the `setName(String), setGroup(String)` and `setJobClass(Class)` methods must be called before the job can be placed into a `Scheduler`.

---

JobDetailImpl

public JobDetailImpl(String name, Class<? extends Job> jobClass)

Deprecated. use JobBuilder

Create a JobDetail with the given name, given class, default group, and the default settings of all the other properties.

Parameters:

- group - if null, Scheduler.DEFAULT_GROUP will be used.

Throws:

- `IllegalArgumentException` - if name is null or empty, or the group is an empty string.

---

JobDetailImpl

public JobDetailImpl(String name, String group, Class<? extends Job> jobClass)

Deprecated. use JobBuilder
Create a JobDetail with the given name, group and class, and the default settings of all the other properties.

**Parameters:**
- group - if null, Scheduler.DEFAULT_GROUP will be used.

**Throws:**
- IllegalArgumentException - if name is null or empty, or the group is an empty string.

---

**JobDetailImpl**

```java
public JobDetailImpl(String name,
                     String group,
                     Class<? extends Job> jobClass,
                     boolean durability,
                     boolean recover)
```

**Deprecated. use JobBuilder**

Create a JobDetail with the given name, and group, and the given settings of all the other properties.

**Parameters:**
- group - if null, Scheduler.DEFAULT_GROUP will be used.

**Throws:**
- IllegalArgumentException - if name is null or empty, or the group is an empty string.

---

**Method Detail**

**getName**

```java
public String getName()
```

Get the name of this Job.

**setName**

public void setName(String name)

    Set the name of this Job.

    **Throws:**
    [IllegalArgumentException](#) - if name is null or empty.

---

**getGroup**

public String getGroup()

    Get the group of this Job.

---

**setGroup**

public void setGroup(String group)

    Set the group of this Job.

    **Parameters:**
    group - if null, Scheduler.DEFAULT_GROUP will be used.

    **Throws:**
    [IllegalArgumentException](#) - if the group is an empty string.

---

**getFullName**

public String getFullName()

    Returns the 'full name' of the JobDetail in the format "group.name".

---

**getKey**

public JobKey getKey()

    **Specified by:**
    [getKey](#) in interface JobDetail
**setKey**

public void setKey(JobKey key)

**getDescription**

public String getDescription()

Description copied from interface: JobDetail

Return the description given to the Job instance by its creator (if any).

Specified by: getDescription in interface JobDetail

Returns:
null if no description was set.

**setDescription**

public void setDescription(String description)

Set a description for the Job instance - may be useful for remembering/displaying the purpose of the job, though the description has no meaning to Quartz.

**getJobClass**

public Class<? extends Job> getJobClass()

Description copied from interface: JobDetail

Get the instance of Job that will be executed.

Specified by:
getJobClass in interface JobDetail
**setJobClass**

public void **setJobClass**(Class<? extends Job> jobClass)

Set the instance of Job that will be executed.

**Throws:**
   IllegalArgumentOutOfRangeException - if jobClass is null or the class is not a Job.

---

**getJobDataMap**

public **JobDataMap** getJobDataMap()

   Description copied from interface: JobDetail

   Get the JobDataMap that is associated with the Job.

   **Specified by:**
      getJobDataMap in interface JobDetail

---

**setJobDataMap**

public void **setJobDataMap**(JobDataMap jobDataMap)

   Set the JobDataMap to be associated with the Job.

---

**setDurability**

public void **setDurability**(boolean durability)

   Set whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

   If not explicitly set, the default value is false.
**setRequestsRecovery**

```java
public void setRequestsRecovery(boolean shouldRecover)
```

Set whether or not the the Scheduler should re-execute the Job if a 'recovery' or 'fail-over' situation is encountered.

If not explicitly set, the default value is false.

**See Also:**
`JobExecutionContext.isRecovering()`

---

**isDurable**

```java
public boolean isDurable()
```

**Description copied from interface:** `JobDetail`

Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

If not explicitly set, the default value is false.

**Specified by:**
`isDurable` in interface `JobDetail`

**Returns:**
true if the Job should remain persisted after being orphaned.

---

**isPersistJobDataAfterExecution**

```java
public boolean isPersistJobDataAfterExecution()
```

**Specified by:**
`isPersistJobDataAfterExecution` in interface `JobDetail`

**Returns:**
whether the associated Job class carries the `PersistJobDataAfterExecution` annotation.

**See Also:**
isConcurrentExecutionDisallowed

public boolean isConcurrentExecutionDisallowed()

Specified by:  
isConcurrentExecutionDisallowed in interface JobDetail

Returns:  
whether the associated Job class carries the  
DisallowConcurrentExecution annotation.

See Also:  
DisallowConcurrentExecution

requestsRecovery

public boolean requestsRecovery()

Description copied from interface: JobDetail

Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.

If not explicitly set, the default value is \texttt{false}.

Specified by:  
requestsRecovery in interface JobDetail

See Also:  
JobExecutionContext.isRecovering()

toString

public String toString()

Return a simple string representation of this object.

Overrides:
null

equals

public boolean equals(Object obj)

Overrides:
equals in class Object

hashCode

public int hashCode()

Overrides:
hashCode in class Object

clon

public Object clone()

Specified by:
clone in interface JobDetail

Overrides:
clone in class Object

getJobBuilder

public JobBuilder getJobBuilder()

Description copied from interface: JobDetail
Get a JobBuilder that is configured to produce a JobDetail identical to this one.

Specified by:
getJobBuilder in interface JobDetail
org.quartz.impl  Class JobExecutionContextImpl

java.lang.Object
   U org.quartz.impl.JobExecutionContextImpl

All Implemented Interfaces:
   Serializable, JobExecutionContext

public class JobExecutionContextImpl
extends Object
implements Serializable, JobExecutionContext

See Also:
   Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Constructor Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobExecutionContextImpl</td>
<td>(Scheduler scheduler, org.quartz.spi.TriggerFiredBundle firedBundle, Job job)</td>
</tr>
</tbody>
</table>

Create a JobExecutionContext with the given context data.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Get the value with the given key from the context's data map.</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>Object key</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Get a handle to the Calendar referenced by the Trigger instance that fired the Job.</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCalendar</td>
<td>()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>The actual time the trigger fired.</th>
</tr>
</thead>
<tbody>
<tr>
<td>getFireTime</td>
<td>()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Get the JobDetail associated with the Job.</th>
</tr>
</thead>
<tbody>
<tr>
<td>getJobDetail</td>
<td>()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Get the instance of the Job that was created for this execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>getJobInstance</td>
<td>()</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>long getJobRunTime()</td>
<td>The amount of time the job ran for (in milliseconds).</td>
</tr>
<tr>
<td>JobDataMap getMergedJobDataMap()</td>
<td>Get the convenience JobDataMap of this execution context.</td>
</tr>
<tr>
<td>Date getNextFireTime()</td>
<td></td>
</tr>
<tr>
<td>Date getPreviousFireTime()</td>
<td></td>
</tr>
<tr>
<td>int getRefireCount()</td>
<td></td>
</tr>
<tr>
<td>Object getResult()</td>
<td>Returns the result (if any) that the Job set before its execution completed (the type of object set as the result is entirely up to the particular job).</td>
</tr>
<tr>
<td>Date get ScheduledFireTime()</td>
<td>The scheduled time the trigger fired for.</td>
</tr>
<tr>
<td>Scheduler getScheduled()</td>
<td>Get a handle to the Scheduler instance that fired the Job.</td>
</tr>
<tr>
<td>Trigger getTrigger()</td>
<td>Get a handle to the Trigger instance that fired the Job.</td>
</tr>
<tr>
<td>void incrementRefireCount()</td>
<td></td>
</tr>
<tr>
<td>boolean isRecovering()</td>
<td>If the Job is being re-executed because of a 'recovery' situation, this method will return true.</td>
</tr>
<tr>
<td>void put(Object key, Object value)</td>
<td>Put the specified value into the context's data map with the given key.</td>
</tr>
<tr>
<td>void setJobRunTime(long jobRunTime)</td>
<td></td>
</tr>
<tr>
<td>void setResult(Object result)</td>
<td>Set the result (if any) of the Job's execution (the type of object set as the result is entirely up to the particular job).</td>
</tr>
<tr>
<td>String toString()</td>
<td></td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

JobExecutionContextImpl

public JobExecutionContextImpl(Scheduler scheduler, org.quartz.spi.TriggerFiredBundle firedBundle, Job job)

Create a JobExcecutionContext with the given context data.

Method Detail

getscheduler

public Scheduler getscheduler()

Description copied from interface: JobExecutionContext

Get a handle to the Scheduler instance that fired the Job.

Specified by:
getscheduler in interface JobExecutionContext

getTrigger

public Trigger getTrigger()

Description copied from interface: JobExecutionContext

Get a handle to the Trigger instance that fired the Job.
**getCalendar**

```java
public Calendar getCalendar()
```

**Description copied from interface: JobExecutionContext**

Get a handle to the Calendar referenced by the Trigger instance that fired the Job.

**Specified by:**

getCalendar in interface JobExecutionContext

---

**isRecovering**

```java
public boolean isRecovering()
```

**Description copied from interface: JobExecutionContext**

If the Job is being re-executed because of a 'recovery' situation, this method will return true.

**Specified by:**

isRecovering in interface JobExecutionContext

---

**incrementRefireCount**

```java
public void incrementRefireCount()
```

---

**getRefireCount**

```java
public int getRefireCount()
```

**Specified by:**
getRefireCount in interface JobExecutionContext

getMergedJobDataMap

public JobDataMap getMergedJobDataMap()

Description copied from interface: JobExecutionContext

Get the convenience JobDataMap of this execution context.

The JobDataMap found on this object serves as a convenience - it is a merge of the JobDataMap found on the JobDetail and the one found on the Trigger, with the value in the latter overriding any same-named values in the former. It is thus considered a 'best practice' that the execute code of a Job retrieve data from the JobDataMap found on this object.

NOTE: Do not expect value 'set' into this JobDataMap to somehow be set back onto a StatefulJob's own JobDataMap.

Attempts to change the contents of this map typically result in an IllegalStateException.

Specified by:

getAddressedJobDataMap in interface JobExecutionContext

getJobDetail

public JobDetail getJobDetail()

Description copied from interface: JobExecutionContext

Get the JobDetail associated with the Job.

Specified by:

getAddressedJobDetail in interface JobExecutionContext

getJobInstance
public Job getJobInstance()

**Description copied from interface: JobExecutionContext**

Get the instance of the Job that was created for this execution.

Note: The Job instance is not available through remote scheduler interfaces.

**Specified by:**

getJobInstance in interface JobExecutionContext

---

getFireTime

public Date getFireTime()

**Description copied from interface: JobExecutionContext**

The actual time the trigger fired. For instance the scheduled time may have been 10:00:00 but the actual fire time may have been 10:00:03 if the scheduler was too busy.

**Specified by:**

getFireTime in interface JobExecutionContext

**Returns:**

Returns the fireTime.

**See Also:**

JobExecutionContext.getScheduledFireTime()

---

getscheduledFireTime

public Date getScheduledFireTime()

**Description copied from interface: JobExecutionContext**

The scheduled time the trigger fired for. For instance the scheduled time may have been 10:00:00 but the actual fire time may have been 10:00:03 if the scheduler was too busy.

**Specified by:**

getScheduledFireTime in interface JobExecutionContext
Returns:
Returns the scheduledFireTime.

See Also:
JobExecutionContext.getFireTime()

getPreviousFireTime

public Date getPreviousFireTime()

Specified by:
getPreviousFireTime in interface JobExecutionContext

getNextFireTime

public Date getNextFireTime()

Specified by:
getNextFireTime in interface JobExecutionContext

toString

public String toString()

Overrides:
toString in class Object

getResult

public Object getResult()

Description copied from interface: JobExecutionContext
Returns the result (if any) that the Job set before its execution completed (the type of object set as the result is entirely up to the particular job).

The result itself is meaningless to Quartz, but may be informative to
JobListeners or TriggerListeners that are watching the job's execution.

**Specified by:**
```
getResult in interface JobExecutionContext
```

**Returns:**
Returns the result.

---

**setResult**

```
public void setResult(Object result)
```

**Description copied from interface:** JobExecutionContext
Set the result (if any) of the Job's execution (the type of object set as the result is entirely up to the particular job).

The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

**Specified by:**
```
setResult in interface JobExecutionContext
```

---

**getJobRunTime**

```
public long getJobRunTime()
```

**Description copied from interface:** JobExecutionContext
The amount of time the job ran for (in milliseconds). The returned value will be -1 until the job has actually completed (or thrown an exception), and is therefore generally only useful to JobListeners and TriggerListeners.

**Specified by:**
```
getJobRunTime in interface JobExecutionContext
```

**Returns:**
Returns the jobRunTime.

---

**setJobRunTime**
public void setJobRunTime(long jobRunTime)

Parameters:
jobRunTime - The jobRunTime to set.

put

public void put(Object key,
                Object value)

Description copied from interface: JobExecutionContext
Put the specified value into the context's data map with the given key. Possibly useful for sharing data between listeners and jobs.

NOTE: this data is volatile - it is lost after the job execution completes, and all TriggerListeners and JobListeners have been notified.

Specified by:
put in interface JobExecutionContext

get

public Object get(Object key)

Description copied from interface: JobExecutionContext
Get the value with the given key from the context's data map.

Specified by:
get in interface JobExecutionContext

Overview  Package  Use  Tree  Deprecated  Index  Help
org.quartz.impl Classes

DirectSchedulerFactory
JobDetailImpl
JobExecutionContextImpl
QuartzServer
RemoteMBeanScheduler
RemoteScheduler
SchedulerRepository
StdJobRunShellFactory
StdScheduler
StdSchedulerFactory
# Package org.quartz.impl

Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.

See: [Description](#)

## Class Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DirectSchedulerFactory</strong></td>
<td>A singleton implementation of SchedulerFactory.</td>
</tr>
<tr>
<td><strong>JobDetailImpl</strong></td>
<td>Conveys the detail properties of a given Job instance.</td>
</tr>
<tr>
<td><strong>JobExecutionContextImpl</strong></td>
<td></td>
</tr>
<tr>
<td><strong>QuartzServer</strong></td>
<td>Instantiates an instance of Quartz Scheduler as a stand-alone program, if the scheduler is configured for RMI it will be made available.</td>
</tr>
<tr>
<td><strong>RemoteMBeanScheduler</strong></td>
<td>An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JMX.</td>
</tr>
<tr>
<td><strong>RemoteScheduler</strong></td>
<td>An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via RMI.</td>
</tr>
<tr>
<td><strong>SchedulerRepository</strong></td>
<td>Holds references to Scheduler instances - ensuring uniqueness, and preventing garbage collection, and allowing 'global' lookups - all within a ClassLoader space.</td>
</tr>
<tr>
<td><strong>StdJobRunShellFactory</strong></td>
<td>Responsible for creating the instances of JobRunShell to be used within the QuartzScheduler instance.</td>
</tr>
<tr>
<td><strong>StdScheduler</strong></td>
<td>An implementation of the Scheduler interface that directly proxies all method calls to the equivalent call on a given QuartzScheduler instance.</td>
</tr>
</tbody>
</table>

An implementation of SchedulerFactory that
| **StdSchedulerFactory** | does all of its work of creating a QuartzScheduler instance based on the contents of a Properties file. |
Package org.quartz.impl Description

Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.

Classes in this package may have dependencies on third-party packages.

See the Quartz project for more information.

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.impl

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.**DirectSchedulerFactory** (implements org.quartz.**SchedulerFactory**)
  - org.quartz.impl.**JobDetailImpl** (implements java.lang.**Cloneable**, org.quartz.**JobDetail**, java.io.**Serializable**)
  - org.quartz.impl.**JobExecutionContextImpl** (implements org.quartz.**JobExecutionContext**, java.io.**Serializable**)
  - org.quartz.impl.**RemoteMBeanScheduler** (implements org.quartz.**Scheduler**)
  - org.quartz.impl.**RemoteScheduler** (implements org.quartz.**Scheduler**)
  - org.quartz.listeners.**SchedulerListenerSupport** (implements org.quartz.**SchedulerListener**)
    - org.quartz.impl.**QuartzServer**
  - org.quartz.impl.**SchedulerRepository**
  - org.quartz.impl.**StdJobRunShellFactory** (implements org.quartz.core.**JobRunShellFactory**)
  - org.quartz.impl.**StdScheduler** (implements org.quartz.**Scheduler**)
  - org.quartz.impl.**StdSchedulerFactory** (implements org.quartz.**SchedulerFactory**)

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.impl

<table>
<thead>
<tr>
<th>Packages that use org.quartz.impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
</tr>
<tr>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.ee.jmx.jboss</td>
</tr>
<tr>
<td>org.quartz.impl</td>
</tr>
<tr>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.quartz.impl used by org.quartz.core</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobExecutionContextImpl</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.quartz.impl used by org.quartz.ee.jmx.jboss</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoteMBeanScheduler</td>
</tr>
<tr>
<td>An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JMX.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.quartz.impl used by org.quartz.impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectSchedulerFactory</td>
</tr>
<tr>
<td>A singleton implementation of SchedulerFactory.</td>
</tr>
<tr>
<td>SchedulerRepository</td>
</tr>
<tr>
<td>Holds references to Scheduler instances - ensuring uniqueness, and preventing garbage collection, and allowing 'global' lookups - all within a ClassLoader space.</td>
</tr>
</tbody>
</table>
org.quartz.impl Class QuartzServer

java.lang.Object
  └ org.quartz.listeners.SchedulerListenerSupport
     └ org.quartz.impl.QuartzServer

All Implemented Interfaces:
   SchedulerListener

public class QuartzServer
extends SchedulerListenerSupport

Instantiates an instance of Quartz Scheduler as a stand-alone program, if the scheduler is configured for RMI it will be made available.

The main() method of this class currently accepts 0 or 1 arguments, if there is an argument, and its value is "console", then the program will print a short message on the console (std-out) and wait for the user to type "exit" - at which time the scheduler will be shutdown.

Future versions of this server should allow additional configuration for responding to scheduler events by allowing the user to specify JobListener, TriggerListener and SchedulerListener classes.

Please read the Quartz FAQ entries about RMI before asking questions in the forums or mail-lists.

Author:
   James House

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void main(String[] args)</td>
</tr>
<tr>
<td>void schedulerError(String msg, SchedulerException cause)</td>
</tr>
<tr>
<td>Called by the Scheduler when a serious error has occurred</td>
</tr>
</tbody>
</table>
within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

```java
void schedulerShutdown()
    Called by the Scheduler to inform the listener that it has shutdown.

void serve(SchedulerFactory schedFact, boolean console)
```

**Methods inherited from class**
org.quartz.listeners.SchedulerListenerSupport
getLog, jobAdded, jobDeleted, jobPaused, jobResumed, jobScheduled, jobsPaused, jobsResumed, jobUnscheduled, schedulerInStandbyMode, schedulerShuttingdown, schedulerStarted, schedulingDataCleared, triggerFinalized, triggerPaused, triggerResumed, triggersPaused, triggersResumed

**Methods inherited from class** java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Method Detail

**serve**

```java
public void serve(SchedulerFactory schedFact, boolean console)
```

**Throws:**

Exception

**schedulerError**

```java
public void schedulerError(String msg,
    SchedulerException cause)
```
Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

The `getErrorCode()` method of the given SchedulerException can be used to determine more specific information about the type of error that was encountered.

**Specified by:**

`schedulerError` in interface `SchedulerListener`

**Overrides:**

`schedulerError` in class `SchedulerListenerSupport`

---

**schedulerShutdown**

`public void schedulerShutdown()`

Called by the Scheduler to inform the listener that it has shutdown.

**Specified by:**

`schedulerShutdown` in interface `SchedulerListener`

**Overrides:**

`schedulerShutdown` in class `SchedulerListenerSupport`

---

**main**

`public static void main(String[] args)`

throws `Exception`

**Throws:**

`Exception`

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl Class RemoteMBeanScheduler

java.lang.Object
   └ org.quartz.impl.RemoteMBeanScheduler

All Implemented Interfaces:
   Scheduler

Direct Known Subclasses:
   JBoss4RMIRemoteMBeanScheduler

public abstract class RemoteMBeanScheduler
   extends Object
   implements Scheduler

An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JMX.

A user must create a subclass to implement the actual connection to the remote MBeanServer using their application specific connector. For example JBoss4RMIRemoteMBeanScheduler.

See Also:
   Scheduler, QuartzScheduler, org.quartz.core.SchedulingContext

Field Summary

Fields inherited from interface org.quartz.Scheduler

DEFAULT_FAIL_OVER_GROUP, DEFAULT_GROUP, DEFAULT_RECOVERY_GROUP,
FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS,
FAILED_JOB_ORIGINAL_TRIGGER_GROUP,
FAILED_JOB_ORIGINAL_TRIGGER_NAME

Constructor Summary
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>void</td>
<td><code>addJob(JobDetail jobDetail, boolean replace)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>checkExists(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>checkExists(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><code>clear()</code></td>
<td>Clears (deletes!) all scheduling data - all Jobs, Trigger Calendars.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>deleteCalendar(String calName)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>deleteJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>deleteJobs(List&amp;lt;JobKey&amp;gt; jobKeys)</code></td>
<td>Delete the identified Jobs from the Scheduler - and associated Triggers.</td>
</tr>
<tr>
<td>protected abstract</td>
<td><code>getAttribute(String attribute)</code></td>
<td>Get the given attribute of the remote Scheduler MBean.</td>
</tr>
<tr>
<td>AttributeList</td>
<td><code>getAttributes(String[] attributes)</code></td>
<td>Get the given attributes of the remote Scheduler MBean.</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Calendar</td>
<td><code>getCalendar(String calName)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code></td>
<td><code>getCalendarNames()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>SchedulerContext</code></td>
<td><code>getContext()</code></td>
<td>Returns the SchedulerContext of the Scheduler.</td>
</tr>
<tr>
<td><code>List&lt;JobExecutionContext&gt;</code></td>
<td><code>getCurrentlyExecutingJobs()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>JobDetail</code></td>
<td><code>getJobDetail(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code></td>
<td><code>getJobGroupNames()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code></td>
<td><code>getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>ListenerManager</code></td>
<td><code>getListenerManager()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>SchedulerMetaData</code></td>
<td><code>getMetaData()</code></td>
<td>Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.</td>
</tr>
<tr>
<td><code>Set&lt;String&gt;</code></td>
<td><code>getPausedTriggerGroups()</code></td>
<td>Get the names of all Trigger groups that are paused.</td>
</tr>
<tr>
<td><code>String</code></td>
<td><code>getSchedulerInstanceId()</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>String getSchedulerName()</code></td>
<td>Returns the name of the Scheduler.</td>
<td></td>
</tr>
<tr>
<td><code>protected ObjectName getSchedulerObjectName()</code></td>
<td>Get the name under which the Scheduler MBean is registered on the remote MBean server.</td>
<td></td>
</tr>
<tr>
<td><code>Trigger getTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>List&lt;String&gt; getTriggerGroupNames()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
<td></td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt; getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
<td></td>
</tr>
<tr>
<td><code>List&lt;Trigger&gt; getTriggersOfJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
<td></td>
</tr>
<tr>
<td><code>Trigger.TriggerState getTriggerState(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
<td></td>
</tr>
<tr>
<td><code>abstract void initialize()</code></td>
<td>Initialize this RemoteMBeanScheduler instance, connecting to the remote MBean server.</td>
<td></td>
</tr>
<tr>
<td><code>boolean interrupt(JobKey jobKey)</code></td>
<td>Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which implements the InterruptableJob interface.</td>
<td></td>
</tr>
<tr>
<td><code>protected abstract Object invoke(String operationName, Object[] params, String[] signature)</code></td>
<td>Invoke the given operation on the remote Scheduler.</td>
<td></td>
</tr>
<tr>
<td><code>boolean isInStandbyMode()</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Calls the equivalent method on the 'proxied' QuartzScheduler.

boolean isShutdown()
    Calls the equivalent method on the 'proxied' QuartzScheduler.

boolean isStarted()
    Whether the scheduler has been started.

void pauseAll()
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

void pauseJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

void pauseJobs(GroupMatcher<JobKey> matcher)
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

void pauseTrigger(TriggerKey triggerKey)
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

void pauseTriggers(GroupMatcher<TriggerKey> matcher)
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Date rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

void resumeAll()
    Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

void resumeJob(JobKey jobKey)
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

```java
void resumeJobs(GroupMatcher<JobKey> matcher)
```
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

```java
void resumeTrigger(TriggerKey triggerKey)
```
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

```java
void resumeTriggers(GroupMatcher<TriggerKey> matcher)
```
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

```java
Date scheduleJob(JobDetail jobDetail, Trigger trigger)
```
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

```java
Date scheduleJob(Trigger trigger)
```
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

```java
void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace)
```
Schedule all of the given jobs with the related set of triggers.

```java
void setJobFactory(org.quartz.spi.JobFactory factory)
```
Set the JobFactory that will be responsible for producing instances of Job classes.

```java
void setSchedulerObjectName(ObjectName schedulerObjectName)
```
Set the name under which the Scheduler MBean is registered on the remote MBean server.

```java
void setSchedulerObjectName(String schedulerObjectName)
```
Set the name under which the Scheduler MBean is registered on the remote MBean server.

```java
void shutdown()
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void shutdown(boolean waitForJobsToComplete)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>void standby()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>void start()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>void startDelayed(int seconds)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><code>protected Boolean toBoolean(boolean bool)</code></td>
<td></td>
</tr>
<tr>
<td><code>void triggerJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>void triggerJob(JobKey jobKey, JobDataMap data)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>boolean unscheduleJob(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.</td>
</tr>
<tr>
<td><code>boolean unscheduleJobs(List&lt;TriggerKey&gt; triggerKeys)</code></td>
<td>Remove all of the indicated Triggers from the scheduler.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.**Object**

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Detail

RemoteMBeanScheduler

class RemoteMBeanScheduler

| Method Detail |

getSchedulerObjectName

```java
protected ObjectName getSchedulerObjectName()
```

Get the name under which the Scheduler MBean is registered on the remote MBean server.

setSchedulerObjectName

```java
public void setSchedulerObjectName(String schedulerObjectName)
```

Set the name under which the Scheduler MBean is registered on the remote MBean server.

**Throws:**

`SchedulerException`

setSchedulerObjectName

```java
public void setSchedulerObjectName(ObjectName schedulerObjectName)
```

Set the name under which the Scheduler MBean is registered on the remote MBean server.

**Throws:**

`SchedulerException`
**initialize**

```java
public abstract void initialize()
    throws SchedulerException
```

Initialize this RemoteMBeanScheduler instance, connecting to the remote MBean server.

** Throws:**

```
SchedulerException
```

---

**getAttribute**

```java
protected abstract Object getAttribute(String attribute)
    throws SchedulerException
```

Get the given attribute of the remote Scheduler MBean.

** Throws:**

```
SchedulerException
```

---

**getAttributes**

```java
protected abstract AttributeList getAttributes(String[] attributes)
    throws SchedulerException
```

Get the given attributes of the remote Scheduler MBean.

** Throws:**

```
SchedulerException
```

---

**invoke**

```java
protected abstract Object invoke(String operationName,
    Object[] params,
    String[] signature)
    throws SchedulerException
```

---
Invoke the given operation on the remote Scheduler MBean.

**Throws:**

SchedulerException

---

**getSchedulerName**

```java
public String getSchedulerName()
```

Throws: SchedulerException

Returns the name of the Scheduler.

**Specified by:**

getSchedulerName in interface Scheduler

**Throws:**

SchedulerException

---

**getSchedulerInstanceId**

```java
public String getSchedulerInstanceId()
```

throws SchedulerException

Returns the instance Id of the Scheduler.

**Specified by:**

getSchedulerInstanceId in interface Scheduler

**Throws:**

SchedulerException

---

**getMetaData**

```java
public SchedulerMetaData getMetaData()
```

throws SchedulerException

**Description copied from interface: Scheduler**

Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.
Note that the data returned is an 'instantaneous' snap-shot, and that as soon as it's returned, the meta data values may be different.

Specified by:

getMetaData in interface Scheduler

Throws:

SchedulerException

getContext

public SchedulerContext getContext() throws SchedulerException

Returns the SchedulerContext of the Scheduler.

Specified by:

getContext in interface Scheduler

Throws:

SchedulerException

start

public void start() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:

start in interface Scheduler

Throws:

SchedulerException - if shutdown() has been called, or there is an error within the Scheduler.

See Also:

Scheduler.startDelayed(int), Scheduler.standby(), Scheduler.shutdown()

startDelayed
public void **startDelayed**(int seconds)  
    throws **SchedulerException**

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**  
**startDelayed** in interface **Scheduler**

**Throws:**  
**SchedulerException** - if shutdown() has been called, or there is an error within the Scheduler.

**See Also:**  
**Scheduler.start(), Scheduler.standby(), Scheduler.shutdown()**

---

**standby**

public void **standby**()  
    throws **SchedulerException**

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**  
**standby** in interface **Scheduler**

**Throws:**  
**SchedulerException**

**See Also:**  
**Scheduler.start(), Scheduler.pauseAll()**

---

**isStarted**

public boolean **isStarted**()  
    throws **SchedulerException**

Whether the scheduler has been started.

Note: This only reflects whether start() has ever been called on this Scheduler, so it will return true even if the Scheduler is currently in standby mode or has been since shutdown.
isInStandbyMode

public boolean isInStandbyMode()
    throws SchedulerException

    Calls the equivalent method on the 'proxied' QuartzScheduler.

    Specified by: 
    isInStandbyMode in interface Scheduler
    Throws: 
    SchedulerException 
    See Also: 
    Scheduler.standby(), Scheduler.start()

shutdown

public void shutdown()
    throws SchedulerException

    Calls the equivalent method on the 'proxied' QuartzScheduler.

    Specified by: 
    shutdown in interface Scheduler
    Throws: 
    SchedulerException 
    See Also: 
    Scheduler.shutdown(boolean)
public void shutdown(boolean waitForJobsToComplete)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    shutdown in interface Scheduler
Parameters:
    waitForJobsToComplete - if true the scheduler will not allow this method to return until all currently executing jobs have completed.
Throws:
    SchedulerException
See Also:
    Scheduler.shutdown()

isShutdown

public boolean isShutdown()
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    isShutdown in interface Scheduler
Throws:
    SchedulerException

getCurrentlyExecutingJobs

public List&lt;JobExecutionContext&gt; getCurrentlyExecutingJobs()
    throws Scheduler

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    getCurrentlyExecutingJobs in interface Scheduler
Throws:
    SchedulerException
See Also:
JobExecutionContext

scheduleJob

public Date scheduleJob(JobDetail jobDetail, Trigger trigger) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
   scheduleJob in interface Scheduler

Throws:
   SchedulerException - if the Job or Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

addJob

public void addJob(JobDetail jobDetail, boolean replace) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

addJob

public Date scheduleJob(Trigger trigger) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
   scheduleJob in interface Scheduler

Throws:
   SchedulerException - if the indicated Job does not exist, or the Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

addJob

public void addJob(JobDetail jobDetail, boolean replace) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.
Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
`addJob` in interface `Scheduler`

**Throws:**
`SchedulerException` - if there is an internal Scheduler error, or if the Job is not durable, or a Job with the same name already exists, and replace is false.

---

**deleteJob**

public boolean `deleteJob(JobKey jobKey)`

throws `SchedulerException`

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
`deleteJob` in interface `Scheduler`

**Returns:**
true if the Job was found and deleted.

**Throws:**
`SchedulerException` - if there is an internal Scheduler error.

---

**unscheduleJob**

public boolean `unscheduleJob(TriggerKey triggerKey)`

throws `SchedulerException`

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
`unscheduleJob` in interface `Scheduler`

**Throws:**
`SchedulerException`
**deleteJobs**

```java
class java.util.Scheduler {
    public boolean deleteJobs(List<JobKey> jobKeys) throws SchedulerException;
}
```

**Description copied from interface: Scheduler**
Delete the identified Jobs from the Scheduler - and any associated Triggers.

Note that while this bulk operation is likely more efficient than invoking `deleteJob(JobKey jobKey)` several times, it may have the adverse affect of holding data locks for a single long duration of time (rather than lots of small durations of time).

**Specified by:**
`deleteJobs` in interface `Scheduler`

**Returns:**
true if all of the Jobs were found and deleted, false if one or more were not deleted.

**Throws:**
`SchedulerException` - if there is an internal Scheduler error.

**scheduleJobs**

```java
class java.util.Scheduler {
    public void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace) throws SchedulerException;
}
```

**Description copied from interface: Scheduler**
Schedule all of the given jobs with the related set of triggers.

If any of the given jobs or triggers already exist (or more specifically, if the keys are not unique) and the replace parameter is not set to true then an exception will be thrown.

**Specified by:**
`scheduleJobs` in interface `Scheduler`

**Throws:**
`ObjectAlreadyExistsException` - if the job/trigger keys are not
unique and the replace flag is not set to true.

SchedulerException

unscheduleJobs

public boolean unscheduleJobs(List<TriggerKey> triggerKeys)
throws SchedulerException

Description copied from interface: Scheduler
Remove all of the indicated Triggers from the scheduler.

If the related job does not have any other triggers, and the job is not durable, then the job will also be deleted.

Note that while this bulk operation is likely more efficient than invoking unscheduleJob(TriggerKey triggerKey) several times, it may have the adverse affect of holding data locks for a single long duration of time (rather than lots of small durations of time).

Specified by:
unscheduleJobs in interface Scheduler
Throws:
SchedulerException

rescheduleJob

public Date rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
rescheduleJob in interface Scheduler
Parameters:
triggerKey - identity of the trigger to replace
newTrigger - The new Trigger to be stored.
Returns:
null if a Trigger with the given name & group was not found and removed from the store, otherwise the first fire time of the newly scheduled trigger.

Throws:
SchedulerException

---

triggerJob

public void triggerJob(JobKey jobKey)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
triggerJob in interface Scheduler

Throws:
SchedulerException

---

triggerJob

public void triggerJob(JobKey jobKey, JobDataMap data)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
triggerJob in interface Scheduler
data - the (possibly null) JobDataMap to be associated with the trigger that fires the job immediately.

Throws:
SchedulerException

---

pauseTrigger
public void pauseTrigger(TriggerKey triggerKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
    pauseTrigger in interface Scheduler

Throws:
    SchedulerException

See Also:
    Scheduler.resumeTrigger(TriggerKey)

-------------------------------------------

pauseTriggers

public void pauseTriggers(GroupMatcher<TriggerKey> matcher) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
    pauseTriggers in interface Scheduler

Parameters:
    matcher - The matcher to evaluate against know groups

Throws:
    SchedulerException

See Also:
    Scheduler.resumeTriggers(org.quartz.impl.matchers.GroupMatcher)

-------------------------------------------

pauseJob

public void pauseJob(JobKey jobKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
**pauseJob** in interface **Scheduler**

**Throws:**
- **SchedulerException**

**See Also:**
- **Scheduler.resumeJob(JobKey)**

---

**pauseJobs**

```java
public void pauseJobs(GroupNameMatcher<JobKey> matcher)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
- **pauseJobs** in interface **Scheduler**

**Parameters:**
- **matcher** - The matcher to evaluate against known groups

**Throws:**
- **SchedulerException** - On error

**See Also:**
- **Scheduler.resumeJobs(org.quartz.impl.matchers.GroupMatcher)**

---

**resumeTrigger**

```java
public void resumeTrigger(TriggerKey triggerKey)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
- **resumeTrigger** in interface **Scheduler**

**Throws:**
- **SchedulerException**

**See Also:**
- **Scheduler.pauseTrigger(TriggerKey)**
resumeTriggers

public void resumeTriggers(GroupMatcher<TriggerKey> matcher)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

Specified by:
    resumeTriggers in interface Scheduler
Parameters:
    matcher - The matcher to evaluate against know paused groups
Throws:
    SchedulerException - On error
See Also:
    Scheduler.pauseTriggers(org.quartz.impl.matchers.GroupMatcher)

resumeJob

public void resumeJob(JobKey jobKey)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

Specified by:
    resumeJob in interface Scheduler
Throws:
    SchedulerException
See Also:
    Scheduler.pauseJob(JobKey)

resumeJobs

public void resumeJobs(GroupMatcher<JobKey> matcher)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

**Specified by:**
- `resumeJobs` in interface `Scheduler`

**Parameters:**
- matcher - The matcher to evaluate against known paused groups

**Throws:**
- `SchedulerException` - On error

**See Also:**
- `Scheduler.pauseJobs(GroupMatcher)`

---

### pauseAll

```java
public void pauseAll()
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
- `pauseAll` in interface `Scheduler`

**Throws:**
- `SchedulerException`

**See Also:**
- `Scheduler.resumeAll()`,
- `Scheduler.pauseTriggers(org.quartz.impl.matchers.GroupMatcher)`
- `Scheduler.standby()`

---

### resumeAll

```java
public void resumeAll()
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
- `resumeAll` in interface `Scheduler`
Throws:
SchedulerException

See Also:
Scheduler.pauseAll()

---

**getJobGroupNames**

```java
public List<String> getJobGroupNames() throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
getJobGroupNames in interface Scheduler

Throws:
SchedulerException

---

**getJobKeys**

```java
public Set<JobKey> getJobKeys(GroupMatcher<JobKey> matcher) throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
getJobKeys in interface Scheduler

Parameters:
matcher - Matcher to evaluate against known groups

Returns:
Set of all keys matching

Throws:
SchedulerException - On error

---

**getTriggersOfJob**
public List<Trigger> getTriggersOfJob(JobKey jobKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
getTriggersOfJob in interface Scheduler

Throws:
SchedulerException

---

getTriggerGroupNames

public List<String> getTriggerGroupNames() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
getTriggerGroupNames in interface Scheduler

Throws:
SchedulerException

---

getTriggerKeys

public Set<TriggerKey> getTriggerKeys(GroupMatcher<TriggerKey> matcher) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

Specified by:
getTriggerKeys in interface Scheduler

Parameters:
matcher - Matcher to evaluate against known groups

Returns:
List of all keys matching
Throws:

SchedulerException - On error

---

**getJobDetail**

public JobDetail getJobDetail(JobKey jobKey)

throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**

getJobDetail in interface Scheduler

**Throws:**

SchedulerException

---

**getTrigger**

public Trigger getTrigger(TriggerKey triggerKey)

throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**

getTrigger in interface Scheduler

**Throws:**

SchedulerException

---

**checkExists**

public boolean checkExists(JobKey jobKey)

throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**

checkExists in interface Scheduler
checkExists

public boolean checkExists(TriggerKey triggerKey)
        throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
        checkExists in interface Scheduler

Parameters:
        triggerKey - the identifier to check for

Returns:
        true if a Trigger exists with the given identifier

Throws:
        SchedulerException

clear

public void clear()
        throws SchedulerException

Description copied from interface: Scheduler
Clears (deletes!) all scheduling data - all Jobs, Triggers Calendars.

Specified by:
        clear in interface Scheduler

Throws:
        SchedulerException

gerTriggerState
public **Trigger.TriggerState** **getTriggerState**(TriggerKey triggerKey) throws **SchedulerException**

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
**getTriggerState** in interface **Scheduler**

**Throws:**
**SchedulerException**

**See Also:**
**Trigger.TriggerState**

---

**addCalendar**

public **void** **addCalendar**(String calName, Calendar calendar, boolean replace, boolean updateTriggers) throws **SchedulerException**

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
**addCalendar** in interface **Scheduler**

**updateTriggers** - whether or not to update existing triggers that referenced the already existing calendar so that they are 'correct' based on the new trigger.

**Throws:**
**SchedulerException** - if there is an internal Scheduler error, or a Calendar with the same name already exists, and replace is false.

---

**deleteCalendar**

public **boolean** **deleteCalendar**(String calName) throws **SchedulerException**

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the
SchedulingContext associated with this instance.

**Specified by:**
`deleteCalendar` in interface `Scheduler`

**Returns:**
true if the Calendar was found and deleted.

**Throws:**
- `SchedulerException` - if there is an internal Scheduler error.

---

### getCalendar

definition
`public Calendar getCalendar(String calName)`

throws `SchedulerException`

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
`getCalendar` in interface `Scheduler`

**Throws:**
- `SchedulerException`

---

### getCalendarNames

definition
`public List<String> getCalendarNames()`

throws `SchedulerException`

Calls the equivalent method on the 'proxied' QuartzScheduler, passing the SchedulingContext associated with this instance.

**Specified by:**
`getCalendarNames` in interface `Scheduler`

**Throws:**
- `SchedulerException`

---

### getPausedTriggerGroups
public Set<String> getPausedTriggerGroups() throws SchedulerException

Description copied from interface: Scheduler
Get the names of all Trigger groups that are paused.

Specified by:
getPausedTriggerGroups in interface Scheduler
Throws:
SchedulerException
See Also:
Scheduler.getPausedTriggerGroups()

getListenerManager

public ListenerManager getListenerManager() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getListenerManager in interface Scheduler
Returns:
the scheduler's ListenerManager
Throws:
SchedulerException - if the scheduler is not local
See Also:
ListenerManager, JobListener, TriggerListener, SchedulerListener

interrupt

public boolean interrupt(JobKey jobKey) throws UnableToInterruptJobException

Description copied from interface: Scheduler
Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which must be an implementor of the InterruptableJob interface.
If more than one instance of the identified job is currently executing, the InterruptableJob#interrupt() method will be called on each instance. However, there is a limitation that in the case that interrupt() on one instance throws an exception, all remaining instances (that have not yet been interrupted) will not have their interrupt() method called.

If you wish to interrupt a specific instance of a job (when more than one is executing) you can do so by calling Scheduler.getCurrentlyExecutingJobs() to obtain a handle to the job instance, and then invoke interrupt() on it yourself.

This method is not cluster aware. That is, it will only interrupt instances of the identified InterruptableJob currently executing in this Scheduler instance, not across the entire cluster.

**Specified by:**

interrupt in interface Scheduler

**Returns:**

true is at least one instance of the identified job was found and interrupted.

**Throws:**

UnableToInterruptJobException - if the job does not implement InterruptableJob, or there is an exception while interrupting the job.

**See Also:**

Scheduler.interrupt(JobKey)

---

**setJobFactory**

```java
public void setJobFactory(org.quartz.spi.JobFactory factory)
throws SchedulerException
```

**Description copied from interface: Scheduler**

Set the JobFactory that will be responsible for producing instances of Job classes.

JobFactories may be of use to those wishing to have their application produce Job instances via some special mechanism, such as to give the opportunity for dependency injection.
Specified by:
    setJobFactory in interface Scheduler

Throws:
    SchedulerException

See Also:
    Scheduler.setJobFactory(org.quartz.spi.JobFactory)

---

**toBoolean**

protected Boolean toBoolean(boolean bool)

---

Copyright 2001-2011, Terracotta, Inc.
public class RemoteScheduler

extends Object

implements Scheduler

An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via RMI.

Author:
James House

See Also:
Scheduler, QuartzScheduler, org.quartz.core.SchedulingContext

Field Summary

Fields inherited from interface org.quartz.Scheduler

DEFAULT_FAIL_OVER_GROUP, DEFAULT_GROUP, DEFAULT_RECOVERY_GROUP,
FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS,
FAILED_JOB_ORIGINAL_TRIGGER_GROUP,
FAILED_JOB_ORIGINAL_TRIGGER_NAME

Constructor Summary

RemoteScheduler (String schedId, String host, int port)

Construct a RemoteScheduler instance to proxy the given RemoteableQuartzScheduler instance, and with the given SchedulingContext.
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>addCalendar</strong> <em>(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><strong>addJob</strong> <em>(JobDetail jobDetail, boolean replace)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>checkExists</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>checkExists</strong> <em>(TriggerKey triggerKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void</td>
<td><strong>clear</strong> ()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>deleteCalendar</strong> <em>(String calName)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>deleteJob</strong> <em>(JobKey jobKey)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>deleteJobs</strong> <em>(List&lt;JobKey&gt; jobKeys)</em></td>
<td>Delete the identified Jobs from the Scheduler - and associated Triggers.</td>
</tr>
<tr>
<td>Calendar</td>
<td><strong>getCalendar</strong> <em>(String calName)</em></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td><strong>getCalendarNames</strong> ()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>SchedulerContext</td>
<td><strong>getContext</strong> ()</td>
<td>Returns the SchedulerContext of the Scheduler.</td>
</tr>
<tr>
<td>List</td>
<td><strong>getCurrentlyExecutingJobs</strong> ()</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>JobDetail.getJobDetail(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>List&lt;String&gt;.getJobGroupNames()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>Set&lt;JobKey&gt;.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>ListenerManager.getListenerManager()</td>
<td>Get a reference to the scheduler's ListenerManager which listeners may be registered.</td>
<td></td>
</tr>
<tr>
<td>SchedulerMetaData.getMetaData()</td>
<td>Get a SchedulerMetaData object describing the set capabilities of the scheduler instance.</td>
<td></td>
</tr>
<tr>
<td>Set.getPausedTriggerGroups()</td>
<td>Get the names of all Trigger groups that are paused.</td>
<td></td>
</tr>
<tr>
<td>protected RemotableQuartzScheduler.getRemoteScheduler()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>String.getSchedulerInstanceId()</td>
<td>Returns the instance Id of the Scheduler.</td>
<td></td>
</tr>
<tr>
<td>String.getSchedulerName()</td>
<td>Returns the name of the Scheduler.</td>
<td></td>
</tr>
<tr>
<td>Trigger.getTrigger(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>List&lt;String&gt;.getTriggerGroupNames()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td>List&lt;? extends Trigger&gt;.getTriggersOfJob(JobKey jobKey)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>Trigger.TriggerState.getTriggerState(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>boolean interrupt(JobKey jobKey)</code></td>
<td>Request the interruption, within this Scheduler instance, of the currently executing instances of the identified Job, which implementor of the InterruptableJob interface.</td>
<td></td>
</tr>
<tr>
<td><code>protected SchedulerException invalidateHandleCreateException(String msg, Exception cause)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>boolean isInStandbyMode()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>boolean isShutdown()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>boolean isStarted()</code></td>
<td>Whether the scheduler has been started.</td>
<td></td>
</tr>
<tr>
<td><code>void pauseAll()</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>void pauseJob(JobKey jobKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>void pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>void pauseTrigger(TriggerKey triggerKey)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>void pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
<tr>
<td><code>Date rescheduleJob(TriggerKey triggerKey, Trigger new</code></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
<td></td>
</tr>
</tbody>
</table>
void resumeAll()
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void resumeJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void resumeJobs(GroupMatcher<JobKey> matcher)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void resumeTrigger(TriggerKey triggerKey)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void resumeTriggers(GroupMatcher<TriggerKey> matcher)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

Date scheduleJob(JobDetail jobDetail, Trigger trigger)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

Date scheduleJob(Trigger trigger)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void scheduleJobs(Map<JobDetail, List<Trigger>> triggerMap, boolean replace)
    Schedule all of the given jobs with the related set of
    triggers.

void setJobFactory(org.quartz.spi.JobFactory factory)
    Set the JobFactory that will be responsible for producing
    instances of Job classes.

void shutdown()
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void shutdown(boolean waitForJobsToComplete)
    Calls the equivalent method on the 'proxied'
    QuartzScheduler.

void standby()
void start()
    Calls the equivalent method on the 'proxied'
QuartzScheduler.

void startDelayed(int seconds)
    Calls the equivalent method on the 'proxied'
QuartzScheduler.

void triggerJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied'
QuartzScheduler.

void triggerJob(JobKey jobKey, JobDataMap data)
    Calls the equivalent method on the 'proxied'
QuartzScheduler.

boolean unscheduleJob(TriggerKey triggerKey)
    Calls the equivalent method on the 'proxied'
QuartzScheduler.

boolean unscheduleJobs(List<TriggerKey> triggerKeys)
    Remove all of the indicated Triggers from the sche

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait

Constructor Detail

RemoteScheduler

public RemoteScheduler(String schedId,
            String host,
            int port)

    Construct a RemoteScheduler instance to proxy the given
RemoteableQuartzScheduler instance, and with the given
SchedulingContext.
Method Detail

getRemoteScheduler

protected RemotableQuartzScheduler getRemoteScheduler() throws SchedulerException

Throws:
    SchedulerException

invalidateHandleCreateException

protected SchedulerException invalidateHandleCreateException(String msg, Exception)

getSchedulerName

public String getSchedulerName() throws SchedulerException

Returns the name of the Scheduler.

Specified by:
    getSchedulerName in interface Scheduler

Throws:
    SchedulerException

getSchedulerInstanceId

public String getSchedulerInstanceId() throws SchedulerException

Returns the instance Id of the Scheduler.

Specified by:
    getSchedulerInstanceId in interface Scheduler

Throws:
SchedulerException

getMetaData

public SchedulerMetaData getMetaData() throws SchedulerException

Description copied from interface: Scheduler
Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.

Note that the data returned is an 'instantaneous' snap-shot, and that as soon as it's returned, the meta data values may be different.

Specified by:
getMetaData in interface Scheduler

Throws:
SchedulerException

getContext

public SchedulerContext getContext() throws SchedulerException

Returns the SchedulerContext of the Scheduler.

Specified by:
getContext in interface Scheduler

Throws:
SchedulerException

start

public void start() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.
Specified by:
   start in interface Scheduler

Throws:
   SchedulerException - if shutdown() has been called, or there is an error within the Scheduler.

See Also:
   Scheduler.startDelayed(int), Scheduler.standby(), Scheduler.shutdown()

startDelayed

public void startDelayed(int seconds)
   throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
   startDelayed in interface Scheduler

Throws:
   SchedulerException - if shutdown() has been called, or there is an error within the Scheduler.

See Also:
   Scheduler.start(), Scheduler.standby(), Scheduler.shutdown()

standby

public void standby()
   throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
   standby in interface Scheduler

Throws:
   SchedulerException

See Also:
   Scheduler.start(), Scheduler.pauseAll()
isStarted

public boolean isStarted() throws SchedulerException

Whether the scheduler has been started.

Note: This only reflects whether start() has ever been called on this Scheduler, so it will return true even if the Scheduler is currently in standby mode or has been since shutdown.

Specified by:
    isStarted in interface Scheduler

Throws:
    SchedulerException

See Also:
    start(), isShutdown(), isInStandbyMode()

--

isInStandbyMode

public boolean isInStandbyMode() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    isInStandbyMode in interface Scheduler

Throws:
    SchedulerException

See Also:
    Scheduler.standby(), Scheduler.start()

--

shutdown

public void shutdown() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.
Specified by:
shutdoun in interface Scheduler

Throws:
SchedulerException

See Also:
Scheduler.shutdown(boolean)

shutdoun

public void shutdoun(boolean waitForJobsToComplete)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
shutdoun in interface Scheduler

Parameters:
waitForJobsToComplete - if true the scheduler will not allow this
method to return until all currently executing jobs have completed.

Throws:
SchedulerException

See Also:
Scheduler.shutdown()

isShutdoun

public boolean isShutdoun()
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
isShutdoun in interface Scheduler

Throws:
SchedulerException

getCurrentlyExecutingJobs
public List getCurrentlyExecutingJobs() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getCurrentlyExecutingJobs in interface Scheduler

Throws:
SchedulerException

See Also:
JobExecutionContext

scheduleJob

public Date scheduleJob(JobDetail jobDetail, Trigger trigger) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
scheduleJob in interface Scheduler

Throws:
SchedulerException - if the Job or Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

scheduleJob

public Date scheduleJob(Trigger trigger) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
scheduleJob in interface Scheduler

Throws:
SchedulerException - if the indicated Job does not exist, or the Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.
addJob

public void addJob(JobDetail jobDetail,
                   boolean replace)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    addJob in interface Scheduler

Throws:
    SchedulerException - if there is an internal Scheduler error, or if the
Job is not durable, or a Job with the same name already exists, and
replace is false.

deleteJobs

public boolean deleteJobs(List<JobKey> jobKeys)
    throws SchedulerException

Description copied from interface: Scheduler
Delete the identified Jobs from the Scheduler - and any associated
Triggers.

Note that while this bulk operation is likely more efficient than invoking
deleteJob(JobKey jobKey) several times, it may have the adverse affect
of holding data locks for a single long duration of time (rather than lots of
small durations of time).

Specified by:
    deleteJobs in interface Scheduler

Returns:
    true if all of the Jobs were found and deleted, false if one or more were
not deleted.

Throws:
    SchedulerException - if there is an internal Scheduler error.
scheduleJobs

public void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace) throws SchedulerException

Description copied from interface: Scheduler
Schedule all of the given jobs with the related set of triggers.

If any of the given jobs or triggers already exist (or more specifically, if the keys are not unique) and the replace parameter is not set to true then an exception will be thrown.

Specified by:
        scheduleJobs in interface Scheduler

Throws:
        ObjectAlreadyExistsException - if the job/trigger keys are not unique and the replace flag is not set to true.
        SchedulerException

unscheduleJobs

public boolean unscheduleJobs(List<TriggerKey> triggerKeys) throws SchedulerException

Description copied from interface: Scheduler
Remove all of the indicated Triggers from the scheduler.

If the related job does not have any other triggers, and the job is not durable, then the job will also be deleted.

Note that while this bulk operation is likely more efficient than invoking unscheduleJob(TriggerKey triggerKey) several times, it may have the adverse affect of holding data locks for a single long duration of time (rather than lots of small durations of time).

Specified by:
        unscheduleJobs in interface Scheduler

Throws:
deleteJob

public boolean deleteJob(JobKey jobKey)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    deleteJob in interface Scheduler
Returns:
    true if the Job was found and deleted.
Throws:
    SchedulerException - if there is an internal Scheduler error.

unscheduleJob

public boolean unscheduleJob(TriggerKey triggerKey)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    unscheduleJob in interface Scheduler
Throws:
    SchedulerException

rescheduleJob

public Date rescheduleJob(TriggerKey triggerKey,
                          Trigger newTrigger)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
rescheduleJob in interface Scheduler

Parameters:
triggerKey - identity of the trigger to replace
newTrigger - The new Trigger to be stored.

Returns:
null if a Trigger with the given name & group was not found and
removed from the store, otherwise the first fire time of the newly
scheduled trigger.

Throws:
SchedulerException

triggerJob

public void triggerJob(JobKey jobKey)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    triggerJob in interface Scheduler

Throws:
    SchedulerException

triggerJob

public void triggerJob(JobKey jobKey, JobDataMap data)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
    triggerJob in interface Scheduler
data - the (possibly null) JobDataMap to be associated with the
trigger that fires the job immediately.

Throws:
    SchedulerException
pauseTrigger

```java
public void pauseTrigger(TriggerKey triggerKey)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
- `pauseTrigger` in interface `Scheduler`

**Throws:**
- `SchedulerException`

**See Also:**
- `Scheduler.resumeTrigger(TriggerKey)`

---------

pauseTriggers

```java
public void pauseTriggers(GroupMatcher<TriggerKey> matcher)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
- `pauseTriggers` in interface `Scheduler`

**Parameters:**
- `matcher` - The matcher to evaluate against know groups

**Throws:**
- `SchedulerException`

**See Also:**
- `Scheduler.resumeTriggers(org.quartz.impl.matchers.GroupMatcher)`

---------

pauseJob

```java
public void pauseJob(JobKey jobKey)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
- `pauseJob` in interface `Scheduler`
Throws:
SchedulerException

See Also:
Scheduler.resumeJob(JobKey)

---

### pauseJobs

```java
class org.quartz.SchedulerProxy

public void pauseJobs(GroupMatcher<JobKey> matcher)
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
pauseJobs in interface Scheduler

**Parameters:**
- matcher - The matcher to evaluate against know groups

**Throws:**
- SchedulerException - On error

**See Also:**
Scheduler.resumeJobs(org.quartz.impl.matchers.GroupMatcher)

---

### resumeTrigger

```java
class org.quartz.SchedulerProxy

public void resumeTrigger(TriggerKey triggerKey)
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
resumeTrigger in interface Scheduler

**Throws:**
- SchedulerException

**See Also:**
Scheduler.pauseTrigger(TriggerKey)

---

### resumeTriggers

```java
class org.quartz.SchedulerProxy

public void resumeTriggers()
throws SchedulerException
```

---
public void `resumeTriggers`(GroupMatcher<TriggerKey> matcher) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
`resumeTriggers` in interface Scheduler

**Parameters:**
matcher - The matcher to evaluate against known paused groups

**Throws:**
SchedulerException - On error

**See Also:**
Scheduler.pauseTriggers(org.quartz.impl.matchers.GroupMatcher)

---

**resumeJob**

public void `resumeJob`(JobKey jobKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
`resumeJob` in interface Scheduler

**Throws:**
SchedulerException

**See Also:**
Scheduler.pauseJob(JobKey)

---

**resumeJobs**

public void `resumeJobs`(GroupMatcher<JobKey> matcher) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
`resumeJobs` in interface Scheduler

**Parameters:**
matcher - The matcher to evaluate against known paused groups
Throws:
- `SchedulerException` - On error

See Also:
- `Scheduler.pauseJobs(GroupMatcher)`

---

**pauseAll**

```java
public void pauseAll() throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
- `pauseAll` in interface `Scheduler`

**Throws:**
- `SchedulerException`

**See Also:**
- `Scheduler.resumeAll()`,
- `Scheduler.pauseTriggers(org.quartz.impl.matchers.GroupMatcher)`
- `Scheduler.standby()`

---

**resumeAll**

```java
public void resumeAll() throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
- `resumeAll` in interface `Scheduler`

**Throws:**
- `SchedulerException`

**See Also:**
- `Scheduler.pauseAll()`

---

**getJobGroupNames**
public List<String> getJobGroupNames() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getJobGroupNames in interface Scheduler

Throws:
SchedulerException

---

getJobKeys

public Set<JobKey> getJobKeys(GroupMatcher<JobKey> matcher) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getJobKeys in interface Scheduler

Parameters:
matcher - Matcher to evaluate against known groups

Returns:
Set of all keys matching

Throws:
SchedulerException - On error

---

getTriggersOfJob

public List<? extends Trigger> getTriggersOfJob(JobKey jobKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getTriggersOfJob in interface Scheduler

Throws:
SchedulerException
getTriggerGroupNames

public List<String> getTriggerGroupNames() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getTriggerGroupNames in interface QuartzScheduler

Throws:
SchedulerException

getTriggerKeys

public Set<TriggerKey> getTriggerKeys(GroupMatcher<TriggerKey> matcher) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getTriggerKeys in interface QuartzScheduler

Parameters:
matcher - Matcher to evaluate against known groups

Returns:
List of all keys matching

Throws:
SchedulerException - On error

getJobDetail

public JobDetail getJobDetail(JobKey jobKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getJobDetail in interface QuartzScheduler

Throws:
checkExists

```java
public boolean checkExists(JobKey jobKey)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
checkExists in interface Scheduler

**Parameters:**
jobKey - the identifier to check for

**Returns:**
true if a Job exists with the given identifier

**Throws:**
SchedulerException

checkExists

```java
public boolean checkExists(TriggerKey triggerKey)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
checkExists in interface Scheduler

**Parameters:**
triggerKey - the identifier to check for

**Returns:**
true if a Trigger exists with the given identifier

**Throws:**
SchedulerException

clear

```java
public void clear()
```
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by: clear in interface Scheduler

Throws: SchedulerException

---

getTrigger

public Trigger getTrigger(TriggerKey triggerKey)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by: getTrigger in interface Scheduler

Throws: SchedulerException

---

getTriggerState

public Trigger.TriggerState getTriggerState(TriggerKey triggerKey)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by: getTriggerState in interface Scheduler

Throws: SchedulerException

See Also: Trigger.TriggerState

---

addCalendar

public void addCalendar(String calName,
addCalendar

```java
public boolean addCalendar(Calendar calendar, boolean replace, boolean updateTriggers)
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
```
addCalendar in interface Scheduler
```

updateTriggers - whether or not to update existing triggers that referenced the already existing calendar so that they are 'correct' based on the new trigger.

**Throws:**
```
SchedulerException - if there is an internal Scheduler error, or a Calendar with the same name already exists, and replace is false.
```

---

deleteCalendar

```java
public boolean deleteCalendar(String calName)
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
```
deleteCalendar in interface Scheduler
```

**Returns:**
true if the Calendar was found and deleted.

**Throws:**
```
SchedulerException - if there is an internal Scheduler error.
```

---

getCalendar

```java
public Calendar getCalendar(String calName)
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
**getCalendar** in interface **Scheduler**

**Throws:**

**SchedulerException**

---

**getCalendarNames**

```java
public List<String> getCalendarNames()
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**

**getCalendarNames** in interface **Scheduler**

**Throws:**

**SchedulerException**

---

**getPausedTriggerGroups**

```java
public Set getPausedTriggerGroups()
throws SchedulerException
```

**Description copied from interface:** **Scheduler**

Get the names of all Trigger groups that are paused.

**Specified by:**

**getPausedTriggerGroups** in interface **Scheduler**

**Throws:**

**SchedulerException**

**See Also:**

**Scheduler.getPausedTriggerGroups()**

---

**getListenerManager**

```java
public ListenerManager getListenerManager()
throws SchedulerException
```

**Description copied from interface:** **Scheduler**

Get a reference to the scheduler's ListenerManager, through which
listeners may be registered.

**Specified by:**
`getListenerManager` in interface `Scheduler`

**Returns:**
the scheduler's `ListenerManager`

**Throws:**
`SchedulerException` - if the scheduler is not local

**See Also:**
`ListenerManager`, `JobListener`, `TriggerListener`, `SchedulerListener`

---

**interrupt**

```java
public boolean interrupt(JobKey jobKey) throws UnableToInterruptJobException
```

**Description copied from interface: `Scheduler`**
Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which must be an implementor of the `InterruptableJob` interface.

If more than one instance of the identified job is currently executing, the `InterruptableJob#interrupt()` method will be called on each instance. However, there is a limitation that in the case that `interrupt()` on one instances throws an exception, all remaining instances (that have not yet been interrupted) will not have their `interrupt()` method called.

If you wish to interrupt a specific instance of a job (when more than one is executing) you can do so by calling `Scheduler.getCurrentlyExecutingJobs()` to obtain a handle to the job instance, and then invoke `interrupt()` on it yourself.

This method is not cluster aware. That is, it will only interrupt instances of the identified `InterruptableJob` currently executing in this Scheduler instance, not across the entire cluster.

**Specified by:**
`interrupt` in interface `Scheduler`
Returns:
true is at least one instance of the identified job was found and interrupted.

Throws:
**UnableToInterruptJobException** - if the job does not implement **InterruptableJob**, or there is an exception while interrupting the job.

See Also:
**Scheduler.interrupt(JobKey)**

---

**setJobFactory**

public void **setJobFactory**(org.quartz.spi.JobFactory factory)
throws **SchedulerException**

Description copied from interface: **Scheduler**
Set the JobFactory that will be responsible for producing instances of Job classes.

JobFactories may be of use to those wishing to have their application produce Job instances via some special mechanism, such as to give the opportunity for dependency injection.

Specified by: **setJobFactory** in interface **Scheduler**

Throws: **SchedulerException**

See Also: **Scheduler.setJobFactory(org.quartz.spi.JobFactory)**
public class SchedulerRepository
extends Object

Holds references to Scheduler instances - ensuring uniqueness, and preventing garbage collection, and allowing 'global' lookups - all within a ClassLoader space.

Author:
James House

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void bind(Scheduler sched)</td>
<td></td>
</tr>
<tr>
<td>static SchedulerRepository getInstance()</td>
<td></td>
</tr>
<tr>
<td>Scheduler lookup(String schedName)</td>
<td></td>
</tr>
<tr>
<td>Collection&lt;Scheduler&gt; lookupAll()</td>
<td></td>
</tr>
<tr>
<td>boolean remove(String schedName)</td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Method Detail
getInstance

public static SchedulerRepository getInstance()

bind

public void bind(Scheduler sched)
    throws SchedulerException

    Throws:
        SchedulerException

remove

public boolean remove(String schedName)

lookup

public Scheduler lookup(String schedName)

lookupAll

public Collection<Scheduler> lookupAll()
org.quartz.impl  Class StdJobRunShellFactory

java.lang.Object
   ↓ org.quartz.impl.StdJobRunShellFactory

All Implemented Interfaces:
   JobRunShellFactory

public class StdJobRunShellFactory
   extends Object
   implements JobRunShellFactory

Responsible for creating the instances of JobRunShell to be used within the QuartzScheduler instance.

Author:
   James House

Constructor Summary

| StdJobRunShellFactory() |

Method Summary

| JobRunShell | createJobRunShell(org.quartz.spi.TriggerFiredBundle bndle) |
|            | Called by the QuartzSchedulerThread to obtain instances of JobRunShell. |

| void        | initialize(Scheduler scheduler) |
|            | Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContexts within it. |

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
**Constructor Detail**

StdJobRunShellFactory

public StdJobRunShellFactory()

**Method Detail**

**initialize**

public void initialize(Scheduler scheduler)

Initialize the factory, providing a handle to the Scheduler that should be made available within the JobRunShell and the JobExecutionContext within it.

**Specified by:**

initialize in interface JobRunShellFactory

**createJobRunShell**

public JobRunShell createJobRunShell(org.quartz.spi.TriggerFiredBundle bndle) throws SchedulerException

Called by the QuartzSchedulerThread to obtain instances of JobRunShell.

**Specified by:**

createJobRunShell in interface JobRunShellFactory

**Throws:**

SchedulerException
org.quartz.impl  Class StdScheduler

java.lang.Object
   ^org.quartz.impl.StdScheduler

All Implemented Interfaces:
   Scheduler

public class StdScheduler
extends Object
implements Scheduler

An implementation of the Scheduler interface that directly proxies all method calls to the equivalent call on a given QuartzScheduler instance.

Author:
   James House

See Also:
   Scheduler, QuartzScheduler, org.quartz.core.SchedulingContext

Field Summary

Fields inherited from interface org.quartz.Scheduler

DEFAULT_FAIL_OVER_GROUP, DEFAULT_GROUP, DEFAULT_RECOVERY_GROUP,
FAILED_JOB_ORIGINAL_TRIGGER_FIRETIME_IN_MILLISECONDS,
FAILED_JOB_ORIGINAL_TRIGGER_GROUP,
FAILED_JOB_ORIGINAL_TRIGGER_NAME

Constructor Summary

StdScheduler(QuartzScheduler sched)

Construct a StdScheduler instance to proxy the given QuartzScheduler instance, and with the given SchedulingContext.
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void addJob(JobDetail jobDetail, boolean replace)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean checkExists(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean checkExists(TriggerKey triggerKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>void clear()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean deleteCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean deleteJob(JobKey jobKey)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>boolean deleteJobs(List&lt;JobKey&gt; jobKeys)</td>
<td>Delete the identified Jobs from the Scheduler - and associated Triggers.</td>
</tr>
<tr>
<td>Calendar getCalendar(String calName)</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>List&lt;String&gt; getCalendarNames()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>SchedulerContext getContext()</td>
<td>Returns the SchedulerContext of the Scheduler.</td>
</tr>
<tr>
<td>List&lt;JobExecutionContext&gt; getCurrentlyExecutingJobs()</td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>getJobDetail(JobKey jobKey)</strong></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>getJobGroupNames()</strong></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</strong></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>getListenerManager()</strong></td>
<td>Get a reference to the scheduler's ListenerManager which listeners may be registered.</td>
</tr>
<tr>
<td><strong>getMetaData()</strong></td>
<td>Get a SchedulerMetaData object describing the settings capabilities of the scheduler instance.</td>
</tr>
<tr>
<td><strong>getPausedTriggerGroups()</strong></td>
<td>Get the names of all Trigger groups that are paused.</td>
</tr>
<tr>
<td><strong>getSchedulerInstanceId()</strong></td>
<td>Returns the instance Id of the Scheduler.</td>
</tr>
<tr>
<td><strong>getSchedulerName()</strong></td>
<td>Returns the name of the Scheduler.</td>
</tr>
<tr>
<td><strong>getTrigger(TriggerKey triggerKey)</strong></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>getTriggerGroupNames()</strong></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</strong></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
<tr>
<td><strong>getTriggersOfJob(JobKey jobKey)</strong></td>
<td>Calls the equivalent method on the 'proxied' QuartzScheduler.</td>
</tr>
</tbody>
</table>
- `getTriggerState(TriggerKey triggerKey)`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `interrupt(JobKey jobKey)`
  Request the interruption, within this Scheduler instance, of currently executing instances of the identified Job, which implementor of the InterruptableJob interface.

- `isInStandbyMode()`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `isShutdown()`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `isStarted()`
  Whether the scheduler has been started.

- `pauseAll()`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `pauseJob(JobKey jobKey)`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `pauseTrigger(TriggerKey triggerKey)`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `rescheduleJob(TriggerKey triggerKey, Trigger newTrigger)`
  Calls the equivalent method on the 'proxied' QuartzScheduler.

- `resumeAll()`
  Calls the equivalent method on the 'proxied' QuartzScheduler.
void resumeJob(JobKey jobKey)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void resumeJobs(GroupMatcher<JobKey> matcher)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void resumeTrigger(TriggerKey triggerKey)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void resumeTriggers(GroupMatcher<TriggerKey> matcher)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

Date scheduleJob(JobDetail jobDetail, Trigger trigger)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

Date scheduleJob(Trigger trigger)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace)
    Schedule all of the given jobs with the related set of triggers.

void setJobFactory(org.quartz.spi.JobFactory factory)
    Set the JobFactory that will be responsible for producing instances of Job classes.

void shutdown()
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void shutdown(boolean waitForJobsToComplete)
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void standby()
    Calls the equivalent method on the 'proxied' QuartzScheduler.

void start()
    Calls the equivalent method on the 'proxied' QuartzScheduler.
void **startDelayed** (int seconds)
Calls the equivalent method on the 'proxied'
QuartzScheduler.

void **triggerJob** (JobKey jobKey)
Calls the equivalent method on the 'proxied'
QuartzScheduler.

void **triggerJob** (JobKey jobKey, JobDataMap data)
Calls the equivalent method on the 'proxied'
QuartzScheduler.

boolean **unscheduleJob** (TriggerKey triggerKey)
Calls the equivalent method on the 'proxied'
QuartzScheduler.

boolean **unscheduleJobs** (List<TriggerKey> triggerKeys)
Remove all of the indicated Triggers from the sch

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Constructor Detail**

**StdScheduler**

public **StdScheduler** (QuartzScheduler sched)

Construct a StdScheduler instance to proxy the given QuartzScheduler instance, and with the given SchedulingContext.

**Method Detail**

**getSchedulerName**

public **String** getSchedulerName()

Returns the name of the Scheduler.
Specified by:
getSchedulerName in interface Scheduler

getSchedulerInstanceId

public String getSchedulerInstanceId()

Returns the instance Id of the Scheduler.

Specified by:
getSchedulerInstanceId in interface Scheduler

getMetaData

public SchedulerMetaData getMetaData()

Description copied from interface: Scheduler
Get a SchedulerMetaData object describing the settings and capabilities of the scheduler instance.

Note that the data returned is an 'instantaneous' snap-shot, and that as soon as it's returned, the meta data values may be different.

Specified by:
getMetaData in interface Scheduler

getContext

public SchedulerContext getContext() throws SchedulerException

Returns the SchedulerContext of the Scheduler.

Specified by:
getContext in interface Scheduler

Throws:
SchedulerException
**start**

```java
class:

public void start()
    throws SchedulerException

calls the equivalent method on the 'proxied' QuartzScheduler.

specified by:

- start in interface Scheduler

throws:

- SchedulerException - if shutdown() has been called, or there is an error within the Scheduler.

see also:

- Scheduler.startDelayed(int), Scheduler.standby(), Scheduler.shutdown()
```

**startDelayed**

```java
class:

public void startDelayed(int seconds)
    throws SchedulerException

calls the equivalent method on the 'proxied' QuartzScheduler.

specified by:

- startDelayed in interface Scheduler

throws:

- SchedulerException - if shutdown() has been called, or there is an error within the Scheduler.

see also:

- Scheduler.start(), Scheduler.standby(), Scheduler.shutdown()
```

**standby**

```java
class:

public void standby()

calls the equivalent method on the 'proxied' QuartzScheduler.
```
isStarted

public boolean isStarted()

Whether the scheduler has been started.

Note: This only reflects whether start() has ever been called on this Scheduler, so it will return true even if the Scheduler is currently in standby mode or has been since shutdown.

Specified by: isStarted in interface Scheduler
See Also: start(), isShutdown(), isInStandbyMode()
Specified by:

`shutdown` in interface `Scheduler`

See Also:

`Scheduler.shutdown(boolean)`

---

**shutdown**

```java
public void shutdown(boolean waitForJobsToComplete)
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:

`shutdown` in interface `Scheduler`

Parameters:

- `waitForJobsToComplete` - if true the scheduler will not allow this method to return until all currently executing jobs have completed.

See Also:

`Scheduler.shutdown()`

---

**isShutdown**

```java
public boolean isShutdown()
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:

`isShutdown` in interface `Scheduler`

---

**getCurrentlyExecutingJobs**

```java
public List&lt;JobExecutionContext&gt; getCurrentlyExecutingJobs()
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:

`getCurrentlyExecutingJobs` in interface `Scheduler`

See Also:
clear

```java
public void clear()
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
- ```clear``` in interface ```Scheduler```

**Throws:**
- ```SchedulerException```
Throws:

- **SchedulerException** - if the indicated Job does not exist, or the Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.

---

**addJob**

```java
class NameSpace
public void addJob(JobDetail jobDetail,
                    boolean replace)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
- `addJob` in interface `Scheduler`

**Throws:**

- **SchedulerException** - if there is an internal Scheduler error, or if the Job is not durable, or a Job with the same name already exists, and replace is false.

---

**deleteJobs**

```java
class NameSpace
public boolean deleteJobs(List<JobKey> jobKeys)
    throws SchedulerException
```

**Description copied from interface:** `Scheduler`
Delete the identified Jobs from the Scheduler - and any associated Triggers.

Note that while this bulk operation is likely more efficient than invoking `deleteJob(JobKey jobKey)` several times, it may have the adverse affect of holding data locks for a single long duration of time (rather than lots of small durations of time).

**Specified by:**
- `deleteJobs` in interface `Scheduler`

**Returns:**
true if all of the Jobs were found and deleted, false if one or more were
Throws: 
SchedulerException - if there is an internal Scheduler error.

---

**scheduleJobs**

```java
public void scheduleJobs(Map<JobDetail, List<Trigger>> triggersAndJobs,
                          boolean replace)
  throws SchedulerException
```

Description copied from interface: Scheduler
Schedule all of the given jobs with the related set of triggers.

If any of the given jobs or triggers already exist (or more specifically, if the keys are not unique) and the replace parameter is not set to true then an exception will be thrown.

Specified by: 
scheduleJobs in interface Scheduler

Throws: 
ObjectAlreadyExistsException - if the job(trigger keys are not unique and the replace flag is not set to true.
SchedulerException

---

**unscheduleJobs**

```java
public boolean unscheduleJobs(List<TriggerKey> triggerKeys)
  throws SchedulerException
```

Description copied from interface: Scheduler
Remove all of the indicated Triggers from the scheduler.

If the related job does not have any other triggers, and the job is not durable, then the job will also be deleted.

Note that while this bulk operation is likely more efficient than invoking unscheduleJob(TriggerKey triggerKey) several times, it may have the adverse affect of holding data locks for a single long duration of time.
(rather than lots of small durations of time).

**Specified by:**
unscheduleJobs in interface Scheduler

**Throws:**
SchedulerException

---

**deleteJob**

```java
public boolean deleteJob(JobKey jobKey)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
deleteJob in interface Scheduler

**Returns:**
true if the Job was found and deleted.

**Throws:**
SchedulerException - if there is an internal Scheduler error.

---

**unscheduleJob**

```java
public boolean unscheduleJob(TriggerKey triggerKey)
    throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
unscheduleJob in interface Scheduler

**Throws:**
SchedulerException

---

**rescheduleJob**

```java
public Date rescheduleJob(TriggerKey triggerKey,
    Trigger newTrigger)
```
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

 Specified by:
   rescheduleJob in interface Scheduler

 Parameters:
   triggerKey - identity of the trigger to replace
   newTrigger - The new Trigger to be stored.

 Returns:
   null if a Trigger with the given name & group was not found and
   removed from the store, otherwise the first fire time of the newly
   scheduled trigger.

 Throws:
   SchedulerException

triggerJob

public void triggerJob(JobKey jobKey)
   throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

 Specified by:
   triggerJob in interface Scheduler

 Throws:
   SchedulerException

triggerJob

public void triggerJob(JobKey jobKey, JobDataMap data)
   throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

 Specified by:
   triggerJob in interface Scheduler
data - the (possibly null) JobDataMap to be associated with the trigger that fires the job immediately.

Throws:
SchedulerException

-----------------------------

pauseTrigger

public void pauseTrigger(TriggerKey triggerKey)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
pauseTrigger in interface Scheduler

Throws:
SchedulerException

See Also:
Scheduler.resumeTrigger(TriggerKey)

-----------------------------

pauseTriggers

public void pauseTriggers(GroupMatcher<TriggerKey> matcher)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
pauseTriggers in interface Scheduler

Parameters:
matcher - The matcher to evaluate against know groups

Throws:
SchedulerException

See Also:
Scheduler.resumeTriggers(org.quartz.impl.matchers.GroupMatcher)

-----------------------------

pauseJob
public void pauseJob(JobKey jobKey)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
pauseJob in interface Scheduler

Throws:
SchedulerException

See Also:
Scheduler.resumeJob(JobKey)

public Set getPausedTriggerGroups()
    throws SchedulerException

Description copied from interface: Scheduler
Get the names of all Trigger groups that are paused.

Specified by:
getPausedTriggerGroups in interface Scheduler

Throws:
SchedulerException

See Also:
Scheduler.getPausedTriggerGroups()

public void pauseJobs(GroupMatcher<JobKey> matcher)
    throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
pauseJobs in interface Scheduler

Parameters:
matcher - The matcher to evaluate against know groups

Throws:
**resumeTrigger**

```java
public void resumeTrigger(TriggerKey triggerKey)
        throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
`resumeTrigger` in interface `Scheduler`

**Throws:**
`SchedulerException`

**See Also:**
`Scheduler.pauseTrigger(TriggerKey)`

---

**resumeTriggers**

```java
public void resumeTriggers(GroupName matcher)
        throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**
`resumeTriggers` in interface `Scheduler`

**Parameters:**
matcher - The matcher to evaluate against know paused groups

**Throws:**
`SchedulerException` - On error

**See Also:**
`Scheduler.pauseTriggers(org.quartz.impl.matchers.GroupMatcher)`

---

**resumeJob**

```java
public void resumeJob(JobKey jobKey)
```
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
   resumeJob in interface Scheduler

Throws:
   SchedulerException

See Also:
   Scheduler.pauseJob(JobKey)

---

resumeJobs

public void resumeJobs(GroupMatcher<JobKey> matcher)
   throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
   resumeJobs in interface Scheduler

Parameters:
   matcher - The matcher to evaluate against known paused groups

Throws:
   SchedulerException - On error

See Also:
   Scheduler.pauseJobs(GroupMatcher)

---

pauseAll

public void pauseAll()
   throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
   pauseAll in interface Scheduler

Throws:
   SchedulerException

See Also:
resumeAll

public void resumeAll() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
resumeAll in interface Scheduler

Throws:
SchedulerException

See Also:
Scheduler.pauseAll()

getJobGroupNames

public List<String> getJobGroupNames() throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getJobGroupNames in interface Scheduler

Throws:
SchedulerException

getTriggersOfJob

public List<? extends Trigger> getTriggersOfJob(JobKey jobKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getTriggersOfJob in interface Scheduler

Throws:
SchedulerException

---

getJobKeys

```java
public Set&lt;JobKey&gt; getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getJobKeys in interface Scheduler

Parameters:
matcher - Matcher to evaluate against known groups

Returns:
Set of all keys matching

Throws:
SchedulerException - On error

---

getTriggerGroupNames

```java
public List&lt;String&gt; getTriggerGroupNames()
throws SchedulerException
```

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getTriggerGroupNames in interface Scheduler

Throws:
SchedulerException

---

getTriggerKeys

```java
public Set&lt;TriggerKey&gt; getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)
throws SchedulerException
```
Calls the equivalent method on the 'proxied' QuartzScheduler.

**getTriggerKeys**

getTriggerKeys in interface Scheduler

**Specified by:**

getTriggerKeys in interface Scheduler

**Parameters:**

matcher - Matcher to evaluate against known groups

**Returns:**

List of all keys matching

**Throws:**

SchedulerException - On error

---

**getJobDetail**

public JobDetail getJobDetail(JobKey jobKey)

throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**

getJobDetail in interface Scheduler

**Throws:**

SchedulerException

---

**getTrigger**

public Trigger getTrigger(TriggerKey triggerKey)

throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

**Specified by:**

getTrigger in interface Scheduler

**Throws:**

SchedulerException

---

**getTriggerState**
public Trigger.TriggerState getTriggerState(TriggerKey triggerKey) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getTriggerState in interface Scheduler

Throws:
SchedulerException

See Also:
Trigger.TriggerState

addCalendar

public void addCalendar(String calName, Calendar calendar, boolean replace, boolean updateTriggers) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
addCalendar in interface Scheduler

updateTriggers - whether or not to update existing triggers that referenced the already existing calendar so that they are 'correct' based on the new trigger.

Throws:
SchedulerException - if there is an internal Scheduler error, or a Calendar with the same name already exists, and replace is false.

deleteCalendar

public boolean deleteCalendar(String calName) throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
deleteCalendar in interface Scheduler

Returns:
true if the Calendar was found and deleted.

Throws:
SchedulerException - if there is an internal Scheduler error.

----------------------------------

getCalendar

public Calendar getCalendar(String calName)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getCalendar in interface Scheduler

Throws:
SchedulerException

----------------------------------

getCalendarNames

public List<String> getCalendarNames()
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
getCalendarNames in interface Scheduler

Throws:
SchedulerException

----------------------------------

checkExists

public boolean checkExists(JobKey jobKey)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.
Specified by:
   checkExists in interface Scheduler

Parameters:
   jobKey - the identifier to check for

Returns:
   true if a Job exists with the given identifier

Throws:
   SchedulerException

---

checkExists

public boolean checkExists(TriggerKey triggerKey)
throws SchedulerException

Calls the equivalent method on the 'proxied' QuartzScheduler.

Specified by:
   checkExists in interface Scheduler

Parameters:
   triggerKey - the identifier to check for

Returns:
   true if a Trigger exists with the given identifier

Throws:
   SchedulerException

---

setJobFactory

public void setJobFactory(org.quartz.spi.JobFactory factory)
throws SchedulerException

Description copied from interface: Scheduler
Set the JobFactory that will be responsible for producing instances of Job classes.

JobFactories may be of use to those wishing to have their application produce Job instances via some special mechanism, such as to give the opportunity for dependency injection.
 Specified by:  
   `setJobFactory` in interface `Scheduler`

 Throws:  
   `SchedulerException`

 See Also:  
   `Scheduler.setJobFactory(org.quartz.spi.JobFactory)`

---

### getListenerManager

```java
public ListenerManager getListenerManager()
```

Throws: `SchedulerException` - if the scheduler is not local

Description copied from interface: `Scheduler`
Get a reference to the scheduler's `ListenerManager`, through which listeners may be registered.

Specified by:  
   `getListenerManager` in interface `Scheduler`

Returns:  
   the scheduler's `ListenerManager`

See Also:  
   `Scheduler.getListenerManager()`

---

### interrupt

```java
public boolean interrupt(JobKey jobKey)
```

Throws: `UnableToInterruptJobException`

Description copied from interface: `Scheduler`
Request the interruption, within this Scheduler instance, of all currently executing instances of the identified Job, which must be an implementor of the `InterruptableJob` interface.

If more than one instance of the identified job is currently executing, the `InterruptableJob#interrupt()` method will be called on each instance. However, there is a limitation that in the case that `interrupt()` on one
instances throws an exception, all remaining instances (that have not yet been interrupted) will not have their `interrupt()` method called.

If you wish to interrupt a specific instance of a job (when more than one is executing) you can do so by calling `Scheduler.getCurrentlyExecutingJobs()` to obtain a handle to the job instance, and then invoke `interrupt()` on it yourself.

This method is not cluster aware. That is, it will only interrupt instances of the identified `InterruptableJob` currently executing in this `Scheduler` instance, not across the entire cluster.

**Specified by:**

`interrupt` in interface `Scheduler`

**Returns:**

true is at least one instance of the identified job was found and interrupted.

**Throws:**

`UnableToInterruptJobException` - if the job does not implement `InterruptableJob`, or there is an exception while interrupting the job.

**See Also:**

`InterruptableJob.interrupt()`, `Scheduler.getCurrentlyExecutingJobs()`
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS   NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

<table>
<thead>
<tr>
<th>CLASS</th>
<th>USE</th>
<th>TREE</th>
<th>DEPRECATED</th>
<th>INDEX</th>
<th>HELP</th>
</tr>
</thead>
</table>

FRAMES | NO FRAMES
**org.quartz.impl**  
**Class StdSchedulerFactory**

**java.lang.Object**  
└**org.quartz.impl.StdSchedulerFactory**

**All Implemented Interfaces:**  
SchedulerFactory

```java
public class StdSchedulerFactory
extends Object
implements SchedulerFactory
```

An implementation of SchedulerFactory that does all of its work of creating a QuartzScheduler instance based on the contents of a Properties file.

By default a properties file named "quartz.properties" is loaded from the 'current working directory'. If that fails, then the "quartz.properties" file located (as a resource) in the org/quartz package is loaded. If you wish to use a file other than these defaults, you must define the system property 'org.quartz.properties' to point to the file you want.

Alternatively, you can explicitly initialize the factory by calling one of the `initialize(xx)` methods before calling `getScheduler()`.

See the sample properties files that are distributed with Quartz for information about the various settings available within the file. Full configuration documentation can be found at http://www.quartz-scheduler.org/docs/configuration/index.html

Instances of the specified JobStore, ThreadPool, and other SPI classes will be created by name, and then any additional properties specified for them in the config file will be set on the instance by calling an equivalent 'set' method. For example if the properties file contains the property 'org.quartz.jobStore.myProp = 10' then after the JobStore class has been instantiated, the method 'setMyProp()' will be called on it. Type conversion to primitive Java types (int, long, float, double, boolean, and String) are performed before calling the property's setter method.
One property can reference another property's value by specifying a value following the convention of "$@other.property.name", for example, to reference the scheduler's instance name as the value for some other property, you would use "$@org.quartz.scheduler.instanceName".

Author:
James House, Anthony Eden, Mohammad Rezaei

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String AUTO_GENERATE_INSTANCE_ID</td>
</tr>
<tr>
<td>static String DEFAULT_INSTANCE_ID</td>
</tr>
<tr>
<td>static String PROP_CONNECTION_PROVIDER_CLASS</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_DRIVER</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_JNDI_ALWAYS_LOOKUP</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_JNDI_CREDENTIALS</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_JNDI_INITIAL</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_JNDI_PRINCIPAL</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_JNDI_PROVIDER</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_JNDI_URL</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_MAX_CONNECTIONS</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_PASSWORD</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_PREFIX</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_URL</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_USER</td>
</tr>
<tr>
<td>static String PROP_DATASOURCE_VALIDATION_QUERY</td>
</tr>
<tr>
<td>static String PROP_JOB_LISTENER_PREFIX</td>
</tr>
<tr>
<td>static String PROP_JOB_STORE_CLASS</td>
</tr>
<tr>
<td>static String PROP_JOB_STORE_LOCK_HANDLER_CLASS</td>
</tr>
<tr>
<td>static String PROP_JOB_STORE_LOCK_HANDLER_PREFIX</td>
</tr>
<tr>
<td>static String PROP_JOB_STORE_PREFIX</td>
</tr>
<tr>
<td>static String PROP_JOB_STORE_USE_PROP</td>
</tr>
<tr>
<td>static String PROP_LISTENER_CLASS</td>
</tr>
<tr>
<td>static String PROP_PLUGIN_CLASS</td>
</tr>
<tr>
<td>static String PROP_PLUGIN_PREFIX</td>
</tr>
<tr>
<td>static String PROP_SCHED_BATCH_TIME_WINDOW</td>
</tr>
<tr>
<td>static String PROP_SCHED_CLASS_LOAD_HELPER_CLASS</td>
</tr>
<tr>
<td>static String PROP_SCHED_CONTEXT_PREFIX</td>
</tr>
<tr>
<td>static String PROP_SCHED_DB_FAILURE_RETRY_INTERVAL</td>
</tr>
<tr>
<td>static String PROP_SCHED_IDLE_WAIT_TIME</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
</tbody>
</table>
### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdSchedulerFactory()</td>
<td>Create an uninitialized StdSchedulerFactory.</td>
</tr>
<tr>
<td>StdSchedulerFactory(Properties props)</td>
<td>Create a StdSchedulerFactory that has been initialized via initialize(Properties).</td>
</tr>
<tr>
<td>StdSchedulerFactory(String fileName)</td>
<td>Create a StdSchedulerFactory that has been initialized via initialize(String).</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection&lt;Scheduler&gt; getAllSchedulers()</td>
<td>Returns a handle to all known Schedulers (made by any StdSchedulerFactory instance.).</td>
</tr>
<tr>
<td>static Scheduler getDefaultScheduler()</td>
<td>Returns a handle to the default Scheduler, creating it if it does not yet exist.</td>
</tr>
<tr>
<td>org.slf4j.Logger getLog()</td>
<td></td>
</tr>
<tr>
<td>Scheduler getScheduler()</td>
<td>Returns a handle to the Scheduler produced by this factory.</td>
</tr>
<tr>
<td>Scheduler getScheduler(String schedName)</td>
<td>Returns a handle to the Scheduler with the given name, if it exists (if it has already been instantiated).</td>
</tr>
<tr>
<td>void initialize()</td>
<td>Initialize the SchedulerFactory with the contents of a Properties file and overriding System properties.</td>
</tr>
<tr>
<td>void initialize(InputStream propertiesStream)</td>
<td>Initialize the SchedulerFactory with the contents of the Properties file opened with the given InputStream.</td>
</tr>
<tr>
<td>void initialize(Properties props)</td>
<td>Initialize the SchedulerFactory with the contents</td>
</tr>
</tbody>
</table>
of the given Properties object.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>initialize</code></td>
<td>Initialize the SchedulerFactory with the contents of the Properties file with the given name.</td>
</tr>
<tr>
<td><code>instantiate</code></td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.**Object**
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`  

Field Detail

**PROPERTIES_FILE**

public static final `String` **PROPERTIES_FILE**

**See Also:**
- **Constant Field Values**

**PROP_SCHED_INSTANCE_NAME**

public static final `String` **PROP_SCHED_INSTANCE_NAME**

**See Also:**
- **Constant Field Values**

**PROP_SCHED_INSTANCE_ID**

public static final `String` **PROP_SCHED_INSTANCE_ID**

**See Also:**
- **Constant Field Values**
PROP_SCHED_INSTANCE_ID_GENERATOR_PREFIX

public static final String PROP_SCHED_INSTANCE_ID_GENERATOR_PREFIX

See Also:
   Constant Field Values

PROP_SCHED_INSTANCE_ID_GENERATOR_CLASS

public static final String PROP_SCHED_INSTANCE_ID_GENERATOR_CLASS

See Also:
   Constant Field Values

PROP_SCHED_THREAD_NAME

public static final String PROP_SCHED_THREAD_NAME

See Also:
   Constant Field Values

PROP_SCHED_SKIP_UPDATE_CHECK

public static final String PROP_SCHED_SKIP_UPDATE_CHECK

See Also:
   Constant Field Values

PROP_SCHED_BATCH_TIME_WINDOW

public static final String PROP_SCHED_BATCH_TIME_WINDOW

See Also:
   Constant Field Values
PROP_SCHED_MAX_BATCH_SIZE

public static final String PROP_SCHED_MAX_BATCH_SIZE

See Also:
Constant Field Values

PROP_SCHED_JMX_EXPORT

public static final String PROP_SCHED_JMX_EXPORT

See Also:
Constant Field Values

PROP_SCHED_JMX_OBJECT_NAME

public static final String PROP_SCHED_JMX_OBJECT_NAME

See Also:
Constant Field Values

PROP_SCHED_JMX_PROXY

public static final String PROP_SCHED_JMX_PROXY

See Also:
Constant Field Values

PROP_SCHED_JMX_PROXY_CLASS

public static final String PROP_SCHED_JMX_PROXY_CLASS

See Also:
Constant Field Values
PROP_SCHED_RMI_EXPORT

public static final String PROP_SCHED_RMI_EXPORT

See Also:
   Constant Field Values

PROP_SCHED_RMI_PROXY

public static final String PROP_SCHED_RMI_PROXY

See Also:
   Constant Field Values

PROP_SCHED_RMI_HOST

public static final String PROP_SCHED_RMI_HOST

See Also:
   Constant Field Values

PROP_SCHED_RMI_PORT

public static final String PROP_SCHED_RMI_PORT

See Also:
   Constant Field Values

PROP_SCHED_RMI_SERVER_PORT

public static final String PROP_SCHED_RMI_SERVER_PORT

See Also:
   Constant Field Values
PROP_SCHED_RMI_CREATE_REGISTRY
public static final String PROP_SCHED_RMI_CREATE_REGISTRY

See Also:
Constant Field Values

PROP_SCHED_RMI_BIND_NAME
public static final String PROP_SCHED_RMI_BIND_NAME

See Also:
Constant Field Values

PROP_SCHED_WRAP_JOB_IN_USER_TX
public static final String PROP_SCHED_WRAP_JOB_IN_USER_TX

See Also:
Constant Field Values

PROP_SCHED_USER_TX_URL
public static final String PROP_SCHED_USER_TX_URL

See Also:
Constant Field Values

PROP_SCHED_IDLE_WAIT_TIME
public static final String PROP_SCHED_IDLE_WAIT_TIME

See Also:
Constant Field Values
PROP_SCHED_DB_FAILURE_RETRY_INTERVAL
public static final String PROP_SCHED_DB_FAILURE_RETRY_INTERVAL

See Also:
Constant Field Values

PROP_SCHED_MAKE_SCHEDULER_THREAD_DAEMON
public static final String PROP_SCHED_MAKE_SCHEDULER_THREAD_DAEMON

See Also:
Constant Field Values

PROP_SCHED_SCHEDULER_THREADS_INHERIT_CONTEXT_CLASS_LOADER_OF_INITIALIZING_THREAD
public static final String PROP_SCHED_SCHEDULER_THREADS_INHERIT_CONTEXT_CLASS_LOADER_OF_INITIALIZING_THREAD

See Also:
Constant Field Values

PROP_SCHED_CLASS_LOAD_HELPER_CLASS
public static final String PROP_SCHED_CLASS_LOAD_HELPER_CLASS

See Also:
Constant Field Values

PROP_SCHED_JOB_FACTORY_CLASS
public static final String PROP_SCHED_JOB_FACTORY_CLASS

See Also:
Constant Field Values
PROP_SCHED_JOB_FACTORY_PREFIX

public static final String PROP_SCHED_JOB_FACTORY_PREFIX

See Also:
Constant Field Values

PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN

public static final String PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN

See Also:
Constant Field Values

PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN_WITH_WAIT

public static final String PROP_SCHED_INTERRUPT_JOBS_ON_SHUTDOWN_WITH_WAIT

See Also:
Constant Field Values

PROP_SCHED_CONTEXT_PREFIX

public static final String PROP_SCHED_CONTEXT_PREFIX

See Also:
Constant Field Values

PROP THREAD_POOL_PREFIX

public static final String PROP THREAD_POOL_PREFIX

See Also:
Constant Field Values
PROP_THREAD_POOL_CLASS

public static final String PROP_THREAD_POOL_CLASS

See Also:
   Constant Field Values

PROP_JOB_STORE_PREFIX

public static final String PROP_JOB_STORE_PREFIX

See Also:
   Constant Field Values

PROP_JOB_STORE_LOCK_HANDLER_PREFIX

public static final String PROP_JOB_STORE_LOCK_HANDLER_PREFIX

See Also:
   Constant Field Values

PROP_JOB_STORE_LOCK_HANDLER_CLASS

public static final String PROP_JOB_STORE_LOCK_HANDLER_CLASS

See Also:
   Constant Field Values

PROP_TABLE_PREFIX

public static final String PROP_TABLE_PREFIX

See Also:
   Constant Field Values
PROP_SCHED_NAME

public static final String PROP_SCHED_NAME

See Also:
Constant Field Values

PROP_JOB_STORE_CLASS

public static final String PROP_JOB_STORE_CLASS

See Also:
Constant Field Values

PROP_JOB_STORE_USE_PROP

public static final String PROP_JOB_STORE_USE_PROP

See Also:
Constant Field Values

PROP_DATASOURCE_PREFIX

public static final String PROP_DATASOURCE_PREFIX

See Also:
Constant Field Values

PROP_CONNECTION_PROVIDER_CLASS

public static final String PROP_CONNECTION_PROVIDER_CLASS

See Also:
Constant Field Values
PROP_DATASOURCE_DRIVER

public static final String PROP_DATASOURCE_DRIVER

See Also:
   Constant Field Values

PROP_DATASOURCE_URL

public static final String PROP_DATASOURCE_URL

See Also:
   Constant Field Values

PROP_DATASOURCE_USER

public static final String PROP_DATASOURCE_USER

See Also:
   Constant Field Values

PROP_DATASOURCE_PASSWORD

public static final String PROP_DATASOURCE_PASSWORD

See Also:
   Constant Field Values

PROP_DATASOURCE_MAX_CONNECTIONS

public static final String PROP_DATASOURCE_MAX_CONNECTIONS

See Also:
   Constant Field Values
PROP_DATASOURCE_VALIDATION_QUERY

public static final String PROP_DATASOURCE_VALIDATION_QUERY

See Also:
  Constant Field Values

PROP_DATASOURCE_JNDI_URL

public static final String PROP_DATASOURCE_JNDI_URL

See Also:
  Constant Field Values

PROP_DATASOURCE_JNDI_ALWAYS_LOOKUP

public static final String PROP_DATASOURCE_JNDI_ALWAYS_LOOKUP

See Also:
  Constant Field Values

PROP_DATASOURCE_JNDI_INITIAL

public static final String PROP_DATASOURCE_JNDI_INITIAL

See Also:
  Constant Field Values

PROP_DATASOURCE_JNDI_PROVIDER

public static final String PROP_DATASOURCE_JNDI_PROVIDER

See Also:
  Constant Field Values
PROP_DATASOURCE_JNDI_PRINCIPAL

public static final String PROP_DATASOURCE_JNDI_PRINCIPAL

See Also:

Constant Field Values

PROP_DATASOURCE_JNDI_CREDENTIALS

public static final String PROP_DATASOURCE_JNDI_CREDENTIALS

See Also:

Constant Field Values

PROP_PLUGIN_PREFIX

public static final String PROP_PLUGIN_PREFIX

See Also:

Constant Field Values

PROP_PLUGIN_CLASS

public static final String PROP_PLUGIN_CLASS

See Also:

Constant Field Values

PROP_JOB_LISTENER_PREFIX

public static final String PROP_JOB_LISTENER_PREFIX

See Also:

Constant Field Values
PROP_TRIGGER_LISTENER_PREFIX

public static final String PROP_TRIGGER_LISTENER_PREFIX

See Also:
  Constant Field Values

PROP_LISTENER_CLASS

public static final String PROP_LISTENER_CLASS

See Also:
  Constant Field Values

DEFAULT_INSTANCE_ID

public static final String DEFAULT_INSTANCE_ID

See Also:
  Constant Field Values

AUTO_GENERATE_INSTANCE_ID

public static final String AUTO_GENERATE_INSTANCE_ID

See Also:
  Constant Field Values

SYSTEM_PROPERTY_AS_INSTANCE_ID

public static final String SYSTEM_PROPERTY_AS_INSTANCE_ID

See Also:
  Constant Field Values
Constructor Detail

StdSchedulerFactory

public StdSchedulerFactory()

Create an uninitialized StdSchedulerFactory.

Throws:

SchedulerException

See Also:
initialize(Properties)

StdSchedulerFactory

public StdSchedulerFactory(Properties props)
    throws SchedulerException

Create a StdSchedulerFactory that has been initialized via initialize(Properties).

Throws:

SchedulerException

See Also:
initialize(Properties)

StdSchedulerFactory

public StdSchedulerFactory(String fileName)
    throws SchedulerException

Create a StdSchedulerFactory that has been initialized via initialize(String).

Throws:

SchedulerException

See Also:
initialize(String)

Method Detail

getLog
public org.slf4j.Logger getLog()

### initialize

public void initialize() throws SchedulerException

Initialize the SchedulerFactory with the contents of a Properties file and overriding System properties.

By default a properties file named "quartz.properties" is loaded from the 'current working directory'. If that fails, then the "quartz.properties" file located (as a resource) in the org/quartz package is loaded. If you wish to use a file other than these defaults, you must define the system property 'org.quartz.properties' to point to the file you want.

System properties (environment variables, and -D definitions on the command-line when running the JVM) override any properties in the loaded file. For this reason, you may want to use a different initialize() method if your application security policy prohibits access to System.getProperties().

**Throws:**

SchedulerException

### initialize

public void initialize(String filename) throws SchedulerException

Initialize the SchedulerFactory with the contents of the Properties file with the given name.

**Throws:**

SchedulerException

### initialize
public void initialize(InputStream propertiesStream)
   throws SchedulerException

   Initialize the SchedulerFactory with the contents of the Properties file opened with the given InputStream.

   Throws:
   SchedulerException

initialize

public void initialize(Properties props)
   throws SchedulerException

   Initialize the SchedulerFactory with the contents of the given Properties object.

   Throws:
   SchedulerException

instantiate

protected Scheduler instantiate(QuartzSchedulerResources rsrcs, QuartzScheduler qs)

getscheduler

public Scheduler getscheduler()
   throws SchedulerException

   Returns a handle to the Scheduler produced by this factory.

   If one of the initialize methods has not be previously called, then the default (no-arg) initialize() method will be called by this method.

   Specified by:
   getscheduler in interface SchedulerFactory

   Throws:
SchedulerException - if there is a problem with the underlying Scheduler.

### getDefaultScheduler

```java
defaultScheduler() throws SchedulerException
```

Returns a handle to the default Scheduler, creating it if it does not yet exist.

**Throws:**  
SchedulerException

**See Also:**  
initialize()

### getScheduler

```java
getScheduler(String schedName) throws SchedulerException
```

Returns a handle to the Scheduler with the given name, if it exists (if it has already been instantiated).

**Specified by:**  
getScheduler in interface SchedulerFactory

**Throws:**  
SchedulerException

### getAllSchedulers

```java
getAllSchedulers() throws SchedulerException
```

Returns a handle to all known Schedulers (made by any StdSchedulerFactory instance.).

**Specified by:**
getAllSchedulers in interface SchedulerFactory

Throws:
SchedulerException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
public class AnnualCalendar
extends BaseCalendar
implements Calendar, Serializable

This implementation of the Calendar excludes a set of days of the year. You may use it to exclude bank holidays which are on the same date every year.

Author:
Juergen Donnerstag

See Also:
Calendar, BaseCalendar, Serialized Form
**AnnualCalendar**(*TimeZone* timeZone)

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td><strong>clone</strong>()</td>
</tr>
<tr>
<td><strong>ArrayList&lt;Calendar&gt;</strong></td>
<td><strong>getDaysExcluded</strong>()</td>
</tr>
<tr>
<td></td>
<td>Get the array which defines the exclude-value of each day of month</td>
</tr>
<tr>
<td><strong>long</strong></td>
<td><strong>getNextIncludedTime</strong>(<em>long</em> <em>timeStamp</em>)</td>
</tr>
<tr>
<td></td>
<td>Determine the next time (in milliseconds) that is 'included' by the Calendar</td>
</tr>
<tr>
<td><strong>boolean</strong></td>
<td><strong>isDayExcluded</strong>(<em>Calendar</em> <em>day</em>)</td>
</tr>
<tr>
<td></td>
<td>Return true, if day is defined to be excluded</td>
</tr>
<tr>
<td><strong>boolean</strong></td>
<td><strong>isTimeIncluded</strong>(<em>long</em> <em>timeStamp</em>)</td>
</tr>
<tr>
<td></td>
<td>Determine whether the given time (in milliseconds) is 'included' by the</td>
</tr>
<tr>
<td></td>
<td>Calendar</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>removeExcludedDay</strong>(<em>Calendar</em> <em>day</em>)</td>
</tr>
<tr>
<td></td>
<td>Remove the given day from the list of excluded days</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>setDayExcluded</strong>(<em>Calendar</em> <em>day</em>, <em>boolean</em> <em>exclude</em>)</td>
</tr>
<tr>
<td></td>
<td>Redefine a certain day to be excluded (true) or included (false).</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>setDaysExcluded</strong>(<em>ArrayList</em> <em>days</em>)</td>
</tr>
<tr>
<td></td>
<td>Redefine the list of days excluded</td>
</tr>
</tbody>
</table>

**Methods inherited from class** org.quartz.impl.calendar.**BaseCalendar**

createJavaCalendar, createJavaCalendar, getBaseCalendar, getDescription, getEndOfDayJavaCalendar, getStartOfDayJavaCalendar, getTimeZone, setBaseCalendar, setDescription, setTimeZone

**Methods inherited from class** java.lang.**Object**

equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Methods inherited from interface org.quartz.Calendar
getBaseCalendar, getDescription, setBaseCalendar, setDescription

Constructor Detail

AnnualCalendar

public AnnualCalendar()

AnnualCalendar

public AnnualCalendar(Calendar baseCalendar)

AnnualCalendar

public AnnualCalendar(TimeZone timeZone)

AnnualCalendar

public AnnualCalendar(Calendar baseCalendar, TimeZone timeZone)

Method Detail

clonen

public Object clone()

Specified by:
clonen in interface Calendar

Overrides:
clonen in class BaseCalendar
**getDaysExcluded**

public `ArrayList<Calendar>` `getDaysExcluded()`

Get the array which defines the exclude-value of each day of month

**isDayExcluded**

public boolean `isDayExcluded(Calendar day)`

Return true, if day is defined to be excluded.

**setDaysExcluded**

public void `setDaysExcluded(ArrayList days)`

Redefine the list of days excluded. The `ArrayList` should contain `java.util.Calendar` objects.

**setDayExcluded**

public void `setDayExcluded(Calendar day, boolean exclude)`

Redefine a certain day to be excluded (true) or included (false).

**removeExcludedDay**

public void `removeExcludedDay(Calendar day)`

Remove the given day from the list of excluded days

**Parameters:**

day -
isTimeIncluded

public boolean isTimeIncluded(long timeSstamp)

Determine whether the given time (in milliseconds) is 'included' by the Calendar.

Note that this Calendar is only has full-day precision.

Specified by:
   isTimeIncluded in interface Calendar
Overrides:
   isTimeIncluded in class BaseCalendar
See Also:
   Calendar.isTimeIncluded(long)

getNextIncludedTime

public long getNextIncludedTime(long timeSstamp)

Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time. Return the original value if timeSstamp is included. Return 0 if all days are excluded.

Note that this Calendar is only has full-day precision.

Specified by:
   getNextIncludedTime in interface Calendar
Overrides:
   getNextIncludedTime in class BaseCalendar
See Also:
   Calendar.getNextIncludedTime(long)

Copyright 2001-2011, Terracotta, Inc.
**Class BaseCalendar**

Extends `Object` and implements `Calendar`, `Serializable`, `Cloneable`.

This implementation of the Calendar may be used (you don't have to) as a base class for more sophisticated one's. It merely implements the base functionality required by each Calendar.

Regarded as base functionality is the treatment of base calendars. Base calendar allow you to chain (stack) as much calendars as you may need. For example to exclude weekends you may use WeeklyCalendar. In order to exclude holidays as well you may define a WeeklyCalendar instance to be the base calendar for HolidayCalendar instance.

**Author:**
Juergen Donnerstag, James House

**See Also:**
`Calendar`, `Serialized Form`

---

### Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from interface org.quartz.Calendar</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONTH</td>
</tr>
</tbody>
</table>
## Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BaseCalendar</strong>()</td>
<td></td>
</tr>
<tr>
<td><strong>BaseCalendar</strong>(Calendar baseCalendar)</td>
<td></td>
</tr>
<tr>
<td><strong>BaseCalendar</strong>(Calendar baseCalendar, TimeZone timeZone)</td>
<td></td>
</tr>
<tr>
<td><strong>BaseCalendar</strong>(TimeZone timeZone)</td>
<td></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong>.clone()</td>
<td></td>
</tr>
<tr>
<td>protected Calendar.createJavaCalendar()</td>
<td>Build a Calendar with the current time.</td>
</tr>
<tr>
<td>protected Calendar.createJavaCalendar(long timeStamp)</td>
<td>Build a Calendar for the given timeStamp.</td>
</tr>
<tr>
<td>Calendar.getBaseCalendar()</td>
<td>Get the base calendar.</td>
</tr>
<tr>
<td>String.getDescription()</td>
<td>Return the description given to the Calendar instance by its creator (if any).</td>
</tr>
<tr>
<td>protected Calendar.getEndOfDayJavaCalendar(long timeInMillis)</td>
<td>Returns the end of the given day Calendar.</td>
</tr>
<tr>
<td>long getNextIncludedTime(long timeStamp)</td>
<td>Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.</td>
</tr>
<tr>
<td>protected Calendar.getStartOfDayJavaCalendar(long timeInMillis)</td>
<td>Returns the start of the given day as a Calendar.</td>
</tr>
<tr>
<td>TimeZone.getTimeZone()</td>
<td>Returns the time zone for which this Calendar will be resolved.</td>
</tr>
<tr>
<td>boolean isTimeIncluded(long timeStamp)</td>
<td></td>
</tr>
</tbody>
</table>
Check if date/time represented by timeStamp is included.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void setBaseCalendar(Calendar baseCalendar)</td>
<td>Set a new base calendar or remove the existing one</td>
</tr>
<tr>
<td>void setDescription(String description)</td>
<td>Set a description for the Calendar instance - may be useful for remembering/displaying the purpose of the calendar, though the description has no meaning to Quartz.</td>
</tr>
<tr>
<td>void setTimeZone(TimeZone timeZone)</td>
<td>Sets the time zone for which this Calendar will be resolved.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

- equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

**BaseCalendar**

public BaseCalendar()

---

**BaseCalendar**

public BaseCalendar(Calendar baseCalendar)

---

**BaseCalendar**

public BaseCalendar(TimeZone timeZone)

**Parameters:**

- timeZone - The time zone to use for this Calendar, null if TimeZone.getDefault() should be used

---

**BaseCalendar**
public BaseCalendar(Calendar baseCalendar, TimeZone timeZone)

**Parameters:**

- timeZone - The time zone to use for this Calendar, null if TimeZone.getDefault() should be used

### Method Detail

#### clone

```java
class clone()
```

**Specified by:**
- clone in interface Calendar

**Overrides:**
- clone in class Object

#### setBaseCalendar

```java
public void setBaseCalendar(Calendar baseCalendar)
```

Set a new base calendar or remove the existing one

**Specified by:**
- setBaseCalendar in interface Calendar

#### getBaseCalendar

```java
public Calendar getBaseCalendar()
```

Get the base calendar. Will be null, if not set.

**Specified by:**
- getBaseCalendar in interface Calendar
**getDescription**

```java
public String getDescription()
```

Return the description given to the calendar instance by its creator (if any).

**Specified by:**
- `getDescription` in interface `Calendar`

**Returns:**
- null if no description was set.

---

**setDescription**

```java
public void setDescription(String description)
```

Set a description for the calendar instance - may be useful for remembering/displaying the purpose of the calendar, though the description has no meaning to Quartz.

**Specified by:**
- `setDescription` in interface `Calendar`

---

**getTimeZone**

```java
public TimeZone getTimeZone()
```

Returns the time zone for which this calendar will be resolved.

**Returns:**
- This Calendar's timezone, null if Calendar should use the `TimeZone.getDefault()`

---

**setTimeZone**

```java
public void setTimeZone(TimeZone timeZone)
```

Sets the time zone for which this calendar will be resolved.
**Parameters:**

timeZone - The time zone to use for this Calendar, null if TimeZone.getDefault() should be used

---

**isTimeIncluded**

```java
public boolean isTimeIncluded(long timeStamp)
```

Check if date/time represented by timeStamp is included. If included return true. The implementation of BaseCalendar simply calls the base calendars isTimeIncluded() method if base calendar is set.

**Specified by:**

`isTimeIncluded` in interface `Calendar`

**See Also:**

`Calendar.isTimeIncluded(long)`

---

**getNextIncludedTime**

```java
public long getNextIncludedTime(long timeStamp)
```

Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time. Return the original value if timeStamp is included. Return 0 if all days are excluded.

**Specified by:**

`getNextIncludedTime` in interface `Calendar`

**See Also:**

`Calendar.getNextIncludedTime(long)`

---

**createJavaCalendar**

```java
protected Calendar createJavaCalendar(long timeStamp)
```

Build a Calendar for the given timeStamp. The new Calendar will use the BaseCalendar time zone if it is not null.
createJavaCalendar

protected Calendar createJavaCalendar()

Build a Calendar with the current time. The new Calendar will use the BaseCalendar time zone if it is not null.

getStartOfDayJavaCalendar

protected Calendar getStartOfDayJavaCalendar(long timeInMillis)

Returns the start of the given day as a calendar. This calculation will take the BaseCalendar time zone into account if it is not null.

Parameters:
    timeInMillis - A time containing the desired date for the start-of-day time

Returns:
    A Calendar set to the start of the given day.

getEndOfDayJavaCalendar

protected Calendar getEndOfDayJavaCalendar(long timeInMillis)

Returns the end of the given day Calendar. This calculation will take the BaseCalendar time zone into account if it is not null.

Parameters:
    timeInMillis - A time containing the desired date for the end-of-day time.

Returns:
    A Calendar set to the end of the given day.
Class CronCalendar

java.lang.Object
  └ org.quartz.impl.calendar.BaseCalendar
      └ org.quartz.impl.calendar.CronCalendar

All Implemented Interfaces:
  Serializable, Cloneable, Calendar

public class CronCalendar
extends BaseCalendar

This implementation of the Calendar excludes the set of times expressed by a
given CronExpression. For example, you could use this calendar to exclude all
but business hours (8AM - 5PM) every day using the expression "* * 0-7,18-23
? * *".

It is important to remember that the cron expression here describes a set of times
to be excluded from firing. Whereas the cron expression in CronTrigger
describes a set of times that can be included for firing. Thus, if a CronTrigger
has a given cron expression and is associated with a CronCalendar with the
same expression, the calendar will exclude all the times the trigger includes, and
they will cancel each other out.

Author:
  Aaron Craven
See Also:
  Serialized Form

Field Summary

Fields inherited from interface org.quartz.Calendar
MONTH
## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CronCalendar</strong>&lt;br&gt;(Calendar baseCalendar, String expression)</td>
<td>Create a CronCalendar with the given cron expression and baseCalendar.</td>
</tr>
<tr>
<td><strong>CronCalendar</strong>&lt;br&gt;(Calendar baseCalendar, String expression, TimeZone timeZone)</td>
<td>Create a CronCalendar with the given cron expression, baseCalendar, and TimeZone.</td>
</tr>
<tr>
<td><strong>CronCalendar</strong>&lt;br&gt;(String expression)</td>
<td>Create a CronCalendar with the given cron expression and no baseCalendar.</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>clone()</td>
<td></td>
</tr>
<tr>
<td>CronExpression</td>
<td>getCronExpression()</td>
<td>Returns the object representation of the cron expression that defines the dates and times this calendar excludes.</td>
</tr>
<tr>
<td>long</td>
<td>getNextIncludedTime(long timeInMillis)</td>
<td>Determines the next time included by the CronCalendar after the specified time.</td>
</tr>
<tr>
<td>TimeZone</td>
<td>getTimeZone()</td>
<td>Returns the time zone for which the CronExpression of this CronCalendar will be resolved.</td>
</tr>
<tr>
<td>boolean</td>
<td>isTimeIncluded(long timeInMillis)</td>
<td>Determines whether the given time (in milliseconds) is 'included' by the BaseCalendar.</td>
</tr>
<tr>
<td>void</td>
<td>setCronExpression(CronExpression expression)</td>
<td>Sets the cron expression for the calendar to a new value.</td>
</tr>
<tr>
<td>void</td>
<td>setCronExpression(String expression)</td>
<td>Sets the cron expression for the calendar to a new value.</td>
</tr>
<tr>
<td>void</td>
<td>setTimeZone(TimeZone timeZone)</td>
<td>Sets the time zone for which the CronExpression of this CronCalendar will be resolved.</td>
</tr>
</tbody>
</table>
**CronCalendar**

```java
public CronCalendar(String expression)
    throws ParseException
```

Create a CronCalendar with the given cron expression and no baseCalendar.

**Parameters:**
- expression - a String representation of the desired cron expression

**Throws:**
- ParseException

```java
public CronCalendar(Calendar baseCalendar, String expression)
    throws ParseException
```

Create a CronCalendar with the given cron expression and baseCalendar.

**Parameters:**
baseCalendar - the base calendar for this calendar instance – see BaseCalendar for more information on base calendar functionality
expression - a String representation of the desired cron expression

Throws:  ParseException

CronCalendar

public CronCalendar(Calendar baseCalendar,
                      String expression,
                      TimeZone timeZone)
     throws ParseException

Create a CronCalendar with the given cron expression, baseCalendar, and TimeZone.

Parameters:
baseCalendar - the base calendar for this calendar instance – see BaseCalendar for more information on base calendar functionality
expression - a String representation of the desired cron expression
timeZone - Specifies for which time zone the expression should be interpreted, i.e. the expression 0 0 10 * * ?, is resolved to 10:00 am in this time zone. If timeZone is null then TimeZone.getDefault() will be used.

Throws:  ParseException

Method Detail

clone

public Object clone()

Specified by:
clone in interface Calendar

Overrides:
clone in class BaseCalendar
**getTimeZone**

public *TimeZone* getTimeZone()

Returns the time zone for which the CronExpression of this CronCalendar will be resolved.

Overrides BaseCalendar.getTimeZone() to defer to its CronExpression.

**Overrides:**

g getTimeZone in class BaseCalendar

**Returns:**

This Calendar's timezone, null if Calendar should use the TimeZone.getDefault()
Specified by:
   isTimeIncluded in interface Calendar
Overrides:
   isTimeIncluded in class BaseCalendar
Parameters:
   timeInMillis - the date/time to test
Returns:
a boolean indicating whether the specified time is 'included' by the
CronCalendar
See Also:
   Calendar.isTimeIncluded(long)

---

g getNextIncludedTime

public long getNextIncludedTime(long timeInMillis)

Determines the next time included by the CronCalendar after the specified
time.

Specified by:
   getNextIncludedTime in interface Calendar
Overrides:
   getNextIncludedTime in class BaseCalendar
Parameters:
   timeInMillis - the initial date/time after which to find an included
time
Returns:
   the time in milliseconds representing the next time included after the
specified time.
See Also:
   Calendar.getNextIncludedTime(long)

---

toString

public String toString()

Returns a string representing the properties of the CronCalendar
Overrides:

toString in class Object

Returns:
the properties of the CronCalendar in a String format

---

getCronExpression

public CronExpression getCronExpression()

Returns the object representation of the cron expression that defines the
dates and times this calendar excludes.

Returns:
the cron expression
See Also:
CronExpression

---

setCronExpression

public void setCronExpression(String expression)
throws ParseException

Sets the cron expression for the calendar to a new value

Parameters:
expression - the new string value to build a cron expression from

Throws:
ParseException - if the string expression cannot be parsed

---

setCronExpression

public void setCronExpression(CronExpression expression)

Sets the cron expression for the calendar to a new value

Parameters:
expression - the new cron expression
Class **DailyCalendar**

```java
public class DailyCalendar
extends BaseCalendar
```

This implementation of the Calendar excludes (or includes - see below) a specified time range each day. For example, you could use this calendar to exclude business hours (8AM - 5PM) every day. Each `DailyCalendar` only allows a single time range to be specified, and that time range may not cross daily boundaries (i.e. you cannot specify a time range from 8PM - 5AM). If the property `invertTimeRange` is `false` (default), the time range defines a range of times in which triggers are not allowed to fire. If `invertTimeRange` is `true`, the time range is inverted – that is, all times outside the defined time range are excluded.

Note when using `DailyCalendar`, it behaves on the same principals as, for example, `WeeklyCalendar`. `WeeklyCalendar` defines a set of days that are excluded every week. Likewise, `DailyCalendar` defines a set of times that are excluded every day.

**Author:**
Mike Funk, Aaron Craven

**See Also:**
Serialized Form

---

### Field Summary

**Fields inherited from interface org.quartz.Calendar**
 Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DailyCalendar</td>
<td>Calendar rangeStartingCalendar, Calendar rangeEndingCalendar</td>
<td>Create a DailyCalendar with a time range defined by the specified java.util.Calendar and no baseCalendar.</td>
</tr>
<tr>
<td>DailyCalendar</td>
<td>Calendar baseCalendar, Calendar rangeStartingCalendar, Calendar rangeEndingCalendar</td>
<td>Create a DailyCalendar with a time range defined by the specified java.util.Calendars and the specified baseCalendar.</td>
</tr>
<tr>
<td>DailyCalendar</td>
<td>Calendar baseCalendar, int rangeStartingHourOfDay, int rangeStartingMinute, int rangeStartingSecond, int rangeStartingMillis, int rangeEndingHourOfDay, int rangeEndingMinute, int rangeEndingSecond, int rangeEndingMillis</td>
<td>Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.</td>
</tr>
<tr>
<td>DailyCalendar</td>
<td>Calendar baseCalendar, long rangeStartingTimeInMillis, long rangeEndingTimeInMillis</td>
<td>Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.</td>
</tr>
<tr>
<td>DailyCalendar</td>
<td>Calendar baseCalendar, String rangeStartingTime, String rangeEndingTime</td>
<td>Create a DailyCalendar with a time range defined by the specified strings and the specified baseCalendar.</td>
</tr>
<tr>
<td>DailyCalendar</td>
<td>Calendar baseCalendar, TimeZone timeZone, long rangeStartingTimeInMillis, long rangeEndingTimeInMillis</td>
<td>Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar.</td>
</tr>
<tr>
<td>DailyCalendar</td>
<td>int rangeStartingHourOfDay, int rangeStartingMinute, int rangeStartingSecond, int rangeStartingMillis, int rangeEndingHourOfDay, int rangeEndingMinute, int rangeEndingSecond, int rangeEndingMillis</td>
<td>Create a DailyCalendar with a time range defined by the specified values and no baseCalendar.</td>
</tr>
<tr>
<td>DailyCalendar</td>
<td>long rangeStartingTimeInMillis, long rangeEndingTimeInMillis</td>
<td>Create a DailyCalendar with a time range defined by the specified values and no baseCalendar.</td>
</tr>
</tbody>
</table>
Create a DailyCalendar with a time range defined by the specified values and no baseCalendar.

**DailyCalendar** *(String rangeStartingTime, String rangeEndingTime)*
  - Create a DailyCalendar with a time range defined by the specified strings and no baseCalendar.

**DailyCalendar** *(TimeZone timeZone, long rangeStartingTimeInMillis, long rangeEndingTimeInMillis)*
  - Create a DailyCalendar with a time range defined by the specified values and no baseCalendar.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object clone()</td>
<td></td>
</tr>
<tr>
<td>boolean getInvertTimeRange()</td>
<td>Indicates whether the time range represents an inverted time range (see class description).</td>
</tr>
<tr>
<td>long getNextIncludedTime(long timeInMillis)</td>
<td>Determines the next time included by the DailyCalendar after the specified time.</td>
</tr>
<tr>
<td>long getTimeRangeEndingTimeInMillis(long timeInMillis)</td>
<td>Returns the end time of the time range (in milliseconds) of the day specified in timeInMillis</td>
</tr>
<tr>
<td>long getTimeRangeStartingTimeInMillis(long timeInMillis)</td>
<td>Returns the start time of the time range (in milliseconds) of the day specified in timeInMillis</td>
</tr>
<tr>
<td>boolean isTimeIncluded(long timeInMillis)</td>
<td>Determines whether the given time (in milliseconds) is 'included' by the BaseCalendar</td>
</tr>
<tr>
<td>void setInvertTimeRange(boolean flag)</td>
<td>Indicates whether the time range represents an inverted time range (see class description).</td>
</tr>
<tr>
<td>void setTimeRange(Calendar rangeStartingCalendar, Calendar rangeEndingCalendar)</td>
<td>Sets the time range for the DailyCalendar to the times represented in the specified java.util.Calendar.</td>
</tr>
</tbody>
</table>
void **setTimeRange**(int rangeStartingHourOfDay,  
    int rangeStartingMinute, int rangeStartingSecond,  
    int rangeStartingMillis, int rangeEndingHourOfDay,  
    int rangeEndingMinute, int rangeEndingSecond,  
    int rangeEndingMillis)  

    Sets the time range for the DailyCalendar to the times  
    represented in the specified values.

void **setTimeRange**(long rangeStartingTime, long rangeEndingTime)  

    Sets the time range for the DailyCalendar to the times  
    represented in the specified values.

void **setTimeRange**(String rangeStartingTimeString,  
    String rangeEndingTimeString)  

    Sets the time range for the DailyCalendar to the times  
    represented in the specified Strings.

String **toString**()  

    Returns a string representing the properties of the DailyCalendar

Methods inherited from class org.quartz.impl.calendar.**BaseCalendar**

createJavaCalendar, createJavaCalendar, getBaseCalendar,  
getDescription, getEndOfDayJavaCalendar,  
getStartOfDayJavaCalendar, getTimeZone, setBaseCalendar,  
setDescription, setTimeZone

Methods inherited from class java.lang.**Object**
equals, finalize, getClass, hashCode, notify, notifyAll, wait,  
wait, wait

**Constructor Detail**

DailyCalendar

public DailyCalendar(String rangeStartingTime,  
    String rangeEndingTime)

    Create a DailyCalendar with a time range defined by the specified strings  
    and no baseCalendar. rangeStartingTime and rangeEndingTime must be  
    in the format "HH:MM[:SS[:mmm]]" where:
- HH is the hour of the specified time. The hour should be specified using military (24-hour) time and must be in the range 0 to 23.
- MM is the minute of the specified time and must be in the range 0 to 59.
- SS is the second of the specified time and must be in the range 0 to 59.
- mmm is the millisecond of the specified time and must be in the range 0 to 999.
- items enclosed in brackets (',']') are optional.
- The time range starting time must be before the time range ending time. Note this means that a time range may not cross daily boundaries (10PM - 2AM)

**Note:** This DailyCalendar will use the TimeZone.getDefault() time zone unless an explicit time zone is set via BaseCalendar.setTimeZone(TimeZone)

**Parameters:**
- rangeStartingTime - a String representing the starting time for the time range
- rangeEndingTime - a String representing the ending time for the time range

---

**DailyCalendar**

```java
public DailyCalendar(Calendar baseCalendar,
                     String rangeStartingTime,
                     String rangeEndingTime)
```

Create a DailyCalendar with a time range defined by the specified strings and the specified baseCalendar. rangeStartingTime and rangeEndingTime must be in the format "HH:MM[:SS[:mmm]]" where:

- HH is the hour of the specified time. The hour should be specified using military (24-hour) time and must be in the range 0 to 23.
- MM is the minute of the specified time and must be in the range 0 to 59.
- SS is the second of the specified time and must be in the range 0 to 59.
- mmm is the millisecond of the specified time and must be in the range 0 to 999.
- items enclosed in brackets (',']') are optional.
- The time range starting time must be before the time range ending time. Note this means that a time range may not cross daily boundaries (10PM - 2AM)

**Note:** This *DailyCalendar* will use the *TimeZone.getDefault()* time zone unless an explicit time zone is set via *BaseCalendar.setTimeZone(TimeZone)*

**Parameters:**
- `baseCalendar` - the base calendar for this calendar instance – see *BaseCalendar* for more information on base calendar functionality
- `rangeStartingTime` - a String representing the starting time for the time range
- `rangeEndingTime` - a String representing the ending time for the time range

---

### DailyCalendar

```java
public DailyCalendar(int rangeStartingHourOfDay, 
    int rangeStartingMinute, 
    int rangeStartingSecond, 
    int rangeStartingMillis, 
    int rangeEndingHourOfDay, 
    int rangeEndingMinute, 
    int rangeEndingSecond, 
    int rangeEndingMillis)
```

Create a *DailyCalendar* with a time range defined by the specified values and no `baseCalendar`. Values are subject to the following validations:
- Hours must be in the range 0-23 and are expressed using military (24-hour) time.
- Minutes must be in the range 0-59
- Seconds must be in the range 0-59
- Milliseconds must be in the range 0-999
- The time range starting time must be before the time range ending time. Note this means that a time range may not cross daily boundaries (10PM - 2AM)

**Note:** This *DailyCalendar* will use the *TimeZone.getDefault()* time zone unless an explicit time zone is set via
BaseCalendar.setTimeZone(TimeZone)

Parameters:

- rangeStartingHourOfDay - the hour of the start of the time range
- rangeStartingMinute - the minute of the start of the time range
- rangeStartingSecond - the second of the start of the time range
- rangeStartingMillis - the millisecond of the start of the time range
- rangeEndingHourOfDay - the hour of the end of the time range
- rangeEndingMinute - the minute of the end of the time range
- rangeEndingSecond - the second of the end of the time range
- rangeEndingMillis - the millisecond of the start of the time range

DailyCalendar

public DailyCalendar(Calendar baseCalendar, 
  int rangeStartingHourOfDay, 
  int rangeStartingMinute, 
  int rangeStartingSecond, 
  int rangeStartingMillis, 
  int rangeEndingHourOfDay, 
  int rangeEndingMinute, 
  int rangeEndingSecond, 
  int rangeEndingMillis)

Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar. Values are subject to the following validations:

- Hours must be in the range 0-23 and are expressed using military (24-hour) time.
- Minutes must be in the range 0-59
- Seconds must be in the range 0-59
- Milliseconds must be in the range 0-999
- The time range starting time must be before the time range ending time. Note this means that a time range may not cross daily boundaries (10PM - 2AM)

Note: This DailyCalendar will use the TimeZone.getDefault() time zone unless an explicit time zone is set via BaseCalendar.setTimeZone(TimeZone)
Parameters:
baseCalendar - the base calendar for this calendar instance – see BaseCalendar for more information on base calendar functionality
rangeStartingHourOfDay - the hour of the start of the time range
rangeStartingMinute - the minute of the start of the time range
rangeStartingSecond - the second of the start of the time range
rangeStartingMillis - the millisecond of the start of the time range
rangeEndingHourOfDay - the hour of the end of the time range
rangeEndingMinute - the minute of the end of the time range
rangeEndingSecond - the second of the end of the time range
rangeEndingMillis - the millisecond of the end of the time range

DailyCalendar

public DailyCalendar(Calendar rangeStartingCalendar, Calendar rangeEndingCalendar)

Create a DailyCalendar with a time range defined by the specified java.util.Calendar s and no baseCalendar. The Calendars are subject to the following considerations:

- Only the time-of-day fields of the specified Calendars will be used (the date fields will be ignored)
- The starting time must be before the ending time of the defined time range. Note this means that a time range may not cross daily boundaries (10PM - 2AM). *(because only time fields are used, it is possible for two Calendars to represent a valid time range and rangeStartingCalendar.after(rangeEndingCalendar) == true)*

Note: This DailyCalendar will use the TimeZone.getDefault() time zone unless an explicit time zone is set via BaseCalendar.setTimeZone(TimeZone)

Parameters:
rangeStartingCalendar - a java.util.Calendar representing the starting time for the time range
rangeEndingCalendar - a java.util.Calendar representing the ending time for the time range
Create a DailyCalendar with a time range defined by the specified java.util.Calendar and the specified baseCalendar. The Calendars are subject to the following considerations:

- Only the time-of-day fields of the specified Calendars will be used (the date fields will be ignored)
- The starting time must be before the ending time of the defined time range. Note this means that a time range may not cross daily boundaries (10PM - 2AM). *(because only time fields are are used, it is possible for two Calendars to represent a valid time range and rangeStartingCalendar.after(rangeEndingCalendar) == true)*

**Note:** This DailyCalendar will use the TimeZone.getDefault() time zone unless an explicit time zone is set via BaseCalendar.setTimeZone(TimeZone)

**Parameters:**
- baseCalendar - the base calendar for this calendar instance – see BaseCalendar for more information on base calendar functionality
- rangeStartingCalendar - a java.util.Calendar representing the starting time for the time range
- rangeEndingCalendar - a java.util.Calendar representing the ending time for the time range

Create a DailyCalendar with a time range defined by the specified values and no baseCalendar. The values are subject to the following considerations:

- Only the time-of-day portion of the specified values will be used
- The starting time must be before the ending time of the defined time range
range. Note this means that a time range may not cross daily boundaries (10PM - 2AM). *(because only time value are are used, it is possible for the two values to represent a valid time range and rangeStartingTime > rangeEndingTime)*

**Note:** This DailyCalendar will use the TimeZone.getDefault() time zone unless an explicit time zone is set via BaseCalendar.setTimeZone(TimeZone). You should use #DailyCalendar(String, TimeZone, long, long) if you don't want the given rangeStartingTimeInMillis and rangeEndingTimeInMillis to be evaluated in the default time zone.

**Parameters:**
- rangeStartingTimeInMillis - a long representing the starting time for the time range
- rangeEndingTimeInMillis - a long representing the ending time for the time range

---

**DailyCalendar**

```java
public DailyCalendar(Calendar baseCalendar,
                      long rangeStartingTimeInMillis,
                      long rangeEndingTimeInMillis)
```

Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar. The values are subject to the following considerations:
- Only the time-of-day portion of the specified values will be used
- The starting time must be before the ending time of the defined time range. Note this means that a time range may not cross daily boundaries (10PM - 2AM). *(because only time value are are used, it is possible for the two values to represent a valid time range and rangeStartingTime > rangeEndingTime)*

**Note:** This DailyCalendar will use the TimeZone.getDefault() time zone unless an explicit time zone is set via BaseCalendar.setTimeZone(TimeZone). You should use #DailyCalendar(String, Calendar, TimeZone, long, long) if you don't want the given rangeStartingTimeInMillis and
rangeEndingTimeInMillis to be evaluated in the default time zone.

**Parameters:**
- baseCalendar - the base calendar for this calendar instance – see `BaseCalendar` for more information on base calendar functionality
- rangeStartingTimeInMillis - a long representing the starting time for the time range
- rangeEndingTimeInMillis - a long representing the ending time for the time range

---

**DailyCalendar**

```java
public DailyCalendar(TimeZone timeZone,
                      long rangeStartingTimeInMillis,
                      long rangeEndingTimeInMillis)
```

Create a `DailyCalendar` with a time range defined by the specified values and no `baseCalendar`. The values are subject to the following considerations:
- Only the time-of-day portion of the specified values will be used
- The starting time must be before the ending time of the defined time range. Note this means that a time range may not cross daily boundaries (10PM - 2AM). *(because only time value are are used, it is possible for the two values to represent a valid time range and rangeStartingTime > rangeEndingTime)*

**Parameters:**
- timeZone - the time zone for of the `DailyCalendar` which will also be used to resolve the given start/end times.
- rangeStartingTimeInMillis - a long representing the starting time for the time range
- rangeEndingTimeInMillis - a long representing the ending time for the time range

---

**DailyCalendar**

```java
public DailyCalendar(Calendar baseCalendar,
                      TimeZone timeZone,
                      ...)
```
long rangeStartingTimeInMillis,
long rangeEndingTimeInMillis)

Create a DailyCalendar with a time range defined by the specified values and the specified baseCalendar. The values are subject to the following considerations:

- Only the time-of-day portion of the specified values will be used
- The starting time must be before the ending time of the defined time range. Note this means that a time range may not cross daily boundaries (10PM - 2AM). *(because only time value are are used, it is possible for the two values to represent a valid time range and rangeStartingTime > rangeEndingTime)*

Parameters:
- baseCalendar - the base calendar for this calendar instance – see BaseCalendar for more information on base calendar functionality
- timeZone - the time zone for of the DailyCalendar which will also be used to resolve the given start/end times.
- rangeStartingTimeInMillis - a long representing the starting time for the time range
- rangeEndingTimeInMillis - a long representing the ending time for the time range

### Method Detail

**clone**

public Object clone()

**Specified by:**

clone in interface Calendar

**Overrides:**

clone in class BaseCalendar

**isTimeIncluded**

public boolean isTimeIncluded(long timeInMillis)
Determines whether the given time (in milliseconds) is 'included' by the BaseCalendar

**Specified by:**

`isTimeIncluded` in interface Calendar

**Overrides:**

`isTimeIncluded` in class BaseCalendar

**Parameters:**

timeInMillis - the date/time to test

**Returns:**

a boolean indicating whether the specified time is 'included' by the BaseCalendar

**See Also:**

Calendar.isTimeIncluded(long)

---

**getNextIncludedTime**

```java
public long getNextIncludedTime(long timeInMillis)
```

Determines the next time included by the DailyCalendar after the specified time.

**Specified by:**

`getNextIncludedTime` in interface Calendar

**Overrides:**

`getNextIncludedTime` in class BaseCalendar

**Parameters:**

timeInMillis - the initial date/time after which to find an included time

**Returns:**

the time in milliseconds representing the next time included after the specified time.

**See Also:**

Calendar.getNextIncludedTime(long)

---

**getTimeRangeStartingTimeInMillis**

```java
public long getTimeRangeStartingTimeInMillis(long timeInMillis)
```

---

**getTimeRangeStartingTimeInMillis**

```java
public long getTimeRangeStartingTimeInMillis(long timeInMillis)
```
Returns the start time of the time range (in milliseconds) of the day specified in timeInMillis

**Parameters:**
- timeInMillis - a time containing the desired date for the starting time of the time range.

**Returns:**
- a date/time (in milliseconds) representing the start time of the time range for the specified date.

---

**getTimeRangeEndingTimeInMillis**

public long getTimeRangeEndingTimeInMillis(long timeInMillis)

Returns the end time of the time range (in milliseconds) of the day specified in timeInMillis

**Parameters:**
- timeInMillis - a time containing the desired date for the ending time of the time range.

**Returns:**
- a date/time (in milliseconds) representing the end time of the time range for the specified date.

---

**getInvertTimeRange**

public boolean getInvertTimeRange()

Indicates whether the time range represents an inverted time range (see class description).

**Returns:**
- a boolean indicating whether the time range is inverted
public void setInvertTimeRange(boolean flag)

    Indicates whether the time range represents an inverted time range (see class description).

    Parameters:
    flag - the new value for the invertTimeRange flag.

---

toString

public String toString()

    Returns a string representing the properties of the DailyCalendar

    Overrides:
    toString in class Object

    Returns:
    the properties of the DailyCalendar in a String format

---

setTimeRange

public void setTimeRange(String rangeStartingTimeString, String rangeEndingTimeString)

    Sets the time range for the DailyCalendar to the times represented in the specified Strings.

    Parameters:
    rangeStartingTimeString - a String representing the start time of the time range
    rangeEndingTimeString - a String representing the end time of the excluded time range

---

setTimeRange

public void setTimeRange(int rangeStartingHourOfDay, int rangeStartingMinute,
int rangeStartingSecond,
int rangeStartingMillis,
int rangeEndingHourOfDay,
int rangeEndingMinute,
int rangeEndingSecond,
int rangeEndingMillis)

Sets the time range for the DailyCalendar to the times represented in the specified values.

Parameters:
rangeStartingHourOfDay - the hour of the start of the time range
rangeStartingMinute - the minute of the start of the time range
rangeStartingSecond - the second of the start of the time range
rangeStartingMillis - the millisecond of the start of the time range
rangeEndingHourOfDay - the hour of the end of the time range
rangeEndingMinute - the minute of the end of the time range
rangeEndingSecond - the second of the end of the time range
rangeEndingMillis - the millisecond of the start of the time range

setTimeRange

public void setTimeRange(Calendar rangeStartingCalendar,
Calendar rangeEndingCalendar)

Sets the time range for the DailyCalendar to the times represented in the specified java.util.Calendar.

Parameters:
rangeStartingCalendar - a Calendar containing the start time for the DailyCalendar
rangeEndingCalendar - a Calendar containing the end time for the DailyCalendar

setTimeRange

public void setTimeRange(long rangeStartingTime,
long rangeEndingTime)
Sets the time range for the DailyCalendar to the times represented in the specified values.

**Parameters:**
- `rangeStartingTime` - the starting time (in milliseconds) for the time range
- `rangeEndingTime` - the ending time (in milliseconds) for the time range

Copyright 2001-2011, Terracotta, Inc.
public class HolidayCalendar

extends BaseCalendar
implements Calendar, Serializable

This implementation of the Calendar stores a list of holidays (full days that are excluded from scheduling).

The implementation DOES take the year into consideration, so if you want to exclude July 4th for the next 10 years, you need to add 10 entries to the exclude list.

Author:
    Sharada Jambula, Juergen Donnerstag

See Also:
    Serialized Form

Field Summary

Fields inherited from interface org.quartz.Calendar
MONTH

Constructor Summary

HolidayCalendar()
## HolidayCalendar

- **HolidayCalendar** *(Calendar baseCalendar)*
- **HolidayCalendar** *(Calendar baseCalendar, TimeZone timeZone)*
- **HolidayCalendar** *(TimeZone timeZone)*

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void addExcludedDate(Date excludedDate)</code></td>
<td>Add the given Date to the list of excluded days.</td>
</tr>
<tr>
<td><code>Object clone()</code></td>
<td></td>
</tr>
<tr>
<td><code>SortedSet getExcludedDates()</code></td>
<td>Returns a SortedSet of Dates representing the excluded days.</td>
</tr>
<tr>
<td><code>long getNextIncludedTime(long timeStamp)</code></td>
<td>Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.</td>
</tr>
<tr>
<td><code>boolean isTimeIncluded(long timeStamp)</code></td>
<td>Determine whether the given time (in milliseconds) is 'included' by the Calendar.</td>
</tr>
<tr>
<td><code>void removeExcludedDate(Date dateToRemove)</code></td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class org.quartz.impl.calendar.BaseCalendar

- `createJavaCalendar`, `createJavaCalendar`, `getBaseCalendar`, `getDescription`, `getEndOfDayJavaCalendar`, `getStartOfDayJavaCalendar`, `getTimeZone`, `setBaseCalendar`, `setDescription`, `setTimeZone`

### Methods inherited from class java.lang.Object

- `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Methods inherited from interface org.quartz.Calendar

- `getBaseCalendar`, `getDescription`, `setBaseCalendar`, `setDescription`
Constructor Detail

HolidayCalendar

public HolidayCalendar()

HolidayCalendar

public HolidayCalendar(Calendar baseCalendar)

HolidayCalendar

public HolidayCalendar(TimeZone timeZone)

HolidayCalendar

public HolidayCalendar(Calendar baseCalendar, TimeZone timeZone)

Method Detail

clone

public Object clone()

Specified by:
clone in interface Calendar

Overrides:
clone in class BaseCalendar

isTimeIncluded
public boolean isTimeIncluded(long time_stamp)  

Determine whether the given time (in milliseconds) is 'included' by the Calendar. 

Note that this Calendar is only has full-day precision. 

Specified by:  
isTimeIncluded in interface Calendar  
Overrides:  
isTimeIncluded in class BaseCalendar  
See Also:  
Calendar.isTimeIncluded(long)  

---

getNextIncludedTime  

public long getNextIncludedTime(long time_stamp)  

Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time. 

Note that this Calendar is only has full-day precision. 

Specified by:  
getNextIncludedTime in interface Calendar  
Overrides:  
getNextIncludedTime in class BaseCalendar  
See Also:  
Calendar.getNextIncludedTime(long)  

---

addExcludedDate  

public void addExcludedDate(Date excludedDate)  

Add the given Date to the list of excluded days. Only the month, day and year of the returned dates are significant.
removeExcludedDate

public void removeExcludedDate(Date dateToRemove)

getExcludedDates

public SortedSet getExcludedDates()

Returns a SortedSet of Dates representing the excluded days. Only the month, day and year of the returned dates are significant.

Copyright 2001-2011, Terracotta, Inc.
Overview  Package  Use  Tree  Deprecated  Index  Help
PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public class MonthlyCalendar
extends BaseCalendar
implements Calendar, Serializable

This implementation of the Calendar excludes a set of days of the month. You may use it to exclude every first day of each month for example. But you may define any day of a month.

Author:
Juergen Donnerstag

See Also:
Calendar, BaseCalendar, Serialized Form

Field Summary

Fields inherited from interface org.quartz.Calendar
MONTH

Constructor Summary

MonthlyCalendar()

MonthlyCalendar(Calendar baseCalendar)

MonthlyCalendar(Calendar baseCalendar, TimeZone timeZone)
**MonthlyCalendar** *(TimeZone timeZone)*

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>areAllDaysExcluded()</td>
</tr>
<tr>
<td>Object</td>
<td>clone()</td>
</tr>
<tr>
<td>boolean[]</td>
<td>getDaysExcluded()</td>
</tr>
<tr>
<td>long</td>
<td>getNextIncludedTime(long time)</td>
</tr>
<tr>
<td>boolean</td>
<td>isDayExcluded(int day)</td>
</tr>
<tr>
<td>boolean</td>
<td>isTimeIncluded(long time)</td>
</tr>
<tr>
<td>void</td>
<td>setDayExcluded(int day, boolean exclude)</td>
</tr>
<tr>
<td>void</td>
<td>setDaysExcluded(boolean[] days)</td>
</tr>
</tbody>
</table>

**Methods inherited from class org.quartz.impl.calendar.BaseCalendar**

- createJavaCalendar, createJavaCalendar, getBaseCalendar, getDescription, getEndOfDayJavaCalendar, getStartOfDayJavaCalendar, getTimeZone, setBaseCalendar, setDescription, setTimeZone

**Methods inherited from class java.lang.Object**

- equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Methods inherited from interface org.quartz.Calendar
getBaseCalendar, getDescription, setBaseCalendar, setDescription

Constructor Detail

MonthlyCalendar

public MonthlyCalendar()

MonthlyCalendar

public MonthlyCalendar(Calendar baseCalendar)

MonthlyCalendar

public MonthlyCalendar(TimeZone timeZone)

MonthlyCalendar

public MonthlyCalendar(Calendar baseCalendar, TimeZone timeZone)

Method Detail

cloned

public Object clone()

Specified by:
cloned in interface Calendar

Overrides:
cloned in class BaseCalendar
**getDaysExcluded**

public boolean[] getDaysExcluded()

Get the array which defines the exclude-value of each day of month. Only the first 31 elements of the array are relevant, with the 0 index element representing the first day of the month.

**isDayExcluded**

public boolean isDayExcluded(int day)

Return true, if day is defined to be excluded.

**Parameters:**

day - The day of the month (from 1 to 31) to check.

**setDaysExcluded**

public void setDaysExcluded(boolean[] days)

Redefine the array of days excluded. The array must non-null and of size greater or equal to 31. The 0 index element represents the first day of the month.

**setDayExcluded**

public void setDayExcluded(int day, boolean exclude)

Redefine a certain day of the month to be excluded (true) or included (false).

**Parameters:**

day - The day of the month (from 1 to 31) to set.
areAllDaysExcluded

public boolean areAllDaysExcluded()

Check if all days are excluded. That is no day is included.

isTimeIncluded

public boolean isTimeIncluded(long timeStamp)

Determine whether the given time (in milliseconds) is 'included' by the Calendar.

Note that this Calendar is only has full-day precision.

Specified by:
   isTimeIncluded in interface Calendar
Overrides:
   isTimeIncluded in class BaseCalendar
See Also:
   Calendar.isTimeIncluded(long)

g getNextIncludedTime

public long getNextIncludedTime(long timeStamp)

Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time. Return the original value if time Stamp is included. Return 0 if all days are excluded.

Note that this Calendar is only has full-day precision.

Specified by:
   getNextIncludedTime in interface Calendar
Overrides:
   getNextIncludedTime in class BaseCalendar
See Also:
Calendar getNextIncludedTime(long)
org.quartz.impl.calendar Classes
AnnualCalendar
BaseCalendar
CronCalendar
DailyCalendar
HolidayCalendar
MonthlyCalendar
WeeklyCalendar
## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnnualCalendar</strong></td>
<td>This implementation of the Calendar excludes a set of days of the year.</td>
</tr>
<tr>
<td><strong>BaseCalendar</strong></td>
<td>This implementation of the Calendar may be used (you don't have to) as a base class for more sophisticated one's.</td>
</tr>
<tr>
<td><strong>CronCalendar</strong></td>
<td>This implementation of the Calendar excludes the set of times expressed by a given CronExpression.</td>
</tr>
<tr>
<td><strong>DailyCalendar</strong></td>
<td>This implementation of the Calendar excludes (or includes - see below) a specified time range each day.</td>
</tr>
<tr>
<td><strong>HolidayCalendar</strong></td>
<td>This implementation of the Calendar stores a list of holidays (full days that are excluded from scheduling).</td>
</tr>
<tr>
<td><strong>MonthlyCalendar</strong></td>
<td>This implementation of the Calendar excludes a set of days of the month.</td>
</tr>
<tr>
<td><strong>WeeklyCalendar</strong></td>
<td>This implementation of the Calendar excludes a set of days of the week.</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, [Terracotta, Inc.](https://www.terracotta.com)
Hierarchy For Package org.quartz.impl.calendar

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.calendar.**BaseCalendar** (implements org.quartz.**Calendar**, java.lang.**Cloneable**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**AnnualCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**CronCalendar**
    - org.quartz.impl.calendar.**DailyCalendar**
    - org.quartz.impl.calendar.**HolidayCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**MonthlyCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)
    - org.quartz.impl.calendar.**WeeklyCalendar** (implements org.quartz.**Calendar**, java.io.**Serializable**)

Copyright 2001-2011, Terracotta, Inc.
# Uses of Package

**org.quartz.impl.calendar**

<table>
<thead>
<tr>
<th>Packages that use org.quartz.impl.calendar</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.calendar</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.quartz.impl.calendar used by org.quartz.impl.calendar</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BaseCalendar</strong></td>
</tr>
<tr>
<td>This implementation of the Calendar may be used (you don't have to) as a base class for more sophisticated one's.</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.calendar  Class WeeklyCalendar

java.lang.Object
   ▼ org.quartz.impl.calendar.BaseCalendar
      ▼ org.quartz.impl.calendar.WeeklyCalendar

All Implemented Interfaces:
   Serializable, Cloneable, Calendar

public class WeeklyCalendar
   extends BaseCalendar
   implements Calendar, Serializable

This implementation of the Calendar excludes a set of days of the week. You may use it to exclude weekends for example. But you may define any day of the week. By default it excludes SATURDAY and SUNDAY.

Author:
   Juergen Donnerstag

See Also:
   Calendar, BaseCalendar, Serialized Form

Field Summary

Fields inherited from interface org.quartz.Calendar

MONTH

Constructor Summary

WeeklyCalendar()

WeeklyCalendar(Calendar baseCalendar)

WeeklyCalendar(Calendar baseCalendar, TimeZone timeZone)
**WeeklyCalendar** *(TimeZone timeZone)*

### Method Summary

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td><code>areAllDaysExcluded()</code></td>
<td>Check if all week days are excluded.</td>
</tr>
<tr>
<td>Object</td>
<td><code>clone()</code></td>
<td></td>
</tr>
<tr>
<td>boolean[]</td>
<td><code>getDaysExcluded()</code></td>
<td>Get the array with the week days</td>
</tr>
<tr>
<td>long</td>
<td><code>getNextIncludedTime(long time)</code></td>
<td>Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>isDayExcluded(int wday)</code></td>
<td>Return true, if wday (see Calendar.get()) is defined to be excluded.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>isTimeIncluded(long time)</code></td>
<td>Determine whether the given time (in milliseconds) is 'included' by the Calendar.</td>
</tr>
<tr>
<td>void</td>
<td><code>setDayExcluded(int wday, boolean exclude)</code></td>
<td>Redefine a certain day of the week to be excluded (true) or included (false).</td>
</tr>
<tr>
<td>void</td>
<td><code>setDaysExcluded(boolean[] weekDays)</code></td>
<td>Redefine the array of days excluded.</td>
</tr>
</tbody>
</table>

### Methods inherited from class org.quartz.impl.calendar.BaseCalendar

- `createJavaCalendar`, `createJavaCalendar`, `getBaseCalendar`, `getDescription`, `getEndOfDayJavaCalendar`, `getStartOfDayJavaCalendar`, `getTimeZone`, `setBaseCalendar`, `setDescription`, `setTimeZone`

### Methods inherited from class java.lang.Object

- `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
Methods inherited from interface org.quartz.Calendar
getBaseCalendar, getDescription, setBaseCalendar, setDescription

Constructor Detail

WeeklyCalendar

public WeeklyCalendar()

WeeklyCalendar

public WeeklyCalendar(Calendar baseCalendar)

WeeklyCalendar

public WeeklyCalendar(TimeZone timeZone)

WeeklyCalendar

public WeeklyCalendar(Calendar baseCalendar, TimeZone timeZone)

Method Detail

clone

public Object clone()

Specified by:
    clone in interface Calendar
Overrides:
    clone in class BaseCalendar
**getDaysExcluded**

```java
public boolean[] getDaysExcluded()
```

Get the array with the week days

---

**isDayExcluded**

```java
public boolean isDayExcluded(int wday)
```

Return true, if `wday` (see `Calendar.get()`) is defined to be excluded. E. g. saturday and sunday.

---

**setDaysExcluded**

```java
public void setDaysExcluded(boolean[] weekDays)
```

Redefine the array of days excluded. The array must of size greater or equal 8. `java.util.Calendar`'s constants like `MONDAY` should be used as index. A value of true is regarded as: exclude it.

---

**setDayExcluded**

```java
public void setDayExcluded(int wday, boolean exclude)
```

Redefine a certain day of the week to be excluded (true) or included (false). Use `java.util.Calendar`'s constants like `MONDAY` to determine the `wday`.

---

**areAllDaysExcluded**

```java
public boolean areAllDaysExcluded()
```

Check if all week days are excluded. That is no day is included.
Returns:
boolean

isTimeIncluded

public boolean isTimeIncluded(long time)

Determine whether the given time (in milliseconds) is 'included' by the Calendar.

Note that this Calendar is only has full-day precision.

Specified by:
    isTimeIncluded in interface Calendar
Overrides:
    isTimeIncluded in class BaseCalendar
See Also:
    Calendar.isTimeIncluded(long)

ggetNextIncludedTime

public long getNextIncludedTime(long time)

Determine the next time (in milliseconds) that is 'included' by the Calendar after the given time. Return the original value if time is included. Return 0 if all days are excluded.

Note that this Calendar is only has full-day precision.

Specified by:
    getNextIncludedTime in interface Calendar
Overrides:
    getNextIncludedTime in class BaseCalendar
See Also:
    Calendar.getNextIncludedTime(long)
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.calendar.AnnualCalendar

No usage of org.quartz.impl.calendar.AnnualCalendar

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class
*org.quartz.impl.calendar.BaseCalendar*

## Packages that use **BaseCalendar**

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>org.quartz.impl.calendar</em></td>
</tr>
</tbody>
</table>

## Uses of **BaseCalendar** in *org.quartz.impl.calendar*

### Subclasses of **BaseCalendar** in *org.quartz.impl.calendar*

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AnnualCalendar</strong></td>
<td>This implementation of the Calendar excludes a set of days of the year.</td>
</tr>
<tr>
<td><strong>CronCalendar</strong></td>
<td>This implementation of the Calendar excludes the set of times expressed by a given <em>CronExpression</em>.</td>
</tr>
<tr>
<td><strong>DailyCalendar</strong></td>
<td>This implementation of the Calendar excludes (or includes - see below) a specified time range each day.</td>
</tr>
<tr>
<td><strong>HolidayCalendar</strong></td>
<td>This implementation of the Calendar stores a list of holidays (full days that are excluded from scheduling).</td>
</tr>
<tr>
<td><strong>MonthlyCalendar</strong></td>
<td>This implementation of the Calendar excludes a set of days of the month.</td>
</tr>
<tr>
<td><strong>WeeklyCalendar</strong></td>
<td>This implementation of the Calendar excludes a set of days of the week.</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.calendar.CronCalendar

No usage of org.quartz.impl.calendar.CronCalendar

Copyright 2001-2011, Terracotta, Inc.
Uses of Class

org.quartz.impl.calendar.DailyCalendar

No usage of org.quartz.impl.calendar.DailyCalendar

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.calendar.HolidayCalendar

No usage of org.quartz.impl.calendar.HolidayCalendar

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Class</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV</td>
<td>NEXT</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.impl.calendar.MonthlyCalendar

No usage of org.quartz.impl.calendar.MonthlyCalendar

<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Class</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.calendar.WeeklyCalendar

No usage of org.quartz.impl.calendar.WeeklyCalendar

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.DirectSchedulerFactory

Packages that use DirectSchedulerFactory

<table>
<thead>
<tr>
<th>org.quartz.impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

Uses of DirectSchedulerFactory in org.quartz.impl

Methods in org.quartz.impl that return DirectSchedulerFactory

| static DirectSchedulerFactory DirectSchedulerFactory.getInstance() |

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.JobDetailImpl

No usage of org.quartz.impl.JobDetailImpl

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class org.quartz.impl.JobExecutionContextImpl

## Packages that use `JobExecutionContextImpl`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
</tbody>
</table>

## Uses of `JobExecutionContextImpl` in `org.quartz.core`

## Fields in `org.quartz.core` declared as `JobExecutionContextImpl`

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>JobRunShell.jec</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.QuartzServer

No usage of org.quartz.impl.QuartzServer

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.RemoteMBeanScheduler

Packages that use **RemoteMBeanScheduler**

| org.quartz.ee.jmx.jboss |

Uses of **RemoteMBeanScheduler** in
org.quartz.ee.jmx.jboss

<table>
<thead>
<tr>
<th>Subclasses of <strong>RemoteMBeanScheduler</strong> in org.quartz.ee.jmx.jboss</th>
</tr>
</thead>
<tbody>
<tr>
<td>class JBoss4RMIRemoteMBeanScheduler</td>
</tr>
<tr>
<td>An implementation of the Scheduler interface that remotely proxies all method calls to the equivalent call on a given QuartzScheduler instance, via JBoss's JMX RMIAdpator.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.RemoteScheduler

No usage of org.quartz.impl.RemoteScheduler

Copyright 2001-2011, Terracotta, Inc.
## Uses of Class

**org.quartz.impl.SchedulerRepository**

### Packages that use `SchedulerRepository`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.impl</code></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

### Uses of `SchedulerRepository` in `org.quartz.impl`

### Methods in `org.quartz.impl` that return `SchedulerRepository`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static SchedulerRepository getInstance()</code></td>
<td>SchedulerRepository.getInstance()</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.StdJobRunShellFactory

No usage of org.quartz.impl.StdJobRunShellFactory

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.StdScheduler

No usage of org.quartz.impl.StdScheduler

Copyright 2001-2011, Terracotta, Inc.
Uses of Class org.quartz.impl.StdSchedulerFactory

No usage of org.quartz.impl.StdSchedulerFactory

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS   NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES   NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
org.quartz.impl.jdbcjobstore Class

AttributeRestoringConnectionInvocationHandler

java.lang.Object

org.quartz.impl.jdbcjobstore.AttributeRestoringConnectionInvocationHandler

All Implemented Interfaces:

InvocationHandler

public class AttributeRestoringConnectionInvocationHandler

extends Object

implements InvocationHandler

Protects a Connection's attributes from being permanently modified.

Wraps a provided Connection such that its auto commit and transaction isolation attributes can be overwritten, but will automatically restored to their original values when the connection is actually closed (and potentially returned to a pool for reuse).

See Also:

JobStoreSupport.getConnection(),
JobStoreCMT.getNonManagedTXConnection()

Constructor Summary

| AttributeRestoringConnectionInvocationHandler | (Connection conn) |

Method Summary

| void | close() |

Attempts to restore the auto commit and transaction isolation connection attributes of the wrapped connection to their original values (if they were overwritten), before finally
actually closing the wrapped connection.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getLog()</code></td>
<td>Provides access to the SLF4J Logger</td>
</tr>
<tr>
<td><code>getWrappedConnection()</code></td>
<td>Gets the underlying connection to which all operations ultimately defer.</td>
</tr>
<tr>
<td><code>invoke(Object proxy, Method method, Object[] args)</code></td>
<td>Dynamic method invocation handler for AttributeRestoringConnectionInvocationHandler.</td>
</tr>
<tr>
<td><code>restoreOriginalAttributes()</code></td>
<td>Attempts to restore the auto commit and transaction isolation connection attributes to their original values.</td>
</tr>
<tr>
<td><code>setAutoCommit(boolean autoCommit)</code></td>
<td>Sets this connection's auto-commit mode to the given state, saving the original mode.</td>
</tr>
<tr>
<td><code>setTransactionIsolation(int level)</code></td>
<td>Attempts to change the transaction isolation level to the given level, saving the original level.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AttributeRestoringConnectionInvocationHandler

public AttributeRestoringConnectionInvocationHandler(Connection conn)

Method Detail

getLog

protected org.slf4j.Logger getLog()
invoke

```java
public Object invoke(Object proxy, Method method, Object[] args)
throws Throwable
```

**Specified by:**
```
invoke in interface InvocationHandler
```

**Throws:**
```
Throwable
```

---

setAutoCommit

```java
public void setAutoCommit(boolean autoCommit)
throws SQLException
```

Sets this connection's auto-commit mode to the given state, saving the original mode. The connection's original auto commit mode is restored when the connection is closed.

**Throws:**
```
SQLException
```

---

setTransactionIsolation

```java
public void setTransactionIsolation(int level)
throws SQLException
```

Attempts to change the transaction isolation level to the given level, saving the original level. The connection's original transaction isolation level is restored when the connection is closed.

**Throws:**
```
SQLException
```
getWrappedConnection

public Connection getWrappedConnection()

Gets the underlying connection to which all operations ultimately defer. This is provided in case a user ever needs to punch through the wrapper to access vendor specific methods outside of the standard java.sql.Connection interface.

Returns:
The underlying connection to which all operations ultimately defer.

restoreOriginalAttributes

public void restoreOriginalAttributes()

Attempts to restore the auto commit and transaction isolation connection attributes of the wrapped connection to their original values (if they were overwritten).

close

public void close() throws SQLException

Attempts to restore the auto commit and transaction isolation connection attributes of the wrapped connection to their original values (if they were overwritten), before finally actually closing the wrapped connection.

Throws:
SQLException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>
org.quartz.impl.jdbcjobstore  Class  
CalendarIntervalTriggerPersistenceDelegate

java.lang.Object  
  org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegate  
  org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate

All Implemented Interfaces: 
  Constants, StdJDBCConstants, TriggerPersistenceDelegate

public class CalendarIntervalTriggerPersistenceDelegate  
extends SimplePropertiesTriggerPersistenceDelegateSupport

Nested Class Summary

Nested classes/interfaces inherited from interface 
org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate
  TriggerPersistenceDelegate.TriggerPropertyBundle

Field Summary

Fields inherited from class 
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport
  COL_BOOL_PROP_1, COL_BOOL_PROP_2, COL_DEC_PROP_1, COL_DEC_PROP_2, 
  COL_INT_PROP_1, COL_INT_PROP_2, COL_LONG_PROP_1, COL_LONG_PROP_2, 
  COL_STR_PROP_1, COL_STR_PROP_2, COL_STR_PROP_3, 
  DELETE_SIMPLE_PROPS_TRIGGER, INSERT_SIMPLE_PROPS_TRIGGER, 
  schedNameLiteral, SELECT_SIMPLE_PROPS_TRIGGER, 
  TABLE_SIMPLE_PROPERTIES_TRIGGERS, tablePrefix, 
  UPDATE_SIMPLE_PROPS_TRIGGER

Fields inherited from interface 
org.quartz.impl.jdbcjobstore.StdJDBCConstants
COUNT MISFIRED TRIGGERS IN STATE, DELETE ALL BLOB TRIGGERS,
DELETE ALL CALENDARS, DELETE ALL CRON TRIGGERS,
DELETE ALL JOB DETAILS, DELETE ALL PAUSED TRIGGER GRPS,
DELETE ALL SIMPLE TRIGGERS, DELETE ALL SIMPROP TRIGGERS,
DELETE ALL TRIGGERS, DELETE BLOB TRIGGER, DELETE CALENDAR,
DELETE CRON TRIGGER, DELETE FIRED TRIGGER, DELETE FIRED TRIGGERS,
DELETE INSTANCES FIRED TRIGGERS, DELETE JOB DETAIL,
DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT BLOB TRIGGER,
INSERT CALENDAR, INSERT CRON TRIGGER, INSERT FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT BLOB_TRIGGER, SELECT CALENDAR,
SELECT CALENDAR_EXISTENCE, SELECT CALENDARS, SELECT_CRON_TRIGGER,
SELECT FIRED_TRIGGER, SELECT FIRED_TRIGGER_GROUP,
SELECT FIRED_TRIGGER_INSTANCE_NAMES, SELECT FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FOR CALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS</td>
<td>COL NEXT FIRE TIME, ALL GROUPS PAUSED, COL BLOB, COL CALENDAR, COL CALENDAR_NAME, COL CHECKIN INTERVAL, COL CRON EXPRESSION, COL DESCRIPTION, COL END TIME, COL ENTRY ID, COL ENTRY STATE, COL FIRED TIME, COL INSTANCE_NAME, COL IS DURABLE, COL IS NONCONCURRENT, COL IS UPDATE DATA, COL IS VOLATILE, COL JOB CLASS, COL JOB DATAMAP, COL JOB GROUP, COL JOB NAME, COL LAST CHECKIN TIME, COL LOCK NAME, COL MISFIRE INSTRUCTION, COL NEXT FIRE TIME, COL PREV FIRE TIME, COL PRIORITY, COL REPEAT_COUNT, COL REPEAT_INTERVAL, COL REQUESTS RECOVERY, COL SCHEDULER NAME, COL START TIME, COL TIME_ZONE_ID, COL TIMES_TRIGGERED, COL TRIGGER GROUP, COL TRIGGER NAME, COL TRIGGER_STATE, COL TRIGGER_TYPE, DEFAULT TABLE PREFIX, STATE ACQUIRED, STATE BLOCKED, STATE COMPLETE, STATE DELETED, STATE ERROR, STATE EXECUTING, STATE MISFIRED, STATE PAUSED, STATE PAUSED BLOCKED, STATE WAITING, TABLE BLOB TRIGGERS, TABLE CALENDARS, TABLE CRON TRIGGERS, TABLE FIRED TRIGGERS, TABLE JOB DETAILS, TABLE LOCKS, TABLE PAUSED TRIGGERS, TABLE_SCHEDULER_STATE, TABLE_SIMPLE_TRIGGERS, TABLE_TRIGGERS, TTYPE BLOB, TTYPE CAL INT, TTYPE CRON, TTYPE SIMPLE</td>
</tr>
</tbody>
</table>

### Constructor Summary

**CalendarIntervalTriggerPersistenceDelegate()**

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean canHandleTriggerType</td>
<td>(org.quartz.spi.OperableTrigger trigger)</td>
</tr>
<tr>
<td>String getHandledTriggerTypeDiscriminator</td>
<td></td>
</tr>
<tr>
<td>protected SimplePropertiesTriggerProperties getTriggerProperties</td>
<td>(org.quartz.spi.OperableTrigger trigger)</td>
</tr>
<tr>
<td>protected TriggerPersistenceDelegate.TriggerPropertyBundle getTriggerPropertyBundle</td>
<td>(SimplePropertiesTriggerProperties triggerProperties)</td>
</tr>
</tbody>
</table>
Methods inherited from class `org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport`
- `deleteExtendedTriggerProperties`
- `initialize`
- `insertExtendedTriggerProperties`
- `loadExtendedTriggerProperties`
- `updateExtendedTriggerProperties`

Methods inherited from class `java.lang.Object`
- `clone`
- `equals`
- `finalize`
- `getClass`
- `hashCode`
- `notify`
- `notifyAll`
- `toString`
- `wait`
- `wait`
- `wait`

Constructor Detail

`CalendarIntervalTriggerPersistenceDelegate`

```java
public CalendarIntervalTriggerPersistenceDelegate()
```

Method Detail

`canHandleTriggerType`

```java
public boolean canHandleTriggerType(org.quartz.spi.OperableTrigger trigger)
```

`getHandledTriggerTypeDiscriminator`

```java
public String getHandledTriggerTypeDiscriminator()
```

`getTriggerProperties`

```java
protected SimplePropertiesTriggerProperties getTriggerProperties(org.quartz.spi.OperableTrigger trigger)
```

Specified by:
- `getTriggerProperties` in class `SimplePropertiesTriggerPersistenceDelegateSupport`
getTriggerPropertyBundle

protected TriggerPersistenceDelegate.TriggerPropertyBundle getTriggerPropertyBundle

Specified by:

getTriggerPropertyBundle in class SimplePropertiesTriggerPersistenceDelegateSupport

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS  SUMMARY: NESTED | FIELD | CONSTR | METHOD  FRAMES  NO FRAMES  DETAIL: FIELD | CONSTR | METHOD
Class CloudscapeDelegate

java.lang.Object
  \org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  \org.quartz.impl.jdbcjobstore.CloudscapeDelegate

All Implemented Interfaces:
  Constants, DriverDelegate, StdJDBCConstants

Deprecated. Use the StdJDBCDelegate for latest versions of Derby

public class CloudscapeDelegate
  extends StdJDBCDelegate

This is a driver delegate for the Cloudscape database, not surprisingly, it is known to work with Derby as well. Though later versions of Derby may simply use StdJDBCDelegate.

Author:
  James House, Sridhar Jawaharlal, Srinivas Venkataramaiah

Field Summary

Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
  classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
  COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS,
  DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS,
  DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS,
  DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS,
  DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETECALENDAR,
  DELETE_CRON_TRIGGER, DELETEFIRED_TRIGGER, DELETEFIRED_TRIGGERS,
  DELETE_INSTANCESFIRED_TRIGGERS, DELETE_JOB_DETAIL,
DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERT_CALANDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALANDAR,
SELECT_CALANDAR_EXISTENCE, SELECT_CALANDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FOR_CALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALANDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants
ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL_CALENDAR, COL_CALANDAR_NAME, COL_CHECKIN_INTERVAL,
### Constructor Summary

**CloudscapeDelegate**
```
(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)
```

*Deprecated.* Create new CloudscapeDelegate instance.

**CloudscapeDelegate**
```
(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)
```

*Deprecated.* Create new CloudscapeDelegate instance.

### Method Summary

**protected Object**
```
getObjectFromBlob(ResultSet rs, String colName)
```

*Deprecated.* This method should be overridden by any delegate subclasses that need special handling for BLOBs.

### Methods inherited from class

```
org.quartz.impl.jdbcjobstore.StdJDBCDelegate
```

```
addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet,
```
closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getJobDataFromBlob, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectMisfiredTriggersInState, selectNextFireTime, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Detail

CloudscapeDelegate

public CloudscapeDelegate(org.slf4j.Logger log,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper)

Deprecated.

Create new CloudscapeDelegate instance.

Parameters:
   log - the logger to use during execution
   tablePrefix - the prefix of all table names

CloudscapeDelegate

public CloudscapeDelegate(org.slf4j.Logger log,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper,
Boolean useProperties)

Deprecated.

Create new CloudscapeDelegate instance.

Parameters:
   log - the logger to use during execution
   tablePrefix - the prefix of all table names
   useProperties - useProperties flag
protected Object getObjectFromBlob(ResultSet rs, String colName)
throws ClassNotFoundException, IOException, SQLException

Deprecated.

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:

getObjectFromBlob in class StdJDBCDelegate

Parameters:

rs - the result set, already queued to the correct row
colName - the column name for the BLOB

Returns:

the deserialized Object from the ResultSet BLOB

Throws:

ClassNotFoundException - if a class found during deserialization cannot be found
IOException - if deserialization causes an error
SQLException
org.quartz.impl.jdbcjobstore Interface Constants

All Known Subinterfaces:
  StdJDBCConstants

All Known Implementing Classes:
  CalendarIntervalTriggerPersistenceDelegate, CloudscapeDelegate,
  CronTriggerPersistenceDelegate, DB2v6Delegate, DB2v7Delegate,
  DB2v8Delegate, DBSemaphore, HSQLDBDelegate, JobStoreCMT,
  JobStoreSupport, JobStoreTX, MSSQLDelegate, OracleDelegate,
  PointbaseDelegate, PostgreSQLDelegate,
  SimplePropertiesTriggerPersistenceDelegateSupport,
  SimpleTriggerPersistenceDelegate, StdJDBCDelegate,
  StdRowLockSemaphore, SybaseDelegate, UpdateLockRowSemaphore,
  WebLogicDelegate, WebLogicOracleDelegate

---

public interface Constants

This interface can be implemented by any DriverDelegate class that needs to use the constants contained herein.

Author:
  Jeffrey Wescott, James House

---

### Field Summary

<table>
<thead>
<tr>
<th>static String</th>
<th>ALIAS_COL_NEXT_FIRE_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>ALL_GROUPS_PAUSED</td>
</tr>
<tr>
<td>static String</td>
<td>COL_BLOB</td>
</tr>
<tr>
<td>static String</td>
<td>COLCALENDAR</td>
</tr>
<tr>
<td>static String</td>
<td>COLCALENDAR_NAME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_CHECKIN_INTERVAL</td>
</tr>
<tr>
<td>static String</td>
<td>COL_CRON_EXPRESSION</td>
</tr>
<tr>
<td>static String</td>
<td>COL_DESCRIPTION</td>
</tr>
<tr>
<td>static String</td>
<td>COL_END_TIME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_ENTRY_ID</td>
</tr>
<tr>
<td>static String</td>
<td>COL_ENTRY_STATE</td>
</tr>
<tr>
<td>static String</td>
<td>COL_FIRED_TIME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_INSTANCE_NAME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_IS_DURABLE</td>
</tr>
<tr>
<td>static String</td>
<td>COL_IS_NONCONCURRENT</td>
</tr>
<tr>
<td>static String</td>
<td>COL_IS_UPDATE_DATA</td>
</tr>
<tr>
<td>static String</td>
<td>COL_IS_VOLATILE</td>
</tr>
<tr>
<td>static String</td>
<td>COL_JOB_CLASS</td>
</tr>
<tr>
<td>static String</td>
<td>COL_JOB_DATAMAP</td>
</tr>
<tr>
<td>static String</td>
<td>COL_JOB_GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>COL_JOB_NAME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_LAST_CHECKIN_TIME</td>
</tr>
<tr>
<td>Java Code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>static String</td>
<td>COL_LOCK_NAME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_MISFIRE_INSTRUCTION</td>
</tr>
<tr>
<td>static String</td>
<td>COL_NEXT_FIRE_TIME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_PREV_FIRE_TIME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_PRIORITY</td>
</tr>
<tr>
<td>static String</td>
<td>COL_REPEAT_COUNT</td>
</tr>
<tr>
<td>static String</td>
<td>COL_REPEAT_INTERVAL</td>
</tr>
<tr>
<td>static String</td>
<td>COL_REQUESTS_RECOVERY</td>
</tr>
<tr>
<td>static String</td>
<td>COL_SCHEDULER_NAME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_START_TIME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_TIME_ZONE_ID</td>
</tr>
<tr>
<td>static String</td>
<td>COL_TIMES_TRIGGERED</td>
</tr>
<tr>
<td>static String</td>
<td>COL_TRIGGER_GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>COL_TRIGGER_NAME</td>
</tr>
<tr>
<td>static String</td>
<td>COL_TRIGGER_STATE</td>
</tr>
<tr>
<td>static String</td>
<td>COL_TRIGGER_TYPE</td>
</tr>
<tr>
<td>static String</td>
<td>DEFAULT_TABLE_PREFIX</td>
</tr>
</tbody>
</table>
static String STATE_ACQUIRED
static String STATE_BLOCKED
static String STATE_COMPLETE
static String STATE_DELETED
static String STATE_ERROR
static String STATE_EXECUTING

static String STATE_MISFIRED
  Deprecated. Whether a trigger has misfired is no longer a state, but rather now identified dynamically by whether the trigger's next fire time is more than the misfire threshold time in the past.

static String STATE_PAUSED
static String STATE_PAUSED_BLOCKED
static String STATE_WAITING

static String TABLE_BLOB_TRIGGERS
static String TABLE_CALENDARS
static String TABLE_CRON_TRIGGERS
static String TABLE_FIRED_TRIGGERS
static String TABLE_JOB_DETAILS
<table>
<thead>
<tr>
<th>Field</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_LOCKS</td>
<td>static String</td>
</tr>
<tr>
<td>TABLE_PAUSED_TRIGGERS</td>
<td>static String</td>
</tr>
<tr>
<td>TABLE_SCHEDULER_STATE</td>
<td>static String</td>
</tr>
<tr>
<td>TABLE_SIMPLE_TRIGGERS</td>
<td>static String</td>
</tr>
<tr>
<td>TABLE_TRIGGERS</td>
<td>static String</td>
</tr>
<tr>
<td>TTYPE_BLOB</td>
<td>static String</td>
</tr>
<tr>
<td>TTYPE_CAL_INT</td>
<td>static String</td>
</tr>
<tr>
<td>TTYPE_CRON</td>
<td>static String</td>
</tr>
<tr>
<td>TTYPE_SIMPLE</td>
<td>static String</td>
</tr>
</tbody>
</table>

### Field Detail

**TABLE_JOB_DETAILS**

static final String TABLE_JOB_DETAILS

See Also:

Constant Field Values

**TABLE_TRIGGERS**

static final String TABLE_TRIGGERS

See Also:
**Constant Field Values**

---

**TABLE_SIMPLE_TRIGGERS**

static final `String` `TABLE_SIMPLE_TRIGGERS`

See Also:

Constant Field Values

---

**TABLE_CRON_TRIGGERS**

static final `String` `TABLE_CRON_TRIGGERS`

See Also:

Constant Field Values

---

**TABLE_BLOB_TRIGGERS**

static final `String` `TABLE_BLOB_TRIGGERS`

See Also:

Constant Field Values

---

**TABLE_FIRED_TRIGGERS**

static final `String` `TABLE_FIRED_TRIGGERS`

See Also:

Constant Field Values

---

**TABLECALENDRS**

static final `String` `TABLECALENDRS`
See Also:
Constant Field Values

---

**TABLE_PAUSED_TRIGGERS**

static final `String` `TABLE_PAUSED_TRIGGERS`

See Also:
Constant Field Values

---

**TABLE_LOCKS**

static final `String` `TABLE_LOCKS`

See Also:
Constant Field Values

---

**TABLE_SCHEDULER_STATE**

static final `String` `TABLE_SCHEDULER_STATE`

See Also:
Constant Field Values

---

**COL_SCHEDULER_NAME**

static final `String` `COL_SCHEDULER_NAME`

See Also:
Constant Field Values

---

**COL_JOB_NAME**

static final `String` `COL_JOB_NAME`
COL_JOB_GROUP

static final String COL_JOB_GROUP

See Also:
Constant Field Values

COL_IS_DURABLE

static final String COL_IS_DURABLE

See Also:
Constant Field Values

COL_IS_VOLATILE

static final String COL_IS_VOLATILE

See Also:
Constant Field Values

COL_IS_NONCONCURRENT

static final String COL_IS_NONCONCURRENT

See Also:
Constant Field Values

COL_IS_UPDATE_DATA

static final String COL_IS_UPDATE_DATA
See Also:
Constant Field Values

---

**COL_REQUESTS_RECOVERY**

static final `String` `COL_REQUESTS_RECOVERY`

See Also:
Constant Field Values

---

**COL_JOB_DATAMAP**

static final `String` `COL_JOB_DATAMAP`

See Also:
Constant Field Values

---

**COL_JOB_CLASS**

static final `String` `COL_JOB_CLASS`

See Also:
Constant Field Values

---

**COL_DESCRIPTION**

static final `String` `COL_DESCRIPTION`

See Also:
Constant Field Values

---

**COL_TRIGGER_NAME**

static final `String` `COL_TRIGGER_NAME`
See Also:

Constant Field Values

---

**COL_TRIGGER_GROUP**

static final `String` `COL_TRIGGER_GROUP`

See Also:

Constant Field Values

---

**COL_NEXT_FIRE_TIME**

static final `String` `COL_NEXT_FIRE_TIME`

See Also:

Constant Field Values

---

**COL_PREV_FIRE_TIME**

static final `String` `COL_PREV_FIRE_TIME`

See Also:

Constant Field Values

---

**COL_TRIGGER_STATE**

static final `String` `COL_TRIGGER_STATE`

See Also:

Constant Field Values

---

**COL_TRIGGER_TYPE**

static final `String` `COL_TRIGGER_TYPE`
See Also:
Constant Field Values

COL_START_TIME
static final String COL_START_TIME

See Also:
Constant Field Values

COL_END_TIME
static final String COL_END_TIME

See Also:
Constant Field Values

COL_PRIORITY
static final String COL_PRIORITY

See Also:
Constant Field Values

COL_MISFIRE_INSTRUCTION
static final String COL_MISFIRE_INSTRUCTION

See Also:
Constant Field Values

ALIAS_COL_NEXT_FIRE_TIME
static final String ALIAS_COL_NEXT_FIRE_TIME
See Also:

Constant Field Values

---

**COL_REPEAT_COUNT**

static final `String` `COL_REPEAT_COUNT`

See Also:

Constant Field Values

---

**COL_REPEAT_INTERVAL**

static final `String` `COL_REPEAT_INTERVAL`

See Also:

Constant Field Values

---

**COL_TIMES_TRIGGERED**

static final `String` `COL_TIMES_TRIGGERED`

See Also:

Constant Field Values

---

**COL_CRON_EXPRESSION**

static final `String` `COL_CRON_EXPRESSION`

See Also:

Constant Field Values

---

**COL_BLOB**

static final `String` `COL_BLOB`
See Also:
  Constant Field Values

---

**COL_TIME_ZONE_ID**

static final String COL_TIME_ZONE_ID

See Also:
  Constant Field Values

---

**COL_INSTANCE_NAME**

static final String COL_INSTANCE_NAME

See Also:
  Constant Field Values

---

**COL_FIRED_TIME**

static final String COL_FIRED_TIME

See Also:
  Constant Field Values

---

**COL_ENTRY_ID**

static final String COL_ENTRY_ID

See Also:
  Constant Field Values

---

**COL_ENTRY_STATE**

static final String COL_ENTRY_STATE
See Also:  
Constant Field Values

---

**COL_CALENDAR_NAME**

static final `String` COL_CALENDAR_NAME

See Also:  
Constant Field Values

---

**COL_CALENDAR**

static final `String` COL_CALENDAR

See Also:  
Constant Field Values

---

**COL_LOCK_NAME**

static final `String` COL_LOCK_NAME

See Also:  
Constant Field Values

---

**COL_LAST_CHECKIN_TIME**

static final `String` COL_LAST_CHECKIN_TIME

See Also:  
Constant Field Values

---

**COL_CHECKIN_INTERVAL**

static final `String` COL_CHECKIN_INTERVAL
See Also:
Constant Field Values

DEFAULT_TABLE_PREFIX

static final String DEFAULT_TABLE_PREFIX

See Also:
Constant Field Values

STATE_WAITING

static final String STATE_WAITING

See Also:
Constant Field Values

STATE_ACQUIRED

static final String STATE_ACQUIRED

See Also:
Constant Field Values

STATE_EXECUTING

static final String STATE_EXECUTING

See Also:
Constant Field Values

STATE_COMPLETE

static final String STATE_COMPLETE
See Also:  
Constant Field Values

---

**STATE_BLOCKED**

static final String STATE_BLOCKED

See Also:  
Constant Field Values

---

**STATE_ERROR**

static final String STATE_ERROR

See Also:  
Constant Field Values

---

**STATE_PAUSED**

static final String STATE_PAUSED

See Also:  
Constant Field Values

---

**STATE_PAUSED_BLOCKED**

static final String STATE_PAUSED_BLOCKED

See Also:  
Constant Field Values

---

**STATE_DELETED**

static final String STATE_DELETED
See Also:
Constant Field Values

STATE_MISFIRED

static final String STATE_MISFIRED

Deprecated. Whether a trigger has misfired is no longer a state, but rather now identified dynamically by whether the trigger's next fire time is more than the misfire threshold time in the past.
See Also:
Constant Field Values

ALL_GROUPS_PAUSED

static final String ALL_GROUPS_PAUSED

See Also:
Constant Field Values

TTYPE_SIMPLE

static final String TTYPE_SIMPLE

See Also:
Constant Field Values

TTYPE_CRON

static final String TTYPE_CRON

See Also:
Constant Field Values
TTYTYPE_CAL_INT

static final String TTYPE_CAL_INT

See Also:
    Constant Field Values

TTYTYPE_BLOB

static final String TTYPE_BLOB

See Also:
    Constant Field Values

Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.jdbcjobstore Class CronTriggerPersistenceDelegate

java.lang.Object
   org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

All Implemented Interfaces:
   Constants, StdJDBCConstants, TriggerPersistenceDelegate

public class CronTriggerPersistenceDelegate
extends Object
implements TriggerPersistenceDelegate, StdJDBCConstants

Nested Class Summary

Nested classes/interfaces inherited from interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate
TriggerPersistenceDelegate.TriggerPropertyBundle

Field Summary

protected String schedNameLiteral
protected String tablePrefix

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOB_DETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETE_CALENDAR, DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS,
DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL,
DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERT_CALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR,
SELECT_CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FOR_CALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants
ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CronTriggerPersistenceDelegate()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean canHandleTriggerType(org.quartz.spi.OperableTrigger trigger)</td>
</tr>
<tr>
<td>int deleteExtendedTriggerProperties(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>String getHandledTriggerTypeDiscriminator()</td>
</tr>
<tr>
<td>void initialize(String tablePrefix, JobDetail jobDetail)</td>
</tr>
<tr>
<td>int insertExtendedTriggerProperties(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail)</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle loadExtendedTriggerProperties()</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object
close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

tablePrefix

protected String tablePrefix

schedNameLiteral

protected String schedNameLiteral

Constructor Detail

CronTriggerPersistenceDelegate

public CronTriggerPersistenceDelegate()

Method Detail

initialize

public void initialize(String tablePrefix, String schedName)
Specified by:
initialize in interface TriggerPersistenceDelegate

getHandledTriggerTypeDiscriminator

public String getHandledTriggerTypeDiscriminator()

Specified by:
getHandledTriggerTypeDiscriminator in interface TriggerPersistenceDelegate

canHandleTriggerType

public boolean canHandleTriggerType(org.quartz.spi.OperableTrigger trigger)

Specified by:
canHandleTriggerType in interface TriggerPersistenceDelegate

deleteExtendedTriggerProperties

public int deleteExtendedTriggerProperties(Connection conn, TriggerKey triggerKey)
    throws SQLException

Specified by:
deleteExtendedTriggerProperties in interface TriggerPersistenceDelegate

Throws:
SQLException

insertExtendedTriggerProperties

public int insertExtendedTriggerProperties(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
    throws SQLException,
IOException

Specified by:
   insertExtendedTriggerProperties in interface
   TriggerPersistenceDelegate

Throws:
   SQLException
   IOException

loadExtendedTriggerProperties

public TriggerPersistenceDelegate.TriggerPropertyBundle loadExtended

Specified by:
   loadExtendedTriggerProperties in interface
   TriggerPersistenceDelegate

Throws:
   SQLException

updateExtendedTriggerProperties

public int updateExtendedTriggerProperties(Connection conn,
                                           org.quartz.spi.OperableTrigger trigger,
                                           String state,
                                           JobDetail jobDetail)

throws SQLException,
         IOException

Specified by:
   updateExtendedTriggerProperties in interface
   TriggerPersistenceDelegate

Throws:
   SQLException
   IOException
Class DB2v6Delegate

java.lang.Object
   ▼ org.quartz.impl.jdbcjobstore.StdJDBCDelegate
      ▼ org.quartz.impl.jdbcjobstore.DB2v6Delegate

All Implemented Interfaces:
   Constants, DriverDelegate, StdJDBCConstants

public class DB2v6Delegate
   extends StdJDBCDelegate

Quartz JDBC delegate for DB2 v6 databases. select count(name) had to be replaced with select count(*).

Author:
   Martin Renner, James House

Field Summary

<table>
<thead>
<tr>
<th>static String</th>
<th>SELECT_NUM_CALENDARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>SELECT_NUM_JOBS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NUM_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NUM_TRIGGERS_FOR_JOB</td>
</tr>
</tbody>
</table>

Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

Fields inherited from interface
org.quartz.impl.jdbcjobstore.StdJDBCConstants

COUNT MISFIRED TRIGGERS IN STATE, DELETE ALL BLOB TRIGGERS,
DELETE ALL CALENDARS, DELETE ALL CRON TRIGGERS,
DELETE ALL JOB DETAILS, DELETE ALL PAUSED TRIGGER GRPS,
DELETE ALL SIMPLE TRIGGERS, DELETE ALL SIMPROP TRIGGERS,
DELETE ALL TRIGGERS, DELETE BLOB TRIGGER, DELETE CALENDAR,
DELETE CRON TRIGGER, DELETE FIRED TRIGGER, DELETE FIRED TRIGGERS,
DELETE INSTANCES FIRED TRIGGERS, DELETE JOB DETAIL,
DELETE NO RECOVERY FIRED TRIGGERS, DELETE PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERT_CALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR,
SELECT CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_TRIGGERS_IN_GROUP,
SELECT_PAUSED_TRIGGER_GROUP, SELECT_PAUSED_TRIGGER_GROUPS,
SELECT_REFERENCEDCALENDAR, SELECT_SCHEDULER_STATE,
SELECT_SCHEDULER_STATES, SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER,
SELECT_TRIGGER_DATA, SELECT_TRIGGER_EXISTENCE,
SELECT_TRIGGER_FOR_FIRE_TIME, SELECT_TRIGGER_GROUPS,
SELECT_TRIGGER_GROUPS_FILTERED, SELECT_TRIGGER_STATE,
SELECT_TRIGGER_STATUS, SELECT_TRIGGERS_FORCALENDAR,
SELECT_TRIGGERS FOR JOB, SELECT_TRIGGERS_IN_GROUP,
SELECT_TRIGGERS_IN_STATE, TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER,
UPDATE_CALENDAR, UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants
ALIAS	COL	NEXT	FIRE	TIME,	ALL	GROUPS	PAUSED,	COL	BLOB,
COL	CALENDAR,	COL	CALENDAR	NAME,	COL	CHECKIN	INTERVAL,
COL	CRON	EXPRESSION,	COL	DESCRIPTION,	COL	END	TIME,	COL	ENTRY	ID,
COL	ENTRY	STATE,	COL	FIRE	TIME,	COL	INSTANCE	NAME,
COL	IS	DURABLE,	COL	IS	NONCONCURRENT,	COL	IS	UPDATE	DATA,
COL	IS	VOLATILE,	COL	JOB	CLASS,	COL	JOB	DATAMAP,	COL	JOB	GROUP,
COL	JOB	NAME,	COL	LAST	CHECKIN	TIME,	COL	LOCK	NAME,
COL	MISFIRE	INSTRUCTION,	COL	NEXT	FIRE	TIME,	COL	PREV	FIRE	TIME,
COL	PRIORITY,	COL	REPEAT	COUNT,	COL	REPEAT	INTERVAL,
COL	REQUESTS	RECOVERY,	COL	SCHEDULER	NAME,	COL	START	TIME,
COL	TIME	ZONE	ID,	COL	TIMES	TRIGGERED,	COL	TRIGGER	GROUP,
COL	TRIGGER	NAME,	COL	TRIGGER	STATE,	COL	TRIGGER	TYPE,
DEFAULT	TABLE	PREFIX,	STATE	ACQUIRED,	STATE	BLOCKED,
STATE	COMPLETE,	STATE	DELETED,	STATE	ERROR,	STATE	EXECUTING,
STATE	MISFIRE,	STATE	PAUSED,	STATE	PAUSED	BLOCKED,	STATE	WAITING,
TABLE	BLOB	TRIGGERS,	TABLE	CALENDARS,	TABLE	CRON	TRIGGERS,
TABLE	FIRED	TRIGGERS,	TABLE	JOB	DETAILS,	TABLE	LOCKS,
TABLE	PAUSED	TRIGGERS,	TABLE	SCHEDULER	STATE,
TABLE	SIMPLE	TRIGGERS,	TABLE	TRIGGERS,	TTYPE	BLOB,	TTYPE	CAL	INT,
TTYPE	CRON,	TTYPE	SIMPLE

Constructor Summary

DB2v6Delegate(org.slf4j.Logger	logger,	String	tablePrefix,
String	schedName,	String	instanceId,
org.quartz.spi.ClassLoadHelper
classLoadHelper)

DB2v6Delegate(org.slf4j.Logger	logger,	String	tablePrefix,
String	schedName,	String	instanceId,
org.quartz.spi.ClassLoadHelper
classLoadHelper,
Boolean	useProperties)

Method Summary

int	selectNumCalendars(Connection	conn)
   Select the total number of calendars stored.

int	selectNumJobs(Connection	conn)
Select the total number of jobs stored.

```java
int selectNumTriggers(Connection conn)
```

Select the total number of triggers stored.

```java
int selectNumTriggersForJob(Connection conn, JobKey jobKey)
```

Select the number of triggers associated with a given job.

---

**Methods inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate**

- addDefaultTriggerPersistenceDelegates
- addTriggerPersistenceDelegate, calendarExists
- calendarIsReferenced, canUseProperties, clearData, closeResultSet
- closeStatement, convertFromProperty, convertToProperty
- countMisfiredTriggersInState, deleteAllPausedTriggerGroups
- deleteBlobTrigger, deleteCalendar, deleteFiredTrigger
- deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail
- deletePausedTriggerGroup, deletePausedTriggerGroup
- deleteSchedulerState, deleteTrigger, deleteTriggerExtension
- findTriggerPersistenceDelegate, findTriggerPersistenceDelegate
- getBoolean, getBoolean, getJobDataFromBlob
- getKeyOfNonSerializableValue, getObjectFromBlob
- getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize
- insertBlobTrigger, insertCalendar, insertFiredTrigger
- insertJobDetail, insertPausedTriggerGroup, insertSchedulerState
- insertTrigger, isExistingTriggerGroup, isJobNonConcurrent
- isTriggerGroupPaused, jobExists, rtp, selectCalendar
- selectCalendars, selectFiredTriggerInstanceNames
- selectFiredTriggerRecords, selectFiredTriggerRecordsByJob
- selectInstancesFiredTriggerRecords, selectJobDetail
- selectJobExecutionCount, selectJobForTrigger, selectJobGroups
- selectJobsInGroup, selectMisfiredTriggers
- selectMisfiredTriggersInGroupInState
- selectMisfiredTriggersInState, selectNextFireTime
- selectPausedTriggerGroups, selectSchedulerStateRecords
- selectTrigger, selectTriggerForFireTime, selectTriggerGroups
- selectTriggerGroups, selectTriggerJobDataMap
- selectTriggerKeysForJob, selectTriggersForCalendar
- selectTriggersForJob, selectTriggersForRecoveringJobs
- selectTriggersInGroup, selectTriggersInState, selectTriggerState
- selectTriggerStatus, selectTriggerToAcquire, serializeJobData
- serializeObject, setBoolean, setBytes, toSqlLikeClause
- triggerExists, updateBlobTrigger, updateCalendar
- updateFiredTrigger, updateJobData, updateJobDetail
- updateSchedulerState, updateTrigger
- updateTriggerGroupStateFromOtherState

---
**updateTriggerGroupStateFromOtherStates**, **updateTriggerState**, **updateTriggerStateFromOtherState**, **updateTriggerStateFromOtherStates**, **updateTriggerStatesForJob**, **updateTriggerStatesForJobFromOtherState**, **updateTriggerStatesFromOtherStates**

**Methods inherited from class java.lang.**[Object](#)

| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait |

---

**Field Detail**

**SELECT_NUM_JOBS**

public static final [String](#) **SELECT_NUM_JOBS**

See Also:

[Constant Field Values](#)

---

**SELECT_NUM_TRIGGERS_FOR_JOB**

public static final [String](#) **SELECT_NUM_TRIGGERS_FOR_JOB**

See Also:

[Constant Field Values](#)

---

**SELECT_NUM_TRIGGERS**

public static final [String](#) **SELECT_NUM_TRIGGERS**

See Also:

[Constant Field Values](#)

---

**SELECT_NUMCALENDARS**
public static final String SELECT_NUM_CALENDARS

See Also:
Constant Field Values

Constructor Detail

DB2v6Delegate

public DB2v6Delegate (org.slf4j.Logger logger,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper)

DB2v6Delegate

public DB2v6Delegate (org.slf4j.Logger logger,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper,
Boolean useProperties)

Method Detail

selectNumJobs

public int selectNumJobs (Connection conn)
throws SQLException

Description copied from class: StdJDBCDelegate

Select the total number of jobs stored.

Specified by:
selectNumJobs in interface DriverDelegate

Overrrides:
selectNumJobs in class StdJDBCDelegate
Parameters:
    conn - the DB Connection

Returns:
    the total number of jobs stored

Throws:
    SQLException

selectNumTriggersForJob

public int selectNumTriggersForJob(Connection conn,
                                    JobKey jobKey)
    throws SQLException

Description copied from class: StdJDBCDelegate

Select the number of triggers associated with a given job.

Specified by:
    selectNumTriggersForJob in interface DriverDelegate

Overrides:
    selectNumTriggersForJob in class StdJDBCDelegate

Parameters:
    conn - the DB Connection

Returns:
    the number of triggers for the given job

Throws:
    SQLException

selectNumTriggers

public int selectNumTriggers(Connection conn)
    throws SQLException

Description copied from class: StdJDBCDelegate

Select the total number of triggers stored.

Specified by:
selectNumTriggers in interface DriverDelegate

Overrides:
selectNumTriggers in class StdJDBCDelegate

Parameters:
conn - the DB Connection

Returns:
the total number of triggers stored

Throws:
SQLException

selectNumCalendars

public int selectNumCalendars(Connection conn)
throws SQLException

Description copied from class: StdJDBCDelegate

Select the total number of calendars stored.

Specified by:
selectNumCalendars in interface DriverDelegate

Overrides:
selectNumCalendars in class StdJDBCDelegate

Parameters:
conn - the DB Connection

Returns:
the total number of calendars stored

Throws:
SQLException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Class DB2v7Delegate

public class DB2v7Delegate
extends StdJDBCDelegate

Quartz JDBC delegate for DB2 v7 databases.

This differs from the StdJDBCDelegate in that it stores boolean values in an varchar(1) column, and saves serialized data in a byte array using PreparedStatement.setObject(int, java.lang.Object, int) rather than PreparedStatement.setBytes(int, byte[]).

Author:
  Blair Jensen

Field Summary

Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETE_CALENDAR,
Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants
### Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB2v7Delegate</strong></td>
<td>(org.slf4j.Logger logger, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)</td>
</tr>
<tr>
<td><strong>DB2v7Delegate</strong></td>
<td>(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td><strong>setBoolean</strong> (PreparedStatement ps, int index, boolean val)</td>
</tr>
<tr>
<td>void</td>
<td>Sets the designated parameter to the given Java boolean value.</td>
</tr>
<tr>
<td>protected</td>
<td><strong>setBytes</strong> (PreparedStatement ps, int index, ByteArrayOutputStream baos)</td>
</tr>
<tr>
<td>void</td>
<td>Sets the designated parameter to the byte array of the given ByteArrayOutputStream.</td>
</tr>
</tbody>
</table>
Methods inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet, closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getJobDataFromBlob, getKeyOfNonSerializableValue, getObjectFromBlob, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectMisfiredTriggersInState, selectNextFireTime, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates
Methods inherited from class java.lang.**Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Constructor Detail**

**DB2v7Delegate**

public **DB2v7Delegate**(org.slf4j.Logger logger,
                       String tablePrefix,
                       String schedName,
                       String instanceId,
                       org.quartz.spi.ClassLoadHelper classLoadHelper)

**DB2v7Delegate**

public **DB2v7Delegate**(org.slf4j.Logger log,
                        String tablePrefix,
                        String schedName,
                        String instanceId,
                        org.quartz.spi.ClassLoadHelper classLoadHelper,
                        Boolean useProperties)

**Method Detail**

**setBytes**

protected void **setBytes**(PreparedStatement ps,
                            int index,
                            ByteArrayOutputStream baos)

throws SQLException

Sets the designated parameter to the byte array of the given ByteArrayOutputStream. Will set parameter value to null if the ByteArrayOutputStream is null. Wraps PreparedStatement.setObject(int, java.lang.Object, int) rather than PreparedStatement.setBytes(int, byte[]) as required by the DB2
v7 database.

Overrides:

    setBytes in class StdJDBCDelegate

Throws:

    SQLException

setBoolean

protected void setBoolean(PreparedStatement ps,
                          int index,
                          boolean val)
                      throws SQLException

Sets the designated parameter to the given Java boolean value. This translates the boolean to 1/0 for true/false.

Overrides:

    setBoolean in class StdJDBCDelegate

Throws:

    SQLException

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
</tr>
</tbody>
</table>
public class **DB2v8Delegate**

extends **StdJDBCDelegate**

Quartz JDBC delegate for DB2 v8 databases.

This differs from the StdJDBCDelegate in that it stores boolean values in an integer column.

**Author:**
Blair Jensen

---

**Field Summary**

**Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate**

classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

**Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants**

COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETECALENDAR, DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERT_CALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR,
SELECT_CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUMCALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCEDCALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FORCALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.**Constants**

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,</td>
</tr>
<tr>
<td>COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL,</td>
</tr>
<tr>
<td>COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID</td>
</tr>
</tbody>
</table>
## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB2v8Delegate</strong></td>
<td>(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, Boolean useProperties, org.quartz.spi.ClassLoadHelper classLoadHelper)</td>
</tr>
<tr>
<td><strong>DB2v8Delegate</strong></td>
<td>(org.slf4j.Logger logger, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protected void setBoolean</strong></td>
<td>(PreparedStatement ps, int index, boolean val) Sets the designated parameter to the given Java boolean value.</td>
</tr>
</tbody>
</table>

## Methods inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet, closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger,</td>
<td></td>
</tr>
<tr>
<td>deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getJobDataFromBlob, getKeyOfNonSerializableValue, getObjectFromBlob, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectMisfiredTriggersInState, selectNextFireTime, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.**Object**  
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail
DB2v8Delegate

public DB2v8Delegate(org.slf4j.Logger logger,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper)

DB2v8Delegate

public DB2v8Delegate(org.slf4j.Logger log,
String tablePrefix,
String schedName,
String instanceId,
Boolean useProperties,
org.quartz.spi.ClassLoadHelper classLoadHelper)

Method Detail

setBoolean

protected void setBoolean(PreparedStatement ps,
int index,
boolean val)
throws SQLException

Sets the designated parameter to the given Java boolean value. This translates the boolean to 1/0 for true/false.

Overrides:
setBoolean in class StdJDBCDelegate

Throws:
SQLException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
org.quartz.impl.jdbcjobstore  Class DBSemaphore

java.lang.Object
   ↓ org.quartz.impl.jdbcjobstore.DBSemaphore

All Implemented Interfaces:
   Constants, Semaphore, StdJDBCClConstants, TablePrefixAware

Direct Known Subclasses:
   StdRowLockSemaphore, UpdateLockRowSemaphore

public abstract class DBSemaphore
extends Object
implements Semaphore, Constants, StdJDBCClConstants, TablePrefixAware

Base class for database based lock handlers for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.

Field Summary

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCClConstants

COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS,
DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS,
DELETE_ALL_JOB_DETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS,
DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS,
DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETECALENDAR,
DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS,
DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL,
DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERTCALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECTCALENDAR,
SELECT CALENDAR_EXISTENCE, SELECT CALENDARS, SELECT CRON TRIGGER, SELECT FIRED_TRIGGER, SELECT FIRED_TRIGGER_GROUP, SELECT FIRED_TRIGGER_INSTANCE_NAMES, SELECT FIRED_TRIGGERS, SELECT FIRED_TRIGGERS_OF_JOB, SELECT FIRED_TRIGGERS_OF_JOB_GROUP, SELECT HAS_MISFIRED_TRIGGERS_IN_STATE, SELECT INSTANCES FIRED_TRIGGERS, SELECT INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT JOB_DETAIL, SELECT JOB_EXECUTION_COUNT, SELECT JOB_EXISTENCE, SELECT JOB_FOR_TRIGGER, SELECT JOB_GROUPS, SELECT JOB_NONCONCURRENT, SELECT JOBS_IN_GROUP, SELECT MISFIRED_TRIGGERS, SELECT MISFIRED_TRIGGERS_IN_GROUP_IN_STATE, SELECT MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME, SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS, SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB, SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP, SELECT_REFERENCED_CALENDAR, SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES, SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA, SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME, SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED, SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS, SELECT_TRIGGERS_FORCALENDAR, SELECT_TRIGGERS_FOR_JOB, SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE, TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATECALENDAR, UPDATE_CRON_TRIGGER, UPDATE FIRED_TRIGGER, UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE JOB_DATA, UPDATEJOB_DETAIL, UPDATE JOB_TRIGGER_STATES, UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE, UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER, UPDATE_TRIGGER_GROUP_STATE_FROM_STATE, UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA, UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE, UPDATE_TRIGGER_STATE_FROMSTATES, UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants

ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL,
COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID,
COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME,
COL_IS_DURABLE, COL_IS_NONCONCURRENT, COL_IS_UPDATE_DATA,
COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP,
COL_JOB_NAME, COL_LAST_CHECKIN_TIME, COL_LOCK_NAME,
COL_MISFIRE_INSTRUCTION, COL_NEXT_FIRE_TIME, COL_PREV_FIRE_TIME,
COL_PRIORITY, COL_REPEAT_COUNT, COL_REPEAT_INTERVAL,
### Constructor Summary

**DBSemaphore**(`String` tablePrefix, `String` schedName, `String` defaultSQL, `String` defaultInsertSQL)

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected abstract void executeSQL(Connection conn, String lockName, String expandedSql, String expandedInsertSql)</code></td>
<td>Execute the SQL that will lock the proper database row.</td>
</tr>
<tr>
<td><code>protected org.slf4j.Logger getLog()</code></td>
<td></td>
</tr>
<tr>
<td><code>String getSchedName()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected String getSchedulerNameLiteral()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected String getSQL()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected String getTablePrefix()</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean isLockOwner(Connection conn, String lockName)</code></td>
<td>Determine whether the calling thread owns a lock on the identified resource.</td>
</tr>
<tr>
<td><code>boolean obtainLock(Connection conn, String lockName)</code></td>
<td>Grants a lock on the identified resource to the calling thread (blocking until it is available).</td>
</tr>
</tbody>
</table>
void releaseLock(Connection conn, String lockName)
    Release the lock on the identified resource if it is held by the calling thread.

boolean requiresConnection()
    This Semaphore implementation does use the database.

protected void setInsertSQL(String insertSql)

void setSchedName(String schedName)

protected void setSQL(String sql)

void setTablePrefix(String tablePrefix)

Methods inherited from class java.lang.**Object**
clon, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Constructor Detail**

**DBSemaphore**

public **DBSemaphore**(String tablePrefix, String schedName, String defaultSQL, String defaultInsertSQL)

**Method Detail**

getLog

protected org.slf4j.Logger **getLog**()

executeSQL
protected abstract void **executeSQL** (**Connection** conn, **String** lockName, **String** expandedSQL, **String** expandedInsertSQL) throws **LockException**

Execute the SQL that will lock the proper database row.

**Throws:**
   **LockException**

---

**obtainLock**

public boolean **obtainLock** (**Connection** conn, **String** lockName) throws **LockException**

Grants a lock on the identified resource to the calling thread (blocking until it is available).

**Specified by:**
   **obtainLock** in interface **Semaphore**

**Parameters:**
   conn - Database connection used to establish lock. Can be null if Semaphore.requiresConnection() returns false.

**Returns:**
   true if the lock was obtained.

**Throws:**
   **LockException**

---

**releaseLock**

public void **releaseLock** (**Connection** conn, **String** lockName)

Release the lock on the identified resource if it is held by the calling thread.

**Specified by:**
   **releaseLock** in interface **Semaphore**
Parameters:

conn - Database connection used to establish lock. Can be null if Semaphore.requiresConnection() returns false.

---

**isLockOwner**

```java
public boolean isLockOwner(Connection conn, String lockName)
```

Determine whether the calling thread owns a lock on the identified resource.

Specified by:

`isLockOwner` in interface `Semaphore`

Parameters:

conn - Database connection used to establish lock. Can be null if Semaphore.requiresConnection() returns false.

---

**requiresConnection**

```java
public boolean requiresConnection()
```

This Semaphore implementation does use the database.

Specified by:

`requiresConnection` in interface `Semaphore`

See Also:

`Semaphore.isLockOwner(Connection, String)`,
`Semaphore.obtainLock(Connection, String)`,
`Semaphore.releaseLock(Connection, String)`

---

**getSQL**

```java
protected String getSQL()
```
setSQL

protected void setSQL(String sql)
______________

setInsertSQL

protected void setInsertSQL(String insertSql)
______________

getSchedulerNameLiteral

protected String getSchedulerNameLiteral()
______________

getSchedName

public String getSchedName()
______________

setSchedName

public void setSchedName(String schedName)

    Specified by:
        setSchedName in interface TablePrefixAware
______________

getTablePrefix

protected String getTablePrefix()
______________

setTablePrefix

public void setTablePrefix(String tablePrefix)

    Specified by:
        setTablePrefix in interface TablePrefixAware
______________
Copyright 2001-2011, Terracotta, Inc.
public interface DriverDelegate

This is the base interface for all driver delegate classes.

This interface is very similar to the JobStore interface except each method has an additional Connection parameter.

Unless a database driver has some **extremely-DB-specific** requirements, any DriverDelegate implementation classes should extend the StdJDBCDelegate class.

**Author:**
Jeffrey Wescott, James House

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean calendarExists(Connection conn, String calendarName)</td>
</tr>
<tr>
<td>Check whether or not a calendar exists.</td>
</tr>
<tr>
<td>boolean calendarIsReferenced(Connection conn, String calendarName)</td>
</tr>
<tr>
<td>Check whether or not a calendar is referenced by any triggers.</td>
</tr>
<tr>
<td>void clearData(Connection conn)</td>
</tr>
<tr>
<td>Clear (delete!) all scheduling data - all Job Calendars.</td>
</tr>
<tr>
<td>int countMisfiredTriggersInState(Connection conn, String state, long ts)</td>
</tr>
<tr>
<td>Get the number of triggers in the given state that have misfired.</td>
</tr>
</tbody>
</table>
misfired - according to the given timestamp.

```java
int deleteAllPausedTriggerGroups(Connection conn)
```  
Delete all paused trigger groups.

```java
int deleteCalendar(Connection conn, String calendarName)
```  
Delete a calendar.

```java
int deleteFiredTrigger(Connection conn, String triggerName)
```  
Delete a fired trigger.

```java
int deleteFiredTriggers(Connection conn)
```  
Delete all fired triggers.

```java
int deleteFiredTriggers(Connection conn, String groupName)
```  
Delete all fired triggers of the given instance.

```java
int deleteJobDetail(Connection conn, JobKey jobKey)
```  
Delete the job detail record for the given job.

```java
int deletePausedTriggerGroup(Connection conn, GroupMatcher<TriggerKey> matcher)
```  
Delete a paused trigger group.

```java
int deletePausedTriggerGroup(Connection conn, String groupName)
```  
Delete a paused trigger group.

```java
int deleteSchedulerState(Connection conn, String state, long ts, int count, List<TriggerKey> resultList)
```  
Delete a scheduler-instance state record.

```java
int deleteTrigger(Connection conn, TriggerKey triggerKey)
```  
Delete the base trigger data for a trigger.

```java
boolean hasMisfiredTriggersInState(Connection conn, String state, long ts, int count, List<TriggerKey> resultList)
```  
Get the names of all of the triggers in the given state that have misfired - according to the given timestamp.

```java
void initialize(String initString)
```  
Initialize the system.

```java
int insertCalendar(Connection conn, String Calendar calendar)
```  
Insert a new calendar.

```java
int insertFiredTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail)
```  
Insert a fired trigger.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>insertJobDetail(Connection conn, JobDetail jobDetail)</code></td>
<td>Insert a job detail record.</td>
</tr>
<tr>
<td><code>insertPausedTriggerGroup(Connection conn, String groupName)</code></td>
<td>Insert a paused trigger group.</td>
</tr>
<tr>
<td><code>insertSchedulerState(Connection conn, long checkInTime, long interval)</code></td>
<td>Insert a scheduler-instance state record.</td>
</tr>
<tr>
<td><code>insertTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail)</code></td>
<td>Insert the base trigger data.</td>
</tr>
<tr>
<td><code>isExistingTriggerGroup(Connection conn, String groupName)</code></td>
<td>Check whether or not the given trigger group exists.</td>
</tr>
<tr>
<td><code>isJobNonConcurrent(Connection conn, JobKey jobKey)</code></td>
<td>Check whether or not the given job disallows concurrent execution.</td>
</tr>
<tr>
<td><code>isTriggerGroupPaused(Connection conn, String groupName)</code></td>
<td>Check whether or not the trigger group is paused.</td>
</tr>
<tr>
<td><code>jobExists(Connection conn, JobKey jobKey)</code></td>
<td>Check whether or not the given job exists.</td>
</tr>
<tr>
<td><code>selectCalendar(Connection conn, String calendarName)</code></td>
<td>Select a calendar.</td>
</tr>
<tr>
<td><code>selectCalendars(Connection conn)</code></td>
<td>Select all of the stored calendars.</td>
</tr>
<tr>
<td><code>selectFiredTriggerInstanceNames(Connection conn)</code></td>
<td>Select the distinct instance names of all fired trigger records.</td>
</tr>
<tr>
<td><code>selectFiredTriggerRecords(Connection conn, String triggerName, String groupName)</code></td>
<td>Select the states of all fired trigger records, or trigger group if trigger name is null.</td>
</tr>
<tr>
<td><code>selectFiredTriggerRecordsByJob(Connection conn, String jobName, String groupName)</code></td>
<td>Select the states of all fired trigger records for a given job or trigger group.</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>selectInstancesFiredTriggerRecords</td>
<td>Select the states of all fired-trigger records for a given scheduler instance.</td>
</tr>
<tr>
<td>selectJobDetail</td>
<td>Select the JobDetail object for a given job name.</td>
</tr>
<tr>
<td>selectJob ExecutionContext</td>
<td>Get the number of instances of the identified job currently executing.</td>
</tr>
<tr>
<td>selectJobForTrigger</td>
<td>Select the job to which the trigger is associated.</td>
</tr>
<tr>
<td>selectJobGroups</td>
<td>Select all of the job group names that are stored.</td>
</tr>
<tr>
<td>selectJobsInGroup</td>
<td>Select all of the jobs contained in a given group.</td>
</tr>
<tr>
<td>selectMisfiredTriggers</td>
<td>Get the names of all of the triggers that have misfired - according to the given timestamp.</td>
</tr>
<tr>
<td>selectMisfiredTriggersInGroupInState</td>
<td>Get the names of all of the triggers in the given group and state that have misfired - according to the given timestamp.</td>
</tr>
<tr>
<td>selectMisfiredTriggersInState</td>
<td>Get the names of all of the triggers in the given state that have misfired - according to the given timestamp.</td>
</tr>
<tr>
<td>selectNextFireTime</td>
<td>Deprecated. Does not account for misfires.</td>
</tr>
<tr>
<td>selectNumCalendars</td>
<td>Select the total number of calendars stored.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>int selectNumJobs(Connection conn)</code></td>
<td>Select the total number of jobs stored.</td>
</tr>
<tr>
<td><code>int selectNumTriggers(Connection conn)</code></td>
<td>Select the total number of triggers stored.</td>
</tr>
<tr>
<td><code>int selectNumTriggersForJob(Connection conn)</code></td>
<td>Select the number of triggers associated with a given job.</td>
</tr>
<tr>
<td><code>Set&lt;String&gt; selectPausedTriggerGroups(Connection conn)</code></td>
<td>Select all of the trigger group names that are stored.</td>
</tr>
<tr>
<td><code>List&lt;SchedulerStateRecord&gt; selectSchedulerStateRecords(Connection String instanceId)</code></td>
<td>A List of all current SchedulerStateRecords.</td>
</tr>
<tr>
<td><code>org.quartz.spi.OperableTrigger selectTrigger(Connection conn, TriggerKey)</code></td>
<td>Select a trigger.</td>
</tr>
<tr>
<td><code>List&lt;String&gt; selectTriggerGroups(Connection conn)</code></td>
<td>Select all of the trigger group names that are stored.</td>
</tr>
<tr>
<td><code>List&lt;String&gt; selectTriggerGroups(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td></td>
</tr>
<tr>
<td><code>JobDataMap selectTriggerJobDataMap(Connection conn, String triggerName, String groupName)</code></td>
<td>Select a trigger's JobDataMap.</td>
</tr>
<tr>
<td><code>List&lt;TriggerKey&gt; selectTriggerKeysForJob(Connection conn)</code></td>
<td>Get the names of all of the triggers associated with a given job.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt; selectTriggersForCalendar(Connection conn, String calName)</code></td>
<td>Select the triggers for a calendar.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt; selectTriggersForJob(Connection conn, JobKey)</code></td>
<td>Select the triggers for a job.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt; selectTriggersForRecoveringJobs(Connection)</code></td>
<td>Select all of the triggers for jobs that are requesting recovery.</td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt; selectTriggersInGroup(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Signature</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>List&lt;TriggerKey&gt;</td>
<td><code>selectTriggersInState(Connection conn, String selectTriggersInState)</code></td>
</tr>
<tr>
<td>String</td>
<td><code>selectTriggerState(Connection conn, TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>TriggerStatus</td>
<td><code>selectTriggerStatus(Connection conn, TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>List&lt;TriggerKey&gt;</td>
<td><code>selectTriggerToAcquire(Connection conn, long noLaterThan, long noEarlierThan)</code></td>
</tr>
<tr>
<td>boolean</td>
<td><code>triggerExists(Connection conn, TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>updateCalendar(Connection conn, String Calendar calendar)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>updateFiredTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>updateJobData(Connection conn, JobDetail jobDetail)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>updateJobDetail(Connection conn, JobDetail jobDetail)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>updateSchedulerState(Connection conn, String long checkInTime)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>updateTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail)</code></td>
</tr>
<tr>
<td>int</td>
<td><code>updateTriggerGroupStateFromOtherState(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher, String String oldState)</code></td>
</tr>
</tbody>
</table>
Update all of the triggers of the given group to the given new state, if they are in the given old state.

```java
public int updateTriggerGroupStateFromOtherStates(
    GroupMatcher<TriggerKey> matcher, String oldState1, String oldState2, String newState)
```

Update all triggers in the given group to the given new state, if they are in one of the given old states.

```java
public int updateTriggerState(Connection conn, TriggerKey triggerKey, String state)
```

Update the state for a given trigger.

```java
public int updateTriggerStateFromOtherState(Connection conn, TriggerKey triggerKey, String newState, String oldState1, String oldState2)
```

Update the given trigger to the given new state, if it is in the given old state.

```java
public int updateTriggerStateFromOtherStates(Connection conn, TriggerKey triggerKey, String newState, String oldState1, String oldState2)
```

Update the given trigger to the given new state, if it is one of the given old states.

```java
public int updateTriggerStatesForJob(Connection conn, JobKey jobKey, String state)
```

Update the states of all triggers associated with the given job.

```java
public int updateTriggerStatesForJobFromOtherState(Connection conn, JobKey jobKey, String state, String oldState)
```

Update the states of any triggers associated with the given job, that are the given current state.

```java
public int updateTriggerStatesFromOtherStates(String newState, String oldState1, String oldState2)
```

Update all triggers having one of the two given states, to the given new state.

---

**Method Detail**

### initialize

```java
void initialize(String initString)
```
Throws:  
NoSuchDelegateException

updateTriggerStatesFromOtherStates

int updateTriggerStatesFromOtherStates(Connection conn,  
String newState,  
String oldState1,  
String oldState2)  
throws SQLException

Update all triggers having one of the two given states, to the given new state.

Parameters:  
conn - the DB Connection  
newState - the new state for the triggers  
oldState1 - the first old state to update  
oldState2 - the second old state to update

Returns:  
number of rows updated

Throws:  
SQLException

selectMisfiredTriggers

List<TriggerKey> selectMisfiredTriggers(Connection conn,  
long ts)  
throws SQLException

Get the names of all of the triggers that have misfired - according to the given timestamp.

Parameters:  
conn - the DB Connection

Returns:  
an array of Key objects
Throws:  
`SQLException`

---

**selectMisfiredTriggersInState**

`List<TriggerKey> selectMisfiredTriggersInState(Connection conn, String state, long ts)`  
throws `SQLException`

Get the names of all of the triggers in the given state that have misfired - according to the given timestamp.

**Parameters:**
- conn - the DB Connection

**Returns:**
- an array of Key objects

**Throws:**  
`SQLException`

---

**hasMisfiredTriggersInState**

`boolean hasMisfiredTriggersInState(Connection conn, String state, long ts, int count, List<TriggerKey> resultList)`  
throws `SQLException`

Get the names of all of the triggers in the given states that have misfired - according to the given timestamp. No more than count will be returned.

**Parameters:**
- conn - the DB Connection
- count - the most misfired triggers to return, negative for all
- resultList - Output parameter. A List of Key objects. Must not be null.

**Returns:**
- Whether there are more misfired triggers left to find beyond the given
count.

**Throws:**

SQLException

---

**countMisfiredTriggersInState**

```java
int countMisfiredTriggersInState(Connection conn, String state1, long ts)
throws SQLException
```

Get the number of triggers in the given state that have misfired - according to the given timestamp.

**Parameters:**

- `conn`: the DB Connection

**Throws:**

SQLException

---

**selectMisfiredTriggersInGroupInState**

```java
List<TriggerKey> selectMisfiredTriggersInGroupInState(Connection conn, String groupName, String state, long ts)
throws SQLException
```

Get the names of all of the triggers in the given group and state that have misfired - according to the given timestamp.

**Parameters:**

- `conn`: the DB Connection

**Returns:**

- an array of Key objects

**Throws:**

SQLException

---

**selectTriggersForRecoveringJobs**
List<org.quartz.spi.OperableTrigger> selectTriggersForRecoveringJobs

Select all of the triggers for jobs that are requesting recovery. The returned trigger objects will have unique "recoverXXX" trigger names and will be in the Scheduler.DEFAULT_RECOVERY_GROUP trigger group.

In order to preserve the ordering of the triggers, the fire time will be set from the COL_FIRED_TIME column in the TABLE_FIRED_TRIGGERS table. The caller is responsible for calling computeFirstFireTime on each returned trigger. It is also up to the caller to insert the returned triggers to ensure that they are fired.

**Parameters:**
- conn - the DB Connection

**Returns:**
- an array of Trigger objects

**Throws:**
- SQLException
- IOException
- ClassNotFoundException

deleteFiredTriggers

```java
int deleteFiredTriggers(Connection conn)
throws SQLException
```

Delete all fired triggers.

**Parameters:**
- conn - the DB Connection

**Returns:**
- the number of rows deleted

**Throws:**
- SQLException

---

deleteFiredTriggers
int deleteFiredTriggers(Connection conn, String instanceId) throws SQLException

Delete all fired triggers of the given instance.

Parameters:
conn - the DB Connection

Returns:
the number of rows deleted

Throws:
SQLException

int insertJobDetail(Connection conn, JobDetail job) throws IOException, SQLException

Insert the job detail record.

Parameters:
conn - the DB Connection
job - the job to insert

Returns:
number of rows inserted

Throws:
IOException - if there were problems serializing the JobDataMap
SQLException

int updateJobDetail(Connection conn, JobDetail job) throws IOException, SQLException

Update the job detail record.
**Parameters:**
- conn - the DB Connection
- job - the job to update

**Returns:**
- number of rows updated

**Throws:**
- `IOException` - if there were problems serializing the JobDataMap
- `SQLException`

---

**selectTriggerKeysForJob**

```java
List<TriggerKey> selectTriggerKeysForJob(Connection conn, JobKey jobKey) throws SQLException
```

Get the names of all of the triggers associated with the given job.

**Parameters:**
- conn - the DB Connection

**Returns:**
- an array of Key objects

**Throws:**
- `SQLException`

---

**deleteJobDetail**

```java
int deleteJobDetail(Connection conn, JobKey jobKey) throws SQLException
```

Delete the job detail record for the given job.

**Parameters:**
- conn - the DB Connection

**Returns:**
- the number of rows deleted

**Throws:**
- `SQLException`
**isJobNonConcurrent**

boolean `isJobNonConcurrent`(*Connection conn, JobKey jobKey*)

throws `SQLException`

Check whether or not the given job disallows concurrent execution.

**Parameters:**
conn - the DB Connection

**Returns:**
true if the job exists and disallows concurrent execution, false otherwise

**Throws:**
`SQLException`

---

**jobExists**

boolean `jobExists`(*Connection conn, JobKey jobKey*)

throws `SQLException`

Check whether or not the given job exists.

**Parameters:**
conn - the DB Connection

**Returns:**
true if the job exists, false otherwise

**Throws:**
`SQLException`

---

**updateJobData**

int `updateJobData`(*Connection conn, JobDetail job*)

throws `IOException`, `SQLException`

---
Update the job data map for the given job.

**Parameters:**
- conn - the DB Connection
- job - the job to update

**Returns:**
- the number of rows updated

**Throws:**
- **IOException** - if there were problems serializing the JobDataMap
- **SQLException**

---

**selectJobDetail**

```java
JobDetail selectJobDetail(Connection conn, JobKey jobKey, org.quartz.spi.ClassLoadHelper loadHelper)
```

Throws:
- **ClassNotFoundException** - if a class found during deserialization cannot be found or if the job class could not be found
- **IOException** - if deserialization causes an error
- **SQLException**

Select the JobDetail object for a given job name / group name.

**Parameters:**
- conn - the DB Connection

**Returns:**
- the populated JobDetail object

---

**selectNumJobs**

```java
int selectNumJobs(Connection conn)
```

Throws:
- **SQLException**

Select the total number of jobs stored.

---
Parameters:

conn - the DB Connection

Returns:

the total number of jobs stored

Throws:

SQLException

selectJobGroups

List<String> selectJobGroups(Connection conn) throws SQLException

Select all of the job group names that are stored.

Parameters:

conn - the DB Connection

Returns:

an array of String group names

Throws:

SQLException

selectJobsInGroup

Set<JobKey> selectJobsInGroup(Connection conn, GroupMatcher<JobKey> matcher) throws SQLException

Select all of the jobs contained in a given group.

Parameters:

conn - the DB Connection
matcher - the group matcher to evaluate against the known jobs

Returns:

an array of String job names

Throws:

SQLException
**insertTrigger**

```java
int insertTrigger(Connection conn,
                   org.quartz.spi.OperableTrigger trigger,
                   String state,
                   JobDetail jobDetail)
throws SQLException,
        IOException
```

Insert the base trigger data.

**Parameters:**
- `conn` - the DB Connection
- `trigger` - the trigger to insert
- `state` - the state that the trigger should be stored in

**Returns:**
- the number of rows inserted

**Throws:**
- `SQLException`
- `IOException`

---

**updateTrigger**

```java
int updateTrigger(Connection conn,
                   org.quartz.spi.OperableTrigger trigger,
                   String state,
                   JobDetail jobDetail)
throws SQLException,
        IOException
```

Update the base trigger data.

**Parameters:**
- `conn` - the DB Connection
- `trigger` - the trigger to insert
- `state` - the state that the trigger should be stored in

**Returns:**
- the number of rows updated

**Throws:**
- `SQLException`
- `IOException`
triggerExists

boolean triggerExists(Connection conn, TriggerKey triggerKey)
throws SQLException

Check whether or not a trigger exists.

Parameters:
  conn - the DB Connection

Returns:
  the number of rows updated

Throws:
  SQLException

updateTriggerState

int updateTriggerState(Connection conn, TriggerKey triggerKey, String state)
throws SQLException

Update the state for a given trigger.

Parameters:
  conn - the DB Connection
  state - the new state for the trigger

Returns:
  the number of rows updated

Throws:
  SQLException

updateTriggerStateFromOtherState

int updateTriggerStateFromOtherState(Connection conn, TriggerKey triggerKey, String newState, String oldState)
Update the given trigger to the given new state, if it is in the given old state.

**Parameters:**
- conn - the DB connection
- newState - the new state for the trigger
- oldState - the old state the trigger must be in

**Returns:**
- int the number of rows updated

**Throws:**
- `SQLException`

---

**updateTriggerStateFromOtherStates**

```java
int updateTriggerStateFromOtherStates(Connection conn,
                                       TriggerKey triggerKey,
                                       String newState,
                                       String oldState1,
                                       String oldState2,
                                       String oldState3)
throws SQLException
```

Update the given trigger to the given new state, if it is one of the given old states.

**Parameters:**
- conn - the DB connection
- newState - the new state for the trigger
- oldState1 - one of the old state the trigger must be in
- oldState2 - one of the old state the trigger must be in
- oldState3 - one of the old state the trigger must be in

**Returns:**
- int the number of rows updated

**Throws:**
- `SQLException`

---

**updateTriggerGroupStateFromOtherStates**
Update all triggers in the given group to the given new state, if they are in one of the given old states.

**Parameters:**
- `conn` - the DB connection
- `matcher` - the group matcher to evaluate against the known triggers
- `newState` - the new state for the trigger
- `oldState1` - one of the old state the trigger must be in
- `oldState2` - one of the old state the trigger must be in
- `oldState3` - one of the old state the trigger must be in

**Returns:**
- `int` the number of rows updated

**Throws:**
- `SQLException`
updateTriggerStatesForJob

```java
int updateTriggerStatesForJob(Connection conn, JobKey jobKey, String state)
```

Throws: `SQLException`

Update the states of all triggers associated with the given job.

**Parameters:**
- `conn` - the DB Connection
- `state` - the new state for the triggers

**Returns:**
- the number of rows updated

**Throws:**
- `SQLException`

updateTriggerStatesForJobFromOtherState

```java
int updateTriggerStatesForJobFromOtherState(Connection conn, JobKey jobKey, String state, String oldState)
```

Throws: `SQLException`

Update the states of any triggers associated with the given job, that are the given current state.

**Parameters:**
- `conn` - the DB Connection
- `state` - the new state for the triggers
- `oldState` - the old state of the triggers

**Returns:**
- the number of rows updated

**Throws:**
SQLException

**deleteTrigger**

```java
int deleteTrigger(Connection conn, TriggerKey triggerKey) throws SQLException
```

Delete the base trigger data for a trigger.

**Parameters:**
- conn - the DB Connection

**Returns:**
- the number of rows deleted

**Throws:**
- SQLException

**selectNumTriggersForJob**

```java
int selectNumTriggersForJob(Connection conn, JobKey jobKey) throws SQLException
```

Select the number of triggers associated with a given job.

**Parameters:**
- conn - the DB Connection

**Returns:**
- the number of triggers for the given job

**Throws:**
- SQLException

**selectJobForTrigger**

```java
JobDetail selectJobForTrigger(Connection conn, org.quartz.spi.ClassLoadHelper loadHelper, TriggerKey triggerKey) throws ClassNotFoundException,
```

**Throws:**
- ClassNotFoundException,
SQLException

Select the job to which the trigger is associated.

**Parameters:**
- conn - the DB Connection

**Returns:**
- the JobDetail object associated with the given trigger

**Throws:**
- ClassNotFoundException
- SQLException

---

selectTriggersForJob

`List<org.quartz.spi.OperableTrigger>` **selectTriggersForJob**(`Connection conn`, `JobKey jobKey`) throws SQLException, IOException, JobPersistenceException

Select the triggers for a job

**Parameters:**
- conn - the DB Connection

**Returns:**
- an array of (@link org.quartz.Trigger) objects associated with a given job.

**Throws:**
- SQLException
- JobPersistenceException
- ClassNotFoundException
- IOException

---

selectTriggersForCalendar

`List<org.quartz.spi.OperableTrigger>` **selectTriggersForCalendar**(`Connection conn`, `String jobName`) throws SQLException, IOException

---
Select the triggers for a calendar

**Parameters:**
- conn - the DB Connection
- calName - the name of the calendar

**Returns:**
an array of (@link org.quartz.Trigger) objects associated with the given calendar.

**Throws:**
- SQLException
- JobPersistenceException
- ClassNotFoundException
- IOException

---

selectTrigger

org.quartz.spi.OperableTrigger selectTrigger(@param conn, TriggerKey triggerKey)

Throws
- SQLException
- ClassNotFoundException
- IOException
- JobPersistenceException

---

Select a trigger.

**Parameters:**
- conn - the DB Connection

**Returns:**
the Trigger object

**Throws:**
- JobPersistenceException
- SQLException
- ClassNotFoundException
- IOException

---

selectTriggerJobDataMap
**selectTriggerJobDataMap**

```java
JobDataMap selectTriggerJobDataMap(Connection conn, String triggerName, String groupName)
throws SQLException, ClassNotFoundException, IOException
```

Select a trigger's JobDataMap.

**Parameters:**
- conn - the DB Connection
- triggerName - the name of the trigger
- groupName - the group containing the trigger

**Returns:**
the JobDataMap of the Trigger, never null, but possibly empty.

**Throws:**
- SQLException
- ClassNotFoundException
- IOException

---

**selectTriggerState**

```java
String selectTriggerState(Connection conn, TriggerKey triggerKey)
throws SQLException
```

Select a trigger's state value.

**Parameters:**
- conn - the DB Connection

**Returns:**
the Trigger object

**Throws:**
- SQLException

---

**selectTriggerStatus**

```java
TriggerStatus selectTriggerStatus(Connection conn, TriggerKey triggerKey)
throws SQLException
```

---
Select a trigger's status (state & next fire time).

**Parameters:**
conn - the DB Connection

**Returns:**
a TriggerStatus object, or null

**Throws:**
SQLException

---

**selectNumTriggers**

```java
int selectNumTriggers(Connection conn)
throws SQLException
```

Select the total number of triggers stored.

**Parameters:**
conn - the DB Connection

**Returns:**
the total number of triggers stored

**Throws:**
SQLException

---

**selectTriggerGroups**

```java
List<String> selectTriggerGroups(Connection conn)
throws SQLException
```

Select all of the trigger group names that are stored.

**Parameters:**
conn - the DB Connection

**Returns:**
an array of String group names

**Throws:**
SQLException
selectTriggerGroups

\[
\text{List}\langle\text{String}\rangle \text{ selectTriggerGroups} (\text{Connection} \ conn, \\
\text{GroupMatcher}\langle\text{TriggerKey}\rangle \ matcher) \quad \text{throws SQLException}
\]

Throws:
\[
\text{SQLException}
\]

selectTriggersInGroup

\[
\text{Set}\langle\text{TriggerKey}\rangle \text{ selectTriggersInGroup} (\text{Connection} \ conn, \\
\text{GroupMatcher}\langle\text{TriggerKey}\rangle \ matcher) \quad \text{throws SQLException}
\]

Select all of the triggers contained in a given group.

Parameters:
- conn - the DB Connection
- matcher - to evaluate against known triggers

Returns:
a Set of TriggerKeys

Throws:
\[
\text{SQLException}
\]

selectTriggersInState

\[
\text{List}\langle\text{TriggerKey}\rangle \text{ selectTriggersInState} (\text{Connection} \ conn, \\
\text{String} \ state) \quad \text{throws SQLException}
\]

Select all of the triggers in a given state.

Parameters:
- conn - the DB Connection
- state - the state the triggers must be in

Returns:
an array of trigger Key s

Throws:
SQLException

insertPausedTriggerGroup

int insertPausedTriggerGroup(Connection conn, String groupName)
throws SQLException

Throws:
SQLException

deletePausedTriggerGroup

int deletePausedTriggerGroup(Connection conn, String groupName)
throws SQLException

Throws:
SQLException

deletePausedTriggerGroup

int deletePausedTriggerGroup(Connection conn,
GroupMatcher<TriggerKey> matcher)
throws SQLException

Throws:
SQLException

deleteAllPausedTriggerGroups

int deleteAllPausedTriggerGroups(Connection conn)
throws SQLException

Throws:
SQLException
isTriggerGroupPaused

boolean isTriggerGroupPaused(Connection conn, String groupName)

Throws:
SQLException

selectPausedTriggerGroups

Set<String> selectPausedTriggerGroups(Connection conn)

throws SQLException

Throws:
SQLException

isExistingTriggerGroup

boolean isExistingTriggerGroup(Connection conn, String groupName)

throws SQLException

Throws:
SQLException

insertCalendar

int insertCalendar(Connection conn, String calendarName, Calendar calendar)

throws IOException, SQLException

Insert a new calendar.

Parameters:
conn - the DB Connection
calendarName - the name for the new calendar
calendar - the calendar

**Returns:**
the number of rows inserted

**Throws:**
*IOException* - if there were problems serializing the calendar
*SQLException*

---

**updateCalendar**

```java
int updateCalendar(Connection conn, String calendarName, Calendar calendar) throws IOException, SQLException
```

Update a calendar.

**Parameters:**
- conn - the DB Connection
- calendarName - the name for the new calendar
- calendar - the calendar

**Returns:**
the number of rows updated

**Throws:**
*IOException* - if there were problems serializing the calendar
*SQLException*

---

**calendarExists**

```java
boolean calendarExists(Connection conn, String calendarName) throws SQLException
```

Check whether or not a calendar exists.

**Parameters:**
- conn - the DB Connection
- calendarName - the name of the calendar
Returns:
true if the trigger exists, false otherwise

Throws:
SQLException

selectCalendar

Calendar selectCalendar(Connection conn, String calendarName)
throws ClassNotFoundException, IOException, SQLException

Select a calendar.

Parameters:
conn - the DB Connection
calendarName - the name of the calendar

Returns:
the Calendar

Throws:
ClassNotFoundException - if a class found during deserialization cannot be found
IOException - if there were problems deserializing the calendar
SQLException

calendarIsReferenced

boolean calendarIsReferenced(Connection conn, String calendarName)
throws SQLException

Check whether or not a calendar is referenced by any triggers.

Parameters:
conn - the DB Connection
calendarName - the name of the calendar

Returns:
true if any triggers reference the calendar, false otherwise
deleteCalendar

```java
int deleteCalendar(Connection conn, String calendarName) throws SQLException
```

Delete a calendar.

**Parameters:**
- `conn` - the DB Connection
- `calendarName` - the name of the trigger

**Returns:**
- the number of rows deleted

**Throws:**
- `SQLException`

selectNumCalendars

```java
int selectNumCalendars(Connection conn) throws SQLException
```

Select the total number of calendars stored.

**Parameters:**
- `conn` - the DB Connection

**Returns:**
- the total number of calendars stored

**Throws:**
- `SQLException`

selectCalendars

```java
List<String> selectCalendars(Connection conn) throws SQLException
```

**Parameters:**
- `conn` - the DB Connection

**Returns:**
- a list of calendars

**Throws:**
- `SQLException`
Select all of the stored calendars.

**Parameters:**
- conn - the DB Connection

**Returns:**
a n array of String calendar names

**Throws:**
- SQLException

---

**selectNextFireTime**

```java
long selectNextFireTime(Connection conn) throws SQLException
```

**Deprecated.** *Does not account for misfires.*

Select the next time that a trigger will be fired.

**Parameters:**
- conn - the DB Connection

**Returns:**
- the next fire time, or 0 if no trigger will be fired

**Throws:**
- SQLException

---

**selectTriggerForFireTime**

```java
Key selectTriggerForFireTime(Connection conn, long fireTime) throws SQLException
```

Select the trigger that will be fired at the given fire time.

**Parameters:**
- conn - the DB Connection
- fireTime - the time that the trigger will be fired

**Returns:**
a Key representing the trigger that will be fired at the given fire time, or
null if no trigger will be fired at that time

**Throws:**

*SQLException*

---

**selectTriggerToAcquire**

*List<TriggerKey> selectTriggerToAcquire(Connection conn, long noLaterThan, long noEarlierThan)*

throws *SQLException*

Select the next trigger which will fire to fire between the two given timestamps in ascending order of fire time, and then descending by priority.

**Parameters:**

- *conn* - the DB Connection
- *noLaterThan* - highest value of `getNextFireTime()` of the triggers (exclusive)
- *noEarlierThan* - highest value of `getNextFireTime()` of the triggers (inclusive)

**Returns:**

A (never null, possibly empty) list of the identifiers (Key objects) of the next triggers to be fired.

**Throws:**

*SQLException*

---

**insertFiredTrigger**

*int insertFiredTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)*

throws *SQLException*

Insert a fired trigger.

**Parameters:**

- *conn* - the DB Connection
- *trigger* - the trigger
state - the state that the trigger should be stored in

Returns:
the number of rows inserted

Throws:
SQLException

updateFiredTrigger

int updateFiredTrigger(Connection conn,
                        org.quartz.spi.OperableTrigger trigger,
                        String state,
                        JobDetail jobDetail)
throws SQLException

Update a fired trigger record. Will update the fields "firing instance", "fire time", and "state".

Parameters:
conn - the DB Connection
trigger - the trigger
state - the state that the trigger should be stored in

Returns:
the number of rows inserted

Throws:
SQLException

selectFiredTriggerRecords

List<FiredTriggerRecord> selectFiredTriggerRecords(Connection conn,
                                                    String triggerName,
                                                    String groupName)
throws SQLException

Select the states of all fired-trigger records for a given trigger, or trigger group if trigger name is null.

Returns:
a List of FiredTriggerRecord objects.

Throws:
SQLException

selectFiredTriggerRecordsByJob

List<FiredTriggerRecord> selectFiredTriggerRecordsByJob(Connection conn, String jobName, String groupName) throws SQLException

Select the states of all fired-trigger records for a given job, or job group if job name is null.

Returns:
a List of FiredTriggerRecord objects.

Throws:
SQLException

selectInstancesFiredTriggerRecords

List<FiredTriggerRecord> selectInstancesFiredTriggerRecords(String instanceName) throws SQLException

Select the states of all fired-trigger records for a given scheduler instance.

Returns:
a List of FiredTriggerRecord objects.

Throws:
SQLException

selectFiredTriggerInstanceNames

Set<String> selectFiredTriggerInstanceNames(Connection conn) throws SQLException

Select the distinct instance names of all fired-trigger records.

This is useful when trying to identify orphaned fired triggers (a fired trigger
without a scheduler state record.)

**Returns:**
a Set of String objects.

**Throws:**
SQLException

-----

**deleteFiredTrigger**

```java
int deleteFiredTrigger(Connection conn, String entryId)
```

Delete a fired trigger.

**Parameters:**
- conn - the DB Connection
- entryId - the fired trigger entry to delete

**Returns:**
the number of rows deleted

**Throws:**
SQLException

-----

**selectJobExecutionCount**

```java
int selectJobExecutionCount(Connection conn, JobKey jobKey)
```

Get the number instances of the identified job currently executing.

**Parameters:**
- conn - the DB Connection

**Returns:**
the number instances of the identified job currently executing.

**Throws:**
SQLException
**insertSchedulerState**

```java
int insertSchedulerState(Connection conn, String instanceId, long checkInTime, long interval)
```

Insert a scheduler-instance state record.

**Parameters:**
`conn` - the DB Connection

**Returns:**
the number of inserted rows.

**Throws:**
`SQLException`

---

**deleteSchedulerState**

```java
int deleteSchedulerState(Connection conn, String instanceId)
```

Delete a scheduler-instance state record.

**Parameters:**
`conn` - the DB Connection

**Returns:**
the number of deleted rows.

**Throws:**
`SQLException`

---

**updateSchedulerState**

```java
int updateSchedulerState(Connection conn, String instanceId, long checkInTime)
```

**Throws:**
`SQLException`
Update a scheduler-instance state record.

**Parameters:**
conn - the DB Connection

**Returns:**
the number of updated rows.

**Throws:**
SQLException

---

selectSchedulerStateRecords

List&lt;SchedulerStateRecord&gt; selectSchedulerStateRecords(Connection conn, String instanceId) throws SQLException

A List of all current SchedulerStateRecords.

If instanceId is not null, then only the record for the identified instance will be returned.

**Parameters:**
conn - the DB Connection

**Throws:**
SQLException

---

clearData

void clearData(Connection conn) throws SQLException

Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

**Throws:**
JobPersistenceException
SQLException
Copyright 2001-2011, Terracotta, Inc.
public class FiredTriggerRecord
extends Object
implements Serializable

Conveys the state of a fired-trigger record.

Author: James House
See Also: Serialized Form

Constructor Summary

FiredTriggerRecord()

Method Summary

String getFireInstanceId()

String getFireInstanceState()

long getFireTimestamp()

JobKey getJobKey()

int
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getPriority()</code></td>
<td>Returns the priority of the job.</td>
</tr>
<tr>
<td><code>getSchedulerInstanceId()</code></td>
<td>Returns the scheduler instance ID of the job.</td>
</tr>
<tr>
<td><code>getTriggerKey()</code></td>
<td>Returns the trigger key of the job.</td>
</tr>
<tr>
<td><code>isJobDisallows ConcurrentExecution()</code></td>
<td>Checks if the job disallows concurrent execution.</td>
</tr>
<tr>
<td><code>isJobRequestsRecovery()</code></td>
<td>Checks if the job requests recovery.</td>
</tr>
<tr>
<td><code>setFireInstanceId(String string)</code></td>
<td>Sets the fire instance ID of the job.</td>
</tr>
<tr>
<td><code>setFireInstanceState(String string)</code></td>
<td>Sets the fire instance state of the job.</td>
</tr>
<tr>
<td><code>setFireTimestamp(long l)</code></td>
<td>Sets the fire timestamp of the job.</td>
</tr>
<tr>
<td><code>setJobDisallows ConcurrentExecution(boolean b)</code></td>
<td>Sets whether the job disallows concurrent execution.</td>
</tr>
<tr>
<td><code>setJobKey(JobKey key)</code></td>
<td>Sets the job key of the job.</td>
</tr>
<tr>
<td><code>setJobRequestsRecovery(boolean b)</code></td>
<td>Sets whether the job requests recovery.</td>
</tr>
<tr>
<td><code>setPriority(int priority)</code></td>
<td>Sets the priority of the job.</td>
</tr>
<tr>
<td><code>setSchedulerInstanceId(String string)</code></td>
<td>Sets the scheduler instance ID of the job.</td>
</tr>
<tr>
<td><code>setTriggerKey(TriggerKey key)</code></td>
<td>Sets the trigger key of the job.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object:
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
### Constructor Detail

**FiredTriggerRecord**

```java
public FiredTriggerRecord()
```

### Method Detail

**getFireInstanceId**

```java
public String getFireInstanceId()
```

**getFireTimestamp**

```java
public long getFireTimestamp()
```

**isJobDisallowsConcurrentExecution**

```java
public boolean isJobDisallowsConcurrentExecution()
```

**getJobKey**

```java
public JobKey getJobKey()
```

**getSchedulerInstanceId**

```java
public String getSchedulerInstanceId()
```

**getTriggerKey**

```java
public TriggerKey getTriggerKey()
```
getFireInstanceState

public String getFireInstanceState()

-----------------------------

setFireInstanceId

public void setFireInstanceId(String string)

-----------------------------

setFireTimestamp

public void setFireTimestamp(long l)

-----------------------------

setJobDisallowsConcurrentExecution

public void setJobDisallowsConcurrentExecution(boolean b)

-----------------------------

setJobKey

public void setJobKey(JobKey key)

-----------------------------

setSchedulerInstanceId

public void setSchedulerInstanceId(String string)

-----------------------------

setTriggerKey

public void setTriggerKey(TriggerKey key)

-----------------------------

setFireInstanceState

public void setFireInstanceState(String string)
isJobRequestsRecovery

public boolean isJobRequestsRecovery()

setJobRequestsRecovery

public void setJobRequestsRecovery(boolean b)

getPriority

public int getPriority()

setPriority

public void setPriority(int priority)
public class **HSQLDBDelegate**

extends **StdJDBCDelegate**

This is a driver delegate for the HSQLDB database.

**Author:**
James House, **Jeffrey Wescott**

---

### Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP, DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE, DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER, INSERTCALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,</td>
</tr>
</tbody>
</table>
Fields inherited from interface org.quartz.impl.jdbcjobstore.**Constants**

**ALIAS COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB, COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL, COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID, COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME, COL_IS_DURABLE, COL_IS_NONCONCURRENT, COL_IS_UPDATE_DATA, COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP,**
Constructor Summary

HSQldbDelegate(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new HSQLDBDelegate instance.

HSQldbDelegate(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)

Create new MSSQLDelegate instance.

Method Summary

protected Object getJobDataFromBlob(ResultSet rs, String colName)

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.

protected Object getObjectFromBlob(ResultSet rs, String colName)

This method should be overridden by any delegate subclasses that need special handling for BLOBs.

Methods inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate

addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet,
Methods inherited from class java.lang.Object

closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

Methods inherited from class java.lang.Object

closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

Methods inherited from class java.lang.Object

closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates
Constructor Detail

HSQLDBDelegate

public **HSQLDBDelegate**(org.slf4j.Logger log,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new HSQLDBDelegate instance.

**Parameters:**
- log - the logger to use during execution
- tablePrefix - the prefix of all table names

HSQLDBDelegate

public **HSQLDBDelegate**(org.slf4j.Logger log,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper,
Boolean useProperties)

Create new MSSQLDelegate instance.

**Parameters:**
- log - the logger to use during execution
- tablePrefix - the prefix of all table names
- useProperties - use java.util.Properties for storage

Method Detail

**getObjectFromBlob**

protected **Object** **getObjectFromBlob**(ResultSet rs,
String colName)

throws ClassNotFoundException
IOException, SQLException

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:

getObjectFromBlob in class StdJDBCDelegate

Parameters:

rs - the result set, already queued to the correct row
colName - the column name for the BLOB

Returns:

the deserialized Object from the ResultSet BLOB

Throws:

ClassNotFoundException - if a class found during deserialization cannot be found
IOException - if deserialization causes an error
SQLException

_____________________________________________________________________

getJobDataFromBlob

protected Object getJobDataFromBlob(ResultSet rs,
                                       String colName)
throws Exception

Description copied from class: StdJDBCDelegate

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:

getJobDataFromBlob in class StdJDBCDelegate

Parameters:

rs - the result set, already queued to the correct row
colName - the column name for the BLOB

Returns:
the deserialized Object from the ResultSet BLOB

**Throws:**

- `ClassNotFoundException` - if a class found during deserialization cannot be found
- `IOException` - if deserialization causes an error
- `SQLException`
Class InvalidConfigurationException

```java
java.lang.Object
    ▼ java.lang.Throwable
        ▼ java.lang.Exception
            ▼ org.quartz.impl.jdbcjobstore.InvalidConfigurationException
```

All Implemented Interfaces:

* Serializable

```java
public class InvalidConfigurationException
    extends Exception

Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.

Author:
    Jeffrey Wescott

See Also:
    Serialized Form
```

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InvalidConfigurationException()</strong></td>
</tr>
<tr>
<td><strong>InvalidConfigurationException(String msg)</strong></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods inherited from class java.lang.Throwable</strong></td>
</tr>
<tr>
<td>fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, setStackTrace, toString</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

InvalidConfigurationException

public InvalidConfigurationException(String msg)

InvalidConfigurationException

public InvalidConfigurationException()
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public class JobStoreCMT extends JobStoreSupport

JobStoreCMT is meant to be used in an application-server environment that provides container-managed-transactions. No commit / rollback will be handled by this class.

If you need commit / rollback, use JobStoreTX instead.

Author:
Jeffrey Wescott, James House, Srinivas Venkatarangaiah

<table>
<thead>
<tr>
<th>Nested Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested classes/interfaces inherited from class org.quartz.impl.jdbcjobstore.JobStoreSupport</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected boolean dontSetNonManagedTXConnectionAutoCommitFalse</td>
</tr>
<tr>
<td>protected String nonManagedTxDsName</td>
</tr>
</tbody>
</table>
protected boolean setTxIsolationLevelReadCommitted

Fields inherited from class org.quartz.impl.jdbcjobstore.JobStoreSupport

calendarCache, delegateClass, delegateClassName, delegateInitString, dsName, firstCheckIn, instanceId, instanceName, lastCheckin, LOCKCALENDARACCESS, LOCKJOBACCESS, LOCKMISFIREACCESS, LOCKSTATEACCESS, LOCKTRIGGERACCESS, maxToRecoverAtATime, sigChangeForTxCompletion, tablePrefix, useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants

ALIAS, COLNEXTFIRETIME, ALLGROUPSSTOPPED, COLBLOB, COLCALENDAR, COLCALENDARNAME, COLCHECKININTERVAL, COLCRONEXPRESSION, COLDESCRIPTION, COLENDTIME, COLENTRYID, COLENTRYSTATE, COLFIREDTIME, COLINSTANCENAME, COLISDURABLE, COLISNONCONCURRENT, COLISUPDATEDATA, COLISVOLATILE, COLJOBCLASS, COLJODATAMAP, COLJOBGROUP, COLJOBNAME, COLLASTCHECKINTIME, COLLOCKNAME, COLMISFIREINSTRUCTION, COLNEXTFIRETIME, COLPREVFIRETIME, COLPRIORITY, COLREPEATCOUNT, COLREPEATINTERVAL, COLREQUESTSRECOVERY, COLSCHEDULERNAME, COLSTARTTIME, COLTIZONEID, COLTIMESTRIGGERED, COLTRIGGERGROUP, COLTRIGGERNAME, COLTRIGGERSTATE, COLTRIGGERTYPE, DEFAULTTABLEPREFIX, STATEACQUIRED, STATEBLOCKED, STATECOMPLETE, STATEDELETED, STATERROR, STATEEXECUTING, STATEMISFIRED, STATEPAUSED, STATEPAUSEDBLOCKED, STATEWAITING, TABLEBLOBTRIGGERS, TABLECALENDARS, TABLECRONTRIGGERS, TABLEFIREDTRIGGERS, TABLEJODETAILS, TABLELOCKS, TABLEPAUSEDTRIGGERS, TABLESCHEDULERSTATE, TABLESIMPLETRIGGERS, TABLETRIGGERS, TTYPEBLOB, TTYPECAL, TTYPEREADABLE, TTYPEREADABLE, TTYPSIMPLE

Constructor Summary

JobStoreCMT()
Execute the given callback having optionally acquired the given lock.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected Connection getNonManagedTXConnection()</code></td>
<td>Get the non-managed transaction connection.</td>
</tr>
<tr>
<td><code>String getNonManagedTXDataSource()</code></td>
<td>Get the name of the DataSource that should be used for performing database functions.</td>
</tr>
<tr>
<td><code>void initialize(org.quartz.spi.ClassLoadHelper loadHelper, org.quartz.spi.SchedulerSignaler signaler)</code></td>
<td>Called by the QuartzScheduler before the JobStore is used, in order to give it a chance to initialize.</td>
</tr>
<tr>
<td><code>boolean isDontSetNonManagedTXConnectionAutoCommitFalse()</code></td>
<td>Don't call set autocommit(false) on connections obtained from the DataSource.</td>
</tr>
<tr>
<td><code>boolean isTxIsolationLevelReadCommitted()</code></td>
<td>Set the transaction isolation level of DB connections to sequential.</td>
</tr>
<tr>
<td><code>void setDontSetNonManagedTXConnectionAutoCommitFalse(boolean b)</code></td>
<td>Set the name of the DataSource that should be used for performing database functions.</td>
</tr>
<tr>
<td><code>void setNonManagedTXDataSource(String nonManagedTxDsName)</code></td>
<td>Called by the QuartzScheduler to inform the JobStore that it should free up all of it's resources because the scheduler is shutting down.</td>
</tr>
</tbody>
</table>

Methods inherited from class org.quartz.impl.jdbcjobstore.JobStoreSupport

acquireNextTrigger, acquireNextTriggers, calcFailedIfAfter, calendarExists, canUseProperties, checkBlockedState, checkExists, checkExists, checkExists, checkExists, cleanupConnection, clearAllSchedulingData, clearAllSchedulingData, clearAndGetSignalSchedulingChangeOnTxCompletion, closeConnection, clusterCheckIn, clusterRecover, commitConnection, doCheckin,
supportsPersistence, triggeredJobComplete, triggeredJobComplete, triggerExists, triggerFired, triggersFired, updateMisfiredTrigger

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

nonManagedTxDsName
protected String nonManagedTxDsName

dontSetNonManagedTXConnectionAutoCommitFalse
protected boolean dontSetNonManagedTXConnectionAutoCommitFalse

setTxIsolationLevelReadCommitted
protected boolean setTxIsolationLevelReadCommitted

Constructor Detail

JobStoreCMT
public JobStoreCMT()

Method Detail

setNonManagedTXDataSource
public void setNonManagedTXDataSource(String nonManagedTxDsName)
Set the name of the dataSource that should be used for performing database functions.

---

**getNonManagedTXDataSource**

```java
public String getNonManagedTXDataSource()
```

Get the name of the dataSource that should be used for performing database functions.

---

**isDontSetNonManagedTXConnectionAutoCommitFalse**

```java
public boolean isDontSetNonManagedTXConnectionAutoCommitFalse()
```

---

**setDontSetNonManagedTXConnectionAutoCommitFalse**

```java
public void setDontSetNonManagedTXConnectionAutoCommitFalse(boolean b)
```

Don't call set autocommit(false) on connections obtained from the DataSource. This can be helpful in a few situations, such as if you have a driver that complains if it is called when it is already off.

**Parameters:**

- `b` -

---

**isTxIsolationLevelReadCommitted**

```java
public boolean isTxIsolationLevelReadCommitted()
```

---

**setTxIsolationLevelReadCommitted**

```java
public void setTxIsolationLevelReadCommitted(boolean b)
```

Set the transaction isolation level of DB connections to sequential.
**Parameters:**

- b

---

**initialize**

```java
public void initialize(org.quartz.spi.ClassLoadHelper loadHelper,
        org.quartz.spi.SchedulerSignaler signaler)
        throws SchedulerConfigException
```

**Description copied from class:** [JobStoreSupport](#)

Called by the QuartzScheduler before the JobStore is used, in order to give it a chance to initialize.

**Specified by:**

- initialize in interface org.quartz.spi.JobStore

**Overrides:**

- initialize in class [JobStoreSupport](#)

**Throws:**

- [SchedulerConfigException](#)

---

**shutdown**

```java
public void shutdown()
```

**Description copied from class:** [JobStoreSupport](#)

Called by the QuartzScheduler to inform the JobStore that it should free up all of its resources because the scheduler is shutting down.

**Specified by:**

- shutdown in interface org.quartz.spi.JobStore

**Overrides:**

- shutdown in class [JobStoreSupport](#)

---

**getNonManagedTXConnection**
protected `Connection` `getNonManagedTXConnection()`

Throws: `JobPersistenceException`

Specified by: `getNonManagedTXConnection` in class `JobStoreSupport`

`executeInLock`

protected `Object` `executeInLock`(`String` `lockName`, `JobStoreSupport.TransactionCallback` `txCallback`)

Throws: `JobPersistenceException`

Execute the given callback having optionally acquired the given lock. Because CMT assumes that the connection is already part of a managed transaction, it does not attempt to commit or rollback the enclosing transaction.

Specified by: `executeInLock` in class `JobStoreSupport`

Parameters:
- `lockName` - The name of the lock to acquire, for example "TRIGGER_ACCESS". If null, then no lock is acquired, but the txCallback is still executed in a transaction.

Throws: `JobPersistenceException`

See Also: `JobStoreSupport.executeInNonManagedTXLock(String, TransactionCallback)`, `JobStoreSupport.executeInLock(String, TransactionCallback)`, `JobStoreSupport.getNonManagedTXConnection()`, `JobStoreSupport.getConnection()`
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS | NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES | NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
org.quartz.impl.jdbcjobstore  **Class JobStoreSupport**

java.lang.Object  
  org.quartz.impl.jdbcjobstore.JobStoreSupport

**All Implemented Interfaces:**
 Constants, org.quartz.spi.JobStore

**Direct Known Subclasses:**
 JobStoreCMT, JobStoreTX

The `JobStoreSupport` class provides a base functionality for JDBC-based JobStore implementations.

**Author:**
Jeffrey Wescott, James House

---

### Nested Class Summary

<table>
<thead>
<tr>
<th>protected static class</th>
<th>JobStoreSupport.RecoverMisfiredJobsResult</th>
</tr>
</thead>
</table>
| Helper class for returning the composite result of trying to recover misfired jobs.

<table>
<thead>
<tr>
<th>protected static interface</th>
<th>JobStoreSupport.TransactionCallback</th>
</tr>
</thead>
</table>
| Implement this interface to provide the code to execute within the a transaction template.

<table>
<thead>
<tr>
<th>protected static interface</th>
<th>JobStoreSupport.VoidTransactionCallback</th>
</tr>
</thead>
</table>
| Implement this interface to provide the code to execute within the a transaction template that has no return value.

### Field Summary

<table>
<thead>
<tr>
<th>protected HashMap&lt;String, Calendar&gt;</th>
<th>calendarCache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected</td>
<td>Type</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Class?</td>
<td>extends</td>
</tr>
<tr>
<td>String</td>
<td>protected</td>
</tr>
<tr>
<td>String</td>
<td>protected</td>
</tr>
<tr>
<td>String</td>
<td>protected</td>
</tr>
<tr>
<td>boolean</td>
<td>protected</td>
</tr>
<tr>
<td>String</td>
<td>protected</td>
</tr>
<tr>
<td>String</td>
<td>protected</td>
</tr>
<tr>
<td>long</td>
<td>protected</td>
</tr>
<tr>
<td>String</td>
<td>protected static</td>
</tr>
<tr>
<td>String</td>
<td>protected static</td>
</tr>
<tr>
<td>String</td>
<td>protected static</td>
</tr>
<tr>
<td>String</td>
<td>protected static</td>
</tr>
<tr>
<td>String</td>
<td>protected static</td>
</tr>
<tr>
<td>int</td>
<td>protected</td>
</tr>
<tr>
<td>ThreadLocal&lt;Long&gt;</td>
<td>protected</td>
</tr>
<tr>
<td>String</td>
<td>protected</td>
</tr>
<tr>
<td>boolean</td>
<td>protected</td>
</tr>
</tbody>
</table>
Fields inherited from interface org.quartz.impl.jdbctjobstore.<strong>Constants</strong>
ALIAS  COL NEXT FIRE TIME, ALL GROUPS PAUSED, COL BLOB,  
COL CALENDAR, COL CALENDAR NAME, COL CHECKIN INTERVAL,  
COL CRON_EXPRESSION, COL DESCRIPTION, COL END TIME, COL_ENTRY_ID,  
COL ENTRY STATE, COL FIRED TIME, COL INSTANCE_NAME,  
COL IS DURABLE, COL IS NONCONCURRENT, COL IS UPDATE_DATA,  
COL IS VOLATILE, COL JOB CLASS, COL JOB_DATAMAP, COL_JOB_GROUP,  
COL JOB_NAME, COL LAST CHECKIN TIME, COL LOCK_NAME,  
COL MISFIRE_INSTRUCTION, COL NEXT FIRE TIME, COL_PREV_FIRE_TIME,  
COL_PRIORITY, COL_REPEAT_COUNT, COL_REPEAT_INTERVAL,  
COL REQUESTS_RECOVERY, COL SCHEDULER_NAME, COL START TIME,  
COL TIME_ZONE_ID, COL TIMES_TRIGGERED, COL_TRIGGER GROUP,  
DEFAULT_TABLE_PREFIX, STATE_ACQUIRED, STATE_BLOCKED,  
STATE_COMPLETE, STATE_DELETED, STATE_ERROR, STATE_EXECUTING,  
STATE_MISFIRED, STATE_PAUSED, STATE_PAUSED_BLOCKED, STATE_WAITING,  
TABLE_BLOB_TRIGGERS, TABLE CALENDARS, TABLE_CRON_TRIGGERS,  
TABLE FIRED TRIGGERS, TABLE_JOBDETAILS, TABLE_LOCKS,  
TABLE_PAUSED_TRIGGERS, TABLE_SCHEDULER_STATE,  
TABLE_SIMPLE_TRIGGERS, TABLE_TRIGGERS, TTYPE_BLOB, TTYPE_CAL_INT,  
TTYPE_CRON, TTYPE_SIMPLE

Constructor Summary

**JobStoreSupport**(())

Method Summary

```java
protected List<org.quartz.spi.OperableTrigger> acquireNextTrigger(Connection conn,)
```

```java
protected List<org.quartz.spi.OperableTrigger> acquireNextTriggers(long noLaterThan,)
```

```java
protected long calcFailedIfAfter(SchedulerStateRecord)
```

```java
protected boolean calendarExists(Connection conn, String )
```

```java
boolean canUseProperties()
```
protected String checkBlockedState(Connection conn, JobKey)
Determine if a Trigger for the given job should be blocked.

protected boolean checkExists(Connection conn, JobKey)

protected boolean checkExists(Connection conn, TriggerKey)
Determine whether a Trigger with the given identifier already exists.

boolean checkExists(JobKey jobKey)
Determine whether a Job with the given identifier already exists.

protected boolean checkExists(JobKey jobKey)

protected void cleanupConnection(Connection conn)
Cleanup the given database connection.

void clearAllSchedulingData()
Clear (delete!) all scheduling data.

protected void clearAllSchedulingData(Connection conn)

protected Long clearAndGetSignalSchedulingChangeOnTxCompletion

protected void closeConnection(Connection conn)
Closes the supplied Connection.

protected List<SchedulerStateRecord> clusterCheckIn(Connection conn)

protected void clusterRecover(Connection conn, List)

protected void commitConnection(Connection conn)
Commit the supplied connection.

protected boolean doCheckin()

protected JobStoreSupport.RecoverMisfiredJobsResult doRecoverMisfires()

protected abstract Object executeInLock(String lockName, JobStoreSupport.TransactionCallback
Execute the given callback having acquired the given lock.

protected void executeInLock(String lockName, JobStoreSupport.VoidTransactionCallback

protected void executeInLock(String lockName, JobStoreSupport.TransactionCallback

Get whether String-only properties will be handled in JobDataMaps.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>executeInNonManagedTXLock(String lock)</code></td>
<td>Execute the given callback having optionally acquired the given lock.</td>
</tr>
<tr>
<td><code>executeInNonManagedTXLock()</code></td>
<td>Execute the given callback having aquired the given lock.</td>
</tr>
<tr>
<td><code>executeWithoutLock(JobStoreSupport.TX)</code></td>
<td>Execute the given callback having optionally acquired the given lock.</td>
</tr>
<tr>
<td><code>findFailedInstances(Connection conn)</code></td>
<td>Get a list of all scheduler instances in the cluster that may have failed.</td>
</tr>
<tr>
<td><code>getAttributeRestoringConnection()</code></td>
<td>Wrap the given connection in a Proxy such that attributes of the connection is closed (and potentially restored).</td>
</tr>
<tr>
<td><code>getCalendarNames()</code></td>
<td>Get the names of all of the Calendar objects.</td>
</tr>
<tr>
<td><code>getCalendarNames(Connection conn)</code></td>
<td>Get the names of all of the Calendar objects from the given connection.</td>
</tr>
<tr>
<td><code>getClassLoadHelper()</code></td>
<td>Get the Quartz ClassLoadHelper.</td>
</tr>
<tr>
<td><code>getClusterCheckinInterval()</code></td>
<td>Get the frequency (in milliseconds) at which this instance “checks-in”.</td>
</tr>
<tr>
<td><code>getConnection()</code></td>
<td>Get the database connection.</td>
</tr>
<tr>
<td><code>getDataSource()</code></td>
<td>Get the name of the DataSource that should be used for DB operations.</td>
</tr>
<tr>
<td><code>getDbRetryInterval()</code></td>
<td>Get the database retry interval.</td>
</tr>
<tr>
<td><code>getDelegate()</code></td>
<td>Get the driver delegate for DB operations.</td>
</tr>
<tr>
<td><code>getDoubleCheckLockMisfireHandler()</code></td>
<td>Get whether to check to see if there are Triggers that have misfired and recover them.</td>
</tr>
<tr>
<td><code>getDriverDelegateClass()</code></td>
<td>Get the JDBC driver delegate class name.</td>
</tr>
<tr>
<td><code>getDriverDelegateInitString()</code></td>
<td>Get the JDB delegate init string.</td>
</tr>
</tbody>
</table>
Get the JDBC driver delegate's initialization string.

```java
long getEstimatedTimeToReleaseAndAcquire()
```

Get the estimated time to release and acquire trigger.

```java
protected String getFiredTriggerRecordId()
```

Get the fired trigger record Id.

```java
String getInstanceId()
```

Get the instance Id of the Scheduler.

```java
String getInstanceName()
```

Get the instance name of the Scheduler.

```java
List<String> getJobGroupNames()
```

Get the names of all of the Job groups.

```java
protected List<String> getJobGroupNames(Connection conn)
```

Get the names of all of the Job groups for a given connection.

```java
Set<JobKey> getJobKeys(GroupMatcher<JobKey> matcher)
```

Get the names of all of the Jobs that match the given matcher.

```java
protected Set<JobKey> getJobNames(Connection conn, GroupMatcher<JobKey> matcher)
```

Get the names of all of the Jobs for a given connection and matcher.

```java
protected Semaphore getLockHandler()
```

Get the lock handler.

```java
protected org.slf4j.Logger getLog()
```

Get the logger.

```java
boolean getMakeThreadsDaemons()
```

Get whether the threads spawned by the JobStore should be daemons.

```java
int getMaxMisfiresToHandleAtATime()
```

Get the maximum number of misfires to handle at a time (within one transaction).

```java
long getMisfireThreshold()
```

Get the misfire threshold.

```java
protected long getMisfireTime()
```

Get the misfire time.

```java
protected abstract Connection getNonManagedTXConnection()
```

Get the non-managed transaction connection.

```java
int getNumberOfCalendars()
```

Get the number of calendars.

```java
protected int...
```

...
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getNumberOfCalendars(Connection conn)</code></td>
<td>Get the number of calendars that are stored.</td>
</tr>
<tr>
<td><code>getNumberOfJobs()</code></td>
<td>Get the number of jobs that are stored.</td>
</tr>
<tr>
<td><code>protected int getNumberOfJobs(Connection conn)</code></td>
<td>Get the number of jobs that are stored.</td>
</tr>
<tr>
<td><code>getNumberOfTriggers()</code></td>
<td>Get the number of triggers that are stored.</td>
</tr>
<tr>
<td><code>protected int getNumberOfTriggers(Connection conn)</code></td>
<td>Get the number of triggers that are stored.</td>
</tr>
<tr>
<td><code>Set&lt;String&gt; getPausedTriggerGroups()</code></td>
<td>Set all of the triggers in the given group to be paused.</td>
</tr>
<tr>
<td><code>Set&lt;String&gt; getPausedTriggerGroups(Connection conn)</code></td>
<td>Set all of the triggers in the given group to be paused.</td>
</tr>
<tr>
<td><code>String getSelectWithLockSQL()</code></td>
<td>Get the SQL to select with lock.</td>
</tr>
<tr>
<td><code>String getTablePrefix()</code></td>
<td>Get the prefix that should be prepended to all table names.</td>
</tr>
<tr>
<td><code>List&lt;String&gt; getTriggerGroupNames()</code></td>
<td>Get the names of all of the trigger groups.</td>
</tr>
<tr>
<td><code>protected List&lt;String&gt; getTriggerGroupNames(Connection conn)</code></td>
<td>Get the names of all of the trigger groups.</td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt; getTriggerKeys(GroupMatcher&lt;TriggerKey&gt;)</code></td>
<td>Get the names of all of the triggers.</td>
</tr>
<tr>
<td><code>protected Set&lt;TriggerKey&gt; getTriggerNames(Connection conn, GroupMatcher&lt;TriggerKey&gt;)</code></td>
<td>Get the names of all of the triggers.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt; getTriggersForJob(Connection conn, JobKey jobKey)</code></td>
<td>Get all of the triggers that are associated to the given job.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt; getTriggersForJob(JobKey jobKey)</code></td>
<td>Get all of the triggers that are associated to the given job.</td>
</tr>
<tr>
<td><code>Trigger.TriggerState getTriggerState(Connection conn, TriggerKey triggerKey)</code></td>
<td>Get the current state of the identified trigger.</td>
</tr>
<tr>
<td><code>Trigger.TriggerState getTriggerState(TriggerKey triggerKey)</code></td>
<td>Get the current state of the identified trigger.</td>
</tr>
<tr>
<td><code>boolean</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>getUseDBLocks()</code></td>
<td>Get whether this instance should use database-based thread locks.</td>
</tr>
<tr>
<td><code>initialize(org.quartz.spi.ClassLoadHelper helper)</code></td>
<td>Called by the QuartzScheduler before the initialization of the instance.</td>
</tr>
<tr>
<td><code>isAcquireTriggersWithinLock()</code></td>
<td>Whether or not the query and update to acquire a Trigger for firing explicit DB lock (to avoid possible race conditions).</td>
</tr>
<tr>
<td><code>isClustered()</code></td>
<td>Get whether this instance is part of a cluster.</td>
</tr>
<tr>
<td><code>isDontSetAutoCommitFalse()</code></td>
<td></td>
</tr>
<tr>
<td><code>isLockOnInsert()</code></td>
<td></td>
</tr>
<tr>
<td><code>isThreadsInheritInitializersClassLoadContext()</code></td>
<td>Get whether to set the class load context of spawned threads to that of the initializers.</td>
</tr>
<tr>
<td><code>isTxIsolationLevelSerializable()</code></td>
<td></td>
</tr>
<tr>
<td><code>jobExists(Connection conn, JobKey jobKey)</code></td>
<td>Check existence of a given job.</td>
</tr>
<tr>
<td><code>logWarnIfNonZero(int val, String warning)</code></td>
<td></td>
</tr>
<tr>
<td><code>pauseAll()</code></td>
<td>Pause all triggers - equivalent of calling <code>pauseAll(Connection conn)</code>.</td>
</tr>
<tr>
<td><code>pauseAll(Connection conn)</code></td>
<td>Pause all triggers - equivalent of calling <code>pauseAll(Connection conn)</code>.</td>
</tr>
<tr>
<td><code>pauseJob(JobKey jobKey)</code></td>
<td>Pause the Job with the given name.</td>
</tr>
<tr>
<td><code>pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Pause all of the Jobs matching the given matcher.</td>
</tr>
<tr>
<td><code>pauseTrigger(Connection conn, TriggerKey triggerKey)</code></td>
<td>Pause the Trigger with the given name.</td>
</tr>
<tr>
<td><code>pauseTrigger(TriggerKey triggerKey)</code></td>
<td>Pause the Trigger with the given name.</td>
</tr>
<tr>
<td><code>pauseTriggerGroup(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Pause all of the Triggers matching the given matcher.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>protected void pauseTriggers(Set&lt;String&gt;)</td>
<td>Pause all of the Triggers matching set.</td>
</tr>
<tr>
<td>protected void recoverJobs()</td>
<td>Recover any failed or misfired jobs.</td>
</tr>
<tr>
<td>protected void recoverJobs(Connection conn)</td>
<td>Will recover any failed or misfired.</td>
</tr>
<tr>
<td>protected void recoverMisfiredJobs(Connection conn)</td>
<td>Recover any failed or misfired.</td>
</tr>
<tr>
<td>protected void releaseAcquiredTrigger()</td>
<td>Inform the JobStore that the scheduler no longer plans to acquire (reserved).</td>
</tr>
<tr>
<td>protected void releaseLock(Connection conn, String lockName)</td>
<td>Release lock.</td>
</tr>
<tr>
<td>protected boolean removeCalendar(Connection conn, String calName)</td>
<td>Remove (delete) the Calendar with given name.</td>
</tr>
<tr>
<td>protected boolean removeJob(Connection conn, JobKey jobKey)</td>
<td>Remove (delete) the Job with the given key.</td>
</tr>
<tr>
<td>protected boolean removeJobs(List&lt;JobKey&gt; jobKeys)</td>
<td>Remove (delete) the Jobs with the given keys.</td>
</tr>
<tr>
<td>protected boolean removeTrigger(Connection conn, TriggerKey triggerKey)</td>
<td>Remove (delete) the Trigger with the given key.</td>
</tr>
<tr>
<td>protected boolean removeTriggers(List&lt;TriggerKey&gt; triggerKeys)</td>
<td>Remove (delete) the Triggers with the given keys.</td>
</tr>
<tr>
<td>protected boolean replaceTrigger(Connection conn, TriggerKey triggerKey)</td>
<td>Replace the Trigger with the given key.</td>
</tr>
</tbody>
</table>
replaceTrigger(TriggerKey triggerKey);

void resumeAll()
    Resume (un-pause) all triggers - equivalent to calling resumeAll(Connection conn);

protected void resumeAll(Connection conn)

void resumeJob(JobKey jobKey)
    Resume (un-pause) the Job with the given JobKey;

Set<String> resumeJobs(GroupMatcher<JobKey> matcher)
    Resume (un-pause) all of the Jobs in the given GroupMatcher;

void resumeTrigger(Connection conn, TriggerKey triggerKey)
    Resume (un-pause) the Trigger with the given TriggerKey;

void resumeTrigger(Connection conn, Trigger trigger)
    Resume (un-pause) the Trigger with the given Trigger;

void resumeTriggerGroup(Connection conn, GroupMatcher<TriggerKey> matcher)
    Resume (un-pause) all of the Triggers in the given GroupMatcher;

Set<String> resumeTriggers(GroupMatcher<TriggerKey> matcher)
    Resume (un-pause) all of the Triggers in the given GroupMatcher;

protected Calendar retrieveCalendar(Connection conn, String calName)
    Retrieve the given Calendar;

protected JobDetail retrieveJob(Connection conn, JobKey jobKey)
    Retrieve the JobDetail for the given JobKey;

protected org.quartz.spi.OperableTrigger retrieveTrigger(Connection conn, TriggerKey triggerKey)
    Retrieve the given Trigger;

protected void rollbackConnection(Connection conn)
    Rollback the supplied connection;

void schedulerStarted()
setAcquireTriggersWithinLock(boolean acquireTriggersWithinLock)
Whether or not the query and update to acquire a Trigger for firing explicit DB lock.

void setClusterCheckinInterval(long l)
Set the frequency (in milliseconds)

void setDataSource(String dsName)
Set the name of the DataSource that should be used for

void setDbRetryInterval(long dbRetryInterval)

void setDontSetAutoCommitFalse(boolean b)
Don't call set autocommit(false) on connections
obtained from the

void setDoubleCheckLockMisfireHandler(boolean doubleCheckLockMisfireHandler)
Set whether to check to see if there are Triggers that have misfired and recover them.

void setDriverDelegateClass(String delegateClassName)
Set the JDBC driver delegate class.

void setDriverDelegateInitString(String delegateInitString)
Set the JDBC driver delegate's initialization string.

void setInstanceId(String instanceId)
Set the instance Id of the Scheduler (must be unique within a cluster).

void setInstanceName(String instanceName)
Set the instance name of the Scheduler (must be unique within this server instance).

void setIsClustered(boolean isClustered)
Set whether this instance is part of a cluster.

void setLockHandler(Semaphore lockHandler)

void setLockOnInsert(boolean lockOnInsert)
Whether or not to obtain locks when inserting new jobs/triggers.

void setMakeThreadsDaemons(boolean makeThreadsDaemons)
Set whether the threads spawned by this JobStore should be

void setMaxMisfiresToHandleAtATime(int maxToRecoverAtATime)
Set the maximum number of misfired jobs that can be recovered (within one transaction).

void setMisfireThreshold(long misfireThreshold)
The the number of milliseconds by which a trigger must have missed its

considered "misfired" and thus have its misfire instruction applied.

```java
void setSelectWithLockSQL(String string)
    set the SQL statement to use to select
```

```java
void setTablePrefix(String prefix)
    Set the prefix that should be pre-pended
to table names.
```

```java
void setThreadPoolSize(int poolSize)
```

```java
void setThreadsInheritInitializersClassLoader(boolean b)
    Set whether to set the class load context
to that of spawned threads.
```

```java
void setTxIsolationLevelSerializable(boolean useDBLocks)
    Set whether this instance should use
database-based thread locks.
```

```java
void setUseDBLocks(String useProp)
    Set whether String-only properties
will be handled in JobDataMaps.
```

```java
void setUseProperties(String useProp)
```

```java
void shutdown()
    Called by the QuartzScheduler to
inform the scheduler is shutting down.
```

```java
protected void signalSchedulingChangeImmediately(long candidateNewNextFireTime)
```

```java
protected void signalSchedulingChangeOnTxCompletion()
```

```java
protected void storeCalendar(Connection conn, String calName, Calendar cal)
    Store the given Calendar.
```

```java
void storeCalendar(String calName, Calendar cal)
    Store the given Calendar.
```

```java
protected void storeJob(Connection conn, JobDetail newJob, boolean replaceExisting)
    Insert or update a job.
```

```java
void storeJob(JobDetail newJob, boolean replaceExisting)
    Store the given JobDetail.
```

```java
void storeJobAndTrigger(JobDetail newJob)
    Store the given JobDetail and Trigger.
```

```java
void storeJobsAndTriggers(Map<JobDetail, Trigger> map)
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void storeTrigger</td>
<td>Connection conn, org.quartz.spi.OperableTrigger newTrigger, boolean replaceExisting, String state</td>
<td>Insert or update a trigger.</td>
</tr>
<tr>
<td>void storeTrigger</td>
<td>org.quartz.spi.OperableTrigger</td>
<td>Store the given Trigger.</td>
</tr>
<tr>
<td>boolean supportsPersistence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>protected void triggeredJobComplete</td>
<td>Connection conn, JobDetail jobDetail, Trigger.CompletionInstruction</td>
<td>Inform the JobStore that the scheduler has completed the associated Job, and that the JobDataMap in the given JobDetail has been modified.</td>
</tr>
<tr>
<td>void triggeredJobComplete</td>
<td>org.quartz.spi.OperableTrigger Trigger.CompletionInstruction</td>
<td>Inform the JobStore that the scheduler has completed the associated Job, and that the JobDataMap in the given trigger has been modified.</td>
</tr>
<tr>
<td>protected boolean triggerExists</td>
<td>Connection conn, TriggerKey</td>
<td>Check existence of a given trigger.</td>
</tr>
<tr>
<td>protected org.quartz.spi.TriggerFiredBundle triggerFired</td>
<td>Connection conn, org.quartz.spi.OperableTrigger trigger</td>
<td></td>
</tr>
<tr>
<td>List&lt;org.quartz.spi.TriggerFiredResult&gt; triggersFired</td>
<td>List&lt;org.quartz.spi.OperableTrigger&gt; triggers</td>
<td>Inform the JobStore that the scheduler is now firing the associated Job that had previously acquired (reserved).</td>
</tr>
<tr>
<td>protected boolean updateMisfiredTrigger</td>
<td>Connection conn, boolean forceState</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

**LOCK_TRIGGER_ACCESS**

protected static final String LOCK_TRIGGER_ACCESS
See Also:
Constant Field Values

LOCK_JOB_ACCESS
protected static final String LOCK_JOB_ACCESS

See Also:
Constant Field Values

LOCK CALENDAR ACCESS
protected static final String LOCK_CALENDAR_ACCESS

See Also:
Constant Field Values

LOCK STATE ACCESS
protected static final String LOCK_STATE_ACCESS

See Also:
Constant Field Values

LOCK MISFIRE ACCESS
protected static final String LOCK_MISFIRE_ACCESS

See Also:
Constant Field Values

dsName
protected String dsName
tablePrefix

protected String tablePrefix

useProperties

protected boolean useProperties

instanceId

protected String instanceId

instanceName

protected String instanceName

delegateClassName

protected String delegateClassName

delegateInitString

protected String delegateInitString

delegateClass

protected Class<? extends DriverDelegate> delegateClass

calendarCache

protected HashMap<String, Calendar> calendarCache
maxToRecoverAtATime
protected int maxToRecoverAtATime

sigChangeForTxCompletion
protected ThreadLocal<Long> sigChangeForTxCompletion

firstCheckIn
protected boolean firstCheckIn

lastCheckin
protected long lastCheckin

**Constructor Detail**

**JobStoreSupport**

public JobStoreSupport()

**Method Detail**

setDataSource

public void setDataSource(String dsName)

Set the name of the DataSource that should be used for performing database functions.
**getDataSource**

```java
public String getDataSource()
```

Get the name of the DataSource that should be used for performing database functions.

---

**setTablePrefix**

```java
public void setTablePrefix(String prefix)
```

Set the prefix that should be pre-pended to all table names.

---

**getTablePrefix**

```java
public String getTablePrefix()
```

Get the prefix that should be pre-pended to all table names.

---

**setUseProperties**

```java
public void setUseProperties(String useProp)
```

Set whether String-only properties will be handled in JobDataMaps.

---

**canUseProperties**

```java
public boolean canUseProperties()
```

Get whether String-only properties will be handled in JobDataMaps.

---

**setInstanceId**

```java
public void setInstanceId(String instanceId)
```
Set the instance Id of the Scheduler (must be unique within a cluster).

**Specified by:**

setInstanceId in interface org.quartz.spi.JobStore

---

**getInstanceId**

public String getInstanceId()

Get the instance Id of the Scheduler (must be unique within a cluster).

---

**setInstanceName**

public void setInstanceName(String instanceName)

Set the instance name of the Scheduler (must be unique within this server instance).

**Specified by:**

setInstanceName in interface org.quartz.spi.JobStore

---

**setThreadPoolSize**

public void setThreadPoolSize(int poolSize)

**Specified by:**

setThreadPoolSize in interface org.quartz.spi.JobStore

---

**getInstanceName**

public String getInstanceName()

Get the instance name of the Scheduler (must be unique within this server instance).
**getEstimatedTimeToReleaseAndAcquireTrigger**

```java
class getEstimatedTimeToReleaseAndAcquireTrigger()

**Specified by:**
getEstimatedTimeToReleaseAndAcquireTrigger in interface org.quartz.spi.JobStore
```

**setIsClustered**

```java
class setIsClustered(boolean isClustered)

Set whether this instance is part of a cluster.
```

**isClustered**

```java
class boolean isClustered()

Get whether this instance is part of a cluster.

**Specified by:**
isClustered in interface org.quartz.spi.JobStore
```

**getClusterCheckinInterval**

```java
class getClusterCheckinInterval()

Get the frequency (in milliseconds) at which this instance "checks-in" with the other instances of the cluster. -- Affects the rate of detecting failed instances.
```

**setClusterCheckinInterval**

```java
class setClusterCheckinInterval(long l)

Set the frequency (in milliseconds) at which this instance "checks-in" with
the other instances of the cluster. -- Affects the rate of detecting failed instances.

getMaxMisfiresToHandleAtATime

```java
public int getMaxMisfiresToHandleAtATime()
```

Get the maximum number of misfired triggers that the misfire handling thread will try to recover at one time (within one transaction). The default is 20.

setMaxMisfiresToHandleAtATime

```java
public void setMaxMisfiresToHandleAtATime(int maxToRecoverAtATime)
```

Set the maximum number of misfired triggers that the misfire handling thread will try to recover at one time (within one transaction). The default is 20.

dgetDbRetryInterval

```java
public long getDbRetryInterval()
```

**Returns:**

Returns the dbRetryInterval.

setDbRetryInterval

```java
public void setDbRetryInterval(long dbRetryInterval)
```

**Parameters:**

dbRetryInterval - The dbRetryInterval to set.

setUseDBLocks
public void setUseDBLocks(boolean useDBLocks)

Set whether this instance should use database-based thread synchronization.

---

getUseDBLocks

public boolean getUseDBLocks()

Get whether this instance should use database-based thread synchronization.

---

isLockOnInsert

public boolean isLockOnInsert()

---

setLockOnInsert

public void setLockOnInsert(boolean lockOnInsert)

Whether or not to obtain locks when inserting new jobs/triggers. Defaults to true, which is safest - some db's (such as MS SQLServer) seem to require this to avoid deadlocks under high load, while others seem to do fine without.

Setting this property to false will provide a significant performance increase during the addition of new jobs and triggers.

**Parameters:**

lockOnInsert -

---

getMisfireThreshold

public long getMisfireThreshold()
setMisfireThreshold

public void setMisfireThreshold(long misfireThreshold)

The number of milliseconds by which a trigger must have missed its next-fire-time, in order for it to be considered "misfired" and thus have its misfire instruction applied.

Parameters:

misfireThreshold -

isDontSetAutoCommitFalse

public boolean isDontSetAutoCommitFalse()

setDontSetAutoCommitFalse

public void setDontSetAutoCommitFalse(boolean b)

Don't call set autocommit(false) on connections obtained from the DataSource. This can be helpful in a few situations, such as if you have a driver that complains if it is called when it is already off.

Parameters:

b -

isTxIsolationLevelSerializable

public boolean isTxIsolationLevelSerializable()

setTxIsolationLevelSerializable

public void setTxIsolationLevelSerializable(boolean b)

Set the transaction isolation level of DB connections to sequential.
isAcquireTriggersWithinLock

public boolean isAcquireTriggersWithinLock()

Whether or not the query and update to acquire a Trigger for firing should be performed after obtaining an explicit DB lock (to avoid possible race conditions on the trigger's db row). This is the behavior prior to Quartz 1.6.3, but is considered unnecessary for most databases (due to the nature of the SQL update that is performed), and therefore a superfluous performance hit.

setAcquireTriggersWithinLock

public void setAcquireTriggersWithinLock(boolean acquireTriggersWithinLock)

Whether or not the query and update to acquire a Trigger for firing should be performed after obtaining an explicit DB lock. This is the behavior prior to Quartz 1.6.3, but is considered unnecessary for most databases, and therefore a superfluous performance hit.

setDriverDelegateClass

public void setDriverDelegateClass(String delegateClassName) throws InvalidConfigurationException

Set the JDBC driver delegate class.

Parameters:

delegateClassName - the delegate class name

Throws:

InvalidConfigurationException
**getDriverDelegateClass**

```java
public String getDriverDelegateClass()
```

Get the JDBC driver delegate class name.

**Returns:**
the delegate class name

**setDriverDelegateInitString**

```java
public void setDriverDelegateInitString(String delegateInitString)
throws InvalidConfigurationException
```

Set the JDBC driver delegate's initialization string.

**Parameters:**
delegateClassName - the delegate init string

**Throws:**
InvalidConfigurationException

**getDriverDelegateInitString**

```java
public String getDriverDelegateInitString()
```

Get the JDBC driver delegate's initialization string.

**Returns:**
the delegate init string

**getSelectWithLockSQL**

```java
public String getSelectWithLockSQL()
```

**setSelectWithLockSQL**
public void setSelectWithLockSQL(String string)

set the SQL statement to use to select and lock a row in the "locks" table.

See Also:
StdRowLockSemaphore

protected org.quartz.spi.ClassLoadHelper getClassLoadHelper()

getMakeThreadsDaemons

class

public boolean getMakeThreadsDaemons()

Get whether the threads spawned by this JobStore should be marked as daemon. Possible threads include the MisfireHandler and the ClusterManager.

See Also:
Thread.setDaemon(boolean)

setMakeThreadsDaemons

public void setMakeThreadsDaemons(boolean makeThreadsDaemons)

Set whether the threads spawned by this JobStore should be marked as daemon. Possible threads include the MisfireHandler and the ClusterManager.

See Also:
Thread.setDaemon(boolean)

isThreadsInheritInitializersClassLoadContext

class

public boolean isThreadsInheritInitializersClassLoadContext()
Get whether to set the class load context of spawned threads to that of the initializing thread.

---

**setThreadsInheritInitializersClassLoadContext**

class method

```
public void setThreadsInheritInitializersClassLoadContext(boolean threadsInheritInitializersClassLoadContext)
```

Set whether to set the class load context of spawned threads to that of the initializing thread.

---

**getDoubleCheckLockMisfireHandler**

class method

```
public boolean getDoubleCheckLockMisfireHandler()
```

Get whether to check to see if there are Triggers that have misfired before actually acquiring the lock to recover them. This should be set to false if the majority of the time, there are are misfired Triggers.

---

**setDoubleCheckLockMisfireHandler**

class method

```
public void setDoubleCheckLockMisfireHandler(boolean doubleCheckLockMisfireHandler)
```

Set whether to check to see if there are Triggers that have misfired before actually acquiring the lock to recover them. This should be set to false if the majority of the time, there are are misfired Triggers.

---

**getLog**

protected method

```
protected org.slf4j.Logger getLog()
```

---

**initialize**

class method

```
public void initialize(org.quartz.spi.ClassLoadHelper loadHelper,
                       org.quartz.spi.SchedulerSignaler signaler)
```
throws SchedulerConfigException

Called by the QuartzScheduler before the JobStore is used, in order to give it a chance to initialize.

Specified by:
   initialize in interface org.quartz.spi.JobStore

Throws:
   SchedulerConfigException

schedulerStarted

public void schedulerStarted() throws SchedulerException

Specified by:
   schedulerStarted in interface org.quartz.spi.JobStore

Throws:
   SchedulerException

See Also:
   JobStore.schedulerStarted()

shutdown

public void shutdown()

Called by the QuartzScheduler to inform the JobStore that it should free up all of its resources because the scheduler is shutting down.

Specified by:
   shutdown in interface org.quartz.spi.JobStore

supportsPersistence

public boolean supportsPersistence()

Specified by:
supportsPersistence in interface org.quartz.spi.JobStore

---

**getNonManagedTXConnection**

protected abstract `Connection getNonManagedTXConnection()`

**Throws:**
`JobPersistenceException`

---

**getAttributeRestoringConnection**

protected `Connection getAttributeRestoringConnection(Connection conn)`

Wrap the given connection in a Proxy such that attributes that might be set will be restored before the connection is closed (and potentially restored to a pool).

---

**getConnection**

protected `Connection getConnection()`

**Throws:**
`JobPersistenceException`

---

**releaseLock**

protected void `releaseLock(Connection conn, String lockName, boolean doIt)`

---

**recoverJobs**

protected void `recoverJobs()`
Recover any failed or misfired jobs and clean up the data store as appropriate.

**Throws:**

*JobPersistenceException* - if jobs could not be recovered

---

**recoverJobs**

```java
protected void recoverJobs(Connection conn) throws JobPersistenceException
```

Will recover any failed or misfired jobs and clean up the data store as appropriate.

**Throws:**

*JobPersistenceException* - if jobs could not be recovered

---

**getMisfireTime**

```java
protected long getMisfireTime()
```

---

**recoverMisfiredJobs**

```java
protected JobStoreSupport.RecoverMisfiredJobsResult recoverMisfiredJobs(boolean recovering) throws JobPersistenceException, SQLException
```

**Throws:**

*JobPersistenceException*, *SQLException*

---

**updateMisfiredTrigger**

```java
protected boolean updateMisfiredTrigger(Connection conn,
```
storeJobAndTrigger

public void storeJobAndTrigger(JobDetail newJob,
org.quartz.spi.OperableTrigger newTrigger)
throws ObjectAlreadyExistsException,
JobPersistenceException

Store the given JobDetail and Trigger.

Specified by:
storeJobAndTrigger in interface org.quartz.spi.JobStore

Parameters:
newJob - The JobDetail to be stored.
newTrigger - The Trigger to be stored.

Throws:
ObjectAlreadyExistsException - if a Job with the same name/group already exists.
JobPersistenceException

storeJob

public void storeJob(JobDetail newJob,
boolean replaceExisting)
throws ObjectAlreadyExistsException,
JobPersistenceException

Store the given JobDetail.

Specified by:
storeJob in interface org.quartz.spi.JobStore

Parameters:
newJob - The JobDetail to be stored.
replaceExisting - If true, any Job existing in the JobStore with the same name & group should be over-written.

**Throws:**
- `ObjectAlreadyExistsException` - if a Job with the same name/group already exists, and replaceExisting is set to false.
- `JobPersistenceException`

---

**storeJob**

protected void **storeJob**(Connection conn, JobDetail newJob, boolean replaceExisting) throws `ObjectAlreadyExistsException`, `JobPersistenceException`

Insert or update a job.

**Throws:**
- `ObjectAlreadyExistsException`
- `JobPersistenceException`

---

**jobExists**

protected boolean **jobExists**(Connection conn, JobKey jobKey) throws `JobPersistenceException`

Check existence of a given job.

**Throws:**
- `JobPersistenceException`

---

**storeTrigger**

public void **storeTrigger**(org.quartz.spi.OperableTrigger newTrigger, boolean replaceExisting) throws `ObjectAlreadyExistsException`,


Store the given Trigger.

**Specified by:**
storeTrigger in interface org.quartz.spi.JobStore

**Parameters:**
newTrigger - The Trigger to be stored.
replaceExisting - If true, any Trigger existing in the JobStore with the same name & group should be over-written.

**Throws:**
- ObjectAlreadyExistsException - if a Trigger with the same name/group already exists, and replaceExisting is set to false.
- JobPersistenceException

---

**storeTrigger**

protected void **storeTrigger**(Connection conn,
org.quartz.spi.OperableTrigger newTrigger,
JobDetail job,
boolean replaceExisting,
String state,
boolean forceState,
boolean recovering)
throws ObjectAlreadyExistsException,
JobPersistenceException

Insert or update a trigger.

**Throws:**
- ObjectAlreadyExistsException
- JobPersistenceException

---

**triggerExists**

protected boolean **triggerExists**(Connection conn,
TriggerKey key)
throws JobPersistenceException

Check existence of a given trigger.
removeJob

public boolean removeJob(JobKey jobKey)
    throws JobPersistenceException

Remove (delete) the Job with the given name, and any Trigger s that reference it.

If removal of the Job results in an empty group, the group should be removed from the JobStore's list of known group names.

Specified by:
    removeJob in interface org.quartz.spi.JobStore

Returns:
    true if a Job with the given name & group was found and removed from the store.

Throws:
    JobPersistenceException

removeJob

protected boolean removeJob(Connection conn,
    JobKey jobKey,
    boolean activeDeleteSafe)
    throws JobPersistenceException

Throws:
    JobPersistenceException

removeJobs

public boolean removeJobs(List<JobKey> jobKeys)
    throws JobPersistenceException

Specified by:
remove Jobs in interface org.quartz.spi.JobStore

Throws: JobPersistenceException

---

remove Triggers

public boolean removeTriggers(List<TriggerKey> triggerKeys)
throws JobPersistenceException

Specified by:
removeTriggers in interface org.quartz.spi.JobStore

Throws: JobPersistenceException

---

store Jobs And Triggers

public void storeJobsAndTriggers(Map<JobDetail, List<Trigger>> triggersAndJobs, boolean replace)
throws ObjectAlreadyExistsException, JobPersistenceException

Specified by:
storeJobsAndTriggers in interface org.quartz.spi.JobStore

Throws: ObjectAlreadyExistsException, JobPersistenceException

---

retrieve Job

public JobDetail retrieveJob(JobKey jobKey)
throws JobPersistenceException

Retrieve the JobDetail for the given Job.

Specified by:
retrieveJob in interface org.quartz.spi.JobStore

Returns:
The desired Job, or null if there is no match.
Throws:

JobPersistenceException

---

**retrieveJob**

protected JobDetail retrieveJob(Connection conn, JobKey key) throws JobPersistenceException

---

**removeTrigger**

public boolean removeTrigger(TriggerKey triggerKey) throws JobPersistenceException

Remove (delete) the Trigger with the given name.

If removal of the Trigger results in an empty group, the group should be removed from the JobStore's list of known group names.

If removal of the Trigger results in an 'orphaned' Job that is not 'durable', then the Job should be deleted also.

**Specified by:**

removeTrigger in interface org.quartz.spi.JobStore

**Returns:**

true if a Trigger with the given name & group was found and removed from the store.

**Throws:**

JobPersistenceException

---

**removeTrigger**

protected boolean removeTrigger(Connection conn, TriggerKey key) throws JobPersistenceException

---
Throws:

   JobPersistenceException

replaceTrigger

public boolean replaceTrigger(TriggerKey triggerKey, org.quartz.spi.OperableTrigger newTrigger)

throws JobPersistenceException

Specified by:

   replaceTrigger in interface org.quartz.spi.JobStore

Throws:

   JobPersistenceException

See Also:


replaceTrigger

protected boolean replaceTrigger(Connection conn, TriggerKey key, org.quartz.spi.OperableTrigger newTrigger)

throws JobPersistenceException

Throws:

   JobPersistenceException

retrieveTrigger

public org.quartz.spi.OperableTrigger retrieveTrigger(TriggerKey triggerKey)

throws JobPersistenceException

Retrieve the given Trigger.

Specified by:

   retrieveTrigger in interface org.quartz.spi.JobStore

Returns:

   The desired Trigger, or null if there is no match.
Throws:  
`JobPersistenceException`

getTriggerState

```java
public Trigger.TriggerState getTriggerState(TriggerKey triggerKey) throws JobPersistenceException
```

Get the current state of the identified Trigger.

Specified by:
  `getTriggerState` in interface `org.quartz.spi.JobStore`

Throws:
  `JobPersistenceException`

See Also:
  Trigger#NORMAL, Trigger#PAUSED, Trigger#COMPLETE, Trigger#ERROR, Trigger#NONE

storeCalendar
public void storeCalendar(String calName,
Calendar calendar,
boolean replaceExisting,
boolean updateTriggers)
throws ObjectAlreadyExistsException,
JobPersistenceException

Store the given Calendar.

Specified by:
storeCalendar in interface org.quartz.spi.JobStore

Parameters:
calName - The name of the calendar.
calendar - The Calendar to be stored.
replaceExisting - If true, any Calendar existing in the JobStore
with the same name & group should be over-written.

Throws:
ObjectAlreadyExistsException - if a Calendar with the same name
already exists, and replaceExisting is set to false.
JobPersistenceException

protected void storeCalendar(Connection conn,
String calName,
Calendar calendar,
boolean replaceExisting,
boolean updateTriggers)
throws ObjectAlreadyExistsException,
JobPersistenceException

Throws:
ObjectAlreadyExistsException
JobPersistenceException

calendarExists

protected boolean calendarExists(Connection conn,
String calName)
throws JobPersistenceException
Throws: 
  JobPersistenceException

removeCalendar

public boolean removeCalendar(String calName) throws JobPersistenceException

Remove (delete) the Calendar with the given name.

If removal of the Calendar would result in Triggers pointing to non-existent calendars, then a JobPersistenceException will be thrown.

* 

Specified by: 
  removeCalendar in interface org.quartz.spi.JobStore

Parameters: 
  calName - The name of the Calendar to be removed.

Returns: 
  true if a Calendar with the given name was found and removed from the store.

Throws: 
  JobPersistenceException

removeCalendar

protected boolean removeCalendar(Connection conn, String calName) throws JobPersistenceException

Throws: 
  JobPersistenceException

retrieveCalendar

public Calendar retrieveCalendar(String calName) throws JobPersistenceException
Retrieve the given Trigger.

**Specified by:**
retrieveCalendar in interface org.quartz.spi.JobStore

**Parameters:**
calName - The name of the Calendar to be retrieved.

**Returns:**
The desired Calendar, or null if there is no match.

**Throws:**
JobPersistenceException

---

**retrieveCalendar**

protected Calendar retrieveCalendar(Connection conn, String calName)
throws JobPersistenceException

**Throws:**
JobPersistenceException

---

**getNumberOfJobs**

public int getNumberOfJobs()
throws JobPersistenceException

Get the number of Job s that are stored in the JobStore.

**Specified by:**
getNumberOfJobs in interface org.quartz.spi.JobStore

**Throws:**
JobPersistenceException

---

**getNumberOfJobs**

protected int getNumberOfJobs(Connection conn)
throws JobPersistenceException
Throws:

   JobPersistenceException

getNumberOfTriggers

public int getNumberOfTriggers()
   throws JobPersistenceException

   Get the number of Trigger s that are stored in the JobsStore.

   Specified by:
       getNumberOfTriggers in interface org.quartz.spi.JobStore

   Throws:
       JobPersistenceException

getNumberOfTriggers

protected int getNumberOfTriggers(Connection conn)
   throws JobPersistenceException

   Throws:
       JobPersistenceException

getNumberOfCalendars

public int getNumberOfCalendars()
   throws JobPersistenceException

   Get the number of Calendar s that are stored in the JobsStore.

   Specified by:
       getNumberOfCalendars in interface org.quartz.spi.JobStore

   Throws:
       JobPersistenceException

getNumberOfCalendars
protected int getNumberOfCalendars(Connection conn) throws JobPersistenceException

Throws:
   JobPersistenceException

---

getJobKeys

public Set<JobKey> getJobKeys(GroupMatcher<JobKey> matcher) throws JobPersistenceException

Get the names of all of the Jobs that matcher the given groupMatcher.

If there are no jobs in the given group name, the result should be an empty Set

Specified by:
   getJobKeys in interface org.quartz.spi.JobStore

Throws:
   JobPersistenceException

---

getJobNames

protected Set<JobKey> getJobNames(Connection conn, GroupMatcher<JobKey> matcher) throws JobPersistenceException

Throws:
   JobPersistenceException

---

checkExists

public boolean checkExists(JobKey jobKey) throws JobPersistenceException

Determine whether a Job with the given identifier already exists within the scheduler.
checkExists

protected boolean checkExists(Connection conn,
                               JobKey jobKey)
    throws JobPersistenceException

    Throws:
    JobPersistenceException

checkExists

public boolean checkExists(TriggerKey triggerKey)
    throws JobPersistenceException

    Determine whether a Trigger with the given identifier already exists within
    the scheduler.

    Specified by:
    checkExists in interface org.quartz.spi.JobStore

    Parameters:
    triggerKey - the identifier to check for

    Returns:
    true if a Trigger exists with the given identifier

    Throws:
    SchedulerException
    JobPersistenceException
checkExists

protected boolean checkExists(Connection conn, TriggerKey triggerKey)
throws JobPersistenceException

Throws:
   JobPersistenceException

clearAllSchedulingData

public void clearAllSchedulingData()
throws JobPersistenceException

Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

Specified by:
   clearAllSchedulingData in interface org.quartz.spi.JobStore

Throws:
   JobPersistenceException

clearAllSchedulingData

protected void clearAllSchedulingData(Connection conn)
throws JobPersistenceException

Throws:
   JobPersistenceException

getTriggerKeys

public Set<TriggerKey> getTriggerKeys(GroupMatcher<TriggerKey> matcher)
throws JobPersistenceException

Get the names of all of the Triggers that match the given group Matcher.
If there are no triggers in the given group name, the result should be an empty Set (not null).
Specified by:
getTriggerKeys in interface org.quartz.spi.JobStore

Throws:
JobPersistenceException

getTriggerNames

protected Set<TriggerKey> getTriggerNames(Connection conn, GroupMatcher<TriggerKey> matcher)

Throws:
JobPersistenceException

getJobGroupNames

public List<String> getJobGroupNames()

Get the names of all of the Job groups.

If there are no known group names, the result should be a zero-length array (not null).

Specified by:
getJobGroupNames in interface org.quartz.spi.JobStore

Throws:
JobPersistenceException

getJobGroupNames

protected List<String> getJobGroupNames(Connection conn)

Throws:
JobPersistenceException
getTriggerGroupNames

public List<String> getTriggerGroupNames()
throws JobPersistenceException

  Get the names of all of the Trigger groups.

  If there are no known group names, the result should be a zero-length array (not null).

  Specified by:
  getTriggerGroupNames in interface org.quartz.spi.JobStore
  Throws:  
  JobPersistenceException

getTriggerGroupNames

protected List<String> getTriggerGroupNames(Connection conn)
throws JobPersistenceException

  Throws:  
  JobPersistenceException

getCalendarNames

public List<String> getCalendarNames()
throws JobPersistenceException

  Get the names of all of the Calendar s in the JobStore.

  If there are no Calendars in the given group name, the result should be a zero-length array (not null).

  Specified by:
  getCalendarNames in interface org.quartz.spi.JobStore
  Throws:  
  JobPersistenceException
**getCalendarNames**

protected `List<String> getCalendarNames(Connection conn)`

Throws: `JobPersistenceException`

---

**getTriggersForJob**

public `List<org.quartz.spi.OperableTrigger> getTriggersForJob(JobKey triggerKey)`

Get all of the Triggers that are associated to the given Job.

If there are no matches, a zero-length array should be returned.

Specified by: `getTriggersForJob in interface org.quartz.spi.JobStore`

Throws: `JobPersistenceException`

---

**getTriggersForJob**

protected `List<org.quartz.spi.OperableTrigger> getTriggersForJob(Connection conn)`

Throws: `JobPersistenceException`

---

**pauseTrigger**

public `void pauseTrigger(TriggerKey triggerKey)`

Pause the Trigger with the given name.
pauseTrigger

```java
public void pauseTrigger(Connection conn,
                          TriggerKey triggerKey)
    throws JobPersistenceException
```

Pause the Trigger with the given name.

Throws:
JobPersistenceException

See Also:
#resumeTrigger(Connection, SchedulingContext, String, String)

pauseJob

```java
public void pauseJob(JobKey jobKey)
    throws JobPersistenceException
```

Pause the Job with the given name - by pausing all of its current Triggers.

Specified by:
pauseJob in interface org.quartz.spi.JobStore

Throws:
JobPersistenceException

See Also:
#resumeJob(SchedulingContext, String, String)

pauseJobs

```java
public Set<String> pauseJobs(GroupMatcher<JobKey> matcher)
```
**Throws:**

`JobPersistenceException`

Pause all of the Jobs matching the given groupMatcher - by pausing all of their Triggers.

**Specified by:**

pauseJobs in interface org.quartz.spi.JobStore

**Throws:**

`JobPersistenceException`

**See Also:**

#resumeJobGroup(SchedulingContext, String)

---

**checkBlockedState**

```java
protected String checkBlockedState(Connection conn, JobKey jobKey, String currentState) throws JobPersistenceException
```

Determines if a Trigger for the given job should be blocked. State can only transition to STATE_PAUSED_BLOCKED/BLOCKED from PAUSED/STATE_WAITING respectively.

**Returns:**

STATE_PAUSED_BLOCKED, BLOCKED, or the currentState.

**Throws:**

`JobPersistenceException`

---

**resumeTrigger**

```java
public void resumeTrigger(TriggerKey triggerKey) throws JobPersistenceException
```

Resume (un-pause) the Trigger with the given name.

If the Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Specified by:**
resumeTrigger in interface org.quartz.spi.JobStore

Throws:
   JobPersistenceException

See Also:
   #pauseTrigger(SchedulingContext, String, String)

---

resumeTrigger

public void resumeTrigger(Connection conn, TriggerKey key)
   throws JobPersistenceException

Resume (un-pause) the Trigger with the given name.

If the Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Throws:
   JobPersistenceException

See Also:
   #pauseTrigger(Connection, SchedulingContext, String, String)

---

resumeJob

public void resumeJob(JobKey jobKey)
   throws JobPersistenceException

Resume (un-pause) the Job with the given name.

If any of the Job's Trigger's missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
   resumeJob in interface org.quartz.spi.JobStore

Throws:
   JobPersistenceException

See Also:
   #pauseJob(SchedulingContext, String, String)
**resumeJobs**

```java
public Set<String> resumeJobs(GroupMatcher<JobKey> matcher)
    throws JobPersistenceException
```

Resume (un-pause) all of the Jobs in the given group.

If any of the Job s had Trigger s that missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Specified by:**
resumeJobs in interface org.quartz.spi.JobStore

**Throws:**
JobPersistenceException

**See Also:**
#pauseJobGroup(SchedulingContext, String)

---

**pauseTriggers**

```java
public Set<String> pauseTriggers(GroupMatcher<TriggerKey> matcher)
    throws JobPersistenceException
```

Pause all of the Triggers matching the given groupMatcher.

**Specified by:**
pauseTriggers in interface org.quartz.spi.JobStore

**Throws:**
JobPersistenceException

**See Also:**
#resumeTriggerGroup(SchedulingContext, String)

---

**pauseTriggerGroup**

```java
public Set<String> pauseTriggerGroup(Connection conn,
    GroupMatcher<TriggerKey> matcher)
    throws JobPersistenceException
```

Pause all of the Triggers matching the given groupMatcher.
getPausedTriggerGroups

public Set<String> getPausedTriggerGroups()
throws JobPersistenceException

Specified by:
getPausedTriggerGroups in interface org.quartz.spi.JobStore

Throws:
JobPersistenceException

getPausedTriggerGroups

public Set<String> getPausedTriggerGroups(Connection conn)
throws JobPersistenceException

Pause all of the Triggers in the given group.

Throws:
JobPersistenceException

See Also:
resumeTriggers(org.quartz.impl.matchers.GroupMatcher)

resumeTriggers

public Set<String> resumeTriggers(GroupMatcher<TriggerKey> matcher)
throws JobPersistenceException

Resume (un-pause) all of the Triggers matching the given groupMatcher.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
resumeTriggers in interface org.quartz.spi.JobStore

Throws:
  JobPersistenceException

See Also:
  pauseTriggers(org.quartz.impl.matchers.GroupMatcher)

---

resumeTriggerGroup

public Set<String> resumeTriggerGroup(Connection conn, GroupMatcher<TriggerKey> matcher)
  throws JobPersistenceException

Resume (un-pause) all of the Triggers matching the given groupMatcher.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Throws:
  JobPersistenceException

See Also:
  pauseTriggers(org.quartz.impl.matchers.GroupMatcher)

---

pauseAll

public void pauseAll()
  throws JobPersistenceException

Pause all triggers - equivalent of calling pauseTriggerGroup(group) on every group.

When resumeAll() is called (to un-pause), trigger misfire instructions WILL be applied.

Specified by:
  pauseAll in interface org.quartz.spi.JobStore

Throws:
  JobPersistenceException

See Also:
  #resumeAll(SchedulingContext),
#pauseTriggerGroup(SchedulingContext, String)

**pauseAll**

public void **pauseAll**(Connection conn)
    throws JobPersistenceException

Pause all triggers - equivalent of calling pauseTriggerGroup(group) on every group.

When resumeAll() is called (to un-pause), trigger misfire instructions WILL be applied.

**Throws:**
    JobPersistenceException

**See Also:**
    #resumeAll(SchedulingContext),
    #pauseTriggerGroup(SchedulingContext, String)

---

**resumeAll**

public void **resumeAll**()
    throws JobPersistenceException

Resume (un-pause) all triggers - equivalent of calling resumeTriggerGroup(group) on every group.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

**Specified by:**
    resumeAll in interface org.quartz.spi.JobStore

**Throws:**
    JobPersistenceException

**See Also:**
    #pauseAll(SchedulingContext)
resumeAll

public void resumeAll(Connection conn)
    throws JobPersistenceException

protected

Resume (un-pause) all triggers - equivalent of calling resumeTriggerGroup(group) on every group.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Throws:
    JobPersistenceException
See Also:
    #pauseAll(SchedulingContext)

getFiredTriggerRecordId

protected String getFiredTriggerRecordId()

acquireNextTriggers

public List<org.quartz.spi.OperableTrigger> acquireNextTriggers(long noLaterThan, int maxCount, long timeWindow)
    throws JobPersistenceException

Get a handle to the next N triggers to be fired, and mark them as 'reserved' by the calling scheduler.

Specified by:
    acquireNextTriggers in interface org.quartz.spi.JobStore

Throws:
    JobPersistenceException
See Also:
    #releaseAcquiredTrigger(SchedulingContext, Trigger)
acquireNextTrigger

protected List<org.quartz.spi.OperableTrigger> acquireNextTrigger(Connection conn, long noLaterThan, int maxCount, long timeWindow) throws JobPersistenceException

Throws:
   JobPersistenceException

---

releaseAcquiredTrigger

public void releaseAcquiredTrigger(org.quartz.spi.OperableTrigger trigger) throws JobPersistenceException

Inform the JobStore that the scheduler no longer plans to fire the given Trigger, that it had previously acquired (reserved).

Specified by:
   releaseAcquiredTrigger in interface org.quartz.spi.JobStore

Throws:
   JobPersistenceException

---

releaseAcquiredTrigger

protected void releaseAcquiredTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger) throws JobPersistenceException

Throws:
   JobPersistenceException

---

triggersFired

public List<org.quartz.spi.TriggerFiredResult> triggersFired(List<org.quartz.spi.OperableTrigger> triggers) throws JobPersistenceException


Inform the JobStore that the scheduler is now firing the given Trigger (executing its associated Job), that it had previously acquired (reserved).

Specified by:
triggersFired in interface org.quartz.spi.JobStore

Returns:
null if the trigger or its job or calendar no longer exist, or if the trigger was not successfully put into the 'executing' state.

Throws:
JobPersistenceException

---

triggerFired

protected org.quartz.spi.TriggerFiredBundle triggerFired(Connection conn, org.quartz.spi.OperableTrigger trigger) throws JobPersistenceException

Throws:
JobPersistenceException

---

triggeredJobComplete

public void triggeredJobComplete(org.quartz.spi.OperableTrigger trig JobDetail jobDetail, Trigger.CompletedExecutionInstruction throws JobPersistenceException

Inform the JobStore that the scheduler has completed the firing of the given Trigger (and the execution its associated Job), and that the JobDataMap in the given JobDetail should be updated if the Job is stateful.

Specified by:
triggeredJobComplete in interface org.quartz.spi.JobStore

Throws:
JobPersistenceException

---

triggeredJobComplete
protected void triggeredJobComplete(
Connection conn, 
org.quartz.spi.OperableTrigger trigger, 
JobDetail jobDetail, 
Trigger.CompletedExecutionInstruction throws JobPersistenceException

Throws: 
JobPersistenceException

getDelegate

protected DriverDelegate getDelegate() throws NoSuchDelegateException

Get the driver delegate for DB operations.

Throws: 
NoSuchDelegateException

getLockHandler

protected Semaphore getLockHandler()

setLockHandler

public void setLockHandler(Semaphore lockHandler)

doRecoverMisfires

protected JobStoreSupport.RecoverMisfiredJobsResult doRecoverMisfire throws

Throws: 
JobPersistenceException

signalSchedulingChangeOnTxCompletion
protected void signalSchedulingChangeOnTxCompletion(long candidateNewNe

---

clearAndGetSignalSchedulingChangeOnTxCompletion

protected Long clearAndGetSignalSchedulingChangeOnTxCompletion()

---

signalSchedulingChangeImmediately

protected void signalSchedulingChangeImmediately(long candidateNewNe

---

doCheckin

protected boolean doCheckin() throws JobPersistenceException

Throws:
JobPersistenceException

---

findFailedInstances

protected List<SchedulerStateRecord> findFailedInstances(Connection conn) throws JobPersistenceException

Get a list of all scheduler instances in the cluster that may have failed. This includes this scheduler if it is checking in for the first time.

Throws:
JobPersistenceException

---

calcFailedIfAfter

protected long calcFailedIfAfter(SchedulerStateRecord rec)

---

clusterCheckIn
protected List<SchedulerStateRecord> clusterCheckIn(Connection conn) throws JobPersistenceException

Throws:
JobPersistenceException

clusterRecover

protected void clusterRecover(Connection conn, List<SchedulerStateRecord> failedInstances) throws JobPersistenceException

Throws:
JobPersistenceException

logWarnIfNonZero

protected void logWarnIfNonZero(int val, String warning)

cleanupConnection

protected void cleanupConnection(Connection conn)

Cleanup the given database connection. This means restoring any modified auto commit or transaction isolation connection attributes, and then closing the underlying connection.

This is separate from closeConnection() because the Spring integration relies on being able to overload closeConnection() and expects the same connection back that it originally returned from the datasource.

See Also:
closeConnection(Connection)

closeConnection
protected void closeConnection(Connection conn)

Closes the supplied Connection.

Ignores a null Connection. Any exception thrown trying to close the Connection is logged and ignored.

Parameters:
conn - The Connection to close (Optional).

rollbackConnection

protected void rollbackConnection(Connection conn)

Rollback the supplied connection.

Logs any SQLException it gets trying to rollback, but will not propagate the exception lest it mask the exception that caused the caller to need to rollback in the first place.

Parameters:
conn - (Optional)

commitConnection

protected void commitConnection(Connection conn)

throws JobPersistenceException

Commit the supplied connection

Parameters:
conn - (Optional)

Throws:
JobPersistenceException - thrown if a SQLException occurs when the connection is committed

executeWithoutLock
public Object executeWithoutLock(JobStoreSupport.TransactionCallback
throws JobPersistenceException

Execute the given callback in a transaction. Depending on the JobStore, the surrounding transaction may be assumed to be already present (managed).

This method just forwards to executeInLock() with a null lockName.

Throws:
   JobPersistenceException
See Also:
   executeInLock(String, TransactionCallback)

executeInLock

protected void executeInLock(String lockName,
JobStoreSupport.VoidTransactionCallback
throws JobPersistenceException

Execute the given callback having acquired the given lock. Depending on the JobStore, the surrounding transaction may be assumed to be already present (managed). This version is just a handy wrapper around executeInLock that doesn't require a return value.

Parameters:
   lockName - The name of the lock to acquire, for example "TRIGGER_ACCESS". If null, then no lock is acquired, but the lockCallback is still executed in a transaction.

Throws:
   JobPersistenceException
See Also:
   executeInLock(String, TransactionCallback)

executeInLock

protected abstract Object executeInLock(String lockName,
JobStoreSupport.TransactionCallback
throws JobPersistenceException
Execute the given callback having acquired the given lock. Depending on the JobStore, the surrounding transaction may be assumed to be already present (managed).

**Parameters:**

lockName - The name of the lock to acquire, for example "TRIGGER_ACCESS". If null, then no lock is acquired, but the lockCallback is still executed in a transaction.

**Throws:**

`JobPersistenceException`

---

**executeInNonManagedTXLock**

protected void `executeInNonManagedTXLock(String lockName, JobStoreSupport.VoidTransactionCallback)`

throws `JobPersistenceException`

Execute the given callback having optionally acquired the given lock. This uses the non-managed transaction connection. This version is just a handy wrapper around `executeInNonManagedTXLock` that doesn't require a return value.

**Parameters:**

lockName - The name of the lock to acquire, for example "TRIGGER_ACCESS". If null, then no lock is acquired, but the lockCallback is still executed in a non-managed transaction.

**Throws:**

`JobPersistenceException`

**See Also:**

`executeInNonManagedTXLock(String, TransactionCallback)`

---

**executeInNonManagedTXLock**

protected `Object executeInNonManagedTXLock(String lockName, JobStoreSupport.TransactionCallback)`

throws `JobPersistenceException`

Execute the given callback having optionally acquired the given lock. This
uses the non-managed transaction connection.

**Parameters:**

lockName - The name of the lock to acquire, for example "TRIGGER_ACCESS". If null, then no lock is acquired, but the lockCallback is still executed in a non-managed transaction.

**Throws:**

*JobPersistenceException*
Enclosing class:
JobStoreSupport

Helper class for returning the composite result of trying to recover misfired jobs.

Field Summary

| static JobStoreSupport.RecoverMisfiredJobsResult NO_OP |

Constructor Summary

JobStoreSupport.RecoverMisfiredJobsResult(boolean hasMoreMisfiredTriggers, int processedMisiredTriggerCount, long earliestNewTime)

Method Summary

<table>
<thead>
<tr>
<th>long getEarliestNewTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getProcessedMisfiredTriggerCount()</td>
</tr>
<tr>
<td>boolean hasMoreMisfiredTriggers()</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

NO_OP

public static final JobStoreSupport.RecoverMisfiredJobsResult NO_OP

Constructor Detail

JobStoreSupport.RecoverMisfiredJobsResult

public JobStoreSupport.RecoverMisfiredJobsResult(boolean hasMoreMisfiredTriggers, int processedMisfiredTriggerCount, long earliestNewTime)

Method Detail

hasMoreMisfiredTriggers

public boolean hasMoreMisfiredTriggers()

getProcessedMisfiredTriggerCount

public int getProcessedMisfiredTriggerCount()

getEarliestNewTime

public long getEarliestNewTime()
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
Implement this interface to provide the code to execute within a transaction template. If no return value is required, `execute` should just return null.

**See Also:**
- `JobStoreSupport.executeInNonManagedTXLock(String, TransactionCallback)
- `JobStoreSupport.executeInLock(String, TransactionCallback)`
- `JobStoreSupport.executeWithoutLock(TransactionCallback)`

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>execute</code></td>
<td><code>Object execute(Connection conn)</code></td>
</tr>
</tbody>
</table>

### Method Detail

**execute**

```java
Object execute(Connection conn)
```

*Throws:* `JobPersistenceException`

**Throws:**
- `JobPersistenceException`
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
Interface

Enclosing class:

`JobStoreSupport`

protected static interface `JobStoreSupport.VoidTransactionCallback`

Implement this interface to provide the code to execute within a transaction template that has no return value.

See Also:

`JobStoreSupport.executeInNonManagedTXLock(String, TransactionCallback)`

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>execute(Connection conn)</code></td>
</tr>
</tbody>
</table>

### Method Detail

**execute**

```java
void execute(Connection conn)
```

Throws:

`JobPersistenceException`

`JobPersistenceException`
| Summary: Nested | Field | Constructor | Method |
| Detail: Field | Constructor | Method |
Class JobStoreTX

All Implemented Interfaces:
   Constants, org.quartz.spi.JobStore

public class JobStoreTX
extends JobStoreSupport

JobStoreTX is meant to be used in a standalone environment. Both commit and rollback will be handled by this class.

If you need a JobStore class to use within an application-server environment, use JobStoreCMT instead.

Author:
   Jeffrey Wescott, James House

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class org.quartz.impl.jdbcjobstore.JobStoreSupport</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobStoreSupport.RecoverMisfiredJobsResult, JobStoreSupport.TransactionCallback,</td>
</tr>
<tr>
<td>JobStoreSupport.VoidTransactionCallback</td>
</tr>
</tbody>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from class org.quartz.impl.jdbcjobstore.JobStoreSupport</th>
</tr>
</thead>
<tbody>
<tr>
<td>calendarCache, delegateClass, delegateClassName,</td>
</tr>
<tr>
<td>delegateInitString, dsName, firstCheckIn, instanceId</td>
</tr>
</tbody>
</table>
instanceName, lastCheckin, LOCK CALENDAR ACCESS, LOCK JOB ACCESS,
LOCK MISFIRE ACCESS, LOCK STATE ACCESS, LOCK_TRIGGER_ACCESS,
maxToRecoverAtATime, sigChangeForTxCompletion, tablePrefix,
useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants

ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL CALENDAR, COL CALENDAR_NAME, COL_CHECKIN_INTERVAL,
COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID,
COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME,
COL_IS_DURABLE, COL_IS_NONCONCURRENT, COL_IS_UPDATE_DATA,
COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP,
COL_JOB_NAME, COL_LAST_CHECKIN_TIME, COL_LOCK_NAME,
COL_MISFIRE_INSTRUCTION, COL_NEXT_FIRE_TIME, COL_PREV_FIRE_TIME,
COL_PRIORITY, COL_REPEAT_COUNT, COL_REPEAT_INTERVAL,
COL_REQUESTS_RECOVERY, COL_SCHEDULER_NAME, COL_START_TIME,
COL_TIME_ZONE_ID, COL_TIMES_TRIGGERED, COL_TRIGGER_GROUP,
COL_TRIGGER_NAME, COL_TRIGGER_STATE, COL_TRIGGER_TYPE,
DEFAULT_TABLE_PREFIX, STATE_ACQUIRED, STATE_BLOCKED,
STATE_COMPLETE, STATE_DELETED, STATE_ERROR, STATE_EXECUTING,
STATE_MISFIRED, STATE_PAUSED, STATE_PAUSED_BLOCKED, STATE_WAITING,
TABLE_BLOB_TRIGGERS, TABLE_CALENDARS, TABLE_CRON_TRIGGERS,
TABLE_FIRED_TRIGGERS, TABLE_JOB_DETAILS, TABLE_LOCKS,
TABLE_PAUSED_TRIGGERS, TABLE_SCHEDULER_STATE,
TABLE_SIMPLE_TRIGGERS, TABLE_TRIGGERS, TTYPE_BLOB, TTYPE_CAL_INT,
TTYPE_CRON, TTYPE_SIMPLE

Constructor Summary

JobStoreTX()
Called by the QuartzScheduler before the JobStore is used, in order to give it a chance to initialize.

<table>
<thead>
<tr>
<th>Methods inherited from class org.quartz.impl.jdbcjobstore.JobStoreSupport</th>
</tr>
</thead>
<tbody>
<tr>
<td>acquireNextTrigger, acquireNextTriggers, calcFailedIfAfter,</td>
</tr>
<tr>
<td>calendarExists, canUseProperties, checkBlockedState, checkExists,</td>
</tr>
<tr>
<td>checkExists, checkExists, checkExists, cleanupConnection,</td>
</tr>
<tr>
<td>clearAllSchedulingData, clearAllSchedulingData,</td>
</tr>
<tr>
<td>clearAndGetSignalSchedulingChangeOnTxCompletion, closeConnection,</td>
</tr>
<tr>
<td>clusterCheckIn, clusterRecover, commitConnection, doCheckin,</td>
</tr>
<tr>
<td>doRecoverMisfires, executeInLock, executeInNonManagedTXLock,</td>
</tr>
<tr>
<td>executeInNonManagedTXLock, executeWithoutLock,</td>
</tr>
<tr>
<td>findFailedInstances, getAttributeRestoringConnection,</td>
</tr>
<tr>
<td>getCalendarNames, getCalendarNames, getClassLoadHelper,</td>
</tr>
<tr>
<td>getClusterCheckinInterval, getConnection, getDataSource,</td>
</tr>
<tr>
<td>getDbRetryInterval, getDelegate, getDoubleCheckLockMisfireHandler,</td>
</tr>
<tr>
<td>getDriverDelegateClass, getDriverDelegateInitString,</td>
</tr>
<tr>
<td>getEstimatedTimeToReleaseAndAcquireTrigger,</td>
</tr>
<tr>
<td>getFiredTriggerRecordId, getInstanceId, getJobGroupName,</td>
</tr>
<tr>
<td>getJobGroupNames, getJobGroupNames, getJobKeys, getJobNames,</td>
</tr>
<tr>
<td>getLockHandler, getLog, getMakeThreadsDaemons,</td>
</tr>
<tr>
<td>getMaxMisfiresToHandleAtATime, getMisfireThreshold,</td>
</tr>
<tr>
<td>getNumberOfCalendars, getNumberOfCalendars,</td>
</tr>
<tr>
<td>getNumberOfJobs, getNumberOfJobs, getNumberOfTriggers,</td>
</tr>
<tr>
<td>getNumberOfTriggers, getPausedTriggerGroups,</td>
</tr>
<tr>
<td>getPausedTriggerGroups, getSelectWithLockSQL, getTablePrefix,</td>
</tr>
<tr>
<td>getTriggerGroupName, getTriggerGroupNames, getTriggerGroupNames,</td>
</tr>
<tr>
<td>getTriggerGroupName, getTriggersForJob, getTriggersForJob,</td>
</tr>
<tr>
<td>getTriggerState, getTriggerState, getUseDBlocks,</td>
</tr>
<tr>
<td>isAcquireTriggersWithinLock, isClustered,</td>
</tr>
<tr>
<td>isDontSetAutoCommitFalse, isLockOnInsert,</td>
</tr>
<tr>
<td>isTxIsolationLevelSerializable, jobExists, logWarnIfNonZero,</td>
</tr>
<tr>
<td>pauseAll, pauseAll, pauseJob, pauseJobs, pauseTrigger,</td>
</tr>
<tr>
<td>pauseTrigger, pauseTriggerGroup, pauseTriggers, recoverJobs,</td>
</tr>
<tr>
<td>recoverJobs, recoverMisfiredJobs, releaseAcquiredTrigger,</td>
</tr>
<tr>
<td>releaseAcquiredTrigger, releaseLock, removeCalendar,</td>
</tr>
<tr>
<td>removeCalendar, removeJob, removeJob, removeJobs, removeTrigger,</td>
</tr>
<tr>
<td>removeTrigger, removeTriggers, replaceTrigger, replaceTriggers, resumeAll,</td>
</tr>
<tr>
<td>resumeAll, resumeAll, resumeJob, resumeJobs, resumeTrigger,</td>
</tr>
<tr>
<td>resumeTrigger, resumeTriggerGroup, resumeTriggers,</td>
</tr>
<tr>
<td>retrieveCalendar, retrieveCalendar, retrieveJob, retrieveJob,</td>
</tr>
<tr>
<td>retrieveTrigger, retrieveTrigger, rollbackConnection,</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.**Object**

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

### Constructor Detail

**JobStoreTX**

```java
public JobStoreTX()
```

### Method Detail

**initialize**

```java
public void initialize(org.quartz.spi.ClassLoadHelper loadHelper,
                      org.quartz.spi.SchedulerSignaler signaler)
          throws SchedulerConfigException
```

**Description copied from class: **JobStoreSupport**

Called by the QuartzScheduler before the JobStore is used, in order to give it a chance to initialize.
Specified by:
initialize in interface org.quartz.spi.JobStore

Overrides:
initialize in class JobStoreSupport

Throws:
SchedulerConfigException

getNonManagedTXConnection

protected Connection getNonManagedTXConnection() throws JobPersistenceException

For JobStoreTX, the non-managed TX connection is just the normal connection because it is not CMT.

Specified by:
getNonManagedTXConnection in class JobStoreSupport

Throws:
JobPersistenceException

See Also:
JobStoreSupport.getConnection()

executeInLock

protected Object executeInLock(String lockName, JobStoreSupport.TransactionCallback txCallback) throws JobPersistenceException

Execute the given callback having optionally aquired the given lock. For JobStoreTX, because it manages its own transactions and only has the one datasource, this is the same behavior as executeInNonManagedTXLock().

Specified by:
executeInLock in class JobStoreSupport

Parameters:
lockName - The name of the lock to aquire, for example "TRIGGER_ACCESS". If null, then no lock is aquired, but the lockCallback is still executed in a transaction.
Throws:

   JobPersistenceException

See Also:

   JobStoreSupport.executeInNonManagedTXLock(String, TransactionCallback),
   JobStoreSupport.executeInLock(String, TransactionCallback),
   JobStoreSupport.getNonManagedTXConnection(),
   JobStoreSupport.getConnection()
org.quartz.impl.jdbcjobstore  Class JTANonClusteredSemaphore

java.lang.Object
  └ org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore

All Implemented Interfaces:
  Semaphore

public class JTANonClusteredSemaphore
  extends Object
  implements Semaphore

Provides in memory thread/resource locking that is JTA Transaction aware. It is most appropriate for use when using JobStoreCMT without clustering.

This Semaphore implementation is not Quartz cluster safe.

When a lock is obtained/released but there is no active JTA Transaction, then this Semaphore operates just like SimpleSemaphore.

By default, this class looks for the TransactionManager in JNDI under name "java:TransactionManager". If this is not where your Application Server registers it, you can modify the JNDI lookup location using the "transactionManagerJNDIName" property.

IMPORTANT: This Semaphore implementation is currently experimental. It has been tested a limited amount on JBoss 4.0.3SP1. If you do choose to use it, any feedback would be most appreciated!

See Also:
  SimpleSemaphore

---

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String DEFAULT_TRANSACTION_MANAGER_LOCATION</td>
</tr>
</tbody>
</table>
### Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTANonClusteredSemaphore()</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected org.slf4j.Logger getLog()</td>
<td></td>
</tr>
<tr>
<td>protected Transaction getTransaction()</td>
<td>Helper method to get the current Transaction from the TransactionManager in JNDI.</td>
</tr>
<tr>
<td>boolean isLockOwner(Connection conn, String lockName)</td>
<td>Determine whether the calling thread owns a lock on the identified resource.</td>
</tr>
<tr>
<td>boolean obtainLock(Connection conn, String lockName)</td>
<td>Grants a lock on the identified resource to the calling thread (blocking until it is available).</td>
</tr>
<tr>
<td>void releaseLock(Connection conn, String lockName)</td>
<td>Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.</td>
</tr>
<tr>
<td>protected void releaseLock(String lockName, boolean fromSynchronization)</td>
<td>Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.</td>
</tr>
<tr>
<td>boolean requiresConnection()</td>
<td>This Semaphore implementation does not use the database.</td>
</tr>
<tr>
<td>void setTransactionManagerJNDIName(String transactionManagerJNDIName)</td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Field Detail
DEFAULT_TRANSACTION_MANANGER_LOCATION

public static final String DEFAULT_TRANSACTION_MANANGER_LOCATION

See Also:
Constant Field Values

Constructor Detail

JTANonClusteredSemaphore

class JTANonClusteredSemaphore

public JTANonClusteredSemaphore()

Method Detail

getLog

protected org.slf4j.Logger getLog()

setTransactionManagerJNDIName

public void setTransactionManagerJNDIName(String transactionManagerJNDIName)

obtainLock

public boolean obtainLock(Connection conn, String lockName)
throws LockException

Grants a lock on the identified resource to the calling thread (blocking until it is available).

Specified by:
obtainLock in interface Semaphore

Parameters:
conn - Database connection used to establish lock. Can be null if
Semaphore.requiresConnection() returns false.

**Returns:**
true if the lock was obtained.

**Throws:**
LockException

---

**getTransaction**

```java
protected Transaction getTransaction() throws LockException
```

Helper method to get the current Transaction from the TransactionManager in JNDI.

**Returns:**
The current Transaction, null if not currently in a transaction.

**Throws:**
LockException

---

**releaseLock**

```java
public void releaseLock(Connection conn, String lockName) throws LockException
```

Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.

**Specified by:**
`releaseLock` in interface Semaphore

**Parameters:**
conn - Database connection used to establish lock. Can be null if Semaphore.requiresConnection() returns false.

**Throws:**
LockException

---

**releaseLock**
protected void **releaseLock**(String lockName, boolean fromSynchronization) throws LockException

Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.

**Parameters:**
fromSynchronization - True if this method is being invoked from Synchronization notified of the enclosing transaction having completed.

**Throws:**
LockException - Thrown if there was a problem accessing the JTA Transaction. Only relevant if fromSynchronization is false.

---

**isLockOwner**

public boolean **isLockOwner**(Connection conn, String lockName)

Determine whether the calling thread owns a lock on the identified resource.

**Specified by:**
isLockOwner in interface Semaphore

**Parameters:**
conn - Database connection used to establish lock. Can be null if Semaphore.requiresConnection() returns false.

---

**requiresConnection**

public boolean **requiresConnection**()

This Semaphore implementation does not use the database.

**Specified by:**
requiresConnection in interface Semaphore

See Also:
Semaphore.isLockOwner(Connection, String), Semaphore.obtainLock(Connection, String), Semaphore.releaseLock(Connection, String)

<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
</tr>
<tr>
<td>Copyright 2001-2011, Terracotta, Inc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
org.quartz.impl.jdbcjobstore Class LockException

java.lang.Object  
   ↘ java.lang.Throwable  
   ↘ java.lang.Exception  
   ↘ org.quartz.SchedulerException  
   ↘ org.quartz.JobPersistenceException  
   ↘ org.quartz.impl.jdbcjobstore.LockException

All Implemented Interfaces:  
   Serializable

public class LockException
   extends JobPersistenceException

Exception class for when there is a failure obtaining or releasing a resource lock.

Author:  
   James House

See Also:  
   Semaphore, Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockException(String msg)</td>
<td></td>
</tr>
<tr>
<td>LockException(String msg, Throwable cause)</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods inherited from class org.quartz.SchedulerException</td>
<td></td>
</tr>
</tbody>
</table>
   getUnderlyingException, toString |
Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockException

public LockException(String msg)

LockException

public LockException(String msg, Throwable cause)
**org.quartz.impl.jdbcjobstore**  
**Class MSSQLDelegate**

```java
does not inherit java.lang.Object
```

**All Implemented Interfaces:**  
- Constants  
- DriverDelegate  
- StdJDBCConstants

---

```java
public class MSSQLDelegate
extends StdJDBCDelegate
```

This is a driver delegate for the MSSQL JDBC driver.

**Author:**  
Jeffrey Wescott

---

## Field Summary

### Fields inherited from class **org.quartz.impl.jdbcjobstore.StdJDBCDelegate**
- `classLoadHelper`, `instanceId`, `logger`, `schedName`, `tablePrefix`,  
- `triggerPersistenceDelegates`, `useProperties`

### Fields inherited from interface **org.quartz.impl.jdbcjobstore.StdJDBCConstants**
- `COUNT_MISFIRED_TRIGGERS_IN_STATE`, `DELETE_ALL_BLOB_TRIGGERS`,  
- `DELETE_ALL.Calendar`, `DELETE_ALL_CRON_TRIGGER`,  
- `DELETE_ALL_JOB_DETAILS`, `DELETE_ALL_PAUSED_TRIGGER_GRPS`,  
- `DELETE_ALL_SIMPLE_TRIGGERS`, `DELETE_ALL_SIMPROP_TRIGGERS`,  
- `DELETE_ALL_TRIGGERS`, `DELETE_BLOB_TRIGGER`, `DELETE_CALENDAR`,  
- `DELETE_CRON_TRIGGER`, `DELETE_FIRED_TRIGGER`, `DELETE_FIRED_TRIGGERS`,  
- `DELETE_INSTANCES_FIRED_TRIGGERS`, `DELETE_JOB_DETAIL`,  
- `DELETE_NO_RECOVERY_FIRED_TRIGGERS`, `DELETE_PAUSED_TRIGGER_GROUP`,  
- `DELETE_PAUSED_TRIGGER_GROUPS`, `DELETE_SCHEDULER_STATE`,  
- `DELETE_SIMPLE_TRIGGER`, `DELETE_TRIGGER`, `INSERT_BLOB_TRIGGER`,  
- `INSERT_CALENDAR`, `INSERT_CRON_TRIGGER`, `INSERT_FIRED_TRIGGER`,
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS_COL_NEXT_FIRE_TIME</td>
<td>Alias for the next fire time column</td>
</tr>
<tr>
<td>ALL_GROUPS_PAUSED</td>
<td>True if all groups are paused</td>
</tr>
<tr>
<td>COL_BLOB</td>
<td>Blob column</td>
</tr>
<tr>
<td>COL_CALENDAR</td>
<td>Calendar column</td>
</tr>
<tr>
<td>COL_CALENDAR_NAME</td>
<td>Calendar name</td>
</tr>
<tr>
<td>COL_CHECKIN_INTERVAL</td>
<td>Check-in interval</td>
</tr>
<tr>
<td>COL_CRON_EXPRESSION</td>
<td>Cron expression</td>
</tr>
<tr>
<td>COL_DESCRIPTION</td>
<td>Description column</td>
</tr>
<tr>
<td>COL_END_TIME</td>
<td>End time column</td>
</tr>
<tr>
<td>COL_ENTRY_ID</td>
<td>Entry ID</td>
</tr>
<tr>
<td>COL_ENTRY_STATE</td>
<td>Entry state column</td>
</tr>
<tr>
<td>COL_FIRED_TIME</td>
<td>Fired time column</td>
</tr>
<tr>
<td>COL_INSTANCE_NAME</td>
<td>Instance name</td>
</tr>
<tr>
<td>COL_IS_DURABLE</td>
<td>True if the job is durable</td>
</tr>
<tr>
<td>COL_IS_NONCONCURRENT</td>
<td>True if the job is non-concurrent</td>
</tr>
<tr>
<td>COL_IS_UPDATE_DATA</td>
<td>True if the job is update data</td>
</tr>
<tr>
<td>COL_IS_VOLATILE</td>
<td>True if the job is volatile</td>
</tr>
<tr>
<td>COL_JOB_CLASS</td>
<td>Job class</td>
</tr>
<tr>
<td>COL_JOB_DATA</td>
<td>Job data</td>
</tr>
<tr>
<td>COL_JOB_DATAMAP</td>
<td>Job data map</td>
</tr>
<tr>
<td>COL_JOB_GROUP</td>
<td>Job group</td>
</tr>
<tr>
<td>COL_JOB_EXECUTION_COUNT</td>
<td>Execution count</td>
</tr>
<tr>
<td>COL_JOB_EXISTENCE</td>
<td>True if the job exists</td>
</tr>
<tr>
<td>COL_JOB_GROUPS</td>
<td>Job groups</td>
</tr>
<tr>
<td>COL_JOB_NONCONCURRENT</td>
<td>True if the job is non-concurrent</td>
</tr>
<tr>
<td>COL_MISFIRED_TRIGGERS</td>
<td>True if the job has misfired triggers</td>
</tr>
<tr>
<td>SELECT_INSTANCES_FIRED_TRIGGER_STATE</td>
<td>Select instances fired trigger state</td>
</tr>
<tr>
<td>SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS</td>
<td>Select instances recoverable fired triggers</td>
</tr>
<tr>
<td>SELECT_JOB_DETAIL</td>
<td>Select job detail</td>
</tr>
<tr>
<td>SELECT_JOB_EXECUTION_COUNT</td>
<td>Select job execution count</td>
</tr>
<tr>
<td>SELECT_JOB_EXISTENCE</td>
<td>Select job existence</td>
</tr>
<tr>
<td>SELECT_JOB_GROUPS</td>
<td>Select job groups</td>
</tr>
<tr>
<td>SELECT_JOB_GROUPS_IN_GROUP_IN_STATE</td>
<td>Select job groups in state</td>
</tr>
<tr>
<td>SELECT_JOB_NONCONCURRENT</td>
<td>Select job non-concurrent</td>
</tr>
<tr>
<td>SELECT_JOBS_IN_GROUP</td>
<td>Select jobs in group</td>
</tr>
<tr>
<td>SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE</td>
<td>Select misfired triggers in group in state</td>
</tr>
<tr>
<td>SELECT_MISFIRED_TRIGGERS_IN_STATE</td>
<td>Select misfired triggers in state</td>
</tr>
<tr>
<td>SELECT_NEXT_FIRE_TIME</td>
<td>Select next fire time</td>
</tr>
<tr>
<td>SELECT_NEXT_TRIGGER_TO_ACQUIRE</td>
<td>Select next trigger to acquire</td>
</tr>
<tr>
<td>SELECT_NUM_CALENDARS</td>
<td>Select number of calendars</td>
</tr>
<tr>
<td>SELECT_NUM_JOBS</td>
<td>Select number of jobs</td>
</tr>
<tr>
<td>SELECT_NUM_TRIGGERS</td>
<td>Select number of triggers</td>
</tr>
<tr>
<td>SELECT_NUM_TRIGGERS_FOR_JOB</td>
<td>Select number of triggers for job</td>
</tr>
<tr>
<td>SELECT_PAUSED_TRIGGER_GROUPS</td>
<td>Select paused trigger groups</td>
</tr>
<tr>
<td>SELECT_REFERENCEDCALENDAR</td>
<td>Select referenced calendar</td>
</tr>
<tr>
<td>SELECT_SCHEDULER_STATE</td>
<td>Select scheduler state</td>
</tr>
<tr>
<td>SELECT_SCHEDULER_STATES</td>
<td>Select scheduler states</td>
</tr>
<tr>
<td>SELECT_SIMPLE_TRIGGER</td>
<td>Select simple trigger</td>
</tr>
<tr>
<td>SELECT_TRIGGER</td>
<td>Select trigger</td>
</tr>
<tr>
<td>SELECT_TRIGGER_DATA</td>
<td>Select trigger data</td>
</tr>
<tr>
<td>SELECT_TRIGGER_GROUPS</td>
<td>Select trigger groups</td>
</tr>
<tr>
<td>SELECT_TRIGGER_GROUPS_FILTERED</td>
<td>Select trigger groups filtered</td>
</tr>
<tr>
<td>SELECT_TRIGGER_STATE</td>
<td>Select trigger state</td>
</tr>
<tr>
<td>SELECT_TRIGGER_STATUS</td>
<td>Select trigger status</td>
</tr>
<tr>
<td>SELECT_TRIGGERS_FORCALENDAR</td>
<td>Select triggers for calendar</td>
</tr>
<tr>
<td>SELECT_TRIGGERS_FOR_JOB</td>
<td>Select triggers for job</td>
</tr>
<tr>
<td>SELECT_TRIGGERS_IN_GROUP</td>
<td>Select triggers in group</td>
</tr>
<tr>
<td>SELECT_TRIGGERS_IN_GROUP_IN_STATE</td>
<td>Select triggers in group in state</td>
</tr>
<tr>
<td>TABLE_PREFIX_SUBST</td>
<td>Table prefix substitution</td>
</tr>
<tr>
<td>UPDATE_INSTANCES_FIRED_TRIGGER_STATE</td>
<td>Update instances fired trigger state</td>
</tr>
<tr>
<td>UPDATE_JOB_DATA</td>
<td>Update job data</td>
</tr>
<tr>
<td>UPDATE_JOB_DETAIL</td>
<td>Update job detail</td>
</tr>
<tr>
<td>UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE</td>
<td>Update job trigger states from other state</td>
</tr>
<tr>
<td>UPDATE_SCHEDULER_STATE</td>
<td>Update scheduler state</td>
</tr>
<tr>
<td>UPDATE_SIMPLE_TRIGGER</td>
<td>Update simple trigger</td>
</tr>
<tr>
<td>UPDATE_TRIGGER</td>
<td>Update trigger</td>
</tr>
<tr>
<td>UPDATE_TRIGGER_GROUP</td>
<td>Update trigger group</td>
</tr>
<tr>
<td>UPDATE_TRIGGER_GROUP_STATE_FROM_STATE</td>
<td>Update trigger group state from state</td>
</tr>
<tr>
<td>UPDATE_TRIGGER_STATE_FROM_STATES</td>
<td>Update trigger state from states</td>
</tr>
<tr>
<td>UPDATE_TRIGGER_STATE_FROM_STATE_FROM_STATE</td>
<td>Update trigger state from state from state</td>
</tr>
<tr>
<td>UPDATE_TRIGGER_STATE_FROM_OTHER_STATES</td>
<td>Update trigger state from other states</td>
</tr>
<tr>
<td>UPDATE_TRIGGER_STATE_FROM_OTHER_STATES</td>
<td>Update trigger state from other states</td>
</tr>
</tbody>
</table>

Fields inherited from interface org.quartz.impl.jdbcjobstore.Const...
Constructor Summary

**MSSQLDelegate**(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)

  Create new MSSQLDelegate instance.

  **MSSQLDelegate**(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)

Method Summary

**protected Object** getJobDataFromBlob(ResultSet rs, String colName)

  This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.

**protected Object** getObjectFromBlob(ResultSet rs, String colName)

  This method should be overridden by any delegate subclasses that need special handling for BLOBs.

Methods inherited from class
org.quartz.impl.jdbcjobstore.StdJDBCDelegate

addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet,
closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

**Methods inherited from class java.lang.Object**
closeStatement, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
**Constructor Detail**

**MSSQLDelegate**

```java
public MSSQLDelegate(org.slf4j.Logger log,
                      String tablePrefix,
                      String schedName,
                      String instanceId,
                      org.quartz.spi.ClassLoadHelper classLoadHelper)
```

Create new MSSQLDelegate instance.

**Parameters:**
- `log` - the logger to use during execution
- `tablePrefix` - the prefix of all table names

---

**Method Detail**

**getObjectFromBlob**

```java
protected Object getObjectFromBlob(ResultSet rs,
                                    String colName)
```

Throws `ClassNotFoundException`, `IOException`, `SQLException`

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard JDBC `java.sql.Blob` operations.
Overrides:
  getObjectFromBlob in class StdJDBCDelegate

Parameters:
  rs - the result set, already queued to the correct row
colName - the column name for the BLOB

Returns:
  the deserialized Object from the ResultSet BLOB

Throws:
  ClassNotFoundException - if a class found during deserialization cannot be found
  IOException - if deserialization causes an error
  SQLException

---

getJobDataFromBlob

protected Object getJobDataFromBlob(ResultSet rs,
          String colName)
throws ClassNotFoundException,
          IOException,
          SQLException

Description copied from class: StdJDBCDelegate

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:
  getJobDataFromBlob in class StdJDBCDelegate

Parameters:
  rs - the result set, already queued to the correct row
colName - the column name for the BLOB

Returns:
  the deserialized Object from the ResultSet BLOB

Throws:
  ClassNotFoundException - if a class found during deserialization cannot be found
  IOException - if deserialization causes an error
  SQLException
Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.jdbcjobstore  Class NoSuchDelegateException

java.lang.Object  
   ↙ java.lang.Throwable  
   ↙ java.lang.Exception  
   ↙ org.quartz.SchedulerException  
   ↙ org.quartz.JobPersistenceException  
   ↙ org.quartz.impl.jdbcjobstore.NoSuchDelegateException

All Implemented Interfaces:
   Serializable

public class NoSuchDelegateException

extends JobPersistenceException

Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.

Author:
   Jeffrey Wescott

See Also:
   Serialized Form

Constructor Summary

NoSuchDelegateException(String msg)

NoSuchDelegateException(String msg, Throwable cause)

Method Summary

Methods inherited from class org.quartz.SchedulerException
getUnderlyingException, toString
Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NoSuchDelegateException

public NoSuchDelegateException(String msg)

NoSuchDelegateException

public NoSuchDelegateException(String msg, Throwable cause)

Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.jdbcjobstore

Interfaces  Constants  
DriverDelegate  JobStoreSupport.TransactionCallback  JobStoreSupport.VoidTransactionCallback  Semaphore  StdJDBCConstants  TablePrefixAware  TriggerPersistenceDelegate

Classes  
AttributeRestoringConnectionInvocationHandler  CalendarIntervalTriggerPersistenceDelegate  CloudscapeDelegate  CronTriggerPersistenceDelegate  DB2v6Delegate  DB2v7Delegate  DB2v8Delegate  DBSemaphore  FiredTriggerRecord  HSQLDBDelegate  JobStoreCMT  JobStoreSupport  JobStoreSupport.RecoverMisfiredJobsResult  JobStoreTX  JTANonClusteredSemaphore  MSSQLDelegate  PointbaseDelegate  PostgreSQLDelegate  SchedulerStateRecord  SimplePropertiesTriggerPersistenceDelegateSupport  SimplePropertiesTriggerProperties  SimpleSemaphore  SimpleTriggerPersistenceDelegate  StdJDBCDelegate  StdRowLockSemaphore  SybaseDelegate  TriggerPersistenceDelegate.TriggerPropertyBundle  TriggerStatus  UpdateLockRowSemaphore  Util  WebLogicDelegate

Exceptions  
InvalidConfigurationException  LockException  NoSuchDelegateException
## Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Constants</strong></td>
<td>This interface can be implemented by any DriverDelegate class that needs to use the constants contained herein.</td>
</tr>
<tr>
<td><strong>DriverDelegate</strong></td>
<td>This is the base interface for all driver delegate classes.</td>
</tr>
<tr>
<td><strong>JobStoreSupport.TransactionCallback</strong></td>
<td>Implement this interface to provide the code to execute within the a transaction template.</td>
</tr>
<tr>
<td><strong>JobStoreSupport.VoidTransactionCallback</strong></td>
<td>Implement this interface to provide the code to execute within the a transaction template that has no return value.</td>
</tr>
<tr>
<td><strong>Semaphore</strong></td>
<td>An interface for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td><strong>StdJDBCConstants</strong></td>
<td>This interface extends Constants to include the query string constants in use by the StdJDBCDelegate class.</td>
</tr>
<tr>
<td><strong>TablePrefixAware</strong></td>
<td>Interface for Quartz objects that need to know what the table prefix of the tables used by a JDBC JobStore is.</td>
</tr>
<tr>
<td><strong>TriggerPersistenceDelegate</strong></td>
<td>An interface which provides an implementation for storing a particular type of Trigger's</td>
</tr>
<tr>
<td>Class Summary</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>AttributeRestoringConnectionInvocationHandler</strong></td>
<td></td>
</tr>
<tr>
<td>Protects a connection's attributes from being permanently modified.</td>
<td></td>
</tr>
<tr>
<td><strong>CalendarIntervalTriggerPersistenceDelegate</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Deprecated. Use the StdJDBCDelegate for latest versions of Derby</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CloudscapeDelegate</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DB2v6Delegate</strong></td>
<td></td>
</tr>
<tr>
<td>Quartz JDBC delegate for DB2 v6 databases.</td>
<td></td>
</tr>
<tr>
<td><strong>DB2v7Delegate</strong></td>
<td></td>
</tr>
<tr>
<td>Quartz JDBC delegate for DB2 v7 databases.</td>
<td></td>
</tr>
<tr>
<td><strong>DB2v8Delegate</strong></td>
<td></td>
</tr>
<tr>
<td>Quartz JDBC delegate for DB2 v8 databases.</td>
<td></td>
</tr>
<tr>
<td><strong>DBSemaphore</strong></td>
<td></td>
</tr>
<tr>
<td>Base class for database lock handlers for providing thread/resource locking</td>
<td></td>
</tr>
<tr>
<td>to protect resources from being altered by multiple threads at the same time.</td>
<td></td>
</tr>
<tr>
<td><strong>FiredTriggerRecord</strong></td>
<td></td>
</tr>
<tr>
<td>Conveys the state of a fired trigger record.</td>
<td></td>
</tr>
<tr>
<td><strong>HSQLDBDelegate</strong></td>
<td></td>
</tr>
<tr>
<td>This is a driver delegate for the HSQLDB database.</td>
<td></td>
</tr>
<tr>
<td><strong>JobStoreCMT</strong></td>
<td></td>
</tr>
<tr>
<td>JobStoreCMT is meant to be used in an application-server environment that</td>
<td></td>
</tr>
<tr>
<td>provides container-managed-transactions.</td>
<td></td>
</tr>
<tr>
<td><strong>JobStoreSupport</strong></td>
<td></td>
</tr>
<tr>
<td>Contains base functionality for JDBC-based JobStore implementations.</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JobStoreSupport.RecoverMisfiredJobsResult</td>
<td>Helper class for returning the composite result of trying to recover misfired jobs.</td>
</tr>
<tr>
<td>JobStoreTX</td>
<td>JobStoreTX is meant to be used in a standalone environment.</td>
</tr>
<tr>
<td>JTANonClusteredSemaphore</td>
<td>Provides in memory thread/resource locking that is JTA Transaction aware.</td>
</tr>
<tr>
<td>MSSQLDelegate</td>
<td>This is a driver delegate for the MSSQL JDBC driver.</td>
</tr>
<tr>
<td>PointbaseDelegate</td>
<td>This is a driver delegate for the Pointbase JDBC driver.</td>
</tr>
<tr>
<td>PostgreSQLDelegate</td>
<td>This is a driver delegate for the PostgreSQL JDBC driver.</td>
</tr>
<tr>
<td>SchedulerStateRecord</td>
<td>Conveys a scheduler-instance state record.</td>
</tr>
<tr>
<td>SimplePropertiesTriggerPersistenceDelegateSupport</td>
<td>A base implementation that persists trigger fields in the &quot;QRTZ_SIMPROP_TRIGGERS&quot; table.</td>
</tr>
<tr>
<td>SimplePropertiesTriggerProperties</td>
<td>Internal in-memory lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td>SimpleSemaphore</td>
<td></td>
</tr>
<tr>
<td>SimpleTriggerPersistenceDelegate</td>
<td>This is meant to be an abstract base class for most, if not all, DriverDelegate implementations.</td>
</tr>
<tr>
<td>StdJDBCDelegate</td>
<td>Internal database based lock handler for providing thread/resource locking to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td>StdRowLockSemaphore</td>
<td></td>
</tr>
</tbody>
</table>
SybaseDelegate

This is a driver delegate for the Sybase database.

TriggerPersistenceDelegate.TriggerPropertyBundle

Object representing a job or trigger key.

UpdateLockRowSemaphore

Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.

Util

This class contains utility functions for use in all delegate classes.

WebLogicDelegate

This is a driver delegate for the WebLogic JDBC driver.

---

Exception Summary

**InvalidConfigurationException**

Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.

**LockException**

Exception class for when there is a failure obtaining or releasing a resource lock.

**NoSuchDelegateException**

Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.
Hierarchy For Package org.quartz.impl.jdbcjobstore

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.jdbcjobstore.**AttributeRestoringConnectionInvocationHandler** (implements java.lang.reflect.**InvocationHandler**)
  - org.quartz.impl.jdbcjobstore.**CronTriggerPersistenceDelegate** (implements org.quartz.impl.jdbcjobstore.**StdJDBCConstants**, org.quartz.impl.jdbcjobstore.**TriggerPersistenceDelegate**)
  - org.quartz.impl.jdbcjobstore.**DBSemaphore** (implements org.quartz.impl.jdbcjobstore.**Constants**, org.quartz.impl.jdbcjobstore.**Semaphore**, org.quartz.impl.jdbcjobstore.**StdJDBCConstants**, org.quartz.impl.jdbcjobstore.**TablePrefixAware**)
    - org.quartz.impl.jdbcjobstore.**StdRowLockSemaphore**
    - org.quartz.impl.jdbcjobstore.**UpdateLockRowSemaphore**
  - org.quartz.impl.jdbcjobstore.**FiredTriggerRecord** (implements java.io.**Serializable**)
  - org.quartz.impl.jdbcjobstore.**JobStoreSupport** (implements org.quartz.impl.jdbcjobstore.**Constants**, org.quartz.spi.**JobStore**)
    - org.quartz.impl.jdbcjobstore.**JobStoreCMT**
    - org.quartz.impl.jdbcjobstore.**JobStoreTX**
  - org.quartz.impl.jdbcjobstore.**JobStoreSupport.RecoverMisfiredJobsI**
  - org.quartz.impl.jdbcjobstore.**JTANonClusteredSemaphore** (implements org.quartz.impl.jdbcjobstore.**Semaphore**)
  - org.quartz.impl.jdbcjobstore.**SchedulerStateRecord** (implements java.io.**Serializable**)
  - org.quartz.impl.jdbcjobstore.**SimplePropertiesTriggerPersistenceDelegate** (implements org.quartz.impl.jdbcjobstore.**StdJDBCConstants**, org.quartz.impl.jdbcjobstore.**TriggerPersistenceDelegate**)
    - org.quartz.impl.jdbcjobstore.**CalendarIntervalTriggerPersistenc**
  - org.quartz.impl.jdbcjobstore.**SimplePropertiesTriggerProperties**
  - org.quartz.impl.jdbcjobstore.**SimpleSemaphore** (implements org.quartz.impl.jdbcjobstore.**Semaphore**)
  - org.quartz.impl.jdbcjobstore.**SimpleTriggerPersistenceDelegate** (implements org.quartz.impl.jdbcjobstore.**StdJDBCConstants**, org.quartz.impl.jdbcjobstore.**TriggerPersistenceDelegate**)
  - org.quartz.impl.jdbcjobstore.**StdJDBCDelegate** (implements
org.quartz.impl.jdbcjobstore.DriverDelegate,
org.quartz.impl.jdbcjobstore.StdJDBCConstants)
  ○ org.quartz.impl.jdbcjobstore.CloudscapeDelegate
  ○ org.quartz.impl.jdbcjobstore.DB2v6Delegate
  ○ org.quartz.impl.jdbcjobstore.DB2v7Delegate
  ○ org.quartz.impl.jdbcjobstore.DB2v8Delegate
  ○ org.quartz.impl.jdbcjobstore.HSQLDBDelegate
  ○ org.quartz.impl.jdbcjobstore.MSSQLDelegate
  ○ org.quartz.impl.jdbcjobstore.PointbaseDelegate
  ○ org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
  ○ org.quartz.impl.jdbcjobstore.SybaseDelegate
  ○ org.quartz.impl.jdbcjobstore.WebLogicDelegate
  ○ java.lang.Throwable (implements java.io.Serializable)
    ○ java.lang.Exception
      ○ org.quartz.impl.jdbcjobstore.InvalidConfigurationException
      ○ org.quartz.SchedulerException
        ○ org.quartz.JobPersistenceException
          ○ org.quartz.impl.jdbcjobstore.LockException
          ○ org.quartz.impl.jdbcjobstore.NoSuchDelegateException
          ○ org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPro
          ○ org.quartz.impl.jdbcjobstore.TriggerStatus
          ○ org.quartz.impl.jdbcjobstore.Util
Interface Hierarchy

- org.quartz.impl.jdbcjobstore. Constants
  - org.quartz.impl.jdbcjobstore. StdJDBCConstants
- org.quartz.impl.jdbcjobstore. DriverDelegate
- org.quartz.impl.jdbcjobstore. JobStoreSupport.TransactionCallback
- org.quartz.impl.jdbcjobstore. Semaphore
- org.quartz.impl.jdbcjobstore. TablePrefixAware
- org.quartz.impl.jdbcjobstore. TriggerPersistenceDelegate

Copyright 2001-2011, Terracotta, Inc.
# Uses of Package

**org.quartz.impl.jdbcjobstore**

## Packages that use `org.quartz.impl.jdbcjobstore`

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.impl.jdbcjobstore</code></td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore.oracle</code></td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore.oracle.weblogic</code></td>
</tr>
</tbody>
</table>

## Classes in `org.quartz.impl.jdbcjobstore` used by `org.quartz.impl.jdbcjobstore`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Constants</strong></td>
<td>This interface can be implemented by any DriverDelegate class that needs to use the constants contained therein.</td>
</tr>
<tr>
<td><strong>DBSemaphore</strong></td>
<td>Base class for database based lock handlers for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td><strong>DriverDelegate</strong></td>
<td>This is the base interface for all driver delegate classes.</td>
</tr>
<tr>
<td><strong>FiredTriggerRecord</strong></td>
<td>Conveys the state of a fired-trigger record.</td>
</tr>
<tr>
<td><strong>InvalidConfigurationException</strong></td>
<td>Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.</td>
</tr>
<tr>
<td><strong>JobStoreSupport</strong></td>
<td>Contains base functionality for JDBC-based JobStore implementations.</td>
</tr>
<tr>
<td><strong>JobStoreSupport.RecoverMisfiredJobsResult</strong></td>
<td>Helper class for returning the composite result of trying to recover misfired jobs.</td>
</tr>
<tr>
<td><strong>JobStoreSupport.TransactionCallback</strong></td>
<td>Implement this interface to provide the code to execute within the a</td>
</tr>
<tr>
<td>Transaction template.</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td><strong>JobStoreSupport.VoidTransactionCallback</strong></td>
<td></td>
</tr>
<tr>
<td>Implement this interface to provide the code to execute within the a transaction template that has no return value.</td>
<td></td>
</tr>
<tr>
<td><strong>LockException</strong></td>
<td></td>
</tr>
<tr>
<td>Exception class for when there is a failure obtaining or releasing a resource lock.</td>
<td></td>
</tr>
<tr>
<td><strong>NoSuchDelegateException</strong></td>
<td></td>
</tr>
<tr>
<td>Exception class for when a driver delegate cannot be found for a given configuration, or lack thereof.</td>
<td></td>
</tr>
<tr>
<td><strong>SchedulerStateRecord</strong></td>
<td></td>
</tr>
<tr>
<td>Conveys a scheduler-instance state record.</td>
<td></td>
</tr>
<tr>
<td><strong>Semaphore</strong></td>
<td></td>
</tr>
<tr>
<td>An interface for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
<td></td>
</tr>
<tr>
<td><strong>SimplePropertiesTriggerPersistenceDelegateSupport</strong></td>
<td></td>
</tr>
<tr>
<td>A base implementation of TriggerPersistenceDelegate that persists trigger fields in the &quot;QRTZ_SIMPROP_TRIGGERS&quot; table.</td>
<td></td>
</tr>
<tr>
<td><strong>SimplePropertiesTriggerProperties</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>StdJDBCConstants</strong></td>
<td></td>
</tr>
<tr>
<td>This interface extends Constants to include the query string constants in use by the StdJDBCDelegate class.</td>
<td></td>
</tr>
<tr>
<td><strong>StdJDBCDelegate</strong></td>
<td></td>
</tr>
<tr>
<td>This is meant to be an abstract base class for most, if not all, DriverDelegate implementations.</td>
<td></td>
</tr>
<tr>
<td><strong>TablePrefixAware</strong></td>
<td></td>
</tr>
<tr>
<td>Interface for Quartz objects that need to know what the table prefix of the tables used by a JDBC JobStore is.</td>
<td></td>
</tr>
<tr>
<td><strong>TriggerPersistenceDelegate</strong></td>
<td></td>
</tr>
<tr>
<td>An interface which provides an implementation for storing a particular type of Trigger's extended properties.</td>
<td></td>
</tr>
<tr>
<td><strong>TriggerPersistenceDelegate.TriggerPropertyBundle</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TriggerStatus</strong></td>
<td></td>
</tr>
</tbody>
</table>
Object representing a job or trigger key.

| Classes in `org.quartz.impl.jdbcjobstore` used by `org.quartz.impl.jdbcjobstore.oracle` |
|---------------------------------|-----------------------------------------------------------------------------------|
| **Constants**                   | This interface can be implemented by any DriverDelegate class that needs to use the constants contained herein. |
| **DriverDelegate**              | This is the base interface for all driver delegate classes.                      |
| **StdJDBCConstants**            | This interface extends Constants to include the query string constants in use by the StdJDBCDelegate class. |
| **StdJDBCDelegate**             | This is meant to be an abstract base class for most, if not all, DriverDelegate implementations. |

| Classes in `org.quartz.impl.jdbcjobstore` used by `org.quartz.impl.jdbcjobstore.oracle.weblogic` |
|---------------------------------|-----------------------------------------------------------------------------------|
| **Constants**                   | This interface can be implemented by any DriverDelegate class that needs to use the constants contained herein. |
| **DriverDelegate**              | This is the base interface for all driver delegate classes.                      |
| **StdJDBCConstants**            | This interface extends Constants to include the query string constants in use by the StdJDBCDelegate class. |
| **StdJDBCDelegate**             | This is meant to be an abstract base class for most, if not all, DriverDelegate implementations. |
Copyright 2001-2011, Terracotta, Inc.
Class PointbaseDelegate

java.lang.Object
  └ org.quartz.impl.jdbcjobstore.StdJDBCDelegate
    └ org.quartz.impl.jdbcjobstore.PointbaseDelegate

All Implemented Interfaces:
  Constants, DriverDelegate, StdJDBCConstants

public class PointbaseDelegate

extends StdJDBCDelegate

This is a driver delegate for the Pointbase JDBC driver.

Author:
  Gregg Freeman

Field Summary

Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate:
classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants:
COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP, DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE, DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER, INSERT.Calendar, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR,
SELECT_CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FOR_CALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES
### Constructor Summary

**PointbaseDelegate**(org.slf4j.Logger logger, **String** tablePrefix, **String** schedName, **String** instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new PointbaseJDBCDelegate instance.

**PointbaseDelegate**(org.slf4j.Logger logger, **String** tablePrefix, **String** schedName, **String** instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, **Boolean** useProperties)

Create new PointbaseJDBCDelegate instance.

### Method Summary

**protected Object** getJobDataFromBlob**(ResultSet rs, **String** colName)**

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.

**protected Object** getObjectFromBlob**(ResultSet rs, **String** colName)**

This method should be overridden by any delegate subclasses that need special handling for BLOBs.

**int** insertCalendar**(Connection conn, **String** calendarName, Calendar calendar)**

Insert a new calendar.

**int** insertJobDetail**(Connection conn, JobDetail job)**

Insert the job detail record.

**int** insertTrigger**(Connection conn,
org.quartz.spi.OperableTrigger trigger, String state,
JobDetail jobDetail)
   Insert the base trigger data.

int updateCalendar(Connection conn, String calendarName,
Calendar calendar)
   Update a calendar.

int updateJobData(Connection conn, JobDetail job)
   Update the job data map for the given job.

int updateJobDetail(Connection conn, JobDetail job)
   Update the job detail record.

int updateTrigger(Connection conn,
org.quartz.spi.OperableTrigger trigger, String state,
JobDetail jobDetail)
   Update the base trigger data.

Methods inherited from class
org.quartz.impl.jdbcjobstore.StdJDBCDelegate
addDefaultTriggerPersistenceDelegates,
addTriggerPersistenceDelegate, calendarExists,
calendarIsReferenced, canUseProperties, clearData, closeResultSet,
closeStatement, convertFromProperty, convertToProperty,
countMisfiredTriggersInState, deleteAllPausedTriggerGroups,
deleteBlobTrigger, deleteCalendar, deleteFiredTrigger,
deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail,
deletePausedTriggerGroup, deletePausedTriggerGroup,
deleteSchedulerState, deleteTrigger, deleteTriggerExtension,
findTriggerPersistenceDelegate, findTriggerPersistenceDelegate,
getBoolean, getBoolean, getFirstChildOfNonSerializableValue,
getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize,
insertBlobTrigger, insertFiredTrigger, insertPausedTriggerGroup,
insertSchedulerState, isExistingTriggerGroup, isJobNonConcurrent,
isTriggerGroupPaused, jobExists, rtp, selectCalendar,
selectCalendars, selectFiredTriggerInstanceNames,
selectFiredTriggerRecords, selectFiredTriggerRecordsByJob,
selectInstancesFiredTriggerRecords, selectJobDetail,
selectJobExecutionCount, selectJobForTrigger, selectJobGroups,
selectJobsInGroup, selectMisfiredTriggers,
selectMisfiredTriggersInGroupInState,
selectMisfiredTriggersInState, selectNextFireTime,
selectNumCalendars, selectNumJobs, selectNumTriggers,
selectNumTriggersForJob, selectPausedTriggerGroups,
selectSchedulerStateRecords, selectTrigger,
selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateFiredTrigger, updateSchedulerState, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PointbaseDelegate

public PointbaseDelegate(org.slf4j.Logger logger, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new PointbaseJDBCDelegate instance.

Parameters:
logger - the logger to use during execution
tablePrefix - the prefix of all table names

PointbaseDelegate

public PointbaseDelegate (org.slf4j.Logger logger, String tablePrefix,
Create new PointbaseJDBCDelegate instance.

**Parameters:**
- logger - the logger to use during execution
- tablePrefix - the prefix of all table names

### Method Detail

**insertJobDetail**

```java
class InsertJobDetail {
    public int insertJobDetail(Connection conn, JobDetail job) throws IOException, SQLException {
        // Insert the job detail record.
        // Specified by: insertJobDetail in interface DriverDelegate
        // Overrides: insertJobDetail in class StdJDBCDelegate
        // Parameters:
        //   conn - the DB Connection
        //   job - the job to insert
        // Returns:
        //   number of rows inserted
        // Throws:
        //   IOException - if there were problems serializing the JobDataMap
        //   SQLException
        //}
    }
}
```

**updateJobDetail**

```java
class UpdateJobDetail {
    public int updateJobDetail(Connection conn, JobDetail job) {
        // Update the job detail record.
        // Specified by: updateJobDetail in interface DriverDelegate
        // Overrides: updateJobDetail in class StdJDBCDelegate
        // Parameters:
        //   conn - the DB Connection
        //   job - the job to update
        // Returns:
        //   number of rows updated
        // Throws:
        //   IOException - if there were problems serializing the JobDataMap
        //   SQLException
        //}
    }
}
```
Update the job detail record.

Specified by:
updateJobDetail in interface DriverDelegate

Overrides:
updateJobDetail in class StdJDBCDelegate

Parameters:
conn - the DB Connection
job - the job to update

Returns:
number of rows updated

Throws:
IOException - if there were problems serializing the JobDataMap
SQLException

insertTrigger

public int insertTrigger(Connection conn,
                        org.quartz.spi.OperableTrigger trigger,
                        String state,
                        JobDetail jobDetail)

throws SQLException,
IOException

Description copied from class: StdJDBCDelegate

Insert the base trigger data.

Specified by:
insertTrigger in interface DriverDelegate

Overrides:
insertTrigger in class StdJDBCDelegate

Parameters:
conn - the DB Connection
trigger - the trigger to insert
state - the state that the trigger should be stored in

Returns:
the number of rows inserted

Throws:
  SQLException
  IOException

updateTrigger

public int updateTrigger(Connection conn,
    org.quartz.spi.OperableTrigger trigger,
    String state,
    JobDetail jobDetail)
throws SQLException,
   IOException

Description copied from class: StdJDBCDelegate

Update the base trigger data.

Specified by:
  updateTrigger in interface DriverDelegate

Overrides:
  updateTrigger in class StdJDBCDelegate

Parameters:
  conn - the DB Connection
  trigger - the trigger to insert
  state - the state that the trigger should be stored in

Returns:
  the number of rows updated

Throws:
  SQLException
  IOException

updateJobData

public int updateJobData(Connection conn,
    JobDetail job)
throws IOException,
   SQLException
Update the job data map for the given job.

Specified by:
   updateJobData in interface DriverDelegate

Overrides:
   updateJobData in class StdJDBCDelegate

Parameters:
   conn - the DB Connection
   job - the job to update

Returns:
   the number of rows updated

Throws:
   IOException - if there were problems serializing the JobDataMap
   SQLException

---

**insertCalendar**

```java
public int insertCalendar(Connection conn,
                           String calendarName,
                           Calendar calendar)
    throws IOException,
            SQLException
```

Insert a new calendar.

Specified by:
   insertCalendar in interface DriverDelegate

Overrides:
   insertCalendar in class StdJDBCDelegate

Parameters:
   conn - the DB Connection
   calendarName - the name for the new calendar
   calendar - the calendar

Returns:
   the number of rows inserted

Throws:
   IOException - if there were problems serializing the calendar
   SQLException
**updateCalendar**

```java
public int updateCalendar(Connection conn,
                          String calendarName,
                          Calendar calendar)
    throws IOException,
            SQLException;
```

Update a calendar.

**Specified by:**
`updateCalendar in interface DriverDelegate`

**Overrides:**
`updateCalendar in class StdJDBCDelegate`

**Parameters:**
- `conn` - the DB Connection
- `calendarName` - the name for the new calendar
- `calendar` - the calendar

**Returns:**
the number of rows updated

**Throws:**
- `IOException` - if there were problems serializing the calendar
- `SQLException`

---

**getObjectFromBlob**

```java
protected Object getObjectFromBlob(ResultSet rs,
                                    String colName)
    throws ClassNotFoundException,
            IOException,
            SQLException;
```

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard JDBC `java.sql.Blob` operations.

**Overrides:**
`getObjectFromBlob in class StdJDBCDelegate`

**Parameters:**
- `rs` - the result set, already queued to the correct row
colName - the column name for the BLOB

**Returns:**
the deserialized Object from the ResultSet BLOB

**Throws:**

- `ClassNotFoundException` - if a class found during deserialization cannot be found
- `IOException` - if deserialization causes an error
- `SQLException`

---

**getJobDataFromBlob**

```java
protected Object getJobDataFromBlob(ResultSet rs, String colName)
throws ClassNotFoundException, IOException, SQLException
```

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC `java.sql.Blob` operations.

**Overrides:**

`getJobDataFromBlob` in class `StdJDBCDelegate`

**Parameters:**

- `rs` - the result set, already queued to the correct row
- `colName` - the column name for the BLOB

**Returns:**

the deserialized Object from the ResultSet BLOB

**Throws:**

- `ClassNotFoundException` - if a class found during deserialization cannot be found
- `IOException` - if deserialization causes an error
- `SQLException`
org.quartz.impl.jdbcjobstore  Class PostgreSQLDelegate

java.lang.Object
   ↓ org.quartz.impl.jdbcjobstore.StdJDBCDelegate
   ↓ org.quartz.impl.jdbcjobstore.PostgreSQLDelegate

All Implemented Interfaces:
   Constants, DriverDelegate, StdJDBCConstants

public class PostgreSQLDelegate
extends StdJDBCDelegate

This is a driver delegate for the PostgreSQL JDBC driver.

Author:
   Jeffrey Wescott

---

Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOB_DETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE, DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER, INSERT_CALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,</td>
</tr>
</tbody>
</table>
Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants

ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB, COLCALENDAR, COLCALENDAR_NAME, COLCHECKIN_INTERVAL, COLCRONEXPRESSION, COLDESCRIPTION, COLENDTIME, COLENTRY_ID, COLENTRYSTATE, COLFIREDTIME, COLINSTANCENAME, COLISDURABLE, COLISNONCONCURRENT, COLISUPDATEDATA, COLISVOLATILE, COLJOCCLASS, COLJODB DATAMAP, COLJOBGROUP,
**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PostgreSQLDelegate</strong>(org.slf4j.Logger log, <strong>String</strong> tablePrefix, <strong>String</strong> schedName, <strong>String</strong> instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)</td>
<td>Create new PostgreSQLDelegate instance.</td>
</tr>
<tr>
<td><strong>PostgreSQLDelegate</strong>(org.slf4j.Logger log, <strong>String</strong> tablePrefix, <strong>String</strong> schedName, <strong>String</strong> instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, <strong>Boolean</strong> useProperties)</td>
<td>Create new PostgreSQLDelegate instance.</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protected Object getJobDataFromBlob</strong>(ResultSet rs, <strong>String</strong> colName)</td>
<td>This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.</td>
</tr>
<tr>
<td><strong>protected Object getObjectFromBlob</strong>(ResultSet rs, <strong>String</strong> colName)</td>
<td>This method should be overridden by any delegate subclasses that need special handling for BLOBs.</td>
</tr>
</tbody>
</table>

**Methods inherited from class**

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore.<strong>StdJDBCDelegate</strong></td>
<td>addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet,</td>
</tr>
</tbody>
</table>
closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</td>
</tr>
</tbody>
</table>
## Constructor Detail

### PostgreSQLDelegate

```java
public PostgreSQLDelegate(org.slf4j.Logger log,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper)
```

Create new PostgreSQLDelegate instance.

**Parameters:**
- `log` - the logger to use during execution
- `tablePrefix` - the prefix of all table names

```java
public PostgreSQLDelegate(org.slf4j.Logger log,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper,
Boolean useProperties)
```

Create new PostgreSQLDelegate instance.

**Parameters:**
- `log` - the logger to use during execution
- `tablePrefix` - the prefix of all table names
- `useProperties` - use java.util.Properties for storage

## Method Detail

### getObjectFromBlob

```java
protected Object getObjectFromBlob(ResultSet rs,
String colName)
throws ClassNotFoundException,
ClassNotFoundException,
```
IOException, SQLException

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard JDBC java.sql.Blob operations.

Overrrides:

getObjectFromBlob in class StdJDBCDelegate

Parameters:

- rs - the result set, already queued to the correct row
- colName - the column name for the BLOB

Returns:

the deserialized Object from the ResultSet BLOB

Throws:

ClassNotFoundException - if a class found during deserialization cannot be found
IOException - if deserialization causes an error
SQLException

getJobDataFromBlob

protected Object getJobDataFromBlob(ResultSet rs, String colName) throws ClassNotFoundException, IOException, SQLException

Description copied from class: StdJDBCDelegate

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC java.sql.Blob operations.

Overrrides:

getJobDataFromBlob in class StdJDBCDelegate

Parameters:

- rs - the result set, already queued to the correct row
- colName - the column name for the BLOB

Returns:
the deserialized Object from the ResultSet BLOB

**Throws:**

- `ClassNotFoundException` - if a class found during deserialization cannot be found
- `IOException` - if deserialization causes an error
- `SQLException`
org.quartz.impl.jdbcjobstore Class SchedulerStateRecord

java.lang.Object
  └ org.quartz.impl.jdbcjobstore.SchedulerStateRecord

All Implemented Interfaces:
    Serializable

public class SchedulerStateRecord
    extends Object
    implements Serializable

Conveys a scheduler-instance state record.

Author:
    James House

See Also:
    Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerStateRecord()</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>long getCheckinInterval()</td>
<td></td>
</tr>
<tr>
<td>long getCheckinTimestamp()</td>
<td></td>
</tr>
<tr>
<td>String getSchedulerInstanceId()</td>
<td></td>
</tr>
<tr>
<td>void setCheckinInterval(long l)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
**Constructor Detail**

**SchedulerStateRecord**

```java
public SchedulerStateRecord()
```

**Method Detail**

**getCheckinInterval**

```
public long getCheckinInterval()
```

**getCheckinTimestamp**

```
public long getCheckinTimestamp()
```

**getSchedulerInstanceId**

```
public String getSchedulerInstanceId()
```

**setCheckinInterval**

```
public void setCheckinInterval(long l)
```
setCheckinTimestamp

public void setCheckinTimestamp(long l)

setSchedulerInstanceId

public void setSchedulerInstanceId(String string)
<table>
<thead>
<tr>
<th>Class</th>
<th>Summary</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PREV CLASS	NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES	NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public interface Semaphore

An interface for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.

Author:
jhouse

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean isLockOwner(Connection conn, String lockName)</td>
</tr>
<tr>
<td>Determines whether the calling thread owns a lock on the identified resource.</td>
</tr>
<tr>
<td>boolean obtainLock(Connection conn, String lockName)</td>
</tr>
<tr>
<td>Grants a lock on the identified resource to the calling thread (blocking until it is available).</td>
</tr>
<tr>
<td>void releaseLock(Connection conn, String lockName)</td>
</tr>
<tr>
<td>Release the lock on the identified resource if it is held by the calling thread.</td>
</tr>
<tr>
<td>boolean requiresConnection()</td>
</tr>
<tr>
<td>Whether this Semaphore implementation requires a database connection for its lock management operations.</td>
</tr>
</tbody>
</table>

Method Detail

obtainLock
boolean obtainLock (Connection conn, String lockName) throws LockException

Grants a lock on the identified resource to the calling thread (blocking until it is available).

**Parameters:**
- conn - Database connection used to establish lock. Can be null if requiresConnection() returns false.

**Returns:**
- true if the lock was obtained.

**Throws:**
- LockException

---

releaseLock

void releaseLock (Connection conn, String lockName) throws LockException

Release the lock on the identified resource if it is held by the calling thread.

**Parameters:**
- conn - Database connection used to establish lock. Can be null if requiresConnection() returns false.

**Throws:**
- LockException

---

isLockOwner

boolean isLockOwner (Connection conn, String lockName) throws LockException

Determine whether the calling thread owns a lock on the identified resource.

**Parameters:**
conn - Database connection used to establish lock. Can be null if
requiresConnection() returns false.

Throws:
LockException

requiresConnection

boolean requiresConnection()

Whether this Semaphore implementation requires a database connection for
its lock management operations.

See Also:
isLockOwner(Connection, String), obtainLock(Connection, String),
releaseLock(Connection, String)
org.quartz.impl.jdbcjobstore Class
SimplePropertiesTriggerPersistenceDelegateSupport

java.lang.Object
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

All Implemented Interfaces:
Constants, StdJDBCConstants, TriggerPersistenceDelegate

Direct Known Subclasses:
CalendarIntervalTriggerPersistenceDelegate

public abstract class SimplePropertiesTriggerPersistenceDelegateSupport
extends Object
implements TriggerPersistenceDelegate, StdJDBCConstants

A base implementation of TriggerPersistenceDelegate that persists trigger fields in the "QRTZ_SIMPROP_TRIGGERS" table. This allows extending concrete classes to simply implement a couple methods that do the work of getting/setting the trigger's fields, and creating the ScheduleBuilder for the particular type of trigger.

Author:
jhouse

See Also:
for an example extension

---

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Field Summary</td>
</tr>
<tr>
<td>--------------</td>
</tr>
</tbody>
</table>
| **protected**  
| **static**  
| **String**  
| **COL_BOOL_PROP_1** |
| **protected**  
| **static**  
| **String**  
| **COL_BOOL_PROP_2** |
| **protected**  
| **static**  
| **String**  
| **COL_DEC_PROP_1** |
| **protected**  
| **static**  
| **String**  
| **COL_DEC_PROP_2** |
| **protected**  
| **static**  
| **String**  
| **COL_INT_PROP_1** |
| **protected**  
| **static**  
| **String**  
| **COL_INT_PROP_2** |
| **protected**  
| **static**  
| **String**  
| **COL_LONG_PROP_1** |
| **protected**  
| **static**  
| **String**  
| **COL_LONG_PROP_2** |
| **protected**  
| **static**  
| **String**  
| **COL_STR_PROP_1** |
| **protected**  
| **static**  
| **String**  
| **COL_STR_PROP_2** |
| **protected**  
| **static**  
| **String**  
| **COL_STR_PROP_3** |
| **protected**  
| **static**  
| **String**  
| **DELETE_SIMPLE_PROPS_TRIGGER** |
| **protected**  
| **static**  
| **String**  
| **INSERT_SIMPLE_PROPS_TRIGGER** |
| **protected**  
| **String**  
| **schedNameLiteral** |
| **protected**  
| **static**  
| **String**  
| **SELECT_SIMPLE_PROPS_TRIGGER** |
| **protected**  
| **static**  
| **String**  
| **TABLE_SIMPLE_PROPERTIES_TRIGGERS** |
| **protected**  
| **static**  
| **String**  
| **tablePrefix** |
protected static String UPDATE_SIMPLE_PROPS_TRIGGER

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS,
DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS,
DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS,
DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS,
DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETE CALENDAR,
DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS,
DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL,
DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERT_CALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR,
SELECT_CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOBGROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOBGROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERSFOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECTREFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGEREXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FORCALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATECALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants
ALIAS COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL,
COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID,
COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME,
COL_IS_DURABLE, COL_IS_NONCONCURRENT, COL_IS_UPDATE_DATA,
COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP,
COL_JOB_NAME, COL_LAST_CHECKIN_TIME, COL_LOCK_NAME,
COL_MISFIRE_INSTRUCTION, COL_NEXT_FIRE_TIME, COL_PREV_FIRE_TIME,
COL_PRIORITY, COL_REPEAT_COUNT, COL_REPEAT_INTERVAL,
COL_REQUESTS_RECOVERY, COL_SCHEDULER_NAME, COL_START_TIME,
COL_TIME_ZONE_ID, COL_TIMES_TRIGGERED, COL_TRIGGER_GROUP,
COL_TRIGGER_NAME, COL_TRIGGER_STATE, COL_TRIGGER_TYPE,
DEFAULT_TABLE_PREFIX, STATE_ACQUIRED, STATE_BLOCKED,
STATE_COMPLETE, STATE_DELETED, STATE_ERROR, STATE_EXECUTING,
STATE_MISFIRED, STATE_PAUSED, STATE_PAUSED_BLOCKED, STATE_WAITING,
TABLE_BLOB_TRIGGERS, TABLE_CALENDARS, TABLE_CRON_TRIGGERS,
TABLE_FIRED_TRIGGERS, TABLE_JOBDETAILS, TABLE_LOCKS,
TABLE_PASSED_TRIGGERS, TABLE_SCHEDULER_STATE,
TABLE_SIMPLE_TRIGGERS, TABLE_TRIGGERS, TTYPE_BLOB, TTYPE_CAL_INT,
TTYPE_CRON, TTYPE_SIMPLE

Constructor Summary

SimplePropertiesTriggerPersistenceDelegateSupport()
protected abstract SimplePropertiesTriggerProperties getTriggerProperties(org.quartz.spi.OperableTrigger trigger)

protected abstract TriggerPersistenceDelegate.TriggerPropertyBundle getTriggerPropertyBundle()

void initialize(String tablePrefix, int insertExtendedTriggerProperties org.quartz.spi.OperableTrigger JobDetail jobDetail)

int insertExtendedTriggerProperties org.quartz.spi.OperableTrigger JobDetail jobDetail)

int updateExtendedTriggerProperties org.quartz.spi.OperableTrigger JobDetail jobDetail)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate
canHandleTriggerType, getHandledTriggerTypeDiscriminator

Field Detail

TABLE_SIMPLE_PROPERTIES_TRIGGERS

protected static final String TABLE_SIMPLE_PROPERTIES_TRIGGERS

See Also:
Constant Field Values
COL_STR_PROP_1
protected static final String COL_STR_PROP_1

See Also:
Constant Field Values

COL_STR_PROP_2
protected static final String COL_STR_PROP_2

See Also:
Constant Field Values

COL_STR_PROP_3
protected static final String COL_STR_PROP_3

See Also:
Constant Field Values

COL_INT_PROP_1
protected static final String COL_INT_PROP_1

See Also:
Constant Field Values

COL_INT_PROP_2
protected static final String COL_INT_PROP_2

See Also:
Constant Field Values
COL_LONG_PROP_1
protected static final String COL_LONG_PROP_1

See Also:
Constant Field Values

COL_LONG_PROP_2
protected static final String COL_LONG_PROP_2

See Also:
Constant Field Values

COL_DEC_PROP_1
protected static final String COL_DEC_PROP_1

See Also:
Constant Field Values

COL_DEC_PROP_2
protected static final String COL_DEC_PROP_2

See Also:
Constant Field Values

COL_BOOL_PROP_1
protected static final String COL_BOOL_PROP_1

See Also:
Constant Field Values
COL_BOOL_PROP_2

protected static final String COL_BOOL_PROP_2

See Also:
Constant Field Values

-----------------------------------

SELECT_SIMPLE_PROPS_TRIGGER

protected static final String SELECT_SIMPLE_PROPS_TRIGGER

See Also:
Constant Field Values

-----------------------------------

DELETE_SIMPLE_PROPS_TRIGGER

protected static final String DELETE_SIMPLE_PROPS_TRIGGER

See Also:
Constant Field Values

-----------------------------------

INSERT_SIMPLE_PROPS_TRIGGER

protected static final String INSERT_SIMPLE_PROPS_TRIGGER

See Also:
Constant Field Values

-----------------------------------

UPDATE_SIMPLE_PROPS_TRIGGER

protected static final String UPDATE_SIMPLE_PROPS_TRIGGER

See Also:
Constant Field Values
tablePrefix

protected String tablePrefix

schedNameLiteral

protected String schedNameLiteral

### Constructor Detail

SimplePropertiesTriggerPersistenceDelegateSupport

public SimplePropertiesTriggerPersistenceDelegateSupport()

### Method Detail

initialize

public void initialize(String tablePrefix, String schedName)

Specified by:
initialize in interface TriggerPersistenceDelegate

getTriggerProperties

protected abstract SimplePropertiesTriggerProperties getTriggerPrope

getTriggerPropertyBundle

protected abstract TriggerPersistenceDelegate.TriggerPropertyBundle getTriggerPrope

deleteExtendedTriggerProperties
public int **deleteExtendedTriggerProperties**(*Connection* conn, *TriggerKey* triggerKey)

**Specified by:**
*deleteExtendedTriggerProperties* in interface *TriggerPersistenceDelegate*

**Throws:**
*SQLException*

---

**insertExtendedTriggerProperties**


**Specified by:**
*insertExtendedTriggerProperties* in interface *TriggerPersistenceDelegate*

**Throws:**
*SQLException*, *IOException*

---

**loadExtendedTriggerProperties**

public **TriggerPersistenceDelegate.TriggerPropertyBundle** **loadExtended**

**Specified by:**
*loadExtendedTriggerProperties* in interface *TriggerPersistenceDelegate*

**Throws:**
*SQLException*

---

**updateExtendedTriggerProperties**
public int updateExtendedTriggerProperties(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
throws SQLException, IOException

Specified by:
updateExtendedTriggerProperties in interface TriggerPersistenceDelegate

Throws:
SQLException
IOException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
org.quartz.impl.jdbcjobstore Class
SimplePropertiesTriggerProperties

java.lang.Object
   ↘ org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

public class SimplePropertiesTriggerProperties
extends Object

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimplePropertiesTriggerProperties()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>BigDecimal getDecimal1()</td>
</tr>
<tr>
<td>BigDecimal getDecimal2()</td>
</tr>
<tr>
<td>int getInt1()</td>
</tr>
<tr>
<td>int getInt2()</td>
</tr>
<tr>
<td>long getLong1()</td>
</tr>
<tr>
<td>long getLong2()</td>
</tr>
<tr>
<td>String getString1()</td>
</tr>
<tr>
<td>String getString2()</td>
</tr>
<tr>
<td>String getString3()</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>boolean isBoolean1()</td>
</tr>
<tr>
<td>boolean isBoolean2()</td>
</tr>
<tr>
<td>void setBoolean1(boolean boolean1)</td>
</tr>
<tr>
<td>void setBoolean2(boolean boolean2)</td>
</tr>
<tr>
<td>void setDecimal1(BigDecimal decimal1)</td>
</tr>
<tr>
<td>void setDecimal2(BigDecimal decimal2)</td>
</tr>
<tr>
<td>void setInt1(int int1)</td>
</tr>
<tr>
<td>void setInt2(int int2)</td>
</tr>
<tr>
<td>void setLong1(long long1)</td>
</tr>
<tr>
<td>void setLong2(long long2)</td>
</tr>
<tr>
<td>void setString1(String string1)</td>
</tr>
<tr>
<td>void setString2(String string2)</td>
</tr>
<tr>
<td>void setString3(String string3)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail
public SimplePropertiesTriggerProperties()

Method Detail

g(getString1)
public String getString1()

setStatus1
public void setStatus1(String string1)

g(getString2)
public String getString2()

setStatus2
public void setStatus2(String string2)

g(getString3)
public String getString3()

setStatus3
public void setStatus3(String string3)

g(int1)
public int getInt1()

setInt1

public void setInt1(int int1)

getInt2

public int getInt2()

setInt2

public void setInt2(int int2)

getLong1

public long getLong1()

setLong1

public void setLong1(long long1)

getLong2

public long getLong2()

setLong2

public void setLong2(long long2)

getDecimal1
public BigDecimal getDecimal1()

setDecimal1

public void setDecimal1(BigDecimal decimal1)

getDecimal2

public BigDecimal getDecimal2()

setDecimal2

public void setDecimal2(BigDecimal decimal2)

isBoolean1

public boolean isBoolean1()

setBoolean1

public void setBoolean1(boolean boolean1)

isBoolean2

public boolean isBoolean2()

setBoolean2

public void setBoolean2(boolean boolean2)
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
org.quartz.impl.jdbcjobstore Class SimpleSemaphore

java.lang.Object
   org.quartz.impl.jdbcjobstore.SimpleSemaphore

All Implemented Interfaces:
   Semaphore

public class SimpleSemaphore
   extends Object
   implements Semaphore

Internal in-memory lock handler for providing thread/resource locking in order
to protect resources from being altered by multiple threads at the same time.

Author:
   jhouse

Constructor Summary

| SimpleSemaphore() |

Method Summary

| protected org.slf4j.Logger getLog() |
| boolean isLockOwner(Connection conn, String lockName) |
|   Determine whether the calling thread owns a lock on the identified resource. |
| boolean obtainLock(Connection conn, String lockName) |
|   Grants a lock on the identified resource to the calling thread (blocking until it is available). |
| void releaseLock(Connection conn, String lockName) |
|   Release the lock on the identified resource if it is held by |

| boolean |
| boolean |
| void |
boolean \texttt{requiresConnection}() \\
This Semaphore implementation does not use the database.

### Methods inherited from class java.lang.\texttt{Object}
\texttt{clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait}

### Constructor Detail

**SimpleSemaphore**

public \texttt{SimpleSemaphore}()

### Method Detail

**getLog**

protected org.slf4j.Logger \texttt{getLog}()

**obtainLock**

public boolean \texttt{obtainLock} (\texttt{Connection} conn, \texttt{String} lockName)

Grants a lock on the identified resource to the calling thread (blocking until it is available).

**Specified by:**
\texttt{obtainLock} in interface \texttt{Semaphore}

**Parameters:**
- conn - Database connection used to establish lock. Can be null if Semaphore.\texttt{requiresConnection}() returns false.

**Returns:**
true if the lock was obtained.

---

**releaseLock**

public void releaseLock(**Connection** conn, **String** lockName)

Release the lock on the identified resource if it is held by the calling thread.

**Specified by:**

releaseLock in interface Semaphore

**Parameters:**

conn - Database connection used to establish lock. Can be null if Semaphore.requiresConnection() returns false.

---

**isLockOwner**

public boolean isLockOwner(**Connection** conn, **String** lockName)

Determine whether the calling thread owns a lock on the identified resource.

**Specified by:**

isLockOwner in interface Semaphore

**Parameters:**

conn - Database connection used to establish lock. Can be null if Semaphore.requiresConnection() returns false.

---

**requiresConnection**

public boolean requiresConnection()

This Semaphore implementation does not use the database.

**Specified by:**

requiresConnection in interface Semaphore
See Also:

Semaphore.isLockOwner(Connection, String),
Semaphore.obtainLock(Connection, String),
Semaphore.releaseLock(Connection, String)
org.quartz.impl.jdbcjobstore Class
SimpleTriggerPersistenceDelegate

java.lang.Object
  \org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

All Implemented Interfaces:
  Constants, StdJDBCConstants, TriggerPersistenceDelegate

public class SimpleTriggerPersistenceDelegate
  extends Object
  implements TriggerPersistenceDelegate, StdJDBCConstants

Nested Class Summary

Nested classes/interfaces inherited from interface
org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate
TriggerPersistenceDelegate.TriggerPropertyBundle

Field Summary

protected String schedNameLiteral

protected String tablePrefix

Fields inherited from interface
org.quartz.impl.jdbcjobstore.StdJDBCConstants
COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS,
DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS,
DELETE_ALL_JOB_DETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS,
DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS,
DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETECALENDAR,
DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS,
DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL,
DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERT.Calendar, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECTCALENDAR,
SELECT_CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE,
SELECT_NEXT_FIRE_TIME_TO_ACQUIRE, SELECT_NUM_CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGERS, SELECT_REFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FORCALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATECALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants

ALIAS COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
| COL CALENDAR, COL CALENDAR_NAME, COL CHECKIN INTERVAL, COL CRON_EXPRESSION, COL DESCRIPTION, COL END TIME, COL ENTRY_ID, COL ENTRY_STATE, COL FIRED TIME, COL INSTANCE_NAME, COL IS DURABLE, COL IS NONCONCURRENT, COL IS UPDATE DATA, COL IS VOLATILE, COL JOB CLASS, COL JOB DATAMAP, COL JOB_GROUP, COL JOB_NAME, COL LAST CHECKIN TIME, COL LOCK NAME, COL MISFIRE_INSTRUCTION, COL NEXT FIRE TIME, COL PREV FIRE TIME, COL PRIORITY, COL REPEAT_COUNT, COL REPEAT_INTERVAL, COL REQUESTS_RECOVERY, COL_SCHEDULER_NAME, COL START_TIME, COL TIME_ZONE_ID, COL TIMES_TRIGGERED, COL_TRIGGER_GROUP, COL_TRIGGER_NAME, COL_TRIGGER_STATE, COL_TRIGGER_TYPE, DEFAULT_TABLE_PREFIX, STATE_ACQUIRED, STATE_BLOCKED, STATE_COMPLETE, STATE_DELETED, STATE_ERROR, STATE EXECUTING, STATE MISFIRED, STATE_PAUSED, STATE_PAUSED_BLOCKED, STATE_WAITING, TABLE_BLOB_TRIGGERS, TABLE_CALENDARS, TABLE_CRON_TRIGGERS, TABLE_FIRED_TRIGGERS, TABLE_JOB_DETAILS, TABLE_LOCKS, TABLE_PAUSED_TRIGGERS, TABLE_SCHEDULER_STATE, TABLE_SIMPLE_TRIGGERS, TABLE_TRIGGERS, TTYPE_BLOB, TTYPE_CAL_INT, TTYPE_CRON, TTYPE_SIMPLE |

## Constructor Summary

| **SimpleTriggerPersistenceDelegate**() |

## Method Summary

| boolean canHandleTriggerType(org.quartz.spi.OperableTrigger trigger) |
| int deleteExtendedTriggerProperties(TriggerKey triggerKey) |
| String getHandledTriggerTypeDiscriminator() |
| void initialize(String tablePrefix, JobDetail jobDetail) |
| int insertExtendedTriggerProperties(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail) |
| TriggerPersistenceDelegate.TriggerPropertyBundle loadExtendedTriggerProperties() |
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

tablePrefix
protected String tablePrefix

schedNameLiteral
protected String schedNameLiteral

Constructor Detail

SimpleTriggerPersistenceDelegate
public SimpleTriggerPersistenceDelegate()

Method Detail

initialize
public void initialize(String tablePrefix, String schedName)
getHandledTriggerTypeDiscriminator

public String getHandledTriggerTypeDiscriminator()

Specified by:
getHandledTriggerTypeDiscriminator in interface TriggerPersistenceDelegate

canHandleTriggerType

public boolean canHandleTriggerType(org.quartz.spi.OperableTrigger trigger)

Specified by:
canHandleTriggerType in interface TriggerPersistenceDelegate

deleteExtendedTriggerProperties

public int deleteExtendedTriggerProperties(Connection conn, TriggerKey triggerKey)
throws SQLException

Specified by:
deleteExtendedTriggerProperties in interface TriggerPersistenceDelegate

Throws:
SQLException

insertExtendedTriggerProperties

public int insertExtendedTriggerProperties(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
throws SQLException,
IOException

Specified by:
insertExtendedTriggerProperties in interface
TriggerPersistenceDelegate

Throws:
SQLException
IOException

loadExtendedTriggerProperties

public TriggerPersistenceDelegate.TriggerPropertyBundle loadExtended

Specified by:
loadExtendedTriggerProperties in interface
TriggerPersistenceDelegate

Throws:
SQLException

updateExtendedTriggerProperties

public int updateExtendedTriggerProperties(Connection conn,
org.quartz.spi.OperableTrigger
String state,
JobDetail jobDetail)
throws SQLException,
IOException

Specified by:
updateExtendedTriggerProperties in interface
TriggerPersistenceDelegate

Throws:
SQLException
IOException
Interface StdJDBCConstants

All Superinterfaces:
   Constants

All Known Implementing Classes:
   CalendarIntervalTriggerPersistenceDelegate, CloudscapeDelegate, CronTriggerPersistenceDelegate, DB2v6Delegate, DB2v7Delegate, DB2v8Delegate, DBSemaphore, HSQLDBDelegate, MSSQLDelegate, OracleDelegate, PointbaseDelegate, PostgreSQLDelegate, SimplePropertiesTriggerPersistenceDelegateSupport, SimpleTriggerPersistenceDelegate, StdJDBCDelegate, StdRowLockSemaphore, SybaseDelegate, UpdateLockRowSemaphore, WebLogicDelegate, WebLogicOracleDelegate

public interface StdJDBCConstants

extends Constants

This interface extends Constants to include the query string constants in use by the StdJDBCDelegate class.

Author:
   Jeffrey Wescott

Field Summary

<table>
<thead>
<tr>
<th>static String</th>
<th>COUNT_MISFIRED_TRIGGERS_IN_STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>DELETE_ALL_BLOB_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td>DELETE_ALL_CALENDARS</td>
</tr>
<tr>
<td>static String</td>
<td>DELETE_ALL_CRON_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>DELETE_ALL_JOBDETAILS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_ALL_PAUSED_TRIGGER_GRPS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_ALL_SIMPLE_TRIGGERS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_ALL_SIMPROP_TRIGGERS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_ALL_TRIGGERS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_BLOB_TRIGGER</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETECALENDAR</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_CRON_TRIGGER</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_FIRED_TRIGGER</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_FIRED_TRIGGERS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_INSTANCES_FIRED_TRIGGERS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_JOB_DETAIL</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_NO_RECOVERY_FIRED_TRIGGERS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_PAUSED_TRIGGER_GROUP</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_PAUSED_TRIGGER_GROUPS</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_SCHEDULER_STATE</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>DELETE_SIMPLE_TRIGGER</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>static String</td>
<td>DELETE_TRIGGER</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_BLOB_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>INSERTCALENDAR</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_CRON_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_FIRED_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_JOB_DETAIL</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_PAUSED_TRIGGER_GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_SCHEDULER_STATE</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_SIMPLE_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>INSERT_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>SCHED_NAME_SUBST</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_BLOB_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>SELECTCALENDAR</td>
</tr>
<tr>
<td>static String</td>
<td>SELECTCALENDAR_EXISTENCE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECTCALENDARS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_CRON_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_FIRED_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>SCHED_NAME_SUBST</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_FIRED_TRIGGER_GROUP</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_FIRED_TRIGGER_INSTANCE_NAMES</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_FIRED_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_FIRED_TRIGGERS_OF_JOB</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_FIRED_TRIGGERS_OF_JOB_GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_INSTANCES_FIRED_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_JOB_DETAIL</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_JOB_EXECUTION_COUNT</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_JOB_EXISTENCE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_JOB_FOR_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_JOB_GROUPS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_JOB_NONCONCURRENT</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_JOBS_IN_GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_MISFIRED_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_MISFIRED_TRIGGERS_IN_STATE</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NEXT_FIRE_TIME</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NEXT_TRIGGER_TO_ACQUIRE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NUM_CALENDARS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NUM_JOBS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NUM_TRIGGERS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NUM_TRIGGERS_FOR_JOB</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_NUM_TRIGGERS_IN_GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_PAUSED_TRIGGER_GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_PAUSED_TRIGGER_GROUPS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_REFERENCED_CALENDAR</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_SCHEDULER_STATE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_SCHEDULER_STATES</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_SIMPLE_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_TRIGGER_DATA</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT_TRIGGEREXISTENCE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGER FOR FIRE TIME</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGER_GROUPS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGER_GROUPS FILTERED</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGER_STATE</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGER_STATUS</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGERS FOR CALENDAR</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGERS FOR JOB</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGERS IN GROUP</td>
</tr>
<tr>
<td>static String</td>
<td>SELECT TRIGGERS IN STATE</td>
</tr>
<tr>
<td>static String</td>
<td>TABLE_PREFIX_SUBST</td>
</tr>
<tr>
<td>static String</td>
<td>UPDATE BLOB TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>UPDATE CALENDAR</td>
</tr>
<tr>
<td>static String</td>
<td>UPDATE CRON TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>UPDATE FIRED TRIGGER</td>
</tr>
<tr>
<td>static String</td>
<td>UPDATE INSTANCES FIRED TRIGGER STATE</td>
</tr>
<tr>
<td>static String</td>
<td>UPDATE JOB DATA</td>
</tr>
<tr>
<td>static String</td>
<td>UPDATE JOB_DETAIL</td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_JOB_TRIGGER_STATES</strong></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_SCHEDULER_STATE</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_SIMPLE_TRIGGER</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER_GROUP_STATE_FROM_STATE</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER_GROUP_STATE_FROM_STATES</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER_SKIP_DATA</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER_STATE</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER_STATE_FROM_STATE</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER_STATE_FROM_STATES</strong></td>
</tr>
<tr>
<td>static String</td>
<td><strong>UPDATE_TRIGGER_STATES_FROM_OTHER_STATES</strong></td>
</tr>
</tbody>
</table>

**Fields inherited from interface org.quartz.impl.jdbcjobstore.**

<table>
<thead>
<tr>
<th>Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS, COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB, COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL, COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID, COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME, COL_IS_DURABLE, COL_IS_NONCONCURRENT, COL_IS_UPDATE_DATA, COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP, COL_JOB_NAME, COL_LAST_CHECKIN_TIME, COL_LOCK_NAME, COL_MISFIRE_INSTRUCTION, COL_NEXT_FIRE_TIME, COL_PREV_FIRE_TIME, COL_PRIORITY, COL_REPEAT_COUNT, COL_REPEAT_INTERVAL, COL_REQUESTS_RECOVERY, COL_SCHEDULER_NAME, COL_START_TIME,</td>
</tr>
</tbody>
</table>
Field Detail

**TABLE_PREFIX_SUBST**

static final `String` TABLE_PREFIX_SUBST

See Also: [Constant Field Values](#)

---

**SCHED_NAME_SUBST**

static final `String` SCHED_NAME_SUBST

See Also: [Constant Field Values](#)

---

**UPDATE_TRIGGER_STATES_FROM_OTHER_STATES**

static final `String` UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

See Also: [Constant Field Values](#)

---

**SELECT_MISFIRED_TRIGGERS**
static final String SELECT_MISFIRED_TRIGGERS

See Also:
Constant Field Values

SELECT_TRIGGERS_IN_STATE

static final String SELECT_TRIGGERS_IN_STATE

See Also:
Constant Field Values

SELECT_MISFIRED_TRIGGERS_IN_STATE

static final String SELECT_MISFIRED_TRIGGERS_IN_STATE

See Also:
Constant Field Values

COUNT_MISFIRED_TRIGGERS_IN_STATE

static final String COUNT_MISFIRED_TRIGGERS_IN_STATE

See Also:
Constant Field Values

SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE

static final String SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE

See Also:
Constant Field Values

SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE
static final String SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE

See Also:
   Constant Field Values

DELETE_FIRED_TRIGGERS

static final String DELETE_FIRED_TRIGGERS

See Also:
   Constant Field Values

INSERT_JOB_DETAIL

static final String INSERT_JOB_DETAIL

See Also:
   Constant Field Values

UPDATE_JOB_DETAIL

static final String UPDATE_JOB_DETAIL

See Also:
   Constant Field Values

SELECT_TRIGGERS_FOR_JOB

static final String SELECT_TRIGGERS_FOR_JOB

See Also:
   Constant Field Values

SELECT_TRIGGERS_FORCALENDAR
static final String SELECT_TRIGGERS_FOR_CALENDAR

See Also: Constant Field Values

DELETE_JOB_DETAIL

static final String DELETE_JOB_DETAIL

See Also: Constant Field Values

SELECT_JOB_NONCONCURRENT

static final String SELECT_JOB_NONCONCURRENT

See Also: Constant Field Values

SELECT_JOB_EXISTENCE

static final String SELECT_JOB_EXISTENCE

See Also: Constant Field Values

UPDATE_JOB_DATA

static final String UPDATE_JOB_DATA

See Also: Constant Field Values

SELECT_JOB_DETAIL
static final String SELECT_JOB_DETAIL

See Also:
Constant Field Values

---------

SELECT_NUM_JOBS

static final String SELECT_NUM_JOBS

See Also:
Constant Field Values

---------

SELECT_JOB_GROUPS

static final String SELECT_JOB_GROUPS

See Also:
Constant Field Values

---------

SELECT_JOBS_IN_GROUP

static final String SELECT_JOBS_IN_GROUP

See Also:
Constant Field Values

---------

INSERT_TRIGGER

static final String INSERT_TRIGGER

See Also:
Constant Field Values

---------

INSERT_SIMPLE_TRIGGER
static final String INSERT_SIMPLE_TRIGGER

See Also:
   Constant Field Values

_____________________________________________________

INSERT_CRON_TRIGGER
static final String INSERT_CRON_TRIGGER

See Also:
   Constant Field Values

_____________________________________________________

INSERT_BLOB_TRIGGER
static final String INSERT_BLOB_TRIGGER

See Also:
   Constant Field Values

_____________________________________________________

UPDATE_TRIGGER_SKIP_DATA
static final String UPDATE_TRIGGER_SKIP_DATA

See Also:
   Constant Field Values

_____________________________________________________

UPDATE_TRIGGER
static final String UPDATE_TRIGGER

See Also:
   Constant Field Values

_____________________________________________________

UPDATE_SIMPLE_TRIGGER
static final String UPDATE_SIMPLE_TRIGGER

See Also: Constant Field Values

UPDATE_CRON_TRIGGER

static final String UPDATE_CRON_TRIGGER

See Also: Constant Field Values

UPDATE_BLOB_TRIGGER

static final String UPDATE_BLOB_TRIGGER

See Also: Constant Field Values

SELECT_TRIGGER_EXISTENCE

static final String SELECT_TRIGGER_EXISTENCE

See Also: Constant Field Values

UPDATE_TRIGGER_STATE

static final String UPDATE_TRIGGER_STATE

See Also: Constant Field Values

UPDATE_TRIGGER_STATE_FROM_STATE
static final String UPDATE_TRIGGER_STATE_FROM_STATE

See Also:
Constant Field Values

-----------------------------

UPDATE_TRIGGER_GROUP_STATE_FROM_STATE

static final String UPDATE_TRIGGER_GROUP_STATE_FROM_STATE

See Also:
Constant Field Values

-----------------------------

UPDATE_TRIGGER_STATE_FROM_STATES

static final String UPDATE_TRIGGER_STATE_FROM_STATES

See Also:
Constant Field Values

-----------------------------

UPDATE_TRIGGER_GROUP_STATE_FROM_STATES

static final String UPDATE_TRIGGER_GROUP_STATE_FROM_STATES

See Also:
Constant Field Values

-----------------------------

UPDATE_JOB_TRIGGER_STATES

static final String UPDATE_JOB_TRIGGER_STATES

See Also:
Constant Field Values

-----------------------------

UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE
static final String UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE

See Also:
Constant Field Values

______________________________________________________________________________

DELETE_SIMPLE_TRIGGER

static final String DELETE_SIMPLE_TRIGGER

See Also:
Constant Field Values

______________________________________________________________________________

DELETE_CRON_TRIGGER

static final String DELETE_CRON_TRIGGER

See Also:
Constant Field Values

______________________________________________________________________________

DELETE_BLOB_TRIGGER

static final String DELETE_BLOB_TRIGGER

See Also:
Constant Field Values

______________________________________________________________________________

DELETE_TRIGGER

static final String DELETE_TRIGGER

See Also:
Constant Field Values

______________________________________________________________________________

SELECT_NUM_TRIGGERS_FOR_JOB
static final String SELECT_NUM_TRIGGERS_FOR_JOB

See Also:
Constant Field Values

SELECT_JOB_FOR_TRIGGER

static final String SELECT_JOB_FOR_TRIGGER

See Also:
Constant Field Values

SELECT_TRIGGER

static final String SELECT_TRIGGER

See Also:
Constant Field Values

SELECT_TRIGGER_DATA

static final String SELECT_TRIGGER_DATA

See Also:
Constant Field Values

SELECT_TRIGGER_STATE

static final String SELECT_TRIGGER_STATE

See Also:
Constant Field Values

SELECT_TRIGGER_STATUS
static final String SELECT_TRIGGER_STATUS

See Also:
Constant Field Values

SELECT_SIMPLE_TRIGGER

static final String SELECT_SIMPLE_TRIGGER

See Also:
Constant Field Values

SELECT_CRON_TRIGGER

static final String SELECT_CRON_TRIGGER

See Also:
Constant Field Values

SELECT_BLOB_TRIGGER

static final String SELECT_BLOB_TRIGGER

See Also:
Constant Field Values

SELECT_NUM_TRIGGERS

static final String SELECT_NUM_TRIGGERS

See Also:
Constant Field Values

SELECT_NUM_TRIGGERS_IN_GROUP
static final String SELECT_NUM_TRIGGERS_IN_GROUP

See Also:
Constant Field Values

SELECT_TRIGGER_GROUPS

static final String SELECT_TRIGGER_GROUPS

See Also:
Constant Field Values

SELECT_TRIGGER_GROUPS_FILTERED

static final String SELECT_TRIGGER_GROUPS_FILTERED

See Also:
Constant Field Values

SELECT_TRIGGERS_IN_GROUP

static final String SELECT_TRIGGERS_IN_GROUP

See Also:
Constant Field Values

INSERTCALENDAR

static final String INSERTCALENDAR

See Also:
Constant Field Values

UPDATECALENDAR
static final String UPDATE_CALENDAR

See Also:
Constant Field Values

SELECT_CALENDAR_EXISTENCE
static final String SELECT_CALENDAR_EXISTENCE

See Also:
Constant Field Values

SELECT_CALENDAR
static final String SELECT_CALENDAR

See Also:
Constant Field Values

SELECT_REFERENCED_CALENDAR
static final String SELECT_REFERENCED_CALENDAR

See Also:
Constant Field Values

DELETE_CALENDAR
static final String DELETE_CALENDAR

See Also:
Constant Field Values

SELECT_NUM_CALENDARS
static final String SELECT_NUM_CALENDARS

See Also:
Constant Field Values

SELECT_CALENDARS
static final String SELECT_CALENDARS

See Also:
Constant Field Values

SELECT_NEXT_FIRE_TIME
static final String SELECT_NEXT_FIRE_TIME

See Also:
Constant Field Values

SELECT_TRIGGER_FOR_FIRE_TIME
static final String SELECT_TRIGGER_FOR_FIRE_TIME

See Also:
Constant Field Values

SELECT_NEXT_TRIGGER_TO_ACQUIRE
static final String SELECT_NEXT_TRIGGER_TO_ACQUIRE

See Also:
Constant Field Values

INSERT_FIRED_TRIGGER
static final String INSERT_FIRED_TRIGGER

See Also:
Constant Field Values

---

UPDATE_FIRED_TRIGGER

static final String UPDATE_FIRED_TRIGGER

See Also:
Constant Field Values

---

UPDATE_INSTANCES_FIRED_TRIGGER_STATE

static final String UPDATE_INSTANCES_FIRED_TRIGGER_STATE

See Also:
Constant Field Values

---

SELECT_INSTANCES_FIRED_TRIGGERS

static final String SELECT_INSTANCES_FIRED_TRIGGERS

See Also:
Constant Field Values

---

SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS

static final String SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS

See Also:
Constant Field Values

---

SELECT_JOB_EXECUTION_COUNT
static final String SELECT_JOB_EXECUTION_COUNT

See Also:
Constant Field Values

-------------------

SELECT_FIRED_TRIGGERS

static final String SELECT_FIRED_TRIGGERS

See Also:
Constant Field Values

-------------------

SELECT_FIRED_TRIGGER

static final String SELECT_FIRED_TRIGGER

See Also:
Constant Field Values

-------------------

SELECT_FIRED_TRIGGER_GROUP

static final String SELECT_FIRED_TRIGGER_GROUP

See Also:
Constant Field Values

-------------------

SELECT_FIRED_TRIGGERS_OF_JOB

static final String SELECT_FIRED_TRIGGERS_OF_JOB

See Also:
Constant Field Values

-------------------

SELECT_FIRED_TRIGGERS_OF_JOB_GROUP
static final String SELECT_FIRED_TRIGGERS_OF_JOB_GROUP

See Also:
  Constant Field Values

---

DELETE_FIRED_TRIGGER

static final String DELETE_FIRED_TRIGGER

See Also:
  Constant Field Values

---

DELETE_INSTANCES_FIRED_TRIGGERS

static final String DELETE_INSTANCES_FIRED_TRIGGERS

See Also:
  Constant Field Values

---

DELETE_NO_RECOVERY_FIRED_TRIGGERS

static final String DELETE_NO_RECOVERY_FIRED_TRIGGERS

See Also:
  Constant Field Values

---

DELETE_ALL_SIMPLE_TRIGGERS

static final String DELETE_ALL_SIMPLE_TRIGGERS

See Also:
  Constant Field Values

---

DELETE_ALL_SIMPROP_TRIGGERS
static final String DELETE_ALL_SIMPROP_TRIGGERS

See Also:
Constant Field Values

DELETE_ALL_CRON_TRIGGERS

static final String DELETE_ALL_CRON_TRIGGERS

See Also:
Constant Field Values

DELETE_ALL_BLOB_TRIGGERS

static final String DELETE_ALL_BLOB_TRIGGERS

See Also:
Constant Field Values

DELETE_ALL_TRIGGERS

static final String DELETE_ALL_TRIGGERS

See Also:
Constant Field Values

DELETE_ALL_JOBDETAILS

static final String DELETE_ALL_JOBDETAILS

See Also:
Constant Field Values

DELETE_ALLCALENDARS
static final String DELETE_ALL_CALENDARS

See Also:
Constant Field Values

DELETE_ALL_PAUSED_TRIGGER_GRPS
static final String DELETE_ALL_PAUSED_TRIGGER_GRPS

See Also:
Constant Field Values

SELECT_FIRED_TRIGGER_INSTANCE_NAMES
static final String SELECT_FIRED_TRIGGER_INSTANCE_NAMES

See Also:
Constant Field Values

INSERT_SCHEDULER_STATE
static final String INSERT_SCHEDULER_STATE

See Also:
Constant Field Values

SELECT_SCHEDULER_STATE
static final String SELECT_SCHEDULER_STATE

See Also:
Constant Field Values

SELECT_SCHEDULER_STATES
static final String SELECT_SCHEDULER_STATES

See Also: Constant Field Values

DELETE_SCHEDULER_STATE

static final String DELETE_SCHEDULER_STATE

See Also: Constant Field Values

UPDATE_SCHEDULER_STATE

static final String UPDATE_SCHEDULER_STATE

See Also: Constant Field Values

INSERT_PAUSED_TRIGGER_GROUP

static final String INSERT_PAUSED_TRIGGER_GROUP

See Also: Constant Field Values

SELECT_PAUSED_TRIGGER_GROUP

static final String SELECT_PAUSED_TRIGGER_GROUP

See Also: Constant Field Values

SELECT_PAUSED_TRIGGER_GROUPS
static final String SELECT_PAUSED_TRIGGER_GROUPS

See Also:
Constant Field Values

DELETE_PAUSED_TRIGGER_GROUP

static final String DELETE_PAUSED_TRIGGER_GROUP

See Also:
Constant Field Values

DELETE_PAUSED_TRIGGER_GROUPS

static final String DELETE_PAUSED_TRIGGER_GROUPS

See Also:
Constant Field Values

Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.jdbcjobstore Class StdJDBCDelegate

java.lang.Object
  ⌈org.quartz.impl.jdbcjobstore.StdJDBCDelegate

All Implemented Interfaces:
    Constants, DriverDelegate, StdJDBCConstants

Direct Known Subclasses:
    CloudscapeDelegate, DB2v6Delegate, DB2v7Delegate, DB2v8Delegate,
    HSQLDBDelegate, MSSQLDelegate, OracleDelegate, PointbaseDelegate,
    PostgreSQLDelegate, SybaseDelegate, WebLogicDelegate

public class StdJDBCDelegate
    extends Object
    implements DriverDelegate, StdJDBCConstants

This is meant to be an abstract base class for most, if not all, DriverDelegate
implementations. Subclasses should override only those methods that need
special handling for the DBMS driver in question.

Author:
    Jeffrey Wescott, James House, Eric Mueller

Field Summary

<p>|protected| org.quartz.spi.ClassLoadHelper|classLoadHelper|
|protected| String|instanceId|
|protected| org.slf4j.Logger|logger|
|protected| String|schedName|
|protected| String|tablePrefix|</p>
<table>
<thead>
<tr>
<th>protected List&lt;TriggerPersistenceDelegate&gt; triggerPersistenceDelegates</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected boolean useProperties</td>
</tr>
</tbody>
</table>

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
SELECT TRIGGERS FOR CALENDAR, SELECT TRIGGERS FOR JOB,
SELECT TRIGGERS IN GROUP, SELECT TRIGGERS IN STATE,
TABLE PREFIX SUBST, UPDATE BLOB TRIGGER, UPDATE CALENDAR,
UPDATE CRON TRIGGER, UPDATE FIRED TRIGGER,
UPDATE INSTANCES FIRED_TRIGGER STATE, UPDATE_JOB_DATA,
UPDATE JOB_DETAIL, UPDATE JOB_TRIGGER STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Const\stants
ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL,
COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID,
COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME,
COL_IS_DURABLE, COL_IS_NONCONCURRENT, COL_IS_UPDATE_DATA,
COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP,
COL_JOB_NAME, COL_LAST_CHECKIN_TIME, COL_LOCK_NAME,
COL_MISFIRE_INSTRUCTION, COL_NEXT_FIRE_TIME, COL_PREV_FIRE_TIME,
COL_PRIORITY, COL_REPEAT_COUNT, COL_REPEAT_INTERVAL,
COL_REQUESTS_RECOVERY, COL_SCHEDULER_NAME, COL_START_TIME,
COL_TIME_ZONE_ID, COL_TIMES_TRIGGERED, COL_TRIGGER_GROUP,
COL_TRIGGER_NAME, COL_TRIGGER_STATE, COL_TRIGGER_TYPE,
DEFAULT_TABLE_PREFIX, STATE_ACQUIRED, STATE_BLOCKED,
STATE_COMPLETE, STATE_DELETED, STATE_ERROR, STATE_EXECUTING,
STATE_MISFIRED, STATE_PAUSED, STATE_PAUSED_BLOCKED, STATE_WAITING,
TABLE_BLOB_TRIGGERS, TABLE_CALENDARS, TABLE_CRON_TRIGGERS,
TABLE_FIRED_TRIGGERS, TABLE_JOB_DETAILS, TABLE_LOCKS,
TABLE_PAUSED_TRIGGERS, TABLE_SCHEDULER_STATE,
TABLE_SIMPLE_TRIGGERS, TABLE_TRIGGERS, TTYPE_BLOB, TTYPE_CAL_INT,
TTYPE_CRON, TTYPE_SIMPLE

Constructor Summary

StdJDBCDelegate\(org.slf4j.Logger logger, String tablePrefix,
String schedName, String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper\)

Create new StdJDBCDelegate instance.

StdJDBCDelegate\(org.slf4j.Logger logger, String tablePrefix,
String schedName, String instanceId,\)
Create new StdJDBCDelegate instance.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void addDefaultTriggerPersistenceDelegates()</td>
<td>Add default trigger persistence delegates.</td>
</tr>
<tr>
<td>void addTriggerPersistenceDelegate(TriggerPersistenceDelegate delegate)</td>
<td>Add a trigger persistence delegate.</td>
</tr>
<tr>
<td>boolean calendarExists(Connection conn, String calendarName)</td>
<td>Check whether or not a calendar exists.</td>
</tr>
<tr>
<td>boolean calendarIsReferenced(Connection conn, String calendarName)</td>
<td>Check whether or not a calendar is referenced by any triggers.</td>
</tr>
<tr>
<td>protected boolean canUseProperties()</td>
<td>Whether to use properties.</td>
</tr>
<tr>
<td>void clearData(Connection conn)</td>
<td>Clear all scheduling data - all Job instances deleted.</td>
</tr>
<tr>
<td>protected static void closeResultSet(ResultSet rs)</td>
<td>Cleanup helper method that closes the given ResultSet.</td>
</tr>
<tr>
<td>protected static void closeStatement(Statement statement)</td>
<td>Cleanup helper method that closes the given Statement.</td>
</tr>
<tr>
<td>protected Map&lt;?,?&gt; convertFromProperty(Properties properties)</td>
<td>Convert the JobDataMap into a list of properties.</td>
</tr>
<tr>
<td>protected Properties convertToProperty(Map&lt;?,?&gt; data)</td>
<td>Convert the JobDataMap into a list of properties.</td>
</tr>
<tr>
<td>int countMisfiredTriggersInState(Connection conn, String[] states, Date timestamp)</td>
<td>Get the number of triggers in the given states that have misfired.</td>
</tr>
<tr>
<td>int deleteAllPausedTriggerGroups(Connection conn)</td>
<td>Delete all paused trigger groups.</td>
</tr>
<tr>
<td>int deleteBlobTrigger(Connection conn, TriggerKey triggerKey)</td>
<td>Delete the cron trigger data for a trigger.</td>
</tr>
</tbody>
</table>
deleteCalendar(Connection conn, String calendarName)
Delete a calendar.

int deleteFiredTrigger(Connection conn, String calendarName)
Delete a fired trigger.

int deleteFiredTriggers(Connection conn)
Delete all fired triggers.

int deleteFiredTriggers(Connection conn, String calendarName)
Delete all fired triggers of the given instance.

int deleteJobDetail(Connection conn, JobKey jobKey)
Delete the job detail record for the given instance.

int deletePausedTriggerGroup(Connection conn, GroupMatcher<TriggerKey> matcher)
Delete all fired triggers of the given instance.

int deleteSchedulerState(Connection conn, String discriminator)
Delete a scheduler-instance state record.

int deleteTrigger(Connection conn, TriggerKey triggerKey)
Delete the base trigger data for a trigger.

protected void deleteTriggerExtension(Connection conn, TriggerPersistenceDelegate triggerPersistenceDelegate)
Find the key of the first non-serializable value in the given Map.

protected Object getKeyOfNonSerializableValue(Map<?, ?> data)
Find the key of the first non-serializable value in the given Map.

protected boolean getBoolean(ResultSet rs, int columnIndex)
Retrieves the value of the designated column index as boolean.

protected boolean getBoolean(ResultSet rs, String columnName)
Retrieves the value of the designated column in the current row as boolean.

protected Object getJobDataFromBlob(ResultSet rs, String columnName)
This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.

protected Object getKeyOfNonSerializableValue(Map<?, ?> data)
Find the key of the first non-serializable value in the given Map.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected <code>Object</code> <code>getObjectFromBlob</code></td>
<td>This method should be overridden by any delegate subclasses that need special handling for BLOBs.</td>
</tr>
<tr>
<td>protected <code>String</code> <code>getSchedulerNameLiteral</code></td>
<td></td>
</tr>
<tr>
<td>boolean <code>hasMisfiredTriggersInState</code></td>
<td>Get the names of all of the triggers in the given state that have according to the given timestamp.</td>
</tr>
<tr>
<td>void <code>initialize</code> <code>initializeString</code></td>
<td>initStrings are of the format: settingName=settingValue</td>
</tr>
<tr>
<td>int <code>insertBlobTrigger</code></td>
<td>Insert the blob trigger data.</td>
</tr>
<tr>
<td>int <code>insertCalendar</code> <code>insertCalendar</code></td>
<td>Insert a new calendar.</td>
</tr>
<tr>
<td>int <code>insertFiredTrigger</code></td>
<td>Insert a fired trigger.</td>
</tr>
<tr>
<td>int <code>insertJobDetail</code> <code>insertJobDetail</code></td>
<td>Insert the job detail record.</td>
</tr>
<tr>
<td>int <code>insertPausedTriggerGroup</code></td>
<td></td>
</tr>
<tr>
<td>int <code>insertSchedulerState</code></td>
<td>Insert a scheduler-instance state record.</td>
</tr>
<tr>
<td>int <code>insertTrigger</code> <code>insertTrigger</code></td>
<td>Insert the base trigger data.</td>
</tr>
<tr>
<td>boolean <code>isExistingTriggerGroup</code></td>
<td></td>
</tr>
<tr>
<td>boolean <code>isJobNonConcurrent</code></td>
<td>Check whether or not the given job is stateful.</td>
</tr>
<tr>
<td>boolean <code>isTriggerGroupPaused</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>boolean jobExists(Connection conn, JobKey jobKey)</td>
<td>Check whether or not the given job exists.</td>
</tr>
<tr>
<td>protected String rtp(String query)</td>
<td>Replace the table prefix in a query by replacing any occurrences of <code>*</code> with the table prefix.</td>
</tr>
<tr>
<td>Calendar selectCalendar(Connection conn, String calendarName)</td>
<td>Select a calendar.</td>
</tr>
<tr>
<td>List&lt;String&gt; selectCalendars(Connection conn)</td>
<td>Select all of the stored calendars.</td>
</tr>
<tr>
<td>Set&lt;String&gt; selectFiredTriggerInstanceNames(Connection conn)</td>
<td>Select the distinct instance names of all fired-trigger records.</td>
</tr>
<tr>
<td>List&lt;FiredTriggerRecord&gt; selectFiredTriggerRecords(Connection conn, String groupName)</td>
<td>Select the states of all fired-trigger records for a given group if trigger name is null.</td>
</tr>
<tr>
<td>List&lt;FiredTriggerRecord&gt; selectFiredTriggerRecordsByJob(Connection conn, String groupName)</td>
<td>Select the states of all fired-trigger records for a given job, or job name is null.</td>
</tr>
<tr>
<td>List&lt;FiredTriggerRecord&gt; selectInstancesFiredTriggerRecords(Connection conn, String instanceName)</td>
<td>Select the states of all fired-trigger records for a given scheduler.</td>
</tr>
<tr>
<td>JobDetail selectJobDetail(Connection conn, JobKey jobKey, org.quartz.spi.ClassLoadHelper loadHelper)</td>
<td>Select the JobDetail object for a given job.</td>
</tr>
<tr>
<td>int selectJobExecutionCount(Connection conn)</td>
<td>Get the number instances of the identified job currently executing.</td>
</tr>
<tr>
<td>JobDetail selectJobForTrigger(Connection conn, org.quartz.spi.ClassLoadHelper loadHelper)</td>
<td>Select the job to which the trigger is associated.</td>
</tr>
<tr>
<td>List&lt;String&gt; selectJobGroups(Connection conn)</td>
<td>Select all of the job group names that are associated with the given connection.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; selectJobsInGroup(Connection conn, GroupMatcher matcher)</td>
<td>Select all of the jobs contained in a given group.</td>
</tr>
<tr>
<td>List&lt;TriggerKey&gt; selectMisfiredTriggers(Connection conn, ...)</td>
<td>Select all of the misfired trigger records.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>selectMisfiredTriggersInGroupInState</code></td>
<td>Get the names of all of the triggers that have misfired.</td>
</tr>
<tr>
<td><code>selectMisfiredTriggersInState</code></td>
<td>Get the names of all of the triggers in the given group and state that misfired.</td>
</tr>
<tr>
<td><code>selectNextFireTime</code></td>
<td>Get the names of all of the triggers in the given state that have misfired according to the given timestamp.</td>
</tr>
<tr>
<td><code>selectNumCalendars</code></td>
<td>Select the total number of calendars stored.</td>
</tr>
<tr>
<td><code>selectNumJobs</code></td>
<td>Select the total number of jobs stored.</td>
</tr>
<tr>
<td><code>selectNumTriggers</code></td>
<td>Select the total number of triggers stored.</td>
</tr>
<tr>
<td><code>selectNumTriggersForJob</code></td>
<td>Select the number of triggers associated with the given job.</td>
</tr>
<tr>
<td><code>selectPausedTriggerGroups</code></td>
<td>Select all of the trigger group names that are stored.</td>
</tr>
<tr>
<td><code>selectSchedulerStateRecords</code></td>
<td>A List of all current SchedulerStateRecords.</td>
</tr>
<tr>
<td><code>selectSchedulerStateRecord</code></td>
<td>A List of all current SchedulerStateRecords.</td>
</tr>
<tr>
<td><code>selectTrigger</code></td>
<td>Select a trigger.</td>
</tr>
<tr>
<td><code>selectTriggerForFireTime</code></td>
<td>Select the trigger that will be fired at the given fire time.</td>
</tr>
<tr>
<td><code>selectTriggerGroups</code></td>
<td>Select all of the trigger group names that are stored.</td>
</tr>
<tr>
<td><code>selectTriggerGroups</code></td>
<td>Select all of the trigger group names that are stored.</td>
</tr>
<tr>
<td><code>selectTriggerJobDataMap</code></td>
<td>Select a trigger's JobDataMap.</td>
</tr>
<tr>
<td><code>selectTriggerKeysForJob</code></td>
<td>Get the names of all of the triggers associated with the given job.</td>
</tr>
<tr>
<td>Method Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>List</strong>&lt;<strong>org.quartz.spi.OperableTrigger</strong>&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggersForCalendar</strong> <em>(Connection conn)</em></td>
<td></td>
</tr>
<tr>
<td>Select the triggers for a calendar</td>
<td></td>
</tr>
<tr>
<td><strong>List</strong>&lt;<strong>org.quartz.spi.OperableTrigger</strong>&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggersForJob</strong> <em>(Connection conn, JobKey jobKey)</em></td>
<td></td>
</tr>
<tr>
<td>Select the triggers for a job</td>
<td></td>
</tr>
<tr>
<td><strong>List</strong>&lt;<strong>org.quartz.spi.OperableTrigger</strong>&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggersForRecoveringJobs</strong> <em>(Connection conn)</em></td>
<td></td>
</tr>
<tr>
<td>Select all of the triggers for jobs that are requesting recovery.</td>
<td></td>
</tr>
<tr>
<td>**Set&lt;*<em>TriggerKey</em>&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggersInGroup</strong> <em>(Connection conn, GroupMatcher&lt;TriggerKey&gt; matcher)</em></td>
<td></td>
</tr>
<tr>
<td>Select all of the triggers contained in a group</td>
<td></td>
</tr>
<tr>
<td>**List&lt;*<em>TriggerKey</em>&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggersInState</strong> <em>(Connection conn, String state)</em></td>
<td></td>
</tr>
<tr>
<td>Select all of the triggers in a given state.</td>
<td></td>
</tr>
<tr>
<td><strong>String</strong></td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggerState</strong> <em>(Connection conn, TriggerKey triggerKey)</em></td>
<td></td>
</tr>
<tr>
<td>Select a trigger’s state value.</td>
<td></td>
</tr>
<tr>
<td><strong>TriggerStatus</strong></td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggerStatus</strong> <em>(Connection conn, TriggerKey triggerKey)</em></td>
<td></td>
</tr>
<tr>
<td>Select a trigger’s status (state &amp; next fire time).</td>
<td></td>
</tr>
<tr>
<td>**List&lt;*<em>TriggerKey</em>&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>selectTriggerToAcquire</strong> <em>(Connection conn, long noEarlierThan)</em></td>
<td></td>
</tr>
<tr>
<td>Select the next trigger which will fire to fire between the two given</td>
<td></td>
</tr>
<tr>
<td>timestamps in ascending order of fire time, and descending by priority.</td>
<td></td>
</tr>
<tr>
<td><strong>protected ByteArrayOutputStream</strong></td>
<td></td>
</tr>
<tr>
<td><strong>serializeJobData</strong> <em>(JobDataMap data)</em></td>
<td></td>
</tr>
<tr>
<td>Remove the transient data from and then create a serialized java.util.ByteArrayOutputStream version of the data.</td>
<td></td>
</tr>
<tr>
<td><strong>protected ByteArrayOutputStream</strong></td>
<td></td>
</tr>
<tr>
<td><strong>serializeObject</strong> <em>(Object obj)</em></td>
<td></td>
</tr>
<tr>
<td>Create a serialized java.util.ByteArrayOutputStream Object.</td>
<td></td>
</tr>
<tr>
<td><strong>protected void</strong></td>
<td></td>
</tr>
<tr>
<td><strong>setBoolean</strong> <em>(PreparedStatement ps, int index)</em></td>
<td></td>
</tr>
<tr>
<td>Sets the designated parameter to the given</td>
<td></td>
</tr>
<tr>
<td><strong>protected void</strong></td>
<td></td>
</tr>
<tr>
<td><strong>setBytes</strong> <em>(PreparedStatement ps, int index)</em></td>
<td></td>
</tr>
<tr>
<td>Sets the designated parameter to the byte</td>
<td></td>
</tr>
<tr>
<td>array of the given ByteArrayOutputStream.</td>
<td></td>
</tr>
<tr>
<td><strong>protected String</strong></td>
<td></td>
</tr>
<tr>
<td><strong>toSqlLikeClause</strong> <em>(GroupMatcher matcher)</em></td>
<td></td>
</tr>
<tr>
<td><strong>boolean</strong></td>
<td></td>
</tr>
<tr>
<td><strong>triggerExists</strong> <em>(Connection conn, TriggerKey triggerKey)</em></td>
<td></td>
</tr>
<tr>
<td>Check whether or not a trigger exists.</td>
<td></td>
</tr>
<tr>
<td><strong>int</strong></td>
<td></td>
</tr>
<tr>
<td><strong>updateBlobTrigger</strong> <em>(Connection conn, org.quartz.spi.OperableTrigger trigger)</em></td>
<td></td>
</tr>
<tr>
<td><strong>updateBlobTrigger</strong> <em>(Connection conn, org.quartz.spi.OperableTrigger trigger)</em></td>
<td></td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>updateCalendar</td>
<td>Update the blob trigger data.</td>
</tr>
<tr>
<td>updateCalendar(Connection conn, String Calendar calendar)</td>
<td>Update a calendar.</td>
</tr>
<tr>
<td>updateFiredTrigger</td>
<td>Update a fired trigger.</td>
</tr>
<tr>
<td>updateFiredTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger)</td>
<td>Update a fired trigger.</td>
</tr>
<tr>
<td>updateJobData</td>
<td>Update the job data map for the given job.</td>
</tr>
<tr>
<td>updateJobDetail</td>
<td>Update the job detail record.</td>
</tr>
<tr>
<td>updateJobDetail(Connection conn, JobDetail jobDetail)</td>
<td>Update the job detail record.</td>
</tr>
<tr>
<td>updateSchedulerState</td>
<td>Update a scheduler-instance state record.</td>
</tr>
<tr>
<td>updateSchedulerState(Connection conn, String checkInTime)</td>
<td>Update a scheduler-instance state record.</td>
</tr>
<tr>
<td>updateTrigger</td>
<td>Update the base trigger data.</td>
</tr>
<tr>
<td>updateTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)</td>
<td>Update the base trigger data.</td>
</tr>
<tr>
<td>updateTriggerGroupStateFromOtherState</td>
<td>Update all of the triggers of the given group to the given new state, if they are in the given old state.</td>
</tr>
<tr>
<td>updateTriggerGroupStateFromOtherStates</td>
<td>Update all triggers in the given group to the given new state, if they one of the given old states.</td>
</tr>
<tr>
<td>updateTriggerState</td>
<td>Update the state for a given trigger.</td>
</tr>
<tr>
<td>updateTriggerStateFromOtherState</td>
<td>Update the given trigger to the given new state.</td>
</tr>
<tr>
<td>updateTriggerStateFromOtherStates</td>
<td>Update the given trigger to the given new state.</td>
</tr>
<tr>
<td>updateTriggerStatesForJob</td>
<td>Update the given trigger to the given new state.</td>
</tr>
</tbody>
</table>
String state)

Update the states of all triggers associated with the given job.

int updateTriggerStatesForJobFromOtherState
JobKey jobKey, String state, String oldState)
Update the states of any triggers associated with the given job, given the current state.

int updateTriggerStatesFromOtherStates(Connection conn,
String oldState1, String oldState2)
Insert the job detail record.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait

Field Detail

logger

protected org.slf4j.Logger logger

tablePrefix

protected String tablePrefix

instanceId

protected String instanceId

schedName

protected String schedName
useProperties
protected boolean useProperties

classLoadHelper
protected org.quartz.spi.ClassLoadHelper classLoadHelper

triggerPersistenceDelegates
protected List<TriggerPersistenceDelegate> triggerPersistenceDelegates

| Constructor Detail |

**StdJDBCDelegate**

public StdJDBCDelegate(org.slf4j.Logger logger,
String tablePrefix,
String schedName,
String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper,
Boolean useProperties)

Create new StdJDBCDelegate instance.

**Parameters:**
- logger - the logger to use during execution
- tablePrefix - the prefix of all table names
Create new StdJDBCDelgate instance.

**Parameters:**
- logger - the logger to use during execution
- tablePrefix - the prefix of all table names

### Method Detail

**initialize**

```java
public void initialize(String initString)
```

Throws: `NoSuchDelegateException`

initStrings are of the format:
settingName=settingValue|otherSettingName=otherSettingValue|...

**Specified by:**
- `initialize` in interface `DriverDelegate`

**Throws:**
- `NoSuchDelegateException`

---

**addDefaultTriggerPersistenceDelegates**

```java
protected void addDefaultTriggerPersistenceDelegates()
```

---

**canUseProperties**

```java
protected boolean canUseProperties()
```

---

**addTriggerPersistenceDelegate**

```java
public void addTriggerPersistenceDelegate(TriggerPersistenceDelegate delegate)
```

---

**findTriggerPersistenceDelegate**
public TriggerPersistenceDelegate findTriggerPersistenceDelegate(org.

findTriggerPersistenceDelegate

public TriggerPersistenceDelegate findTriggerPersistenceDelegate(String.

updateTriggerStatesFromOtherStates

public int updateTriggerStatesFromOtherStates(Connection conn, String newState, String oldState1, String oldState2)
throws SQLException

Insert the job detail record.

Specified by:
updateTriggerStatesFromOtherStates in interface DriverDelegate

Parameters:
- conn - the DB Connection
- newState - the new state for the triggers
- oldState1 - the first old state to update
- oldState2 - the second old state to update

Returns:
- number of rows updated

Throws:
- SQLException

selectMisfiredTriggers

public List<TriggerKey> selectMisfiredTriggers(Connection conn, long ts)
throws SQLException

Get the names of all of the triggers that have misfired.

Specified by:
- selectMisfiredTriggers in interface DriverDelegate
Parameters:

conn - the DB Connection

Returns:

an array of Key objects

Throws:

SQLException

selectTriggersInState

public List<TriggerKey> selectTriggersInState(Connection conn, String state)
throws SQLException

Select all of the triggers in a given state.

Specified by:

selectTriggersInState in interface DriverDelegate

Parameters:

conn - the DB Connection
state - the state the triggers must be in

Returns:

an array of trigger Key s

Throws:

SQLException

selectMisfiredTriggersInState

public List<TriggerKey> selectMisfiredTriggersInState(Connection con, String state, long ts)
throws SQLException

Description copied from interface: DriverDelegate

Get the names of all of the triggers in the given state that have misfired - according to the given timestamp.

Specified by:

selectMisfiredTriggersInState in interface DriverDelegate
**Parameters:**
conn - the DB Connection

**Returns:**
an array of Key objects

**Throws:**
SQLException

---

**hasMisfiredTriggersInState**

```java
public boolean hasMisfiredTriggersInState(Connection conn, String state1, long ts, int count, List<TriggerKey> resultList) throws SQLException
```

Get the names of all of the triggers in the given state that have misfired - according to the given timestamp. No more than count will be returned.

**Specified by:**
hasMisfiredTriggersInState in interface DriverDelegate

**Parameters:**
conn - The DB Connection
count - The most misfired triggers to return, negative for all resultlist - Output parameter. A List of Key objects. Must not be null.

**Returns:**
Whether there are more misfired triggers left to find beyond the given count.

**Throws:**
SQLException

---

**countMisfiredTriggersInState**

```java
public int countMisfiredTriggersInState(Connection conn, String state1, long ts) throws SQLException
```

---
Get the number of triggers in the given states that have misfired - according to the given timestamp.

**Specified by:**
- `countMisfiredTriggersInState` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection

**Throws:**
- `SQLException`

---

**selectMisfiredTriggersInGroupInState**

```java
define function selectMisfiredTriggersInGroupInState
```

Get the names of all of the triggers in the given group and state that have misfired.

**Specified by:**
- `selectMisfiredTriggersInGroupInState` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection

**Returns:**
- an array of Key objects

**Throws:**
- `SQLException`

---

**selectTriggersForRecoveringJobs**

```java
define function selectTriggersForRecoveringJobs
```

Select all of the triggers for jobs that are requesting recovery. The returned
trigger objects will have unique "recoverXXX" trigger names and will be in the Scheduler.DEFAULT_RECOVERY_GROUP trigger group.

In order to preserve the ordering of the triggers, the fire time will be set from the COL_FIRED_TIME column in the TABLE_FIRED_TRIGGERS table. The caller is responsible for calling computeFirstFireTime on each returned trigger. It is also up to the caller to insert the returned triggers to ensure that they are fired.

**Specified by:**

```
selectTriggersForRecoveringJobs in interface DriverDelegate
```

**Parameters:**

- `conn` - the DB Connection

**Returns:**

- an array of `Trigger` objects

**Throws:**

- `SQLException`
- `IOException`
- `ClassNotFoundException`

---

**deleteFiredTriggers**

```
public int deleteFiredTriggers(Connection conn) throws SQLException
```

Delete all fired triggers.

**Specified by:**

```
deleteFiredTriggers in interface DriverDelegate
```

**Parameters:**

- `conn` - the DB Connection

**Returns:**

- the number of rows deleted

**Throws:**

- `SQLException`
public int deleteFiredTriggers(Connection conn, String instanceId) throws SQLException

Description copied from interface: DriverDelegate

Delete all fired triggers of the given instance.

Specified by:
   deleteFiredTriggers in interface DriverDelegate

Parameters:
   conn - the DB Connection

Returns:
   the number of rows deleted

Throws:
   SQLException

clearData

public void clearData(Connection conn) throws SQLException

Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

Specified by:
   clearData in interface DriverDelegate

Throws:
   JobPersistenceException
   SQLException

insertJobDetail

public int insertJobDetail(Connection conn, JobDetail job) throws IOException, SQLException

Insert the job detail record.
Specified by:
   insertJobDetail in interface DriverDelegate

Parameters:
   conn - the DB Connection
   job - the job to insert

Returns:
   number of rows inserted

Throws:
   IOException - if there were problems serializing the JobDataMap
   SQLException

updateJobDetail

public int updateJobDetail(Connection conn, JobDetail job)
   throws IOException, SQLException

Update the job detail record.

Specified by:
   updateJobDetail in interface DriverDelegate

Parameters:
   conn - the DB Connection
   job - the job to update

Returns:
   number of rows updated

Throws:
   IOException - if there were problems serializing the JobDataMap
   SQLException

selectTriggerKeysForJob

public List<TriggerKey> selectTriggerKeysForJob(Connection conn, JobKey jobKey)
   throws SQLException

Get the names of all of the triggers associated with the given job.
Specified by:
    selectTriggerKeysForJob in interface DriverDelegate

Parameters:
    conn - the DB Connection

Returns:
    an array of Key objects

Throws:
    SQLException

---

deleteJobDetail

public int deleteJobDetail(Connection conn,
                                  JobKey jobKey)
    throws SQLException

Delete the job detail record for the given job.

Specified by:
    deleteJobDetail in interface DriverDelegate

Parameters:
    conn - the DB Connection

Returns:
    the number of rows deleted

Throws:
    SQLException

---

isJobNonConcurrent

public boolean isJobNonConcurrent(Connection conn,
                                         JobKey jobKey)
    throws SQLException

Check whether or not the given job is stateful.

Specified by:
    isJobNonConcurrent in interface DriverDelegate

Parameters:
    conn - the DB Connection
jobExists

```java
public boolean jobExists(Connection conn, JobKey jobKey) throws SQLException
```

Check whether or not the given job exists.

**Specified by:**
jobExists in interface DriverDelegate

**Parameters:**
- conn - the DB Connection

**Returns:**
true if the job exists, false otherwise

**Throws:**
SQLException

updateJobData

```java
public int updateJobData(Connection conn, JobDetail job) throws IOException, SQLException
```

Update the job data map for the given job.

**Specified by:**
updateJobData in interface DriverDelegate

**Parameters:**
- conn - the DB Connection
- job - the job to update

**Returns:**
the number of rows updated
Throws:
- IOException - if there were problems serializing the JobDataMap
- SQLException

selectJobDetail

public JobDetail selectJobDetail(Connection conn, JobKey jobKey, org.quartz.spi.ClassLoadHelper loadHelper) throws ClassNotFoundException, IOException, SQLException

Select the JobDetail object for a given job name / group name.

Specified by:
selectJobDetail in interface DriverDelegate

Parameters:
- conn - the DB Connection

Returns:
- the populated JobDetail object

Throws:
- ClassNotFoundException - if a class found during deserialization cannot be found or if the job class could not be found
- IOException - if deserialization causes an error
- SQLException

selectNumJobs

public int selectNumJobs(Connection conn) throws SQLException

Select the total number of jobs stored.

Specified by:
selectNumJobs in interface DriverDelegate

Parameters:
- conn - the DB Connection

Returns:
the total number of jobs stored

Throws:
SQLException

---

selectJobGroups

public List<String> selectJobGroups(Connection conn) throws SQLException

Select all of the job group names that are stored.

Specified by:
selectJobGroups in interface DriverDelegate

Parameters:
conn - the DB Connection

Returns:
an array of String group names

Throws:
SQLException

---

selectJobsInGroup

public Set<JobKey> selectJobsInGroup(Connection conn, GroupMatcher<JobKey> matcher) throws SQLException

Select all of the jobs contained in a given group.

Specified by:
selectJobsInGroup in interface DriverDelegate

Parameters:
conn - the DB Connection
matcher - the groupMatcher to evaluate the jobs against

Returns:
an array of String job names

Throws:
SQLException
**toSqlLikeClause**

protected `String toSqlLikeClause(GroupMatcher matcher)`

---

**insertTrigger**

public int `insertTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)`

throws `SQLException`, `IOException`

Insert the base trigger data.

**Specified by:**

`insertTrigger` in interface `DriverDelegate`

**Parameters:**

- conn - the DB Connection
- trigger - the trigger to insert
- state - the state that the trigger should be stored in

**Returns:**

the number of rows inserted

**Throws:**

`SQLException`

`IOException`

---

**insertBlobTrigger**

public int `insertBlobTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger)`

throws `SQLException`, `IOException`

Insert the blob trigger data.

**Parameters:**

- conn - the DB Connection
- trigger - the trigger to insert
Returns:
the number of rows inserted

Throws:
SQLException
IOException

updateTrigger

public int updateTrigger(Connection conn,
                          org.quartz.spi.OperableTrigger trigger,
                          String state,
                          JobDetail jobDetail)
throws SQLException,
        IOException

Update the base trigger data.

Specified by:
updateTrigger in interface DriverDelegate

Parameters:
conn - the DB Connection
trigger - the trigger to insert
state - the state that the trigger should be stored in

Returns:
the number of rows updated

Throws:
SQLException
IOException

updateBlobTrigger

public int updateBlobTrigger(Connection conn,
                              org.quartz.spi.OperableTrigger trigger)
throws SQLException,
        IOException

Update the blob trigger data.

Parameters:
trigger - the trigger to insert

Returns:
the number of rows updated

Throws:
SQLException
IOException

triggerExists

public boolean triggerExists(Connection conn, TriggerKey triggerKey)
throws SQLException

Check whether or not a trigger exists.

Specified by:
triggerExists in interface DriverDelegate

Parameters:
conn - the DB Connection

Returns:
true if the trigger exists, false otherwise

Throws:
SQLException

updateTriggerState

public int updateTriggerState(Connection conn, TriggerKey triggerKey, String state)
throws SQLException

Update the state for a given trigger.

Specified by:
updateTriggerState in interface DriverDelegate

Parameters:
conn - the DB Connection
state - the new state for the trigger
Returns:
the number of rows updated

Throws:
SQLException

updateTriggerStateFromOtherStates

public int updateTriggerStateFromOtherStates(Connection conn,
                                            TriggerKey triggerKey,
                                            String newState,
                                            String oldState1,
                                            String oldState2,
                                            String oldState3)
    throws SQLException

Update the given trigger to the given new state, if it is one of the given old states.

Specified by:
updateTriggerStateFromOtherStates in interface DriverDelegate

Parameters:
conn - the DB connection
newState - the new state for the trigger
oldState1 - one of the old state the trigger must be in
oldState2 - one of the old state the trigger must be in
oldState3 - one of the old state the trigger must be in

Returns:
int the number of rows updated

Throws:
SQLException

updateTriggerGroupStateFromOtherStates

public int updateTriggerGroupStateFromOtherStates(Connection conn,
                                                    GroupMatcher<TriggerKey> triggerMatcher,
                                                    String newState,
                                                    String oldState1,
                                                    String oldState2,
                                                    String oldState3)
Update all triggers in the given group to the given new state, if they are in one of the given old states.

Specified by:

updateTriggerGroupStateFromOtherStates in interface DriverDelegate

Parameters:
conn - the DB connection
matcher - the groupMatcher to evaluate the triggers against
newState - the new state for the trigger
oldState1 - one of the old state the trigger must be in
oldState2 - one of the old state the trigger must be in
oldState3 - one of the old state the trigger must be in

Returns:
int the number of rows updated

Throws:
SQLException

updateTriggerStateFromOtherState

public int updateTriggerStateFromOtherState(Connection conn,
     TriggerKey triggerKey,
     String newState,
     String oldState)
throws SQLException

Update the given trigger to the given new state, if it is in the given old state.

Specified by:

updateTriggerStateFromOtherState in interface DriverDelegate

Parameters:
conn - the DB connection
newState - the new state for the trigger
oldState - the old state the trigger must be in

Returns:
int the number of rows updated

Throws:
SQLException
**updateTriggerGroupStateFromOtherState**

```
public int updateTriggerGroupStateFromOtherState(Connection conn, GroupMatcher<Trigger> matcher, String newState, String oldState) throws SQLException
```

Update all of the triggers of the given group to the given new state, if they are in the given old state.

**Specified by:**
- `updateTriggerGroupStateFromOtherState` in interface `DriverDelegate`  

**Parameters:**
- `conn` - the DB connection 
- `matcher` - the groupMatcher to evaluate the triggers against 
- `newState` - the new state for the trigger group 
- `oldState` - the old state the triggers must be in

**Returns:**
- `int` the number of rows updated

**Throws:**
- `SQLException`

---

**updateTriggerStatesForJob**

```
public int updateTriggerStatesForJob(Connection conn, JobKey jobKey, String state) throws SQLException
```

Update the states of all triggers associated with the given job.

**Specified by:**
- `updateTriggerStatesForJob` in interface `DriverDelegate`  

**Parameters:**
- `conn` - the DB Connection 
- `state` - the new state for the triggers

**Returns:**
the number of rows updated

Throws:
SQLException

updateTriggerStatesForJobFromOtherState

public int updateTriggerStatesForJobFromOtherState(
    Connection conn,
    JobKey jobKey,
    String state,
    String oldState)
throws SQLException

Description copied from interface: DriverDelegate

Update the states of any triggers associated with the given job, that are the given current state.

Specified by:
updateTriggerStatesForJobFromOtherState in interface DriverDelegate

Parameters:
conn - the DB Connection
state - the new state for the triggers
oldState - the old state of the triggers

Returns:
the number of rows updated

Throws:
SQLException

deleteBlobTrigger

public int deleteBlobTrigger(
    Connection conn,
    TriggerKey triggerKey)
throws SQLException

Delete the cron trigger data for a trigger.

Parameters:
conn - the DB Connection
Returns:
the number of rows deleted

Throws:
SQLException

---

deleteTrigger

public int deleteTrigger(Connection conn,
                        TriggerKey triggerKey)
throws SQLException

Delete the base trigger data for a trigger.

Specified by:
deleteTrigger in interface DriverDelegate

Parameters:
conn - the DB Connection

Returns:
the number of rows deleted

Throws:
SQLException

---

deleteTriggerExtension

protected void deleteTriggerExtension(Connection conn,
                                       TriggerKey triggerKey)
throws SQLException

Throws:
SQLException

---

selectNumTriggersForJob

public int selectNumTriggersForJob(Connection conn,
                                    JobKey jobKey)
throws SQLException

Select the number of triggers associated with a given job.
Specified by:
  selectNumTriggersForJob in interface DriverDelegate

Parameters:
  conn - the DB Connection

Returns:
  the number of triggers for the given job

Throws:
  SQLException

---

**selectJobForTrigger**

```java
public JobDetail selectJobForTrigger(Connection conn,
                                    org.quartz.spi.ClassLoadHelper loadHelper,
                                    TriggerKey triggerKey)
    throws ClassNotFoundException,
            SQLException
```

Select the job to which the trigger is associated.

Specified by:
  selectJobForTrigger in interface DriverDelegate

Parameters:
  conn - the DB Connection

Returns:
  the JobDetail object associated with the given trigger

Throws:
  SQLException
  ClassNotFoundException

---

**selectTriggersForJob**

```java
public List<org.quartz.spi.OperableTrigger> selectTriggersForJob(Connection conn)
    throws SQLException, ClassNotFoundException, IOException
```

Select the triggers for a job

Throws:
  SQLException
  ClassNotFoundException
  IOException

---
Specified by:

selectTriggersForJob in interface DriverDelegate

Parameters:

conn - the DB Connection

Returns:

an array of (@link org.quartz.Trigger) objects associated with a given job.

Throws:

SQLException
JobPersistenceException
ClassNotFoundException
IOException

selectTriggersForCalendar

public List<org.quartz.spi.OperableTrigger> selectTriggersForCalendar

Description copied from interface: DriverDelegate

Select the triggers for a calendar

Specified by:

selectTriggersForCalendar in interface DriverDelegate

Parameters:

conn - the DB Connection
calName - the name of the calendar

Returns:

an array of (@link org.quartz.Trigger) objects associated with the given calendar.

Throws:

SQLException
JobPersistenceException
ClassNotFoundException
IOException

IOException
selectTrigger

public org.quartz.spi.OperableTrigger selectTrigger(Connection conn, TriggerKey triggerKey)
throw SQLException, ClassNotFoundException, IOException

Select a trigger.

Specified by:
selectTrigger in interface DriverDelegate

Parameters:
conn - the DB Connection

Returns:
the Trigger object

Throws:
JobPersistenceException
SQLException
ClassNotFoundException
IOException

selectTriggerJobDataMap

public JobDataMap selectTriggerJobDataMap(Connection conn, String triggerName, String groupName)
throw SQLException, ClassNotFoundException, IOException

Select a trigger's JobDataMap.

Specified by:
selectTriggerJobDataMap in interface DriverDelegate

Parameters:
conn - the DB Connection
triggerName - the name of the trigger
groupName - the group containing the trigger

Returns:
the JobDataMap of the Trigger, never null, but possibly empty.

**Throws:**
- SQLException
- ClassNotFoundException
- IOException

---

### selectTriggerState

```java
public String selectTriggerState(Connection conn, TriggerKey triggerKey) throws SQLException
```

Select a trigger's state value.

**Specified by:**
- `selectTriggerState` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection

**Returns:**
- the Trigger object

**Throws:**
- SQLException

---

### selectTriggerStatus

```java
public TriggerStatus selectTriggerStatus(Connection conn, TriggerKey triggerKey) throws SQLException
```

Select a trigger's status (state & next fire time).

**Specified by:**
- `selectTriggerStatus` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection

**Returns:**
- a `TriggerStatus` object, or null

**Throws:**
- SQLException
selectNumTriggers

```java
public int selectNumTriggers(Connection conn)
    throws SQLException
```

Select the total number of triggers stored.

Specified by:
```
selectNumTriggers in interface DriverDelegate
```

Parameters:
```
conn - the DB Connection
```

Returns:
```
the total number of triggers stored
```

Throws:
```
SQLException
```

selectTriggerGroups

```java
public List<String> selectTriggerGroups(Connection conn)
    throws SQLException
```

Select all of the trigger group names that are stored.

Specified by:
```
selectTriggerGroups in interface DriverDelegate
```

Parameters:
```
conn - the DB Connection
```

Returns:
```
an array of String group names
```

Throws:
```
SQLException
```

selectTriggerGroups

```java
public List<String> selectTriggerGroups(Connection conn,
                                         GroupMatcher<TriggerKey> mat)
    throws SQLException
```
selectTriggersInGroup

public Set<TriggerKey> selectTriggersInGroup(Connection conn, GroupMatcher<TriggerKey> matcher) throws SQLException

Select all of the triggers contained in a given group.

Specified by:
	selctTriggersInGroup in interface DriverDelegate

Parameters:

cconn - the DB Connection
matcher - to evaluate against known triggers

Returns:
a Set of TriggerKeys

Throws:

SQLException

insertPausedTriggerGroup

public int insertPausedTriggerGroup(Connection conn, String groupName) throws SQLException

Specified by:

insertPausedTriggerGroup in interface DriverDelegate

Throws:

SQLException

deletePausedTriggerGroup

public int deletePausedTriggerGroup(Connection conn,
deletePausedTriggerGroup

public int deletePausedTriggerGroup(Connection conn, GroupMatcher<TriggerKey> matcher)
throws SQLException

Specified by:
deletePausedTriggerGroup in interface DriverDelegate
Throws:
SQLException

deleteAllPausedTriggerGroups

public int deleteAllPausedTriggerGroups(Connection conn)
throws SQLException

Specified by:
deleteAllPausedTriggerGroups in interface DriverDelegate
Throws:
SQLException

isTriggerGroupPaused

public boolean isTriggerGroupPaused(Connection conn, String groupName)
throws SQLException

Specified by:
isTriggerGroupPaused in interface DriverDelegate
Throws:
SQLException

**isExistingTriggerGroup**

```java
public boolean isExistingTriggerGroup(Connection conn, String groupName) throws SQLException
```

Specified by:

`isExistingTriggerGroup` in interface `DriverDelegate`

Throws:

`SQLException`

**insertCalendar**

```java
public int insertCalendar(Connection conn, String calendarName, Calendar calendar) throws IOException, SQLException
```

Insert a new calendar.

Specified by:

`insertCalendar` in interface `DriverDelegate`

Parameters:

- `conn` - the DB Connection
- `calendarName` - the name for the new calendar
- `calendar` - the calendar

Returns:

the number of rows inserted

Throws:

- `IOException` - if there were problems serializing the calendar
- `SQLException`

**updateCalendar**

```java
public int updateCalendar(Connection conn,
                          String calendarName, Calendar calendar) throws IOException, SQLException
```

Throws:

- `IOException` - if there were problems serializing the calendar
- `SQLException`
Update a calendar.

**Specified by:**
updateCalendar in interface DriverDelegate

**Parameters:**
- conn - the DB Connection
- calendarName - the name for the new calendar
- calendar - the calendar

**Returns:**
the number of rows updated

**Throws:**
- IOException - if there were problems serializing the calendar
- SQLException

---

calendarExists

public boolean calendarExists(Connection conn, String calendarName)
throws SQLException

Check whether or not a calendar exists.

**Specified by:**
calendarExists in interface DriverDelegate

**Parameters:**
- conn - the DB Connection
- calendarName - the name of the calendar

**Returns:**
true if the trigger exists, false otherwise

**Throws:**
SQLException

---

selectCalendar
public Calendar selectCalendar(Connection conn, String calendarName)
throws ClassNotFoundException, IOException, SQLException

Select a calendar.

Specified by:
   selectCalendar in interface DriverDelegate

Parameters:
   conn - the DB Connection
   calendarName - the name of the calendar

Returns:
   the Calendar

Throws:
   ClassNotFoundException - if a class found during deserialization cannot be found
   IOException - if there were problems deserializing the calendar
   SQLException

calendarIsReferenced

public boolean calendarIsReferenced(Connection conn, String calendarName)
throws SQLException

Check whether or not a calendar is referenced by any triggers.

Specified by:
   calendarIsReferenced in interface DriverDelegate

Parameters:
   conn - the DB Connection
   calendarName - the name of the calendar

Returns:
   true if any triggers reference the calendar, false otherwise

Throws:
   SQLException
**deleteCalendar**

```java
public int deleteCalendar(Connection conn,
                      String calendarName)
    throws SQLException
```

Delete a calendar.

**Specified by:**
- `deleteCalendar` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection
- `calendarName` - the name of the trigger

**Returns:**
- the number of rows deleted

**Throws:**
- `SQLException`

---

**selectNumCalendars**

```java
public int selectNumCalendars(Connection conn)
    throws SQLException
```

Select the total number of calendars stored.

**Specified by:**
- `selectNumCalendars` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection

**Returns:**
- the total number of calendars stored

**Throws:**
- `SQLException`

---

**selectCalendars**

```java
public List<String> selectCalendars(Connection conn)
    throws SQLException
```

---
Select all of the stored calendars.

**Specified by:**

`selectCalendars` in interface `DriverDelegate`

**Parameters:**

- conn - the DB Connection

**Returns:**

- an array of String calendar names

**Throws:**

`SQLException`

---

**selectNextFireTime**

```java
public long selectNextFireTime(Connection conn)
throws SQLException
```

**Deprecated.** *Does not account for misfires.*

Select the next time that a trigger will be fired.

**Specified by:**

`selectNextFireTime` in interface `DriverDelegate`

**Parameters:**

- conn - the DB Connection

**Returns:**

- the next fire time, or 0 if no trigger will be fired

**Throws:**

`SQLException`

---

**selectTriggerForFireTime**

```java
public Key selectTriggerForFireTime(Connection conn, long fireTime)
throws SQLException
```

Select the trigger that will be fired at the given fire time.

**Specified by:**
selectTriggerForFireTime in interface DriverDelegate

**Parameters:**
- conn - the DB Connection
- fireTime - the time that the trigger will be fired

**Returns:**
a Key representing the trigger that will be fired at the given fire time, or null if no trigger will be fired at that time

**Throws:**
SQLException

selectTriggerToAcquire

public List<TriggerKey> selectTriggerToAcquire(Connection conn, long noLaterThan, long noEarlierThan) throws SQLException

Select the next trigger which will fire to fire between the two given timestamps in ascending order of fire time, and then descending by priority.

**Specified by:**
selectTriggerToAcquire in interface DriverDelegate

**Parameters:**
- conn - the DB Connection
- noLaterThan - highest value of getNextFireTime() of the triggers (exclusive)
- noEarlierThan - highest value of getNextFireTime() of the triggers (inclusive)

**Returns:**
A (never null, possibly empty) list of the identifiers (Key objects) of the next triggers to be fired.

**Throws:**
SQLException

insertFiredTrigger

public int insertFiredTrigger(Connection conn, org.quartz.spi.OperableTrigger trigger)
Insert a fired trigger.

**Specified by:**
`insertFiredTrigger` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection
- `trigger` - the trigger
- `state` - the state that the trigger should be stored in

**Returns:**
the number of rows inserted

**Throws:**
`SQLException`

---

**updateFiredTrigger**

```java
public int updateFiredTrigger(Connection conn,
    org.quartz.spi.OperableTrigger trigger,
    String state,
    JobDetail job)
```

Update a fired trigger.

**Specified by:**
`updateFiredTrigger` in interface `DriverDelegate`

**Parameters:**
- `conn` - the DB Connection
- `trigger` - the trigger
- `state` - the state that the trigger should be stored in

**Returns:**
the number of rows inserted

**Throws:**
`SQLException`

---

**selectFiredTriggerRecords**
public List<FiredTriggerRecord> selectFiredTriggerRecords(String triggerName, String groupName) throws SQLException

Select the states of all fired-trigger records for a given trigger, or trigger group if trigger name is null.

Specified by:
    selectFiredTriggerRecords in interface DriverDelegate
Returns:
    a List of FiredTriggerRecord objects.
Throws:
    SQLException

public List<FiredTriggerRecord> selectFiredTriggerRecordsByJob(String triggerName, String groupName) throws SQLException

Select the states of all fired-trigger records for a given job, or job group if job name is null.

Specified by:
    selectFiredTriggerRecordsByJob in interface DriverDelegate
Returns:
    a List of FiredTriggerRecord objects.
Throws:
    SQLException

public List<FiredTriggerRecord> selectInstancesFiredTriggerRecords(String triggerName, String groupName) throws SQLException

Description copied from interface: DriverDelegate
Select the states of all fired-trigger records for a given scheduler instance.

Specified by:

`selectInstancesFiredTriggerRecords` in interface `DriverDelegate`  

Returns:

a List of FiredTriggerRecord objects.

Throws:

`SQLException`  

---

**selectFiredTriggerInstanceNames**

```java
public Set<String> selectFiredTriggerInstanceNames(Connection conn) throws SQLException;
```

Select the distinct instance names of all fired-trigger records.

This is useful when trying to identify orphaned fired triggers (a fired trigger without a scheduler state record.)

Specified by:

`selectFiredTriggerInstanceNames` in interface `DriverDelegate`  

Returns:

a Set of String objects.

Throws:

`SQLException`  

---

**deleteFiredTrigger**

```java
public int deleteFiredTrigger(Connection conn, String entryId) throws SQLException;
```

Delete a fired trigger.

Specified by:

`deleteFiredTrigger` in interface `DriverDelegate`  

Parameters:

conn - the DB Connection
entryId - the fired trigger entry to delete

Returns:
the number of rows deleted

Throws:
SQLException

---

selectJobExecutionCount

public int selectJobExecutionCount(Connection conn,
JobKey jobKey) throws SQLException

Description copied from interface: DriverDelegate

Get the number instances of the identified job currently executing.

Specified by:
selectJobExecutionCount in interface DriverDelegate

Parameters:
conn - the DB Connection

Returns:
the number instances of the identified job currently executing.

Throws:
SQLException

---

insertSchedulerState

public int insertSchedulerState(Connection conn,
String instanceId,
long checkInTime,
long interval) throws SQLException

Description copied from interface: DriverDelegate

Insert a scheduler-instance state record.

Specified by:
insertSchedulerState in interface DriverDelegate
Parameters:
   conn - the DB Connection

Returns:
   the number of inserted rows.

Throws:
   SQLException

---

deleteSchedulerState

```java
public int deleteSchedulerState(Connection conn,
                                    String instanceId)
  throws SQLException
```

Description copied from interface: DriverDelegate

Delete a scheduler-instance state record.

Specified by:
   deleteSchedulerState in interface DriverDelegate

Parameters:
   conn - the DB Connection

Returns:
   the number of deleted rows.

Throws:
   SQLException

---

updateSchedulerState

```java
public int updateSchedulerState(Connection conn,
                                   String instanceId,
                                   long checkInTime)
  throws SQLException
```

Description copied from interface: DriverDelegate

Update a scheduler-instance state record.

Specified by:
   updateSchedulerState in interface DriverDelegate
Parameters:
  conn - the DB Connection

Returns:
  the number of updated rows.

Throws:
  SQLException

selectSchedulerStateRecords

```java
public List<SchedulerStateRecord> selectSchedulerStateRecords(String conn) throws SQLException
```

Description copied from interface: DriverDelegate

A List of all current SchedulerStateRecords.

If instanceId is not null, then only the record for the identified instance will be returned.

Specified by:
  ```javascript
  selectSchedulerStateRecords in interface DriverDelegate
  ```

Parameters:
  conn - the DB Connection

Throws:
  SQLException

rtp

```java
protected final String rtp(String query)
```

Replace the table prefix in a query by replacing any occurrences of "\{0\}" with the table prefix.

Parameters:
  query - the unsubstituted query

Returns:
  the query, with proper table prefix substituted
getSchedulerNameLiteral

protected String getSchedulerNameLiteral()

serializeObject

protected ByteArrayOutputStream serializeObject(Object obj) throws IOException

Create a serialized java.util.ByteArrayOutputStream version of an Object.

Parameters:
obj - the object to serialize

Returns:
the serialized ByteArrayOutputStream

Throws:
IOException - if serialization causes an error

serializeJobData

protected ByteArrayOutputStream serializeJobData(JobDataMap data) throws IOException

Remove the transient data from and then create a serialized java.util.ByteArrayOutputStream version of a JobDataMap.

Parameters:
data - the JobDataMap to serialize

Returns:
the serialized ByteArrayOutputStream

Throws:
IOException - if serialization causes an error

getKeyOfNonSerializableValue
protected Object getgetKeyOfNonSerializableValue(Map<?,?> data)

Find the key of the first non-serializable value in the given Map.

**Returns:**
The key of the first non-serializable value in the given Map or null if all values are serializable.

convertFromProperty

protected Map<?,?> convertFromProperty(Properties properties)
throws IOException

convert the JobDataMap into a list of properties

**Throws:**
IOException

convertToProperty

protected Properties convertToProperty(Map<?,?> data)
throws IOException

convert the JobDataMap into a list of properties

**Throws:**
IOException

ggetObjectFromBlob

protected Object getObjectFromBlob(ResultSet rs, String colName)
throws ClassNotFoundException, IOException, SQLException

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard
JDBC java.sql.Blob operations.

**Parameters:**
- `rs` - the result set, already queued to the correct row
- `colName` - the column name for the BLOB

**Returns:**
- the deserialized Object from the ResultSet BLOB

**Throws:**
- `ClassNotFoundException` - if a class found during deserialization cannot be found
- `IOException` - if deserialization causes an error
- `SQLException`

---

**getJobDataFromBlob**

```java
protected Object getJobDataFromBlob(ResultSet rs, String colName) throws ClassNotFoundException, IOException, SQLException
```

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC java.sql.Blob operations.

**Parameters:**
- `rs` - the result set, already queued to the correct row
- `colName` - the column name for the BLOB

**Returns:**
- the deserialized Object from the ResultSet BLOB

**Throws:**
- `ClassNotFoundException` - if a class found during deserialization cannot be found
- `IOException` - if deserialization causes an error
- `SQLException`

---

**selectPausedTriggerGroups**
public Set<String> selectPausedTriggerGroups(Connection conn) throws SQLException

Specified by: 
selectPausedTriggerGroups in interface DriverDelegate

Throws:
SQLException

See Also:
DriverDelegate.selectPausedTriggerGroups(java.sql.Connection)

closeResultSet

protected static void closeResultSet(ResultSet rs)

Cleanup helper method that closes the given ResultSet while ignoring any errors.

closeStatement

protected static void closeStatement(Statement statement)

Cleanup helper method that closes the given Statement while ignoring any errors.

setBoolean

protected void setBoolean(PreparedStatement ps, 
int index, 
boolean val) 
throws SQLException

Sets the designated parameter to the given Java boolean value. This just wraps PreparedStatement.setBoolean(int, boolean) by default, but it can be overloaded by subclass delegates for databases that don't explicitly support the boolean type.

Throws:
SQLException
**getBoolean**

protected boolean getBoolean(ResultSet rs, String columnName)
    throws SQLException

Retrieves the value of the designated column in the current row as a boolean. This just wraps ResultSet.getBoolean(java.lang.String) by default, but it can be overloaded by subclass delegates for databases that don't explicitly support the boolean type.

**Throws:**
    SQLException

**getBoolean**

protected boolean getBoolean(ResultSet rs, int columnIndex)
    throws SQLException

Retrieves the value of the designated column index in the current row as a boolean. This just wraps ResultSet.getBoolean(java.lang.String) by default, but it can be overloaded by subclass delegates for databases that don't explicitly support the boolean type.

**Throws:**
    SQLException

**setBytes**

protected void setBytes(PreparedStatement ps, int index, ByteArrayOutputStream baos)
    throws SQLException

Sets the designated parameter to the byte array of the given ByteArrayOutputStream. Will set parameter value to null if the ByteArrayOutputStream is null. This just wraps
PreparedStatement.setBytes(int, byte[]) by default, but it can be overloaded by subclass delegates for databases that don't explicitly support storing bytes in this way.

**Throws:**

SQLException
org.quartz.impl.jdbcjobstore Class StdRowLockSemaphore

java.lang.Object
   └ org.quartz.impl.jdbcjobstore.DBSemaphore
       └ org.quartz.impl.jdbcjobstore.StdRowLockSemaphore

All Implemented Interfaces:
   Constants, Semaphore, StdJDBCConstants, TablePrefixAware

public class StdRowLockSemaphore
   extends DBSemaphore

Internal database based lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.

Author:
   jhouse

Field Summary

| static String | INSERT_LOCK |
| static String | SELECT_FOR_LOCK |

Fields inherited from interface
org.quartz.impl.jdbcjobstore.StdJDBCConstants
COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETECALENDAR, DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETEJOBDETAIL, DELETENORECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
Constructor Summary

StdRowLockSemaphore(String tablePrefix, String schedName, String selectWithLockSQL)

Method Summary

protected void executeSQL(Connection conn, String lockName, String expandedSQL, String expandedInsertSQL)

Execute the SQL select for update that will lock the proper database row.

protected String getSelectWithLockSQL()

void setSelectWithLockSQL(String selectWithLockSQL)

Methods inherited from class org.quartz.impl.jdbcjobstore.DBSemaphore
getLog, get_schedName, getSchedulerNameLiteral, getSQL, getTablePrefix, isLockOwner, obtainLock, releaseLock, requiresConnection, setInsertSQL, set_schedName, setSQL, setTablePrefix
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SELECT_FOR_LOCK

public static final String SELECT_FOR.Lock

See Also:
   Constant Field Values

INSERT_LOCK

public static final String INSERT.Lock

See Also:
   Constant Field Values

Constructor Detail

StdRowLockSemaphore

public StdRowLockSemaphore(String tablePrefix,
   String schedName,
   String selectWithLockSQL)

Method Detail

executeSQL

protected void executeSQL(Connection conn,
   String lockName,
   String expandedSQL,
   String expandedInsertSQL)
throws `LockException`

Execute the SQL select for update that will lock the proper database row.

Specified by:
`executeSQL` in class `DBSemaphore`

Throws:
`LockException`

---

**getSelectWithLockSQL**

protected `String` `getSelectWithLockSQL()`

---

**setSelectWithLockSQL**

public void `setSelectWithLockSQL(String selectWithLockSQL)`
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public class SybaseDelegate
extends StdJDBCDelegate

This is a driver delegate for the Sybase database.

Author:
Jeffrey Wescott, jhouse, Ray Case

Field Summary

Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETECALENDAR, DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP, DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE, DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER, INSERTCALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR,
SELECT_CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER,
SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS,
SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_JOB_IN_GROUP, SELECT_MISFIRED_TRIGGERS_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FOR_CALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGER_INSTANCE_IN_STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants
ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL,
COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID,
COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME,
COL_IS_DURABLE, COL_IS_NONCONCurrent, COL_IS_UPDATE_DATA,
COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP,
Constructor Summary

SybaseDelegate(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new SybaseDelegate instance.

SybaseDelegate(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)

Method Summary

protected Object getJobDataFromBlob(ResultSet rs, String colName)

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.

protected Object getObjectFromBlob(ResultSet rs, String colName)

This method should be overridden by any delegate subclasses that need special handling for BLOBs.

protected void setBytes(PreparedStatement ps, int index, ByteArrayOutputStream baos)

Sets the designated parameter to the byte array of the given ByteArrayOutputStream.

Methods inherited from class
<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods of org.quartz.impl.jdbcjobstore.StdJDBCDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet, closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectMisfiredTriggersInState, selectNextFireTime, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates, updateTriggerStatesFromOtherStates</td>
</tr>
</tbody>
</table>
Constructor Detail

SybaseDelegate

public SybaseDelegate(org.slf4j.Logger log,
                       String tablePrefix,
                       String schedName,
                       String instanceId,
                       org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new SybaseDelegate instance.

Parameters:
log - the logger to use during execution
tablePrefix - the prefix of all table names

SybaseDelegate

public SybaseDelegate(org.slf4j.Logger log,
                       String tablePrefix,
                       String schedName,
                       String instanceId,
                       org.quartz.spi.ClassLoadHelper classLoadHelper,
                       Boolean useProperties)

Method Detail

getObjectFromBlob

protected Object getObjectFromBlob(ResultSet rs,
                                    String colName)

throws ClassCastException,
        IOException,
        SQLException

This method should be overridden by any delegate subclasses that need
special handling for BLOBs. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:

**getObjectFromBlob** in class **StdJDBCDelegate**

Parameters:

- **rs** - the result set, already queued to the correct row
- **colName** - the column name for the BLOB

Returns:

the deserialized Object from the ResultSet BLOB

Throws:

- **ClassNotFoundException** - if a class found during deserialization cannot be found
- **IOException** - if deserialization causes an error
- **SQLException**

---

**getJobDataFromBlob**

protected **Object** getJobDataFromBlob(ResultSet **rs**, String **colName**) throws **ClassNotFoundException**, **IOException**, **SQLException**

Description copied from class: **StdJDBCDelegate**

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:

**getJobDataFromBlob** in class **StdJDBCDelegate**

Parameters:

- **rs** - the result set, already queued to the correct row
- **colName** - the column name for the BLOB

Returns:

the deserialized Object from the ResultSet BLOB

Throws:

- **ClassNotFoundException** - if a class found during deserialization
cannot be found

IOException - if deserialization causes an error

SQLException

setBytes

protected void setBytes(PreparedStatement ps,
                        int index,
                        ByteArrayOutputStream baos)
throws SQLException

Sets the designated parameter to the byte array of the given
ByteArrayOutputStream. Will set parameter value to null if the
ByteArrayOutputStream is null. This just wraps
PreparedStatement.setBytes(int, byte[]) by default, but it can be
overloaded by subclass delegates for databases that don't explicitly support
storing bytes in this way.

Overrides:

setBytes in class StdJDBCDelegate

Throws:

SQLException

Copyright 2001-2011, Terracotta, Inc.
Interface TablePrefixAware

All Known Implementing Classes:
   DBSemaphore, StdRowLockSemaphore, UpdateLockRowSemaphore

public interface TablePrefixAware

Interface for Quartz objects that need to know what the table prefix of the tables used by a JDBC JobStore is.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>void setSchedName(String schedName)</td>
<td></td>
</tr>
<tr>
<td>void setTablePrefix(String tablePrefix)</td>
<td></td>
</tr>
</tbody>
</table>

Method Detail

setTablePrefix

void setTablePrefix(String tablePrefix)

setSchedName

void setSchedName(String schedName)
org.quartz.impl.jdbcjobstore Interface TriggerPersistenceDelegate

All Known Implementing Classes:
  CalendarIntervalTriggerPersistenceDelegate,
  CronTriggerPersistenceDelegate,
  SimplePropertiesTriggerPersistenceDelegateSupport,
  SimpleTriggerPersistenceDelegate

public interface TriggerPersistenceDelegate

An interface which provides an implementation for storing a particular type of Trigger's extended properties.

Author:
  jhouse

Nested Class Summary

<table>
<thead>
<tr>
<th>static class</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>boolean canHandleTriggerType(org.quartz.spi.OperableTrigger trigger)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int deleteExtendedTriggerProperty(TriggerKey triggerKey)</td>
</tr>
<tr>
<td>String getHandledTriggerTypeDiscriminator()</td>
</tr>
<tr>
<td>void initialize(String tablePrefix, org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail)</td>
</tr>
</tbody>
</table>
### Method Detail

**initialize**

```java
void initialize(String tablePrefix, String schedulerName)
```

**canHandleTriggerType**

```java
boolean canHandleTriggerType(org.quartz.spi.OperableTrigger trigger)
```

**getHandledTriggerTypeDiscriminator**

```java
String getHandledTriggerTypeDiscriminator()
```

**insertExtendedTriggerProperties**

```java
int insertExtendedTriggerProperties(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)
```

Throws:

```java
SQLException
```
IOException

updateExtendedTriggerProperties

```java
int updateExtendedTriggerProperties(
    Connection conn,
    org.quartz.spi.OperableTrigger t,
    String state,
    JobDetail jobDetail)
throws SQLException,
    IOException
```

Throws:
- SQLException
- IOException

deleteExtendedTriggerProperties

```java
int deleteExtendedTriggerProperties(
    Connection conn,
    TriggerKey triggerKey)
throws SQLException
```

Throws:
- SQLException

loadExtendedTriggerProperties

```java
TriggerPersistenceDelegate.TriggerPropertyBundle loadExtendedTriggerProperties
```

Throws:
- SQLException

Copyright 2001-2011, Terracotta, Inc.
public static class TriggerPersistenceDelegate.TriggerPropertyBundle extends Object

Constructor Summary

TriggerPersistenceDelegate.TriggerPropertyBundle(ScheduleBuilder sb, String[] statePropertyNames, Object[] statePropertyValues)

Method Summary

<table>
<thead>
<tr>
<th>ScheduleBuilder</th>
<th>getScheduleBuilder()</th>
</tr>
</thead>
<tbody>
<tr>
<td>String[]</td>
<td>getStatePropertyNames()</td>
</tr>
<tr>
<td>Object[]</td>
<td>getStatePropertyValues()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail
public TriggerPersistenceDelegate.TriggerPropertyBundle(ScheduleBuilder.ScheduleBuilder, String[] statePropertyNames, Object[] statePropertyValues)

Method Detail

getScheduleBuilder

class ScheduleBuilder getScheduleBuilder()

getStatePropertyNames

public String[] getStatePropertyNames()

getStatePropertyValues

public Object[] getStatePropertyValues()
Class TriggerStatus

public class TriggerStatus
extends Object

Object representing a job or trigger key.

Author:
James House

Constructor Summary

| TriggerStatus(String status, Date nextFireTime) |
| Construct a new TriggerStatus with the status name and nextFireTime. |

Method Summary

| JobKey | getJobKey() |
| TriggerKey | getKey() |
| Date | getNextFireTime() |
| String | getStatus() |
| void | setJobKey(JobKey jobKey) |
| void | setKey(TriggerKey key) |
| String | toString() |

Return the string representation of the TriggerStatus.
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TriggerStatus

public TriggerStatus(String status, Date nextFireTime)

Construct a new TriggerStatus with the status name and nextFireTime.

Parameters:
status - the trigger's status
nextFireTime - the next time the trigger will fire

Method Detail

getJobKey

public JobKey getJobKey()

setJobKey

public void setJobKey(JobKey jobKey)

getKey

public TriggerKey getKey()

setKey
public void setKey(TriggerKey key)

getStatus

public String getStatus()

Get the name portion of the key.

Returns:
the name

getNextFireTime

public Date getNextFireTime()

Get the group portion of the key.

Returns:
the group

toString

public String toString()

Return the string representation of the TriggerStatus.

Overrides:
toString in class Object
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
</tr>
</tbody>
</table>
Class UpdateLockRowSemaphore

java.lang.Object
   org.quartz.impl.jdbcjobstore.DBSemaphore
      org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore

All Implemented Interfaces:
   Constants, Semaphore, StdJDBCConstants, TablePrefixAware

public class UpdateLockRowSemaphore
extends DBSemaphore

Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.

Note: This Semaphore implementation is useful for databases that do not support row locking via "SELECT FOR UPDATE" type syntax, for example Microsoft SQLServer (MSSQL).

Field Summary

| static String | INSERT_LOCK |
| static String | UPDATE_FOR_LOCK |

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS,
DELETE_ALL.Calendar, DELETE_ALL.CRON TRIGGERS,
DELETE_ALL_JOBDETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS,
DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS,
DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETECALENDAR,
DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS,
DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL,
DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP,
DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE, DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER, INSERTCALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER, INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP, INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER, SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CAレンNDAR, SELECT_CAレンNDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER, SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP, SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS, SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP, SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE, SELECT_INSTANCES_FIRED_TRIGGERS, SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL, SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE, SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS, SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP, SELECT_MISFIRED_TRIGGERS, SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE, SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME, SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS, SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB, SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCED_CAレンNDAR, SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES, SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA, SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME, SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED, SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS, SELECT_TRIGGERS_FOR_CAレンNDAR, SELECT_TRIGGERS_FOR_JOB, SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE, TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CAレンNDAR, UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER, UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA, UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES, UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE, UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER, UPDATE_TRIGGER_GROUP_STATE_FROM_STATE, UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA, UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE, UPDATE_TRIGGER_STATE_FROM_STATES, UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants

ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB, COL_CAレンNDAR, COL_CAレンNDAR_NAME, COL_CHECKIN_INTERVAL, COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID,
Constructor Summary

UpdateLockRowSemaphore()
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

UPDATE_FOR_LOCK

class public static final String UPDATE_FOR_LOCK

See Also:
See Also: Constant Field Values

INSERT_LOCK

class public static final String INSERT_LOCK

See Also:
See Also: Constant Field Values

Constructor Detail

UpdateLockRowSemaphore

class public UpdateLockRowSemaphore()

Method Detail

executeSQL

protected void executeSQL(Connection conn,
String lockName,
String expandedSQL,
String expandedInsertSQL)
throws LockException

Execute the SQL select for update that will lock the proper database row.
Specified by:
executeSQL in class DBSemaphore

Throws:
LockException

getUpdateLockRowSQL

protected String getUpdateLockRowSQL()

setUpdateLockRowSQL

public void setUpdateLockRowSQL(String updateLockRowSQL)
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
Object.  

---

public final class Util

extends Object

This class contains utility functions for use in all delegate classes.

Author:

Jeffrey Wescott

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td>closeResultSet(ResultSet rs)</td>
<td>Cleanup helper method that closes the given ResultSet while ignoring any errors.</td>
</tr>
<tr>
<td>static void</td>
<td>closeStatement(Statement statement)</td>
<td>Cleanup helper method that closes the given Statement while ignoring any errors.</td>
</tr>
<tr>
<td>static String</td>
<td>rtp(String query, String tablePrefix, String schedNameLiteral)</td>
<td>Replace the table prefix in a query by replacing any occurrences of &quot;{}&quot; with the table prefix.</td>
</tr>
<tr>
<td>static void</td>
<td>setBeanProps(Object obj, String[] propNames, Object[] propValues)</td>
<td></td>
</tr>
</tbody>
</table>

---

Methods inherited from class java.lang.Object  

clone, equals, finalize,getClass,hashCode,notify,notifyAll,toString,wait,wait,wait
<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
</table>

**rtp**

```java
public static String rtp(String query,
            String tablePrefix,
            String schedNameLiteral)
```

Replace the table prefix in a query by replacing any occurrences of "\{0\}" with the table prefix.

**Parameters:**
- query - the unsubstituted query
- tablePrefix - the table prefix

**Returns:**
- the query, with proper table prefix substituted

---

**closeResultSet**

```java
public static void closeResultSet(ResultSet rs)
```

Cleanup helper method that closes the given ResultSet while ignoring any errors.

---

**closeStatement**

```java
public static void closeStatement(Statement statement)
```

Cleanup helper method that closes the given Statement while ignoring any errors.

---

**setBeanProps**

```java
public static void setBeanProps(Object obj,
                    String[] propNames,
                    Object[] propValues)
```


Throws:

JobPersistenceException
org.quartz.impl.jdbcjobstore  Class WebLogicDelegate

java.lang.Object
  └ org.quartz.impl.jdbcjobstore.StdJDBCDelegate
      └ org.quartz.impl.jdbcjobstore.WebLogicDelegate

All Implemented Interfaces:
   Constants, DriverDelegate, StdJDBCConstants

public class WebLogicDelegate
extends StdJDBCDelegate

This is a driver delegate for the WebLogic JDBC driver.

Author:
   Jeffrey Wescott

See Also:
   WebLogicOracleDelegate

Field Summary

Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants
cOUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOB_DETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETE_CALENDAR, DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP, DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE,
DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER,
INSERT Calendars, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER,
INSERT JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP,
INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER,
SELECT_SCHEDULER_STATE, SELECT_CALENDAR, SELECT_CALENDAR_EXISTENCE,
SELECT CALENDARS, SELECT_CRON_TRIGGER,
SELECT FIRED_TRIGGER, SELECT FIRED_TRIGGER_GROUP,
SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS,
SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP,
SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE,
SELECT_INSTANCES_FIRED_TRIGGERS,
SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL,
SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT_JOBS_IN_GROUP, SELECT_JOBS_IN_GROUP,
SELECT_MISFIRED_TRIGGERS,
SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE,
SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME,
SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS,
SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCED_CALENDAR,
SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES,
SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS_FOR_CALENDAR, SELECT_TRIGGERS_FOR_JOB,
SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
SELECT_TRIGGERS_IN_STATE, SELECT_TRIGGERS_IN_STATE,
Constructor Summary

**WebLogicDelegate**(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new WebLogicDelegate instance.

**WebLogicDelegate**(org.slf4j.Logger log, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)

Create new WebLogicDelegate instance.

Method Summary

```java
protected Object getJobDataFromBlob(ResultSet rs, String colName)

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.
```

```java
protected Object getObjectFromBlob(ResultSet rs, String colName)

This method should be overridden by any delegate subclasses that need special handling for BLOBs.
```

Methods inherited from class

org.quartz.impl.jdbcjobstore.StdJDBCDelegate

addDefaultTriggerPersistenceDelegates,
addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet, closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTriggers, deleteJobDetail, deletePausedTriggerGroup, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertCalendar, insertFiredTrigger, insertJobDetail, insertPausedTriggerGroup, insertSchedulerState, insertTrigger, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectMisfiredTriggersInState, selectNextFireTime, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs, selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateCalendar, updateFiredTrigger, updateJobData, updateJobDetail, updateSchedulerState, updateTrigger, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Detail

WebLogicDelegate

public WebLogicDelegate(org.slf4j.Logger log,
                        String tablePrefix,
                        String schedName,
                        String instanceId,
                        org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new WebLogicDelegate instance.

Parameters:
  log - the logger to use during execution
  tablePrefix - the prefix of all table names

WebLogicDelegate

public WebLogicDelegate(org.slf4j.Logger log,
                        String tablePrefix,
                        String schedName,
                        String instanceId,
                        org.quartz.spi.ClassLoadHelper classLoadHelper,
                        Boolean useProperties)

Create new WebLogicDelegate instance.

Parameters:
  log - the logger to use during execution
  tablePrefix - the prefix of all table names
  useProperties - use java.util.Properties for storage

Method Detail

getObjectFromBlob
protected Object getObjectFromBlob(ResultSet rs, String colName)
throws ClassNotFoundException, IOException, SQLException

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:
  getObjectFromBlob in class StdJDBCDelegate

Parameters:
  rs - the result set, already queued to the correct row
  colName - the column name for the BLOB

Returns:
  the deserialized Object from the ResultSet BLOB

Throws:
  ClassNotFoundException - if a class found during deserialization cannot be found
  IOException - if deserialization causes an error
  SQLException

---

getJobDataFromBlob

protected Object getJobDataFromBlob(ResultSet rs, String colName)
throws ClassNotFoundException, IOException, SQLException

Description copied from class: StdJDBCDelegate

This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:
  getJobDataFromBlob in class StdJDBCDelegate

Parameters:
rs - the result set, already queued to the correct row
colName - the column name for the BLOB

**Returns:**
the deserialized Object from the ResultSet BLOB

**Throws:**
- [ClassNotFoundException](#) - if a class found during deserialization cannot be found
- [IOException](#) - if deserialization causes an error
- [SQLException](#)
Uses of Class
org.quartz.impl.jdbcjobstore.AttributeRestoringConnectionInvocationHandler

No usage of
org.quartz.impl.jdbcjobstore.AttributeRestoringConnectionInvocationHandler

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.CalendarIntervalTrigger

No usage of
org.quartz.impl.jdbcjobstore.CalendarIntervalTriggerPersistenceDelegate

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.CloudscapeDelegate

No usage of org.quartz.impl.jdbcjobstore.CloudscapeDelegate

Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

org.quartz.impl.jdbcjobstore.Constants

## Packages that use Constants

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore.oracle</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore.oracle.weblogic</td>
</tr>
</tbody>
</table>

## Uses of Constants in org.quartz.impl.jdbcjobstore

## Subinterfaces of Constants in org.quartz.impl.jdbcjobstore

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdJDBCConstants</td>
<td>This interface extends Constants to include the query string constants in use by the StdJDBCDelegate class.</td>
</tr>
</tbody>
</table>

## Classes in org.quartz.impl.jdbcjobstore that implement Constants

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalendarIntervalTriggerPersistenceDelegate</td>
<td></td>
</tr>
<tr>
<td>CloudscapeDelegate</td>
<td><strong>Deprecated.</strong> Use the StdJDBCDelegate for latest versions of Derby</td>
</tr>
<tr>
<td>CronTriggerPersistenceDelegate</td>
<td></td>
</tr>
<tr>
<td>DB2v6Delegate</td>
<td>Quartz JDBC delegate for DB2 v6 databases.</td>
</tr>
<tr>
<td>DB2v7Delegate</td>
<td>Quartz JDBC delegate for DB2 v7 databases.</td>
</tr>
<tr>
<td>DB2v8Delegate</td>
<td>Quartz JDBC delegate for DB2 v8 databases.</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>DBSemaphore</code></td>
<td>Base class for database based lock handlers for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td><code>HSQldbDelegate</code></td>
<td>This is a driver delegate for the HSQLDB database.</td>
</tr>
<tr>
<td><code>JobStoreCMT</code></td>
<td>JobStoreCMT is meant to be used in an application-server environment that provides container-managed-transactions.</td>
</tr>
<tr>
<td><code>JobStoreSupport</code></td>
<td>Contains base functionality for JDBC-based JobStore implementations.</td>
</tr>
<tr>
<td><code>JobStoreTX</code></td>
<td>JobStoreTX is meant to be used in a standalone environment.</td>
</tr>
<tr>
<td><code>MSSQLDelegate</code></td>
<td>This is a driver delegate for the MSSQL JDBC driver.</td>
</tr>
<tr>
<td><code>PointbaseDelegate</code></td>
<td>This is a driver delegate for the Pointbase JDBC driver.</td>
</tr>
<tr>
<td><code>PostgreSQLDelegate</code></td>
<td>This is a driver delegate for the PostgreSQL JDBC driver.</td>
</tr>
<tr>
<td><code>SimplePropertiesTriggerPersistenceDelegateSupport</code></td>
<td>A base implementation of <code>TriggerPersistenceDelegate</code> that persists trigger fields in the &quot;QRTZ_SIMPROP_TRIGGERS&quot; table.</td>
</tr>
<tr>
<td><code>SimpleTriggerPersistenceDelegate</code></td>
<td></td>
</tr>
<tr>
<td><code>StdJDBCDelegate</code></td>
<td>This is meant to be an abstract base class for most, if not all, <code>DriverDelegate</code> implementations.</td>
</tr>
<tr>
<td><code>StdRowLockSemaphore</code></td>
<td>Internal database based lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td><code>SybaseDelegate</code></td>
<td>This is a driver delegate for the Sybase database.</td>
</tr>
<tr>
<td><code>UpdateLockRowSemaphore</code></td>
<td></td>
</tr>
</tbody>
</table>
Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.

**class** WebLogicDelegate

This is a driver delegate for the WebLogic JDBC driver.

---

**Uses of Constants in**

**org.quartz.impl.jdbcjobstore.oracle**

---

**Classes in** org.quartz.impl.jdbcjobstore.oracle that implement **Constants**

**class** OracleDelegate

This is a driver delegate for the Oracle 10 and 11 database.

---

**Uses of Constants in**

**org.quartz.impl.jdbcjobstore.oracle.weblogic**

---

**Classes in** org.quartz.impl.jdbcjobstore.oracle.weblogic that implement **Constants**

**class** WebLogicOracleDelegate

Handle Blobs correctly when Oracle is being used inside of Weblogic 8.1, as discussed at:

http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705
Uses of Class
org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate

No usage of org.quartz.impl.jdbcjobstore.CronTriggerPersistenceDelegate
Uses of Class
org.quartz.impl.jdbcjobstore.DB2v6Delegate

No usage of org.quartz.impl.jdbcjobstore.DB2v6Delegate

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.DB2v7Delegate

No usage of org.quartz.impl.jdbcjobstore.DB2v7Delegate

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.DB2v8Delegate

No usage of org.quartz.impl.jdbcjobstore.DB2v8Delegate

Copyright 2001-2011, Terracotta, Inc.
Packages that use **DBSemaphore**

- **org.quartz.impl.jdbcjobstore**

**Uses of DBSemaphore in org.quartz.impl.jdbcjobstore**

<table>
<thead>
<tr>
<th>Subclasses of DBSemaphore in org.quartz.impl.jdbcjobstore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong> StdRowLockSemaphore</td>
</tr>
<tr>
<td>Internal database based lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td><strong>class</strong> UpdateLockRowSemaphore</td>
</tr>
<tr>
<td>Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

**org.quartz.impl.jdbcjobstore.DriverDelegate**

## Packages that use **DriverDelegate**

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore.oracle</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore.oracle.weblogic</td>
</tr>
</tbody>
</table>

## Uses of **DriverDelegate** in **org.quartz.impl.jdbcjobstore**

## Classes in **org.quartz.impl.jdbcjobstore** that implement **DriverDelegate**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CloudscapeDelegate</strong></td>
<td><strong>Deprecated. Use the StdJDBCDelegate for latest versions of Derby</strong></td>
</tr>
<tr>
<td><strong>DB2v6Delegate</strong></td>
<td>Quartz JDBC delegate for DB2 v6 databases.</td>
</tr>
<tr>
<td><strong>DB2v7Delegate</strong></td>
<td>Quartz JDBC delegate for DB2 v7 databases.</td>
</tr>
<tr>
<td><strong>DB2v8Delegate</strong></td>
<td>Quartz JDBC delegate for DB2 v8 databases.</td>
</tr>
<tr>
<td><strong>HSQLDBDelegate</strong></td>
<td>This is a driver delegate for the HSQLDB database.</td>
</tr>
<tr>
<td><strong>MSSQLDelegate</strong></td>
<td>This is a driver delegate for the MSSQL JDBC driver.</td>
</tr>
<tr>
<td><strong>PointbaseDelegate</strong></td>
<td>This is a driver delegate for the Pointbase JDBC driver.</td>
</tr>
<tr>
<td><strong>PostgreSQLDelegate</strong></td>
<td>This is a driver delegate for the PostgreSQL JDBC driver.</td>
</tr>
<tr>
<td><strong>PostgreSQLDelegate</strong></td>
<td>This is a driver delegate for the PostgreSQL JDBC driver.</td>
</tr>
</tbody>
</table>


StdJDBCDelegate
   This is meant to be an abstract base class for most, if not all, DriverDelegate implementations.

class SybaseDelegate
   This is a driver delegate for the Sybase database.

class WebLogicDelegate
   This is a driver delegate for the WebLogic JDBC driver.

Fields in org.quartz.impl.jdbcjobstore with type parameters of type DriverDelegate

<table>
<thead>
<tr>
<th>protected</th>
<th>JobStoreSupport.delegateClass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class&lt;? extends DriverDelegate&gt;</td>
<td>JobStoreSupport.getDelegate()</td>
</tr>
</tbody>
</table>

Methods in org.quartz.impl.jdbcjobstore that return DriverDelegate

<table>
<thead>
<tr>
<th>protected</th>
<th>JobStoreSupport.getDelegate()</th>
</tr>
</thead>
<tbody>
<tr>
<td>DriverDelegate</td>
<td>Get the driver delegate for DB operations.</td>
</tr>
</tbody>
</table>

Uses of DriverDelegate in org.quartz.impl.jdbcjobstore.oracle

Classes in org.quartz.impl.jdbcjobstore.oracle that implement DriverDelegate

<table>
<thead>
<tr>
<th>class</th>
<th>OracleDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This is a driver delegate for the Oracle 10 and 11 database.</td>
</tr>
</tbody>
</table>

Uses of DriverDelegate in org.quartz.impl.jdbcjobstore.oracle.weblogic
Classes in `org.quartz.impl.jdbcjobstore.oracle.weblogic` that implement `DriverDelegate`:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WebLogicOracleDelegate</code></td>
<td>Handle Blobs correctly when Oracle is being used inside of Weblogic 8.1, as discussed at: <a href="http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705">http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705</a></td>
</tr>
</tbody>
</table>
# Uses of Class

**org.quartz.impl.jdbcjobstore.FiredTriggerRecord**

## Packages that use FiredTriggerRecord

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
</tr>
</tbody>
</table>

## Uses of FiredTriggerRecord in

**org.quartz.impl.jdbcjobstore**

## Methods in org.quartz.impl.jdbcjobstore that return types with arguments of FiredTriggerRecord

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdJDBCDelegate.selectFiredTriggerRecords(String triggerName, String groupName)</td>
<td>Select the states of all fired-trigger records for a given group if trigger name is null.</td>
</tr>
<tr>
<td>DriverDelegate.selectFiredTriggerRecords(String triggerName, String groupName)</td>
<td>Select the states of all fired-trigger records for a given group if trigger name is null.</td>
</tr>
<tr>
<td>StdJDBCDelegate.selectFiredTriggerRecordsByJob(String jobName, String groupName)</td>
<td>Select the states of all fired-trigger records for a given job if job name is null.</td>
</tr>
<tr>
<td>DriverDelegate.selectFiredTriggerRecordsByJob(String jobName, String groupName)</td>
<td>Select the states of all fired-trigger records for a given job if job name is null.</td>
</tr>
<tr>
<td>StdJDBCDelegate.selectInstancesFiredTriggerRecords(String instanceName)</td>
<td></td>
</tr>
<tr>
<td>DriverDelegate.selectInstancesFiredTriggerRecords(String instanceName)</td>
<td></td>
</tr>
</tbody>
</table>
Select the states of all fired-trigger records for a given instance.
Uses of Class
org.quartz.impl.jdbcjobstore.HSQLDBDelegate

No usage of org.quartz.impl.jdbcjobstore.HSQLDBDelegate

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.InvalidConfigurationException

Packages that use InvalidConfigurationException
org.quartz.impl.jdbcjobstore

Uses of InvalidConfigurationException in org.quartz.impl.jdbcjobstore

Methods in org.quartz.impl.jdbcjobstore that throw InvalidConfigurationException

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void JobStoreSupport.setDriverDelegateClass(String delegateClassName)</td>
<td>Set the JDBC driver delegate class.</td>
</tr>
<tr>
<td>void JobStoreSupport.setDriverDelegateInitString(String delegateInit)</td>
<td>Set the JDBC driver delegate's initialization string.</td>
</tr>
</tbody>
</table>

Overview  Package  Class  Tree  Deprecated  Index  Help
PREV  NEXT  FRAMES  NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.JobStoreCMT

No usage of org.quartz.impl.jdbcjobstore.JobStoreCMT

Copyright 2001-2011, Terracotta, Inc.
## Uses of Class org.quartz.impl.jdbcjobstore.JobStoreSupport

### Packages that use JobStoreSupport

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
</tr>
</tbody>
</table>

### Uses of JobStoreSupport in org.quartz.impl.jdbcjobstore

### Subclasses of JobStoreSupport in org.quartz.impl.jdbcjobstore

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobStoreCMT</td>
</tr>
<tr>
<td>JobStoreTX</td>
</tr>
</tbody>
</table>

- **JobStoreCMT**
  
  JobStoreCMT is meant to be used in an application-server environment that provides container-managed-transactions.

- **JobStoreTX**
  
  JobStoreTX is meant to be used in a standalone environment.

---

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Uses of Class


Packages that use
JobStoreSupport.RecoverMisfiredJobsResult
org.quartz.impl.jdbcjobstore

Uses of JobStoreSupport.RecoverMisfiredJobsResult in org.quartz.impl.jdbcjobstore

Fields in org.quartz.impl.jdbcjobstore declared as
JobStoreSupport.RecoverMisfiredJobsResult


Methods in org.quartz.impl.jdbcjobstore that return
JobStoreSupport.RecoverMisfiredJobsResult

| protected JobStoreSupport.RecoverMisfiredJobsResult | JobStoreSupport.doRecoverMisfires() |
| protected JobStoreSupport.RecoverMisfiredJobsResult | JobStoreSupport.recoverMisfiredJobs(boolean recovering) |

Copyright 2001-2011, Terracotta, Inc.
### Uses of Interface

**org.quartz.impl.jdbcjobstore.JobStoreSupport.TransactionCallback**

<table>
<thead>
<tr>
<th>Packages that use</th>
<th>org.quartz.impl.jdbcjobstore</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Packages that use</th>
<th>org.quartz.impl.jdbcjobstore</th>
</tr>
</thead>
</table>

### Uses of **JobStoreSupport.TransactionCallback** in **org.quartz.impl.jdbcjobstore**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.impl.jdbcjobstore</strong> with parameters of type <strong>JobStoreSupport.TransactionCallback</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>protected abstract object</td>
</tr>
<tr>
<td>Execute the given callback having aquired the given lock.</td>
</tr>
<tr>
<td>protected object</td>
</tr>
<tr>
<td>Execute the given callback having optionally aquired the given lock.</td>
</tr>
<tr>
<td>protected object</td>
</tr>
<tr>
<td>Execute the given callback having optionally aquired the given lock.</td>
</tr>
<tr>
<td>protected object</td>
</tr>
<tr>
<td>Execute the given callback having optionally aquired the given lock.</td>
</tr>
<tr>
<td>object</td>
</tr>
<tr>
<td>Execute the given callback in a transaction.</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
## Uses of `JobStoreSupport.VoidTransactionCallback` in `org.quartz.impl.jdbcjobstore`

### Packages that use `JobStoreSupport.VoidTransactionCallback`
- `org.quartz.impl.jdbcjobstore`

### Methods in `org.quartz.impl.jdbcjobstore` with parameters of type `JobStoreSupport.VoidTransactionCallback`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected void JobStoreSupport.executeInLock(String lockName, JobStoreSupport.VoidTransactionCallback txCallback)</code></td>
<td>Execute the given callback having aquired the given lock.</td>
</tr>
<tr>
<td><code>protected void JobStoreSupport.executeInNonManagedTXLock(String lockName, JobStoreSupport.VoidTransactionCallback txCallback)</code></td>
<td>Execute the given callback having optionally aquired the given lock.</td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.impl.jdbcjobstore.JobStoreTX

No usage of org.quartz.impl.jdbcjobstore.JobStoreTX

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore

No usage of org.quartz.impl.jdbcjobstore.JTANonClusteredSemaphore

Copyright 2001-2011, Terracotta, Inc.
## Uses of Class

org.quartz.impl.jdbcjobstore.LockException

### Packages that use **LockException**

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.impl.jdbcjobstore</code></td>
</tr>
</tbody>
</table>

### Uses of **LockException** in

org.quartz.impl.jdbcjobstore

### Methods in **org.quartz.impl.jdbcjobstore** that throw **LockException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected abstract void DBSemaphore.executeSQL(Connection conn, String lockName, String expandedSQL, String expandedInsertSQL)</code></td>
<td>Execute the SQL that will lock the proper database row.</td>
</tr>
<tr>
<td><code>protected void StdRowLockSemaphore.executeSQL(Connection conn, String lockName, String expandedSQL, String expandedInsertSQL)</code></td>
<td>Execute the SQL select for update that will lock the proper database row.</td>
</tr>
<tr>
<td><code>protected void UpdateLockRowSemaphore.executeSQL(Connection conn, String lockName, String expandedSQL, String expandedInsertSQL)</code></td>
<td>Execute the SQL select for update that will lock the proper database row.</td>
</tr>
<tr>
<td><code>protected Transaction JTANonClusteredSemaphore.getTransaction()</code></td>
<td>Helper method to get the current Transaction from the TransactionManager in JNDI.</td>
</tr>
<tr>
<td><code>boolean Semaphore.isLockOwner(Connection conn, String lockName)</code></td>
<td>Determine whether the calling thread owns a lock on the identified resource.</td>
</tr>
<tr>
<td><code>boolean DBSemaphore.obtainLock(Connection conn, String lockName)</code></td>
<td>Grants a lock on the identified resource to the calling thread (blocking until it is available).</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>boolean JTANonClusteredSemaphore. obtainLock(Connection conn, String lockName)</td>
<td>Grants a lock on the identified resource to the calling thread (blocking until it is available).</td>
</tr>
<tr>
<td>boolean Semaphore.obtainLock(Connection conn, String lockName)</td>
<td>Grants a lock on the identified resource to the calling thread (blocking until it is available).</td>
</tr>
<tr>
<td>void JTANonClusteredSemaphore. releaseLock(Connection conn, String lockName)</td>
<td>Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.</td>
</tr>
<tr>
<td>void Semaphore.releaseLock(Connection conn, String lockName)</td>
<td>Release the lock on the identified resource if it is held by the calling thread.</td>
</tr>
<tr>
<td>protected void JTANonClusteredSemaphore.releaseLock(String lockName, boolean fromSynchronization)</td>
<td>Release the lock on the identified resource if it is held by the calling thread, unless currently in a JTA transaction.</td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.impl.jdbcjobstore.MSSQLDelegate

No usage of org.quartz.impl.jdbcjobstore.MSSQLDelegate

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.impl.jdbcjobstore.NoSuchDelegateException**

## Packages that use **NoSuchDelegateException**

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
</tr>
</tbody>
</table>

## Uses of **NoSuchDelegateException** in

**org.quartz.impl.jdbcjobstore**

## Methods in **org.quartz.impl.jdbcjobstore** that throw **NoSuchDelegateException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected DriverDelegate getDelegate()</code></td>
<td>Get the driver delegate for DB operations.</td>
</tr>
<tr>
<td><code>void StdJDBCDelegate.initialize(String initString)</code></td>
<td>initStrings are of the format: settingName=settingValue</td>
</tr>
<tr>
<td><code>void DriverDelegate.initialize(String initString)</code></td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.PointbaseDelegate

No usage of org.quartz.impl.jdbcjobstore.PointbaseDelegate

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.PostgreSQLDelegate

No usage of org.quartz.impl.jdbcjobstore.PostgreSQLDelegate

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.SchedulerStateRecord

Packages that use **SchedulerStateRecord**

org.quartz.impl.jdbcjobstore

---

Uses of **SchedulerStateRecord** in
org.quartz.impl.jdbcjobstore

---

Methods in **org.quartz.impl.jdbcjobstore** that return types with arguments of **SchedulerStateRecord**

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protected</strong></td>
<td>JobStoreSupport.clusterCheckIn</td>
<td>Connection conn</td>
</tr>
<tr>
<td><strong>protected</strong></td>
<td>JobStoreSupport.findFailedInstances</td>
<td>Connection conn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Get a list of all scheduler instances in the cluster that may have failed.</td>
</tr>
<tr>
<td></td>
<td>StdJDBCDelegate.selectSchedulerStateRecords</td>
<td>Connection conn, String instanceId</td>
</tr>
<tr>
<td></td>
<td>DriverDelegate.selectSchedulerStateRecords</td>
<td>Connection conn, String instanceId</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A List of all current SchedulerStateRecords.</td>
</tr>
</tbody>
</table>

---

Methods in **org.quartz.impl.jdbcjobstore** with parameters of type **SchedulerStateRecord**

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protected</strong></td>
<td>JobStoreSupport.calcFailedIfAfter</td>
<td>SchedulerStateRecord rec</td>
</tr>
</tbody>
</table>

---

Method parameters in **org.quartz.impl.jdbcjobstore** with type arguments
of type **SchedulerStateRecord**

```java
protected void JobStoreSupport.clusterRecover(Connection conn,
      List<SchedulerStateRecord> failedInstances)
```

Overview  Package  Class  Tree  Deprecated  Index  Help

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.impl.jdbcjobstore.Semaphore

Packages that use Semaphore
org.quartz.impl.jdbcjobstore

Uses of Semaphore in org.quartz.impl.jdbcjobstore

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSemaphore</td>
<td>Base class for database based lock handlers for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td>JTANonClusteredSemaphore</td>
<td>Provides in memory thread/resource locking that is JTA Transaction aware.</td>
</tr>
<tr>
<td>SimpleSemaphore</td>
<td>Internal in-memory lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td>StdRowLockSemaphore</td>
<td>Internal database based lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td>UpdateLockRowSemaphore</td>
<td>Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.</td>
</tr>
</tbody>
</table>

Methods in org.quartz.impl.jdbcjobstore that return Semaphore

protected Semaphore JobStoreSupport.getLockHandler()
Methods in `org.quartz.impl.jdbcjobstore` with parameters of type `Semaphore`

```java
void JobStoreSupport.setLockHandler(Semaphore lockHandler)
```
Uses of Class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerPersistenceDelegateSupport

Packages that use
SimplePropertiesTriggerPersistenceDelegateSupport
in org.quartz.impl.jdbcjobstore

Uses of
SimplePropertiesTriggerPersistenceDelegateSupport
in org.quartz.impl.jdbcjobstore

Subclasses of SimplePropertiesTriggerPersistenceDelegateSupport in org.quartz.impl.jdbcjobstore
- class CalendarIntervalTriggerPersistenceDelegate

Overview Package Class Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.SimplePropertiesTriggerProperties

Packages that use SimplePropertiesTriggerProperties
org.quartz.impl.jdbcjobstore

Uses of SimplePropertiesTriggerProperties in
org.quartz.impl.jdbcjobstore

Methods in org.quartz.impl.jdbcjobstore that return SimplePropertiesTriggerProperties

<table>
<thead>
<tr>
<th>protected abstract SimplePropertiesTriggerProperties</th>
<th>SimplePropertiesTriggerPersistenceDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected SimplePropertiesTriggerProperties</td>
<td>CalendarIntervalTriggerPersistenceDelegate</td>
</tr>
</tbody>
</table>

Methods in org.quartz.impl.jdbcjobstore with parameters of type SimplePropertiesTriggerProperties

<table>
<thead>
<tr>
<th>protected abstract TriggerPersistenceDelegate.TriggerPropertyBundle</th>
<th>SimplePropertiesTriggerPersistenceDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected TriggerPersistenceDelegate.TriggerPropertyBundle</td>
<td>CalendarIntervalTriggerPersistenceDelegate</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
No usage of org.quartz.impl.jdbcjobstore.SimpleSemaphore

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

No usage of org.quartz.impl.jdbcjobstore.SimpleTriggerPersistenceDelegate

Copyright 2001-2011, Terracotta, Inc.
## Uses of Interface

`org.quartz.impl.jdbcjobstore.StdJDBCConstants`

### Packages that use `StdJDBCConstants`

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.impl.jdbcjobstore</code></td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore.oracle</code></td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore.oracle.weblogic</code></td>
</tr>
</tbody>
</table>

### Uses of `StdJDBCConstants` in `org.quartz.impl.jdbcjobstore`

### Classes in `org.quartz.impl.jdbcjobstore` that implement `StdJDBCConstants`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalendarIntervalTriggerPersistenceDelegate</code></td>
<td></td>
</tr>
<tr>
<td><code>CronTriggerPersistenceDelegate</code></td>
<td></td>
</tr>
<tr>
<td><code>CronTriggerPersistenceDelegate</code></td>
<td></td>
</tr>
<tr>
<td><code>DB2v6Delegate</code></td>
<td>Quartz JDBC delegate for DB2 v6 databases.</td>
</tr>
<tr>
<td><code>DB2v7Delegate</code></td>
<td>Quartz JDBC delegate for DB2 v7 databases.</td>
</tr>
<tr>
<td><code>DB2v8Delegate</code></td>
<td>Quartz JDBC delegate for DB2 v8 databases.</td>
</tr>
<tr>
<td><code>DBSemaphore</code></td>
<td>Base class for database based lock handlers for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
</tbody>
</table>

### Deprecated Classes

- `Clou dscapeDelegate`: Use the `StdJDBCDelegate` for latest versions of Derby.
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HSQLDBDelegate</strong></td>
<td>This is a driver delegate for the HSQLDB database.</td>
</tr>
<tr>
<td><strong>MSSQLDelegate</strong></td>
<td>This is a driver delegate for the MSSQL JDBC driver.</td>
</tr>
<tr>
<td><strong>PointbaseDelegate</strong></td>
<td>This is a driver delegate for the Pointbase JDBC driver.</td>
</tr>
<tr>
<td><strong>PostgreSQLDelegate</strong></td>
<td>This is a driver delegate for the PostgreSQL JDBC driver.</td>
</tr>
<tr>
<td><strong>SimplePropertiesTriggerPersistenceDelegateSupport</strong></td>
<td>A base implementation of TriggerPersistenceDelegate that persists trigger fields in the &quot;QRTZ_SIMPROP_TRIGGERS&quot; table.</td>
</tr>
<tr>
<td><strong>SimpleTriggerPersistenceDelegate</strong></td>
<td></td>
</tr>
<tr>
<td><strong>StdJDBCDelegate</strong></td>
<td>This is meant to be an abstract base class for most, if not all, DriverDelegate implementations.</td>
</tr>
<tr>
<td><strong>StdRowLockSemaphore</strong></td>
<td>Internal database based lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td><strong>SybaseDelegate</strong></td>
<td>This is a driver delegate for the Sybase database.</td>
</tr>
<tr>
<td><strong>UpdateLockRowSemaphore</strong></td>
<td>Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.</td>
</tr>
<tr>
<td><strong>WebLogicDelegate</strong></td>
<td>This is a driver delegate for the WebLogic JDBC driver.</td>
</tr>
</tbody>
</table>

**Uses of **StdJDBCConstants** in

**org.quartz.impl.jdbcjobstore.oracle**

**Classes in **org.quartz.impl.jdbcjobstore.oracle** that implement **StdJDBCConstants**
class **OracleDelegate**

This is a driver delegate for the Oracle 10 and 11 database.

Uses of **StdJDBCConstants** in

org.quartz.impl.jdbcjobstore.oracle.weblogic

Classes in org.quartz.impl.jdbcjobstore.oracle.weblogic that implement StdJDBCConstants

class **WebLogicOracleDelegate**

Handle Blobs correctly when Oracle is being used inside of Weblogic 8.1, as discussed at:

http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class `org.quartz.impl.jdbcjobstore.StdJDBCDelegate`

## Packages that use `StdJDBCDelegate`
- `org.quartz.impl.jdbcjobstore`
- `org.quartz.impl.jdbcjobstore.oracle`
- `org.quartz.impl.jdbcjobstore.oracle.weblogic`

## Uses of `StdJDBCDelegate` in `org.quartz.impl.jdbcjobstore`

## Subclasses of `StdJDBCDelegate` in `org.quartz.impl.jdbcjobstore`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CloudscapeDelegate</code></td>
<td>Deprecated. Use the <code>StdJDBCDelegate</code> for latest versions of Derby</td>
</tr>
<tr>
<td><code>DB2v6Delegate</code></td>
<td>Quartz JDBC delegate for DB2 v6 databases.</td>
</tr>
<tr>
<td><code>DB2v7Delegate</code></td>
<td>Quartz JDBC delegate for DB2 v7 databases.</td>
</tr>
<tr>
<td><code>DB2v8Delegate</code></td>
<td>Quartz JDBC delegate for DB2 v8 databases.</td>
</tr>
<tr>
<td><code>HSQLDBDelegate</code></td>
<td>This is a driver delegate for the HSQLDB database.</td>
</tr>
<tr>
<td><code>MSSQLDelegate</code></td>
<td>This is a driver delegate for the MSSQL JDBC driver.</td>
</tr>
<tr>
<td><code>PointbaseDelegate</code></td>
<td>This is a driver delegate for the Pointbase JDBC driver.</td>
</tr>
<tr>
<td><code>PostgreSQLDelegate</code></td>
<td>This is a driver delegate for the PostgreSQL JDBC driver.</td>
</tr>
<tr>
<td><strong>SybaseDelegate</strong></td>
<td>This is a driver delegate for the Sybase database.</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>class <strong>WebLogicDelegate</strong></td>
<td>This is a driver delegate for the WebLogic JDBC driver.</td>
</tr>
</tbody>
</table>

### Uses of **StdJDBCDelegate** in **org.quartz.impl.jdbcjobstore.oracle**

### Subclasses of **StdJDBCDelegate** in **org.quartz.impl.jdbcjobstore.oracle**

| class **OracleDelegate** | This is a driver delegate for the Oracle 10 and 11 database. |

### Uses of **StdJDBCDelegate** in **org.quartz.impl.jdbcjobstore.oracle.weblogic**

### Subclasses of **StdJDBCDelegate** in **org.quartz.impl.jdbcjobstore.oracle.weblogic**

| class **WebLogicOracleDelegate** | Handle Blobs correctly when Oracle is being used inside of Weblogic 8.1, as discussed at: http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705 |

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.StdRowLockSemaphore

No usage of org.quartz.impl.jdbcjobstore.StdRowLockSemaphore

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.SybaseDelegate

No usage of org.quartz.impl.jdbcjobstore.SybaseDelegate

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface  
org.quartz.impl.jdbcjobstore.TablePrefixAware

Packages that use TablePrefixAware

org.quartz.impl.jdbcjobstore

Uses of TablePrefixAware in  
org.quartz.impl.jdbcjobstore

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSemaphore</td>
<td>Base class for database based lock handlers for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td>StdRowLockSemaphore</td>
<td>Internal database based lock handler for providing thread/resource locking in order to protect resources from being altered by multiple threads at the same time.</td>
</tr>
<tr>
<td>UpdateLockRowSemaphore</td>
<td>Provide thread/resource locking in order to protect resources from being altered by multiple threads at the same time using a db row update.</td>
</tr>
</tbody>
</table>

Overview  Package  Class  Tree  Deprecated  Index  Help
PREV  NEXT

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate

<table>
<thead>
<tr>
<th>Packages that use TriggerPersistenceDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of TriggerPersistenceDelegate in org.quartz.impl.jdbcjobstore</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Classes in org.quartz.impl.jdbcjobstore that implement TriggerPersistenceDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong> CalendarIntervalTriggerPersistenceDelegate</td>
</tr>
<tr>
<td><strong>class</strong> CronTriggerPersistenceDelegate</td>
</tr>
<tr>
<td><strong>class</strong> SimplePropertiesTriggerPersistenceDelegateSupport</td>
</tr>
<tr>
<td>A base implementation of TriggerPersistenceDelegate that persists trigger fields in the &quot;QRTZ_SIMPROP_TRIGGERS&quot; table.</td>
</tr>
<tr>
<td><strong>class</strong> SimpleTriggerPersistenceDelegate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields in org.quartz.impl.jdbcjobstore with type parameters of type TriggerPersistenceDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected <code>List&lt;TriggerPersistenceDelegate&gt; StdJDBCDelegate.triggerPersistenceDelegates</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl.jdbcjobstore that return TriggerPersistenceDelegate</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdJDBCDelegate.findTriggerPersistenceDelegate</td>
</tr>
</tbody>
</table>
Methods in `org.quartz.impl.jdbcjobstore` with parameters of type `TriggerPersistenceDelegate`:

- `void StdJDBCDelegate.addTriggerPersistenceDelegate(TriggerPersistenceDelegate)`
Uses of Class

org.quartz.impl.jdbcjobstore.TriggerPersistenceDelegate.TriggerPropertyBundle

Packages that use

TriggerPersistenceDelegate.TriggerPropertyBundle

org.quartz.impl.jdbcjobstore

Uses of

TriggerPersistenceDelegate.TriggerPropertyBundle in

org.quartz.impl.jdbcjobstore

Methods in org.quartz.impl.jdbcjobstore that return TriggerPersistenceDelegate.TriggerPropertyBundle

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected abstract TriggerPersistenceDelegate.TriggerPropertyBundle</td>
<td>SimplePropertiesTriggerPersistenceDelegate</td>
</tr>
<tr>
<td>protected TriggerPersistenceDelegate.TriggerPropertyBundle</td>
<td>CalendarIntervalTriggerPersistenceDelegate</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
<td>SimpleTriggerPersistenceDelegate</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
<td>SimplePropertiesTriggerPersistenceDelegate</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
<td>CronTriggerPersistenceDelegate</td>
</tr>
<tr>
<td>TriggerPersistenceDelegate.TriggerPropertyBundle</td>
<td>TriggerPersistenceDelegate.Loan</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
## Uses of Class

**org.quartz.impl.jdbcjobstore.TriggerStatus**

### Packages that use *TriggerStatus*

- **org.quartz.impl.jdbcjobstore**

### Uses of *TriggerStatus* in **org.quartz.impl.jdbcjobstore**

<table>
<thead>
<tr>
<th>Method in <strong>org.quartz.impl.jdbcjobstore</strong> that return <em>TriggerStatus</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StdJDBCDelegate.selectTriggerStatus</code>&lt;br&gt;<code>(Connection conn, TriggerKey triggerKey)</code></td>
<td>Select a trigger's status (state &amp; next fire time).</td>
</tr>
<tr>
<td><code>DriverDelegate.selectTriggerStatus</code>&lt;br&gt;<code>(Connection conn, TriggerKey triggerKey)</code></td>
<td>Select a trigger's status (state &amp; next fire time).</td>
</tr>
</tbody>
</table>

---

Uses of Class
org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore

No usage of org.quartz.impl.jdbcjobstore.UpdateLockRowSemaphore

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.Util

No usage of org.quartz.impl.jdbcjobstore.Util

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.jdbcjobstore.WebLogicDelegate

No usage of org.quartz.impl.jdbcjobstore.WebLogicDelegate

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>
public class OracleDelegate
extends StdJDBCDelegate

This is a driver delegate for the Oracle 10 and 11 database.

Author:
James House, Patrick Lightbody, Eric Mueller

See Also:
WebLogicDelegate, WebLogicOracleDelegate

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String INSERT_ORACLE_CALEYNDAR</td>
</tr>
<tr>
<td>static String INSERT_ORACLE_JOB_DETAIL</td>
</tr>
<tr>
<td>static String SELECT_ORACLE_CALEYNDAR_BLOB</td>
</tr>
<tr>
<td>static String SELECT_ORACLE_JOB_DETAIL_BLOB</td>
</tr>
<tr>
<td>static String SELECT_ORACLE_TRIGGER_JOB_DETAIL_BLOB</td>
</tr>
<tr>
<td>static String UPDATE_ORACLE_CALEYNDAR_BLOB</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>static String</td>
</tr>
</tbody>
</table>

**Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate**

classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

**Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants**

COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL_CALENDARS, DELETE_ALL_CRON_TRIGGERS, DELETE_ALL_JOB_DETAILS, DELETE_ALL_PAUSED_TRIGGER_GRPS, DELETE_ALL_SIMPLE_TRIGGERS, DELETE_ALL_SIMPROP_TRIGGERS, DELETE_ALL_TRIGGERS, DELETE_BLOB_TRIGGER, DELETE_CALENDAR, DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO_RECOVERY_FIRED_TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP, DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE, DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER, INSERT_CALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER, INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP, INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER, SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR, SELECT_CALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER, SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP, SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS, SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP, SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE, SELECT_INSTANCES_FIRED_TRIGGERS, SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL, SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE,
SELECT JOB FOR TRIGGER, SELECT JOB_GROUPS, 
SELECT JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP,
SELECT MISFIRED TRiggers, 
SELECT MISFIRED TRIGGERS IN GROUP IN STATE,
SELECT MISFIRED_TRIGGERS IN STATE, SELECT_NEXT_FiRE_TIME,
SELECT NEXT TRIGGER TO ACQUIRE, SELECT NUM CALENDARS,
SELECT NUM JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB,
SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP,
SELECT_PAUSED_TRIGGER_GROUPS, SELECT_REFERENCEDCALENDAR,
SELECT SCHEDULER_STATE, SELECT SCHEDULER_STATES,
SELECT SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA,
SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER FOR FIRE TIME,
SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED,
SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS,
SELECT_TRIGGERS FOR CALENDAR, SELECT_TRIGGERS FOR JOB,
SELECT_TRIGGERS IN GROUP, SELECT_TRIGGERS IN STATE,
TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALENDAR,
UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constants

ALIAS COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL CALENDAR, COL.Calendar_NAME, COL.Checkin_INTERVAL,
COL_CRON_EXPRESSION, COL_Description, COL.END_TIME, COL.Entry_ID,
COL_ENTRY_STATE, COL.FIRED_TIME, COL.INSTANCE_NAME,
COL.IS_DURABLE, COL.IS_NONCONCURRENT, COL.IS_UPDATE_DATA,
COL.IS_VOLATILE, COL.JOB_CLASS, COL.Job_Datamap, COL.JOB_GROUP,
COL.JOB_NAME, COL.LAST_CHECKIN_TIME, COL.Lock_NAME,
COL.MISFIRE_INSTRUCTION, COL.Next_FIRE_TIME, COL.Prev_FIRE_TIME,
COL.PRIORITY, COL.Repeat_COUNT, COL.Repeat_INTERVAL,
COL_REQUESTS.RECOVERY, COL.SCHEDULER_NAME, COL.START_TIME,
COL_TIME_ZONE_ID, COL.TIMES_TRIGGERED, COL_TRIGGER_GROUP,
COL_TRIGGER_NAME, COL_TRIGGER_STATE, COL_TRIGGER_TYPE,
DEFAULT_TABLE_PREFIX, STATE_ACQUIRED, STATE_BLOCKED,
STATE_COMPLETE, STATE_DELETED, STATE_ERROR, STATE_EXECUTING,
STATE_MISFIRED, STATE_PAUSED, STATE_PAUSED_BLOCKED, STATE_WAITING,
TABLE_BLOB_TRIGGERS, TABLE_CALENDARS, TABLE_CRON_TRIGGERS,
TABLE_FIRED_TRIGGERS, TABLE_JOBDETAILS, TABLE_LOCKS,
## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OracleDelegate</strong></td>
<td>(org.slf4j.Logger logger, String tablePrefix, String schedName, String instanceId,</td>
<td>Create new OracleDelegate instance.</td>
</tr>
<tr>
<td></td>
<td>org.quartz.spi.ClassLoadHelper classLoadHelper)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(org.slf4j.Logger logger, String tablePrefix, String schedName, String instanceId,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)</td>
<td>Create new OracleDelegate instance.</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getJobDataFromBlob</strong></td>
<td>(ResultSet rs, String colName)</td>
<td>This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details.</td>
</tr>
<tr>
<td><strong>getObjectFromBlob</strong></td>
<td>(ResultSet rs, String colName)</td>
<td>This method should be overridden by any delegate subclasses that need special handling for BLOBs.</td>
</tr>
<tr>
<td><strong>insertCalendar</strong></td>
<td>(Connection conn, String calendarName, Calendar calendar)</td>
<td>Insert a new calendar.</td>
</tr>
<tr>
<td><strong>insertJobDetail</strong></td>
<td>(Connection conn, JobDetail job)</td>
<td>Insert the job detail record.</td>
</tr>
<tr>
<td><strong>insertTrigger</strong></td>
<td>(Connection conn, org.quartz.spi.OperableTrigger trigger, String state, JobDetail jobDetail)</td>
<td>Insert the base trigger data.</td>
</tr>
<tr>
<td><strong>updateCalendar</strong></td>
<td>(Connection conn, String calendarName, Calendar calendar)</td>
<td>Update a calendar.</td>
</tr>
<tr>
<td><strong>updateJobData</strong></td>
<td>(Connection conn, JobDetail job)</td>
<td>Update the job data map for the given job.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>updateJobDetail</td>
<td>Update the job detail record.</td>
<td></td>
</tr>
<tr>
<td>int updateTrigger</td>
<td>Update the base trigger data.</td>
<td></td>
</tr>
<tr>
<td>protected Blob writeDataToBlob</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class `org.quartz.impl.jdbcjobstore.StdJDBCDelegate`:
- addDefaultTriggerPersistenceDelegates
- addTriggerPersistenceDelegate
- calendarExists
- calendarIsReferenced
- canUseProperties
- clearData
- closeResultSet
- closeStatement
- convertFromProperty
- convertToProperty
- countMisfiredTriggersInState
- deleteAllPausedTriggerGroups
- deleteBlobTrigger
- deleteCalendar
- deleteFiredTrigger
- deleteFiredTriggers
- deleteJobDetail
- deletePausedTriggerGroup
- deleteSchedulerState
- deleteTrigger
- deleteTriggerExtension
- findTriggerPersistenceDelegate
- findTriggerPersistenceDelegate
- getBoolean
- getBoolean
- getKeyOfNonSerializableValue
- getSchedulerNameLiteral
- hasMisfiredTriggersInState
- initialize
- insertBlobTrigger
- insertFiredTrigger
- insertPausedTriggerGroup
- insertSchedulerState
- isExistingTriggerGroup
- isJobNonConcurrent
- isTriggerGroupPaused
- jobExists
- rtp
- selectCalendar
- selectCalendars
- selectFiredTriggerInstanceNames
- selectFiredTriggerRecords
- selectFiredTriggerRecordsByJob
- selectInstancesFiredTriggerRecords
- selectJobDetail
- selectJobExecutionCount
- selectJobForTrigger
- selectJobGroups
- selectJobsInGroup
- selectMisfiredTriggers
- selectMisfiredTriggersInGroupInState
- selectMisfiredTriggersInState
- selectNextFireTime
- selectNumCalendars
- selectNumJobs
- selectNumTriggers
- selectNumTriggersForJob
- selectPausedTriggerGroups
- selectSchedulerStateRecords
- selectTrigger
- selectTriggerForFireTime
- selectTriggerGroups
- selectTriggerJobDataMap
- selectTriggerKeysForJob
- selectTriggersForCalendar
- selectTriggersForJob
- selectTriggersForRecoveringJobs
- selectTriggersInGroup
- selectTriggersInState
- selectTriggerState
- selectTriggerStatus
- selectTriggerToAcquire
- serializeJobData
- serializeObject
- setBoolean
- setBytes
- toSqlLikeClause
Method Details

triggerExists, updateBlobTrigger, updateFiredTrigger, updateSchedulerState, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Field Detail

INSERT_ORACLE_JOB_DETAIL

public static final String INSERT_ORACLE_JOB_DETAIL

See Also:
Constant Field Values

UPDATE_ORACLE_JOB_DETAIL

public static final String UPDATE_ORACLE_JOB_DETAIL

See Also:
Constant Field Values

UPDATE_ORACLE_JOB_DETAIL_BLOB

public static final String UPDATE_ORACLE_JOB_DETAIL_BLOB

See Also:
Constant Field Values
SELECT_ORACLE_JOB_DETAIL_BLOB
public static final String SELECT_ORACLE_JOB_DETAIL_BLOB

See Also:
Constant Field Values

-----------------------------------

UPDATE_ORACLE_TRIGGER
public static final String UPDATE_ORACLE_TRIGGER

See Also:
Constant Field Values

-----------------------------------

SELECT_ORACLE_TRIGGER_JOB_DETAIL_BLOB
public static final String SELECT_ORACLE_TRIGGER_JOB_DETAIL_BLOB

See Also:
Constant Field Values

-----------------------------------

UPDATE_ORACLE_TRIGGER_JOB_DETAIL_BLOB
public static final String UPDATE_ORACLE_TRIGGER_JOB_DETAIL_BLOB

See Also:
Constant Field Values

-----------------------------------

UPDATE_ORACLE_TRIGGER_JOB_DETAIL_EMPTY_BLOB
public static final String UPDATE_ORACLE_TRIGGER_JOB_DETAIL_EMPTY_BLOB

See Also:
Constant Field Values
**INSERT_ORACLECALENDAR**

```java
public static final String INSERT_ORACLECALENDAR
```

See Also:
- Constant Field Values

---

**SELECT_ORACLECALENDAR_BLOB**

```java
public static final String SELECT_ORACLECALENDAR_BLOB
```

See Also:
- Constant Field Values

---

**UPDATE_ORACLECALENDAR_BLOB**

```java
public static final String UPDATE_ORACLECALENDAR_BLOB
```

See Also:
- Constant Field Values

---

### Constructor Detail

**OracleDelegate**

```java
public OracleDelegate(org.slf4j.Logger logger,
                      String tablePrefix,
                      String schedName,
                      String instanceId,
                      org.quartz.spi.ClassLoadHelper classLoadHelper)
```

Create new OrcaleDelegate instance.

**Parameters:**
- `logger` - the logger to use during execution
- `tablePrefix` - the prefix of all table names
OracleDelegate

public OracleDelegate(org.slf4j.Logger logger,
                      String tablePrefix,
                      String schedName,
                      String instanceId,
                      org.quartz.spi.ClassLoadHelper classLoadHelper,
                      Boolean useProperties)

Create new OracleDelegate instance.

Parameters:
logger - the logger to use during execution
tablePrefix - the prefix of all table names
useProperties - use java.util.Properties for storage

Method Detail

getObjectFromBlob

protected Object getObjectFromBlob(ResultSet rs,
                                     String colName)
  throws ClassNotFoundException,
           IOException,
           SQLException

Description copied from class: StdJDBCDelegate

This method should be overridden by any delegate subclasses that need special handling for BLOBs. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:
  getBlob in class StdJDBCDelegate

Parameters:
  rs - the result set, already queued to the correct row
  colName - the column name for the BLOB

Returns:
  the deserialized Object from the ResultSet BLOB

Throws:
ClassNotFoundException - if a class found during deserialization cannot be found
IOException - if deserialization causes an error
SQLException

insertJobDetail

public int insertJobDetail(Connection conn, JobDetail job)
    throws IOException, SQLException

Description copied from class: StdJDBCDelegate

Insert the job detail record.

Specified by:
    insertJobDetail in interface DriverDelegate
Overrides:
    insertJobDetail in class StdJDBCDelegate
Parameters:
    conn - the DB Connection
    job - the job to insert
Returns:
    number of rows inserted
Throws:
    IOException - if there were problems serializing the JobDataMap
    SQLException

getJobDataFromBlob

protected Object getJobDataFromBlob(ResultSet rs, String colName)
    throws ClassNotFoundException, IOException, SQLException

Description copied from class: StdJDBCDelegate
This method should be overridden by any delegate subclasses that need special handling for BLOBs for job details. The default implementation uses standard JDBC java.sql.Blob operations.

Overrides:

```
getJobDataFromBlob in class StdJDBCDelegate
```

Parameters:

```
rs - the result set, already queued to the correct row
colName - the column name for the BLOB
```

Returns:

```
the deserialized Object from the ResultSet BLOB
```

Throws:

```
ClassNotFoundException - if a class found during deserialization cannot be found
IOException - if deserialization causes an error
SQLException
```

---

**updateJobDetail**

```
public int updateJobDetail(Connection conn, JobDetail job)
```

Throws: IOException, SQLException

Description copied from class: StdJDBCDelegate

Update the job detail record.

Specified by:

```
updateJobDetail in interface DriverDelegate
```

Overrides:

```
updateJobDetail in class StdJDBCDelegate
```

Parameters:

```
conn - the DB Connection
job - the job to update
```

Returns:

```
number of rows updated
```

Throws:

```
IOException - if there were problems serializing the JobDataMap
```
SQLException

insertTrigger

```java
public int insertTrigger(Connection conn,
                        org.quartz.spi.OperableTrigger trigger,
                        String state,
                        JobDetail jobDetail)
  throws SQLException,
          IOException
```

Description copied from class: StdJDBCDelegate

Insert the base trigger data.

Specified by:
  insertTrigger in interface DriverDelegate

Overrides:
  insertTrigger in class StdJDBCDelegate

Parameters:
  conn - the DB Connection
  trigger - the trigger to insert
  state - the state that the trigger should be stored in

Returns:
  the number of rows inserted

Throws:
  SQLException
  IOException

updateTrigger

```java
public int updateTrigger(Connection conn,
                        org.quartz.spi.OperableTrigger trigger,
                        String state,
                        JobDetail jobDetail)
  throws SQLException,
          IOException
```

Description copied from class: StdJDBCDelegate
Update the base trigger data.

Specified by:
updateTrigger in interface DriverDelegate

Overrides:
updateTrigger in class StdJDBCDelegate

Parameters:
conn - the DB Connection
trigger - the trigger to insert
state - the state that the trigger should be stored in

Returns:
the number of rows updated

Throws:
SQLException
IOException

insertCalendar

public int insertCalendar(Connection conn,
String calendarName,
Calendar calendar)
throws IOException,
SQLException

Description copied from class: StdJDBCDelegate

Insert a new calendar.

Specified by:
insertCalendar in interface DriverDelegate

Overrides:
insertCalendar in class StdJDBCDelegate

Parameters:
conn - the DB Connection
calendarName - the name for the new calendar
calendar - the calendar

Returns:
the number of rows inserted

Throws:
IOException - if there were problems serializing the calendar
SQLException

updateCalendar

public int updateCalendar(Connection conn,
String calendarName,
Calendar calendar)
throws IOException,
SQLException

Description copied from class: StdJDBCDelegate

Update a calendar.

Specified by:
updateCalendar in interface DriverDelegate

Overrides:
updateCalendar in class StdJDBCDelegate

Parameters:
conn - the DB Connection
calendarName - the name for the new calendar
calendar - the calendar

Returns:
the number of rows updated

Throws:
IOException - if there were problems serializing the calendar
SQLException

updateJobData

public int updateJobData(Connection conn,
JobDetail job)
throws IOException,
SQLException

Description copied from class: StdJDBCDelegate

Update the job data map for the given job.
Specified by:
  updateJobData in interface DriverDelegate

Overrides:
  updateJobData in class StdJDBCDelegate

Parameters:
  conn - the DB Connection
  job - the job to update

Returns:
  the number of rows updated

Throws:
  IOException - if there were problems serializing the JobDataMap
  SQLException

writeDataToBlob

protected Blob writeDataToBlob(ResultSet rs,
                               int column,
                               byte[] data)
  throws SQLException

  SQLException
org.quartz.impl.jdbcjobstore.oracle Classes OracleDelegate
Package org.quartz.impl.jdbcjobstore.oracle

Class Summary

| OracleDelegate | This is a driver delegate for the Oracle 10 and 11 database. |

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package
org.quartz.impl.jdbctimerstore.oracle

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.jdbcjobstore.**StdJDBCDelegate** (implements org.quartz.impl.jdbcjobstore.**DriverDelegate**, org.quartz.impl.jdbcjobstore.**StdJDBCConstants**)
  - org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.impl.jdbcjobstore.oracle

Packages that use
org.quartz.impl.jdbcjobstore.oracle
org.quartz.impl.jdbcjobstore.oracle.weblogic

Classes in org.quartz.impl.jdbcjobstore.oracle used by org.quartz.impl.jdbcjobstore.oracle.weblogic
OracleDelegate
   This is a driver delegate for the Oracle 10 and 11 database.
### Uses of Class

**org.quartz.impl.jdbcjobstore.oracle.OracleDelegate**

<table>
<thead>
<tr>
<th>Packages that use <strong>OracleDelegate</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.jdbcjobstore.oracle.weblogic</td>
</tr>
</tbody>
</table>

| Uses of **OracleDelegate** in **org.quartz.impl.jdbcjobstore.oracle.weblogic** |

<table>
<thead>
<tr>
<th>Subclasses of <strong>OracleDelegate</strong> in <strong>org.quartz.impl.jdbcjobstore.oracle.weblogic</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WebLogicOracleDelegate</strong></td>
</tr>
<tr>
<td>Handle Blobs correctly when Oracle is being used inside of Weblogic 8.1, as discussed at: <a href="http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705">http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705</a></td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Package org.quartz.impl.jdbcjobstore.oracle.weblogic

### Class Summary

| Class Summary | WebLogicOracleDelegate | Handle Blobs correctly when Oracle is being used in Weblogic 8.1, as discussed at:  
|               |                        | http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html |

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Hierarchy For Package
org.quartz.impl.jdbcjobstore.oracle.weblogic

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.jdbcjobstore.**StdJDBCDelegate** (implements org.quartz.impl.jdbcjobstore.**DriverDelegate**, org.quartz.impl.jdbcjobstore.**StdJDBCConstants**)
    - org.quartz.impl.jdbcjobstore.oracle.**OracleDelegate**
      - org.quartz.impl.jdbcjobstore.oracle.weblogic.**WebLogicOracle**
Uses of Package
org.quartz.impl.jdbcjobstore.oracle.weblogic

No usage of org.quartz.impl.jdbcjobstore.oracle.weblogic

Copyright 2001-2011, Terracotta, Inc.
java.lang.Object
  └ org.quartz.impl.jdbcjobstore.StdJDBCDelegate
      └ org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
          └ org.quartz.impl.jdbcjobstore.oracle.weblogic.WebLogicOracleDelegate

All Implemented Interfaces:
    Constants, DriverDelegate, StdJDBCConstants

public class WebLogicOracleDelegate
extends OracleDelegate

Handle Blobs correctly when Oracle is being used inside of Weblogic 8.1, as discussed at: http://edocs.bea.com/wls/docs81/jdbc/thirdparty.html#1043705

Author:
    James House, Igor Fedulov igor@fedulov.com
See Also:
    WebLogicDelegate

Field Summary

Fields inherited from class org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
INSERT_ORACLECALENDAR, INSERT_ORACLE_JOB_DETAIL,
SELECT_ORACLECALENDAR_BLOB, SELECT_ORACLE_JOBDETAIL_BLOB,
UPDATE_ORACLECALENDAR_BLOB, UPDATE_ORACLE_JOB_DETAIL,
UPDATE_ORACLE_TRIGGER_JOBDETAIL_BLOB, UPDATE_ORACLE_TRIGGER,
UPDATE_ORACLE_TRIGGER_JOBDETAIL_EMPTY_BLOB

Fields inherited from class org.quartz.impl.jdbcjobstore.StdJDBCDelegate
classLoadHelper, instanceId, logger, schedName, tablePrefix, triggerPersistenceDelegates, useProperties

Fields inherited from interface org.quartz.impl.jdbcjobstore.StdJDBCConstants

| COUNT_MISFIRED_TRIGGERS_IN_STATE, DELETE_ALL_BLOB_TRIGGERS, DELETE_ALL.Calendar, DELETE_ALL.CRON.TRIGGERS, DELETE_ALL_JOB.DETAILS, DELETE_ALL.PAUSED.TRIGGER_GRPS, DELETE_ALL_SIMPLE.TRIGGERS, DELETE_ALL_SIMPROP.TRIGGERS, DELETE_ALL.TRIGGERS, DELETE_BLOB_TRIGGER, DELETE_CALENDAR, DELETE_CRON_TRIGGER, DELETE_FIRED_TRIGGER, DELETE_FIRED_TRIGGERS, DELETE_INSTANCES_FIRED_TRIGGERS, DELETE_JOB_DETAIL, DELETE_NO.RECOVERY.FIRED.TRIGGERS, DELETE_PAUSED_TRIGGER_GROUP, DELETE_PAUSED_TRIGGER_GROUPS, DELETE_SCHEDULER_STATE, DELETE_SIMPLE_TRIGGER, DELETE_TRIGGER, INSERT_BLOB_TRIGGER, INSERT_CALENDAR, INSERT_CRON_TRIGGER, INSERT_FIRED_TRIGGER, INSERT_JOB_DETAIL, INSERT_PAUSED_TRIGGER_GROUP, INSERT_SCHEDULER_STATE, INSERT_SIMPLE_TRIGGER, INSERT_TRIGGER, SCHED_NAME_SUBST, SELECT_BLOB_TRIGGER, SELECT_CALENDAR, SELECTCALENDAR_EXISTENCE, SELECT_CALENDARS, SELECT_CRON_TRIGGER, SELECT_FIRED_TRIGGER, SELECT_FIRED_TRIGGER_GROUP, SELECT_FIRED_TRIGGER_INSTANCE_NAMES, SELECT_FIRED_TRIGGERS, SELECT_FIRED_TRIGGERS_OF_JOB, SELECT_FIRED_TRIGGERS_OF_JOB_GROUP, SELECT_HAS_MISFIRED_TRIGGERS_IN_STATE, SELECT_INSTANCES_FIRED_TRIGGERS, SELECT_INSTANCES_RECOVERABLE_FIRED_TRIGGERS, SELECT_JOB_DETAIL, SELECT_JOB_EXECUTION_COUNT, SELECT_JOB_EXISTENCE, SELECT_JOB_FOR_TRIGGER, SELECT_JOB_GROUPS, SELECT_JOB_NONCONCURRENT, SELECT_JOBS_IN_GROUP, SELECT_MISFIRED_TRIGGERS, SELECT_MISFIRED_TRIGGERS_IN_GROUP_IN_STATE, SELECT_MISFIRED_TRIGGERS_IN_STATE, SELECT_NEXT_FIRE_TIME, SELECT_NEXT_TRIGGER_TO_ACQUIRE, SELECT_NUM_CALENDARS, SELECT_NUM_JOBS, SELECT_NUM_TRIGGERS, SELECT_NUM_TRIGGERS_FOR_JOB, SELECT_NUM_TRIGGERS_IN_GROUP, SELECT_PAUSED_TRIGGER_GROUP, SELECT_PAUSED_TRIGGER_GROUPS, SELECTREFERENCED_CALENDAR, SELECT_SCHEDULER_STATE, SELECT_SCHEDULER_STATES, SELECT_SIMPLE_TRIGGER, SELECT_TRIGGER, SELECT_TRIGGER_DATA, SELECT_TRIGGER_EXISTENCE, SELECT_TRIGGER_FOR_FIRE_TIME, SELECT_TRIGGER_GROUPS, SELECT_TRIGGER_GROUPS_FILTERED, SELECT_TRIGGER_STATE, SELECT_TRIGGER_STATUS, SELECT_TRIGGERS_FOR_CALENDAR, SELECT_TRIGGERS_FOR_JOB, SELECT_TRIGGERS_IN_GROUP, SELECT_TRIGGERS_IN_STATE, TABLE_PREFIX_SUBST, UPDATE_BLOB_TRIGGER, UPDATE_CALENDAR, UPDATE_CRON_TRIGGER, UPDATE_FIRED_TRIGGER,
UPDATE_INSTANCES_FIRED_TRIGGER_STATE, UPDATE_JOB_DATA,
UPDATE_JOB_DETAIL, UPDATE_JOB_TRIGGER_STATES,
UPDATE_JOB_TRIGGER_STATES_FROM_OTHER_STATE,
UPDATE_SCHEDULER_STATE, UPDATE_SIMPLE_TRIGGER, UPDATE_TRIGGER,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATE,
UPDATE_TRIGGER_GROUP_STATE_FROM_STATES, UPDATE_TRIGGER_SKIP_DATA,
UPDATE_TRIGGER_STATE, UPDATE_TRIGGER_STATE_FROM_STATE,
UPDATE_TRIGGER_STATE_FROM_STATES,
UPDATE_TRIGGER_STATES_FROM_OTHER_STATES

Fields inherited from interface org.quartz.impl.jdbcjobstore.Constomts
ALIAS_COL_NEXT_FIRE_TIME, ALL_GROUPS_PAUSED, COL_BLOB,
COL_CALENDAR, COL_CALENDAR_NAME, COL_CHECKIN_INTERVAL,
COL_CRON_EXPRESSION, COL_DESCRIPTION, COL_END_TIME, COL_ENTRY_ID,
COL_ENTRY_STATE, COL_FIRED_TIME, COL_INSTANCE_NAME,
COL_IS_DURABLE, COL_IS_NONCONCURRENT, COL_IS_UPDATE_DATA,
COL_IS_VOLATILE, COL_JOB_CLASS, COL_JOB_DATAMAP, COL_JOB_GROUP,
COL_JOB_NAME, COL_LAST_CHECKIN_TIME, COL_LOCK_NAME,
COL_MISFIRE_INSTRUCTION, COL_NEXT_FIRE_TIME, COL_PREV_FIRE_TIME,
COL_PRIORITY, COL_REPEAT_COUNT, COL_REPEAT_INTERVAL,
COL_REQUESTS_RECOVERY, COL_SCHEDULER_NAME, COL_START_TIME,
COL_TIME_ZONE_ID, COL_TIMES_TRIGGERED, COL_TRIGGER_GROUP,
COL_TRIGGER_NAME, COL_TRIGGER_STATE, COL_TRIGGER_TYPE,
DEFAULT_TABLE_PREFIX, STATE_ACQUIRED, STATE_BLOCKED,
STATE_COMPLETE, STATE_DELETED, STATE_ERROR, STATE_EXECUTING,
STATE_MISFIRED, STATE_PAUSED, STATE_PAUSED_BLOCKED, STATE_WAITING,
TABLE_BLOB_TRIGGERS, TABLE_CALENDARS, TABLE_CRON_TRIGGERS,
TABLE_FIRED_TRIGGERS, TABLE_JOBDETAILS, TABLE_LOCKS,
TABLE_PAOUSED_TRIGGERS, TABLE_SCHEDULER_STATE,
TABLE_SIMPLE_TRIGGERS, TABLE_TRIGGERS, TTYPE_BLOB, TTYPE_CAL_INT,
TTYPE_CRON, TTYPE_SIMPLE

Constructor Summary

WebLogicOracleDelegate (org.slf4j.Logger logger,
String tablePrefix, String schedName, String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new WebLogicOracleDelegate instance.

WebLogicOracleDelegate (org.slf4j.Logger logger,
String tablePrefix, String schedName, String instanceId,
org.quartz.spi.ClassLoadHelper classLoadHelper,
Boolean useProperties)

Create new WebLogicOracleDelegate instance.
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Blob <code>writeDataToBlob(ResultSet rs, int column, byte[] data)</code></td>
<td>Check for the Weblogic Blob wrapper, and handle accordingly...</td>
</tr>
</tbody>
</table>

### Methods inherited from class `org.quartz.impl.jdbcjobstore.oracle.OracleDelegate`

- getJobDataFromBlob, getObjectIdFromBlob, insertCalendar, insertJobDetail, insertTrigger, updateCalendar, updateJobData, updateJobDetail, updateTrigger

### Methods inherited from class `org.quartz.impl.jdbcjobstore.StdJDBCDelegate`

- addDefaultTriggerPersistenceDelegates, addTriggerPersistenceDelegate, calendarExists, calendarIsReferenced, canUseProperties, clearData, closeResultSet, closeStatement, convertFromProperty, convertToProperty, countMisfiredTriggersInState, deleteAllPausedTriggerGroups, deleteBlobTrigger, deleteCalendar, deleteFiredTrigger, deleteFiredTriggers, deleteFiredTrigger, deleteJobDetail, deletePausedTriggerGroup, deleteSchedulerState, deleteTrigger, deleteTriggerExtension, findTriggerPersistenceDelegate, findTriggerPersistenceDelegate, getBoolean, getBoolean, getKeyPressOfNonSerializableValue, getSchedulerNameLiteral, hasMisfiredTriggersInState, initialize, insertBlobTrigger, insertFiredTrigger, insertPausedTriggerGroup, insertSchedulerState, isExistingTriggerGroup, isJobNonConcurrent, isTriggerGroupPaused, jobExists, rtp, selectCalendar, selectCalendars, selectFiredTriggerInstanceNames, selectFiredTriggerRecords, selectFiredTriggerRecordsByJob, selectInstancesFiredTriggerRecords, selectJobDetail, selectJobExecutionCount, selectJobForTrigger, selectJobGroups, selectJobsInGroup, selectMisfiredTriggers, selectMisfiredTriggersInGroupInState, selectMisfiredTriggersInState, selectNextFireTime, selectNumCalendars, selectNumJobs, selectNumTriggers, selectNumTriggersForJob, selectPausedTriggerGroups, selectSchedulerStateRecords, selectTrigger, selectTriggerForFireTime, selectTriggerGroups, selectTriggerGroups, selectTriggerJobDataMap, selectTriggerKeysForJob, selectTriggersForCalendar, selectTriggersForJob, selectTriggersForRecoveringJobs,
selectTriggersInGroup, selectTriggersInState, selectTriggerState, selectTriggerStatus, selectTriggerToAcquire, serializeJobData, serializeObject, setBoolean, setBytes, toSqlLikeClause, triggerExists, updateBlobTrigger, updateFiredTrigger, updateSchedulerState, updateTriggerGroupStateFromOtherState, updateTriggerGroupStateFromOtherStates, updateTriggerState, updateTriggerGroupStateFromOtherState, updateTriggerStateFromOtherStates, updateTriggerStatesForJob, updateTriggerStatesForJobFromOtherState, updateTriggerStatesFromOtherStates

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

WebLogicOracleDelegate

public WebLogicOracleDelegate(org.slf4j.Logger logger, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper)

Create new WebLogicOracleDelegate instance.

Parameters:
logger - the logger to use during execution
tablePrefix - the prefix of all table names

WebLogicOracleDelegate

public WebLogicOracleDelegate(org.slf4j.Logger logger, String tablePrefix, String schedName, String instanceId, org.quartz.spi.ClassLoadHelper classLoadHelper, Boolean useProperties)
Create new WebLogicOracleDelegate instance.

**Parameters:**
- logger - the logger to use during execution
- tablePrefix - the prefix of all table names
- useProperties - use java.util.Properties for storage

**Method Detail**

`writeDataToBlob`

```java
protected Blob writeDataToBlob(ResultSet rs,
    int column,
    byte[] data)
throws SQLException
```

Check for the Weblogic Blob wrapper, and handle accordingly...

**Overrides:**
- `writeDataToBlob` in class `OracleDelegate`

**Throws:**
- `SQLException`

Uses of Class
org.quartz.impl.jdbcjobstore.oracle.weblogic.WebLogi

No usage of
org.quartz.impl.jdbcjobstore.oracle.weblogic.WebLogicOracleDelegate

Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.matchers Class AndMatcher<T extends Key>

java.lang.Object
    \ org.quartz.impl.matchers.AndMatcher<T>

All Implemented Interfaces:
    Serializable, Matcher<T>

public class AndMatcher<T extends Key>
extends Object
implements Matcher<T>

Matches using an AND operator on two Matcher operands.

Author:
jhouse
See Also:
    Serialized Form

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Matcher&lt;T&gt; leftOperand</td>
</tr>
<tr>
<td>protected Matcher&lt;T&gt; rightOperand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected AndMatcher(Matcher&lt;T&gt; leftOperand, Matcher&lt;T&gt; rightOperand)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static &lt;U extends Key&gt; AndMatcher&lt;U&gt;</td>
</tr>
</tbody>
</table>
Create an AndMatcher that depends upon the result of both of the given matchers.

```java
public static <U extends Key> AndMatcher<U> and(Matcher<U> leftOperand, Matcher<U> rightOperand)
```

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, toString, wait, wait

Field Detail

leftOperand

protected Matcher<T> extends Key leftOperand

rightOperand

protected Matcher<T> extends Key rightOperand

Constructor Detail

AndMatcher

protected AndMatcher(Matcher<T> leftOperand, Matcher<T> rightOperand)

Method Detail

and

public static <U extends Key> AndMatcher<U> and(Matcher<U> leftOperand, Matcher<U> rightOperand)
Create an AndMatcher that depends upon the result of both of the given matchers.

**isMatch**

```java
public boolean isMatch(T key)
```

*Specified by:*  
`isMatch` in interface `Matcher<T extends Key>`

**getLeftOperand**

```java
public Matcher<T> getLeftOperand()
```

**getRightOperand**

```java
public Matcher<T> getRightOperand()
```

**hashCode**

```java
public int hashCode()
```

*Specified by:*  
`hashCode` in interface `Matcher<T extends Key>`  
*Overrides:*  
`hashCode` in class `Object`

**equals**

```java
public boolean equals(Object obj)
```

*Specified by:*  
`equals` in interface `Matcher<T extends Key>`  
*Overrides:*
equals in class Object
Class `EverythingMatcher<T extends Key>`

extends `java.lang.Object`

| org.quartz.impl.matchers.EverythingMatcher<T> |

All Implemented Interfaces:

`Serializable, Matcher<T>`

public class `EverythingMatcher<T extends Key>`

extends `Object`

implements `Matcher<T>`

Matches on the complete key being equal (both name and group).

Author:

jhouse

See Also:

`Serialized Form`

Constructor Summary

| protected `EverythingMatcher()` |

Method Summary

| static `EverythingMatcher<JobKey> allJobs()` Create an EverythingMatcher that matches all jobs. |

| static `EverythingMatcher<TriggerKey> allTriggers()` Create an EverythingMatcher that matches all triggers. |

| boolean `equals(Object obj)` |
Methods inherited from class java.lang.**Object**
clone, finalize, getClass, notify, notifyAll, toString, wait, wait

**Constructor Detail**

**EverythingMatcher**

protected **EverythingMatcher()**

**Method Detail**

**allJobs**

class **EverythingMatcher**

public static **EverythingMatcher<JobKey> allJobs()**

Create an EverythingMatcher that matches all jobs.

**allTriggers**

class **EverythingMatcher**

public static **EverythingMatcher<TriggerKey> allTriggers()**

Create an EverythingMatcher that matches all triggers.

**isMatch**

class **EverythingMatcher**

public boolean **isMatch(T key)**

**Specified by:**
equals

public boolean equals(Object obj)

Specified by:
equals in interface Matcher<T extends Key>

Overrides:
equals in class Object

hashCode

public int hashCode()

Specified by:
hashCode in interface Matcher<T extends Key>

Overrides:
hashCode in class Object
Class `GroupMatcher<T extends Key>`

extends `StringMatcher<T>`

Matches on group (ignores name) property of Keys.

Author:
jhouse

See Also:
Serialized Form

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from class org.quartz.impl.matchers.StringMatcher</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StringMatcher.StringOperatorName</code></td>
</tr>
</tbody>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from class org.quartz.impl.matchers.StringMatcher</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>compareTo</code>, <code>compareWith</code></td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>protected `GroupMatcher(String compareTo,</th>
</tr>
</thead>
</table>
**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected String getValue(T key)</code></td>
<td></td>
</tr>
<tr>
<td><code>static GroupMatcher groupContains(String compareTo)</code></td>
<td>Create a GroupMatcher that matches groups containing the given string.</td>
</tr>
<tr>
<td><code>static GroupMatcher groupEndsWith(String compareTo)</code></td>
<td>Create a GroupMatcher that matches groups ending with the given string.</td>
</tr>
<tr>
<td><code>static GroupMatcher groupEquals(String compareTo)</code></td>
<td>Create a GroupMatcher that matches groups equaling the given string.</td>
</tr>
<tr>
<td><code>static GroupMatcher groupStartsWith(String compareTo)</code></td>
<td>Create a GroupMatcher that matches groups starting with the given string.</td>
</tr>
</tbody>
</table>

Methods inherited from class `org.quartz.impl.matchers.StringMatcher`:
- `equals`, `getCompareToValue`, `getCompareWithOperator`, `hashCode`, `isMatch`

Methods inherited from class `java.lang.Object`:
- `clone`, `finalize`, `getClass`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`  

**Constructor Detail**

**GroupMatcher**

`protected GroupMatcher(String compareTo, StringMatcher.StringOperatorName compareWith)`
### Method Detail

**groupEquals**

```java
class GroupMatcher {
  public static GroupMatcher groupEquals(String compareTo) {
    // Create a GroupMatcher that matches groups equaling the given string.
  }
}
```

**groupStartsWith**

```java
class GroupMatcher {
  public static GroupMatcher groupStartsWith(String compareTo) {
    // Create a GroupMatcher that matches groups starting with the given string.
  }
}
```

**groupEndsWith**

```java
class GroupMatcher {
  public static GroupMatcher groupEndsWith(String compareTo) {
    // Create a GroupMatcher that matches groups ending with the given string.
  }
}
```

**groupContains**

```java
class GroupMatcher {
  public static GroupMatcher groupContains(String compareTo) {
    // Create a GroupMatcher that matches groups containing the given string.
  }
}
```

**getValue**

```java
class StringMatcher<T extends Key> {
  protected String getValue(T key) {
    // Specified by:
    // StringMatcher<T extends Key> in class StringMatcher<T extends Key>
  }
}
```
Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.matchers  Class KeyMatcher<T extends Key>

java.lang.Object  
   ↓org.quartz.impl.matchers.KeyMatcher<T>

All Implemented Interfaces:
   Serializable, Matcher<T>

public class KeyMatcher<T extends Key>
   extends Object
   implements Matcher<T>

Matches on the complete key being equal (both name and group).

Author:
   jhouse

See Also:
   Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>protected</th>
<th>compareTo</th>
</tr>
</thead>
</table>

Constructor Summary

| protected | KeyMatcher(I compareTo) |

Method Summary

| boolean | equals(Object obj) |

I getCompareToValue();
**hashCode**

The `hashCode` method is used to determine the hash code of an object. This is an integral part of hash tables and hash maps, which allow for efficient lookup and management of elements.

**isMatch(T key)**

The `isMatch` method checks if the given key matches the criteria set by the KeyMatcher instance.

**KeyMatcher**

A KeyMatcher is a class that can be used to match Keys based on certain criteria. It can be created using the `keyEquals` or `isMatch` methods.

**keyEquals(U compareTo)**

Create a KeyMatcher that matches Keys that equal the given key.

**Methods inherited from class java.lang.Object**

- `clone`, `finalize`, `getClass`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

**Field Detail**

**compareTo**

protected `T` extends `Key` \( compareTo \)

**Constructor Detail**

**KeyMatcher**

protected `KeyMatcher(T compareTo)`

**Method Detail**

**keyEquals**

public static `<U extends Key> KeyMatcher<U> keyEquals(U compareTo)`

Create a KeyMatcher that matches Keys that equal the given key.

**isMatch**
public boolean isMatch(T key)

Specified by:
   isMatch in interface Matcher<T extends Key>

getCompareToValue

public T getCompareToValue()

hashCode

public int hashCode()

Specified by:
   hashCode in interface Matcher<T extends Key>

Overrides:
   hashCode in class Object

equals

public boolean equals(Object obj)

Specified by:
   equals in interface Matcher<T extends Key>

Overrides:
   equals in class Object

Copyright 2001-2011, Terracotta, Inc.
Class `NameMatcher<T extends Key>`

```
java.lang.Object
  ↳ org.quartz.impl.matchers.StringMatcher<T>
  ↳ org.quartz.impl.matchers.NameMatcher<T>
```

**All Implemented Interfaces:**

`Serializable`, `Matcher<T>`

---

```java
public class NameMatcher<T extends Key> extends StringMatcher<T>
```

matches on name (ignores group) property of Keys.

**Author:**

jhouse

**See Also:**

`Serialized Form`

---

### Nested Class Summary

**Nested classes/interfaces inherited from class org.quartz.impl.matchers.StringMatcher**: `StringMatcher.StringOperatorName`

### Field Summary

**Fields inherited from class org.quartz.impl.matchers.StringMatcher**: `compareTo, compareWith`

### Constructor Summary

```java
protected NameMatcher(String compareTo,
```
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected <code>String getValue(T key)</code></td>
<td></td>
</tr>
<tr>
<td>static <code>NameMatcher nameContains(String compareTo)</code></td>
<td>Create a NameMatcher that matches names containing the given string.</td>
</tr>
<tr>
<td>static <code>NameMatcher nameEndsWith(String compareTo)</code></td>
<td>Create a NameMatcher that matches names ending with the given string.</td>
</tr>
<tr>
<td>static <code>NameMatcher nameEquals(String compareTo)</code></td>
<td>Create a NameMatcher that matches names equaling the given string.</td>
</tr>
<tr>
<td>static <code>NameMatcher nameStartsWith(String compareTo)</code></td>
<td>Create a NameMatcher that matches names starting with the given string.</td>
</tr>
</tbody>
</table>

Methods inherited from class `org.quartz.impl.matchers.StringMatcher`:
- `equals`, `getCompareToValue`, `getCompareWithOperator`, `hashCode`, `isMatch`

Methods inherited from class `java.lang.Object`:
- `clone`, `finalize`, `getClass`, `notify`, `notifyAll`, `toString`, `wait`, `wait`  

## Constructor Detail

**NameMatcher**

protected `NameMatcher(String compareTo, StringMatcher.StringOperatorName compareWith)`
Method Detail

nameEquals

public static NameMatcher nameEquals(String compareTo)

Create a NameMatcher that matches names equaling the given string.

nameStartsWith

public static NameMatcher nameStartsWith(String compareTo)

Create a NameMatcher that matches names starting with the given string.

nameEndsWith

public static NameMatcher nameEndsWith(String compareTo)

Create a NameMatcher that matches names ending with the given string.

nameContains

public static NameMatcher nameContains(String compareTo)

Create a NameMatcher that matches names containing the given string.

g getValue

protected String getValue(T key)

Specified by:
g getValue in class StringMatcher<T extends Key>
Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.matchers  Class $\text{NotMatcher}\langle T \text{ extends } \text{Key} \rangle$

java.lang.Object
  └ org.quartz.impl.matchers.NotMatcher<T>

All Implemented Interfaces:
  Serializable, Matcher<T>

---

public class $\text{NotMatcher}\langle T \text{ extends } \text{Key} \rangle$

extends $\text{Object}$
implements $\text{Matcher}\langle T \rangle$

Matches using an NOT operator on another Matcher.

Author:
  jhouse

See Also:
  Serialized Form

---

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Matcher&lt;T&gt; operand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected $\text{NotMatcher}\langle Matcher\langle T \rangle$ operand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(Object obj)</td>
</tr>
<tr>
<td>Matcher&lt;T&gt; getOperand()</td>
</tr>
</tbody>
</table>
int \texttt{hashCode}()

boolean \texttt{isMatch}(T key)

static <U extends Key>
NotMatcher\texttt{<U> not}(Matcher\texttt{<U>} operand)

Create a NotMatcher that reverses the result of the given matcher.

\textbf{Methods inherited from class java.lang.\texttt{Object}}

\texttt{clone, finalize, getClass, notify, notifyAll, toString, wait, wait}

\textbf{Field Detail}

\textbf{operand}

protected Matcher\texttt{<T extends Key>} \texttt{operand}

\textbf{Constructor Detail}

\textbf{NotMatcher}

protected NotMatcher(Matcher\texttt{<T>} operand)

\textbf{Method Detail}

\textbf{not}

public static <U extends Key> NotMatcher\texttt{<U> not(Matcher\texttt{<U>} operand)}

Create a NotMatcher that reverses the result of the given matcher.

\textbf{isMatch}
public boolean isMatch(T key)

Specified by:
    isMatch in interface Matcher<T extends Key>

getOperand

public Matcher<T> getOperand()

hashCode

public int hashCode()

Specified by:
    hashCode in interface Matcher<T extends Key>

Overrides:
    hashCode in class Object

equals

public boolean equals(Object obj)

Specified by:
    equals in interface Matcher<T extends Key>

Overrides:
    equals in class Object
org.quartz.impl.matchers  Class OrMatcher\<T\> extends Key\>

java.lang.Object
  org.quartz.impl.matchers OrMatcher<T>

All Implemented Interfaces:
    Serializable, Matcher<T>

public class OrMatcher\<T\> extends Object
  implements Matcher<T>

Matches using an OR operator on two Matcher operands.

Author:
  jhouse

See Also:
  Serialized Form

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Matcher&lt;T&gt; leftOperand</td>
</tr>
<tr>
<td>protected Matcher&lt;T&gt; rightOperand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected OrMatcher(Matcher&lt;T&gt; leftOperand, Matcher&lt;T&gt; rightOperand)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(Object obj)</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td><code>getLeftOperand()</code></td>
</tr>
<tr>
<td><code>getRightOperand()</code></td>
</tr>
<tr>
<td><code>hashCode()</code></td>
</tr>
<tr>
<td><code>isMatch(T key)</code></td>
</tr>
</tbody>
</table>

```java
<
static
<U extends Key>
OrMatcher<U>
or(Matcher<U> leftOperand, Matcher<U> rightOperand)
```

Create an OrMatcher that depends upon the result of at least one of the given matchers.

**Methods inherited from class java.lang.Object**

`clone, finalize, getClass, notify, notifyAll, toString, wait, wait`

### Field Detail

**leftOperand**

protected `Matcher<T extends Key> leftOperand`

**rightOperand**

protected `Matcher<T extends Key> rightOperand`

### Constructor Detail

**OrMatcher**

protected `OrMatcher(Matcher<T> leftOperand, Matcher<T> rightOperand)`
Method Detail

or

public static <U extends Key> OrMatcher<U> or(Matcher<U> leftOperand, Matcher<U> rightOperand)

Create an OrMatcher that depends upon the result of at least one of the given matchers.

isMatch

public boolean isMatch(T key)

Specified by:
isMatch in interface Matcher<T extends Key>

getLeftOperand

generic Matcher<T> getLeftOperand()

getRightOperand

generic Matcher<T> getRightOperand()

hashCode

public int hashCode()

Specified by:
hashCode in interface Matcher<T extends Key>

Overrides:
hashCode in class Object
public boolean equals(Object obj)

Specified by:
equals in interface Matcher<T extends Key>

Overrides:
equals in class Object
org.quartz.impl.matchers

Classes
- AndMatcher
- EverythingMatcher
- GroupMatcher
- KeyMatcher
- NameMatcher
- NotMatcher
- OrMatcher
- StringMatcher

Enums
- StringMatcher.StringOperatorName
## Class Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndMatcher&lt;T extends Key&gt;</td>
<td>Matches using an AND operator on two Matcher operands.</td>
</tr>
<tr>
<td>EverythingMatcher&lt;T extends Key&gt;</td>
<td>Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td>GroupMatcher&lt;T extends Key&gt;</td>
<td>Matches on group (ignores name) property of Keys.</td>
</tr>
<tr>
<td>KeyMatcher&lt;T extends Key&gt;</td>
<td>Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td>NameMatcher&lt;T extends Key&gt;</td>
<td>Matches on name (ignores group) property of Keys.</td>
</tr>
<tr>
<td>NotMatcher&lt;T extends Key&gt;</td>
<td>Matches using an NOT operator on another Matcher.</td>
</tr>
<tr>
<td>OrMatcher&lt;T extends Key&gt;</td>
<td>Matches using an OR operator on two Matcher operands.</td>
</tr>
<tr>
<td>StringMatcher&lt;T extends Key&gt;</td>
<td>An abstract base class for some types of matchers.</td>
</tr>
</tbody>
</table>

## Enum Summary

| StringMatcher.StringOperatorName | |

---

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.impl.matchers

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.matchers.**AndMatcher**<T> (implements org.quartz.**Matcher**<T>)
  - org.quartz.impl.matchers.**EverythingMatcher**<T> (implements org.quartz.**Matcher**<T>)
  - org.quartz.impl.matchers.**KeyMatcher**<T> (implements org.quartz.**Matcher**<T>)
  - org.quartz.impl.matchers.**NotMatcher**<T> (implements org.quartz.**Matcher**<T>)
  - org.quartz.impl.matchers.**OrMatcher**<T> (implements org.quartz.**Matcher**<T>)
  - org.quartz.impl.matchers.**StringMatcher**<T> (implements org.quartz.**Matcher**<T>)
    - org.quartz.impl.matchers.**GroupMatcher**<T>
    - org.quartz.impl.matchers.**NameMatcher**<T>
Enum Hierarchy

- `java.lang.Object`
  - `org.quartz.impl.matchers.StringMatcher.StringOperatorName`
# Uses of Package

**org.quartz.impl.matchers**

## Packages that use **org.quartz.impl.matchers**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.core</strong></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl</strong></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.matchers</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.simpl</strong></td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Classes in **org.quartz.impl.matchers** used by

**org.quartz**

**GroupMatcher**

Matches on group (ignores name) property of Keys.

## Classes in **org.quartz.impl.matchers** used by

**org.quartz.core**

**GroupMatcher**

Matches on group (ignores name) property of Keys.
### Classes in `org.quartz.impl.matchers` used by
`org.quartz.impl`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupMatcher</td>
<td>Matches on group (ignores name) property of Keys.</td>
</tr>
</tbody>
</table>

### Classes in `org.quartz.impl.matchers` used by
`org.quartz.impl.jdbcjobstore`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupMatcher</td>
<td>Matches on group (ignores name) property of Keys.</td>
</tr>
</tbody>
</table>

### Classes in `org.quartz.impl.matchers` used by
`org.quartz.impl.matchers`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndMatcher</td>
<td>Matches using an AND operator on two Matcher operands.</td>
</tr>
<tr>
<td>EverythingMatcher</td>
<td>Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td>GroupMatcher</td>
<td>Matches on group (ignores name) property of Keys.</td>
</tr>
<tr>
<td>KeyMatcher</td>
<td>Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td>NameMatcher</td>
<td>Matches on name (ignores group) property of Keys.</td>
</tr>
<tr>
<td>NotMatcher</td>
<td>Matches using an NOT operator on another Matcher.</td>
</tr>
<tr>
<td>OrMatcher</td>
<td>Matches using an OR operator on two Matcher operands.</td>
</tr>
<tr>
<td>StringMatcher</td>
<td>An abstract base class for some types of matchers.</td>
</tr>
<tr>
<td><code>StringMatcher.StringOperatorName</code></td>
<td></td>
</tr>
</tbody>
</table>
Classes in `org.quartz.impl.matchers` used by `org.quartz.simpl`

**GroupMatcher**

Matches on group (ignores name) property of Keys.

Class `StringMatcher<T extends Key>`

`java.lang.Object`  
  └ `org.quartz.impl.matchers.StringMatcher<T>`

All Implemented Interfaces:  
  `Serializable, Matcher<T>`

Direct Known Subclasses:  
  `GroupMatcher, NameMatcher`

---

public abstract class `StringMatcher<T extends Key>`

extends `Object`

implements `Matcher<T>`

An abstract base class for some types of matchers.

Author:  
  jhouse

See Also:  
  `Serialized Form`

---

**Nested Class Summary**

<table>
<thead>
<tr>
<th>static class</th>
<th><code>StringMatcher.StringOperatorName</code></th>
</tr>
</thead>
</table>

**Field Summary**

<table>
<thead>
<tr>
<th>protected <code>String</code></th>
<th><code>compareTo</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StringMatcher.StringOperatorName</code></td>
<td><code>compareWith</code></td>
</tr>
</tbody>
</table>
## Constructor Summary

| protected StringMatcher(String compareTo, StringMatcher.StringOperatorName compareWith) |

## Method Summary

| boolean equals(Object obj) |
| String getCompareToValue() |
| StringMatcher.StringOperatorName getCompareWithOperator() |
| protected abstract String getValue(T key) |
| int hashCode() |
| boolean isMatch(T key) |

## Methods inherited from class java.lang.Object

clone, finalize, getClass, notify, notifyAll, toString, wait, wait

## Field Detail

### compareTo

protected String compareTo

### compareWith

protected StringMatcher.StringOperatorName compareWith
Constructor Detail

StringMatcher

protected StringMatcher(String compareTo, StringMatcher.StringOperatorName compareWith)

Method Detail

getValue

protected abstract String getValue(T key)

isMatch

public boolean isMatch(T key)

Specified by:

isMatch in interface Matcher<T extends Key>

hashCode

public int hashCode()

Specified by:

hashCode in interface Matcher<T extends Key>

Overrides:

hashCode in class Object

equals

public boolean equals(Object obj)

Specified by:

equals in interface Matcher<T extends Key>
Overrides:
  equals in class Object

getCompareToValue

public String getCompareToValue()

getCompareWithOperator

public StringMatcher.StringOperatorName getCompareWithOperator()
org.quartz.impl.matchers Enum
StringMatcher.StringOperatorName

java.lang.Object
\_ java.lang.Enum<StringMatcher.StringOperatorName>
\_ org.quartz.impl.matchers.StringMatcher.StringOperatorName

All Implemented Interfaces:
Serializable, Comparable<StringMatcher.StringOperatorName>

Enclosing class:
StringMatcher<T extends Key>

---

public static enum StringMatcher.StringOperatorName
extends Enum<StringMatcher.StringOperatorName>

---

**Enum Constant Summary**

<table>
<thead>
<tr>
<th>CONTAINS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENDS_WITH</td>
</tr>
<tr>
<td>EQUALS</td>
</tr>
<tr>
<td>STARTS_WITH</td>
</tr>
</tbody>
</table>

---

**Method Summary**

<table>
<thead>
<tr>
<th>abstract boolean evaluate(String value, String compareTo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static StringMatcher.StringOperatorName.valueOf(String name)</td>
</tr>
</tbody>
</table>

Returns the enum constant of this
type with the specified name.

<table>
<thead>
<tr>
<th>static StringMatcher.StringOperatorName[] values()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Enum**
close, compareTo, equals, finalize, getDeclaringClass, hashCode,
name, ordinal, toString, valueOf

**Methods inherited from class java.lang.Object**
getClass, notify, notifyAll, wait, wait, wait

---

**Enum Constant Detail**

**EQUALS**

public static final StringMatcher.StringOperatorName EQUALS

**STARTS_WITH**

public static final StringMatcher.StringOperatorName STARTS_WITH

**ENDS_WITH**

public static final StringMatcher.StringOperatorName ENDS_WITH

**CONTAINS**

public static final StringMatcher.StringOperatorName CONTAINS

---
values

public static StringMatcher.StringOperatorName[] values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

for (StringMatcher.StringOperatorName c : StringMatcher.StringOperatorName.values())
    System.out.println(c);

Returns:
an array containing the constants of this enum type, in the order they are declared

valueOf

public static StringMatcher.StringOperatorName.valueOf(String name)

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
name - the name of the enum constant to be returned.

Returns:
the enum constant with the specified name

Throws:
IllegalArgumentException - if this enum type has no constant with the specified name
NullPointerException - if the argument is null

evaluate

public abstract boolean evaluate(String value, String compareTo)
Overview | Package | Use | Tree | Deprecated | Index | Help
--- | --- | --- | --- | --- | --- | ---
PREV CLASS | NEXT CLASS | SUMMARY: NESTED | ENUM CONSTANTS | FIELD | METHOD | FRAMES | NO FRAMES | DETAIL | ENUM CONSTANTS | FIELD | METHOD

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.matchers.AndMatcher

Packages that use AndMatcher

org.quartz.impl.matchers

Uses of AndMatcher in org.quartz.impl.matchers

Methods in org.quartz.impl.matchers that return AndMatcher

**AndMatcher.**

```java
public static <U extends Key>
    AndMatcher and(Matcher<U> leftOperand, Matcher<U> rightOperand)
```

Create an AndMatcher that depends upon the result of both of the given matchers.

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
# Uses of Class

org.quartz.impl.matchers.EverythingMatcher

## Packages that use EverythingMatcher

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.matchers</td>
</tr>
</tbody>
</table>

## Uses of EverythingMatcher in org.quartz.impl.matchers

## Methods in org.quartz.impl.matchers that return EverythingMatcher

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EverythingMatcher.&lt;JobKey&gt;.allJobs()</code></td>
<td>Create an <code>EverythingMatcher</code> that matches all jobs.</td>
</tr>
<tr>
<td><code>EverythingMatcher.&lt;TriggerKey&gt;.allTriggers()</code></td>
<td>Create an <code>EverythingMatcher</code> that matches all triggers.</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.impl.matchers.GroupMatcher**

## Packages that use **GroupMatcher**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz</code></td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td><code>org.quartz.core</code></td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td><code>org.quartz.impl</code></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td><code>org.quartz.impl.jdbcjobstore</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.impl.matchers</code></td>
<td></td>
</tr>
<tr>
<td><code>org.quartz.simpl</code></td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Uses of **GroupMatcher** in **org.quartz**

### Methods in **org.quartz** with parameters of type **GroupMatcher**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Set&lt;JobKey&gt;</code> Scheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)`</td>
<td>Get the keys of all the JobDetails in the matching groups.</td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt;</code> Scheduler.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)`</td>
<td>Get the names of all the Triggers in the given group.</td>
</tr>
<tr>
<td><code>void</code> Scheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)`</td>
<td>Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.</td>
</tr>
<tr>
<td><code>void</code> Scheduler.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)`</td>
<td>Pause all of the Triggers in the groups matching.</td>
</tr>
</tbody>
</table>
void Scheduler.resumeJobs(GroupMatcher<JobKey> matcher)
Resume (un-pause) all of the JobDetails in matching groups.

void Scheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)
Resume (un-pause) all of the Triggers in matching groups.

---

**Uses of `GroupMatcher` in `org.quartz.core`**

<table>
<thead>
<tr>
<th>Methods in <code>org.quartz.core</code> with parameters of type <code>GroupMatcher</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;JobKey&gt; QuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>Get the names of all the Jobs in the matching groups.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt; RemotableQuartzScheduler.getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt; QuartzScheduler.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>Get the names of all the Triggers in the matching groups.</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt; RemotableQuartzScheduler.getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>void QuartzSchedulerMBeanImpl.pauseJobs(GroupMatcher matcher)</td>
</tr>
<tr>
<td>Pause all of the JobDetails in the matching groups - by pausing all of their Triggers.</td>
</tr>
<tr>
<td>void RemotableQuartzScheduler.pauseJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
<tr>
<td>void QuartzScheduler.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>Pause all of the Triggers in the matching groups.</td>
</tr>
<tr>
<td>void RemotableQuartzScheduler.pauseTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>void QuartzSchedulerMBeanImpl.resumeJobs(GroupMatcher matcher)</td>
</tr>
<tr>
<td>Resume (un-pause) all of the JobDetails in the matching groups.</td>
</tr>
<tr>
<td>void RemotableQuartzScheduler.resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</td>
</tr>
</tbody>
</table>

---
void QuartzScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)
Resume (un-pause) all of the Triggers in the matching groups.

void RemotableQuartzScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)

Uses of GroupMatcher in org.quartz.impl

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl with parameters of type GroupMatcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;JobKey&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
</tbody>
</table>

Sets
Calls the equivalent method on the 'proxied' QuartzScheduler SchedulingContext associated with this instance.

void RemoteMBeanScheduler.pauseTriggers(GroupMatcher<TriggerKey> matcher)

void RemoteScheduler.resumeJobs(GroupMatcher<JobKey> matcher)

void StdScheduler.resumeJobs(GroupMatcher<JobKey> matcher)

void RemoteMBeanScheduler.resumeJobs(GroupMatcher<JobKey> matcher)

void RemoteScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)

void StdScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)

void RemoteMBeanScheduler.resumeTriggers(GroupMatcher<TriggerKey> matcher)

Uses of GroupMatcher in org.quartz.impl.jdbcjobstore

Methods in org.quartz.impl.jdbcjobstore with parameters of type GroupMatcher

int StdJDBCDelegate.deletePausedTriggerGroup(Connection conn, GroupMatcher<TriggerKey> matcher)

int DriverDelegate.deletePausedTriggerGroup(Connection conn, GroupMatcher<TriggerKey> matcher)

Set<JobKey> JobStoreSupport.getJobKeys(GroupMatcher<JobKey> matcher)

protected Set<JobKey> JobStoreSupport.getJobNames(Connection conn, GroupMatcher<JobKey> matcher)
<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td>JobStoreSupport.getTriggerKeys</td>
<td>Get the names of all of the Triggers that match the given GroupMatcher.</td>
</tr>
<tr>
<td>protected Set&lt;TriggerKey&gt;</td>
<td>JobStoreSupport.getTriggerNames</td>
<td>Set&lt;TriggerKey&gt; JobStoreSupport.getTriggerKeys(TriggerKey&gt; matcher)</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.pauseJobs</td>
<td>Pause all of the Jobs matching the given groupMatcher - by pausing their Triggers.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.pauseTriggerGroup</td>
<td>Pause all of the Triggers matching the given groupMatcher.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.pauseTriggers</td>
<td>Pause all of the Triggers matching the given groupMatcher.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.resumeJobs</td>
<td>Resume (un-pause) all of the Jobs in the given group.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.resumeTriggerGroup</td>
<td>Resume (un-pause) all of the Triggers matching the given groupMatcher.</td>
</tr>
<tr>
<td>Set&lt;String&gt;</td>
<td>JobStoreSupport.resumeTriggers</td>
<td>Resume (un-pause) all of the Triggers matching the given groupMatcher.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt;</td>
<td>StdJDBCDelegate.selectJobsInGroup</td>
<td>Select all of the jobs contained in a given group.</td>
</tr>
<tr>
<td>Set&lt;JobKey&gt;</td>
<td>DriverDelegate.selectJobsInGroup</td>
<td>Select all of the jobs contained in a given group.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>StdJDBCDelegate.selectTriggerGroups</td>
<td>Select all of the TriggerGroups contained in a given group.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>DriverDelegate.selectTriggerGroups</td>
<td>Select all of the TriggerGroups contained in a given group.</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td>StdJDBCDelegate.selectTriggersInGroup</td>
<td>Select all of the triggers contained in a given group.</td>
</tr>
<tr>
<td>Set&lt;TriggerKey&gt;</td>
<td>DriverDelegate.selectTriggersInGroup</td>
<td>Select all of the triggers contained in a given group.</td>
</tr>
</tbody>
</table>
GroupMatcher<TriggerKey> matcher

Select all of the triggers contained in a given group.

protected String StdJDBCDelegate.toSqlLikeClause(GroupMatcher matcher)

int StdJDBCDelegate.updateTriggerGroupStateFromOtherState(GroupMatcher<TriggerKey> matcher, String newState, String oldState2, String oldState3)

Update all of the triggers of the given group to the given new state, if they are in the given old state.

int DriverDelegate.updateTriggerGroupStateFromOtherState(GroupMatcher<TriggerKey> matcher, String newState, String oldState2, String oldState3)

Update all of the triggers of the given group to the given new state, if they are in the given old state.

int DriverDelegate.updateTriggerGroupStateFromOtherStates(GroupMatcher<TriggerKey> matcher, String newState, String oldState2, String oldState3)

Update all triggers in the given group to the given new state, if they are of the given old states.

Uses of GroupMatcher in org.quartz.impl.matchers

Methods in org.quartz.impl.matchers that return GroupMatcher

<table>
<thead>
<tr>
<th>static GroupMatcher</th>
<th>GroupMatcher.groupContains(String compareTo)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a GroupMatcher that matches groups containing the given string.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static GroupMatcher</th>
<th>GroupMatcher.groupEndsWith(String compareTo)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a GroupMatcher that matches groups ending with the given string.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static GroupMatcher</th>
<th>GroupMatcher.groupEquals(String compareTo)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a GroupMatcher that matches groups equaling the given string.</td>
</tr>
</tbody>
</table>
static GroupMatcher.groupStartsWith(String compareTo)
    Create a GroupMatcher that matches groups starting with the given string.

**Uses of GroupMatcher in org.quartz.simpl**

<table>
<thead>
<tr>
<th>Methods in org.quartz.simpl with parameters of type GroupMatcher</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set&lt;JobKey&gt;</strong></td>
</tr>
<tr>
<td>Get the names of all of the Job s that match the given groupMatcher.</td>
</tr>
<tr>
<td><strong>Set&lt;TriggerKey&gt;</strong></td>
</tr>
<tr>
<td>Get the names of all of the Trigger s that match the given groupMatcher.</td>
</tr>
<tr>
<td><strong>List&lt;String&gt;</strong></td>
</tr>
<tr>
<td>Pause all of the JobDetails in the given group - by pausing all of their Triggers.</td>
</tr>
<tr>
<td><strong>List&lt;String&gt;</strong></td>
</tr>
<tr>
<td>Pause all of the known Triggers matching.</td>
</tr>
<tr>
<td><strong>Collection&lt;String&gt;</strong></td>
</tr>
<tr>
<td>Resume (un-pause) all of the JobDetails in the given group.</td>
</tr>
<tr>
<td><strong>List&lt;String&gt;</strong></td>
</tr>
<tr>
<td>Resume (un-pause) all of the Triggers in the given group.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.matchers.KeyMatcher

Packages that use **KeyMatcher**

org.quartz.impl.matchers

Uses of **KeyMatcher** in org.quartz.impl.matchers

Methods in org.quartz.impl.matchers that return **KeyMatcher**

```java
static <U extends Key> KeyMatcher<br>KeyMatcher.keyEquals(U compareTo)
```

Create a KeyMatcher that matches Keys that equal the given key.

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

org.quartz.impl.matchers.NameMatcher

## Packages that use NameMatcher

<table>
<thead>
<tr>
<th>Package</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.matchers</td>
<td>NameMatcher</td>
</tr>
</tbody>
</table>

## Uses of NameMatcher in org.quartz.impl.matchers

### Methods in org.quartz.impl.matchers that return NameMatcher

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static NameMatcher NameMatcher.&lt;tt&gt;nameContains&lt;/tt&gt;(String compareTo)</td>
<td>Create a NameMatcher that matches names containing the given string.</td>
</tr>
<tr>
<td>static NameMatcher NameMatcher.&lt;tt&gt;nameEndsWith&lt;/tt&gt;(String compareTo)</td>
<td>Create a NameMatcher that matches names ending with the given string.</td>
</tr>
<tr>
<td>static NameMatcher NameMatcher.&lt;tt&gt;nameEquals&lt;/tt&gt;(String compareTo)</td>
<td>Create a NameMatcher that matches names equaling the given string.</td>
</tr>
<tr>
<td>static NameMatcher NameMatcher.&lt;tt&gt;nameStartsWith&lt;/tt&gt;(String compareTo)</td>
<td>Create a NameMatcher that matches names starting with the given string.</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Class org.quartz.impl.matchers.NotMatcher

Packages that use NotMatcher

| org.quartz.impl.matchers |

Uses of NotMatcher in org.quartz.impl.matchers

<table>
<thead>
<tr>
<th>Methods in org.quartz.impl.matchers that return NotMatcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>static &lt;U extends Key&gt; NotMatcher.&lt;U&gt; not (Matcher&lt;U&gt; operand)</td>
</tr>
<tr>
<td>Create a NotMatcher that reverses the result of the given matcher.</td>
</tr>
</tbody>
</table>

Overview Package Class Tree Deprecated Index Help

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.matchers.OrMatcher

Packages that use **OrMatcher**

- org.quartz.impl.matchers

Uses of **OrMatcher** in org.quartz.impl.matchers

Methods in **org.quartz.impl.matchers** that return **OrMatcher**

```java
public static <U extends Key> OrMatcher<U> or(
        Matcher<U> leftOperand, Matcher<U> rightOperand)
```

Create an OrMatcher that depends upon the result of at least one of the given matchers.

---

Copyright 2001-2011, Terracotta, Inc.
Packages that use **StringMatcher**

org.quartz.impl.matchers

Uses of **StringMatcher** in **org.quartz.impl.matchers**

<table>
<thead>
<tr>
<th>Subclasses of <strong>StringMatcher</strong> in <strong>org.quartz.impl.matchers</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>class <strong>GroupMatcher&lt;T extends Key&gt;</strong></td>
</tr>
<tr>
<td>Matches on group (ignores name) property of Keys.</td>
</tr>
<tr>
<td>class <strong>NameMatcher&lt;T extends Key&gt;</strong></td>
</tr>
<tr>
<td>Matches on name (ignores group) property of Keys.</td>
</tr>
</tbody>
</table>

Overview  Package  Class  Tree  Deprecated  Index  Help
PREV  NEXT  FRAMES  NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class org.quartz.impl.matchers.StringMatcher.StringOperatorName

## Packages that use org.quartz.impl.matchers.StringMatcher.StringOperatorName

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.matchers</td>
</tr>
</tbody>
</table>

## Uses of StringMatcher.StringOperatorName in org.quartz.impl.matchers

## Fields in org.quartz.impl.matchers declared as StringMatcher.StringOperatorName

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected StringMatcher.StringOperatorName</td>
<td>StringMatcher.compareWith</td>
</tr>
</tbody>
</table>

## Methods in org.quartz.impl.matchers that return StringMatcher.StringOperatorName

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static StringMatcher.StringOperatorName</td>
<td>StringMatcher.StringOperatorName</td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static StringMatcher.StringOperatorName[]</td>
<td>StringMatcher.StringOperatorName[]</td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

## Constructors in org.quartz.impl.matchers with parameters of type StringMatcher.StringOperatorName

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupMatcher(String compareTo,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>StringMatcher</code></td>
<td><code>StringMatcher(String compareTo, StringMatcher.StringOperatorName compareWith)</code></td>
<td></td>
</tr>
<tr>
<td><code>NameMatcher</code></td>
<td><code>NameMatcher(String compareTo, StringMatcher.StringOperatorName compareWith)</code></td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
org.quartz.impl.triggers Class AbstractTrigger<T extends Trigger>

java.lang.Object
   org.quartz.impl.triggers.AbstractTrigger<T>

All Implemented Interfaces:
   Serializable, Cloneable, Comparable<Trigger>,
   org.quartz.spi.MutableTrigger, org.quartz.spi.OperableTrigger, Trigger

Direct Known Subclasses:
   CalendarIntervalTriggerImpl, CronTriggerImpl, SimpleTriggerImpl

public abstract class AbstractTrigger<T extends Trigger>

extends Object
implements org.quartz.spi.OperableTrigger

The base abstract class to be extended by all Triggers.

Triggers s have a name and group associated with them, which should uniquely identify them within a single Scheduler.

Triggers are the 'mechanism' by which Job s are scheduled. Many Trigger s can point to the same Job, but a single Trigger can only point to one Job.

Triggers can 'send' parameters/data to Jobs by placing contents into the JobDataMap on the Trigger.

Author:
   James House, Sharada Jambula

See Also:
   SimpleTrigger, CronTrigger, NthIncludedDayTrigger, TriggerUtils, JobDataMap, JobExecutionContext, Serialized Form

Nested Class Summary
### Nested classes/interfaces inherited from interface org.quartz.Trigger

- Trigger.CompletionEditedExecutionInstruction
- Trigger.TriggerState
- Trigger.TriggerTimeComparator

### Field Summary

### Fields inherited from interface org.quartz.Trigger

- DEFAULT_PRIORITY
- MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY
- MISFIRE_INSTRUCTION_SMART_POLICY

### Constructor Summary

#### AbstractTrigger()
Create a Trigger with no specified name, group, or JobDetail.

#### AbstractTrigger(String name)
Create a Trigger with the given name, and default group.

#### AbstractTrigger(String name, String group)
Create a Trigger with the given name, and group.

#### AbstractTrigger(String name, String group, String jobName, String jobGroup)
Create a Trigger with the given name, and group.

### Method Summary

#### Object clone()

#### int compareTo(Trigger other)
Compare the next fire time of this Trigger another by comparing their keys, or in other words, sorts them according to the natural (i.e., abstract) date.

#### abstract Date computeFirstFireTime(Calendar calendar)
This method should not be used by the Quartz client.

#### boolean equals(Object o)
Trigger equality is based upon the equality of the TriggerKey.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Trigger.CompletedExecutionInstruction</code></td>
<td><code>executionComplete(JobExecutionContext context, JobExecutionException result)</code></td>
</tr>
<tr>
<td></td>
<td>This method should not be used by the Quartz client.</td>
</tr>
<tr>
<td><code>String getCalendarName()</code></td>
<td>Get the name of the calendar associated with this Trigger.</td>
</tr>
<tr>
<td><code>String getDescription()</code></td>
<td>Return the description given to the Trigger by its creator (if any).</td>
</tr>
<tr>
<td><code>abstract Date getEndTime()</code></td>
<td>Get the time at which the Trigger should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).</td>
</tr>
<tr>
<td><code>abstract Date getFinalFireTime()</code></td>
<td>Returns the last time at which the Trigger will repeat indefinitely, null will be returned if the Trigger will repeat indefinitely.</td>
</tr>
<tr>
<td><code>String getFireInstanceId()</code></td>
<td>This method should not be used by the Quartz client.</td>
</tr>
<tr>
<td><code>abstract Date getFireTimeAfter(Date afterTime)</code></td>
<td>Returns the next time at which the Trigger will fire after the given time.</td>
</tr>
<tr>
<td><code>String getFullJobName()</code></td>
<td>Returns the 'full name' of the Job that the Trigger points to, in the format &quot;group.name&quot;.</td>
</tr>
<tr>
<td><code>String getFullName()</code></td>
<td>Returns the 'full name' of the Trigger in the format &quot;group.name&quot;.</td>
</tr>
<tr>
<td><code>String getGroup()</code></td>
<td>Get the group of this Trigger.</td>
</tr>
<tr>
<td><code>JobDataMap getJobDataMap()</code></td>
<td>Get the JobDataMap that is associated with this Trigger.</td>
</tr>
<tr>
<td><code>String getJobGroup()</code></td>
<td>Get the name of the associated JobDetail.</td>
</tr>
<tr>
<td><code>JobKey getJobKey()</code></td>
<td></td>
</tr>
</tbody>
</table>
### String `getJobName()`
- Get the name of the associated JobDetail.

### `TriggerKey` `getKey()`

### int `getMisfireInstruction()`
- Get the instruction the Scheduler should use for handling misfire situations for this Trigger.
- The concrete Trigger type that you are using will have a set of additional `MISFIRE_INSTRUCTION_XXX` that may be passed to this method.

### String `getName()`
- Get the name of this Trigger.

### Abstract `Date` `getNextFireTime()`
- Returns the next time at which the Trigger scheduled to fire.

### Abstract `Date` `getPreviousFireTime()`
- Returns the previous time at which the Trigger fired.

### int `getPriority()`
- The priority of a Trigger acts as a tiebreaker that if two Triggers have the same scheduled fire time, then the one with the higher priority will get first access to a worker thread.

### Abstract `ScheduleBuilder<T>` `getScheduleBuilder()`
- Get a `ScheduleBuilder` that is configured to produce a schedule identical to this trigger's schedule.

### Abstract `Date` `getStartTime()`
- Get the time at which the Trigger should occur.

### `TriggerBuilder<T>` `getTriggerBuilder()`
- Get a `TriggerBuilder` that is configured to produce a Trigger identical to this one.

### int `hashCode()`

### Abstract boolean `mayFireAgain()`
- Used by the Scheduler to determine whether it is possible for this Trigger to fire again.
void setCalendarName(String calendarName)
    Associate the Calendar with the given name with this Trigger.

void setDescription(String description)
    Set a description for the Trigger instance - may be useful for remembering/displaying the purpose of the trigger, though the description has no meaning to Quartz.

abstract void setEndTime(Date endTime)
    Set the time at which the Trigger should quit repeating - regardless of any remaining repeat settings.

void setFireInstanceId(String id)
    This method should not be used by the Quartz client.

void setGroup(String group)
    Set the name of this Trigger.

void setJobDataMap(JobDataMap jobDataMap)
    Set the JobDataMap to be associated with the Trigger.

void setJobGroup(String jobGroup)
    Set the name of the associated JobDetail.

void setJobKey(JobKey key)

void setJobName(String jobName)
    Set the name of the associated JobDetail.

void setKey(TriggerKey key)

void setMisfireInstruction(int misfireInstruction)
    Set the instruction the Scheduler should follow for handling misfire situations for this Trigger. The concrete Trigger type that you are using will have a set of additional MISFIRE_INSTRUCTION_XXX that may be passed to this method.

void setName(String name)
    Set the name of this Trigger.

void setPriority(int priority)
The priority of a Trigger acts as a tie breaker such that if two Triggers have the same scheduled time, then Quartz will do its best to give the one with higher priority first access to a worker thread.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract void setStartTime (Date startTime)</strong></td>
<td>The time at which the trigger's scheduling should start.</td>
</tr>
<tr>
<td><strong>String toString()</strong></td>
<td>Return a simple string representation of this object.</td>
</tr>
<tr>
<td><strong>Abstract void triggered (Calendar calendar)</strong></td>
<td>This method should not be used by the Quartz client.</td>
</tr>
<tr>
<td><strong>Abstract void updateAfterMisfire (Calendar cal)</strong></td>
<td>This method should not be used by the Quartz client.</td>
</tr>
<tr>
<td><strong>Abstract void updateWithNewCalendar (Calendar cal, long misfireThreshold)</strong></td>
<td>This method should not be used by the Quartz client.</td>
</tr>
<tr>
<td><strong>Void validate()</strong></td>
<td>Validates whether the properties of the trigger are valid for submission into a Scheduler.</td>
</tr>
<tr>
<td><strong>Protected abstract boolean validateMisfireInstruction (int misfireInstruction)</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**
final `Object finalize`, `Object getClass`, `Object notify`, `Object notifyAll`, `Object wait`, `Object wait`, `Object wait`

**Methods inherited from interface org.quartz.spi.OperableTrigger**
`setNextFireTime`, `setPreviousFireTime`

**Constructor Detail**

**AbstractTrigger**

```java
public AbstractTrigger()
```
Create a Trigger with no specified name, group, or JobDetail.

Note that the `setName(String), setGroup(String)` and the `setJobName(String)` and `setJobGroup(String)` methods must be called before the Trigger can be placed into a `Scheduler`.

---

**AbstractTrigger**

```java
public AbstractTrigger(String name)
```

Create a Trigger with the given name, and default group.

Note that the `setJobName(String)` and `setJobGroup(String)` methods must be called before the Trigger can be placed into a `Scheduler`.

**Parameters:**
- `group` - if null, Scheduler.DEFAULT_GROUP will be used.

**Throws:**
- `IllegalArgumentException` - if name is null or empty, or the group is an empty string.

---

**AbstractTrigger**

```java
public AbstractTrigger(String name, String group)
```

Create a Trigger with the given name, and group.

Note that the `setJobName(String)` and `setJobGroup(String)` methods must be called before the Trigger can be placed into a `Scheduler`.

**Parameters:**
- `group` - if null, Scheduler.DEFAULT_GROUP will be used.

**Throws:**
- `IllegalArgumentException` - if name is null or empty, or the group is an empty string.
AbstractTrigger

public AbstractTrigger(String name,
                        String group,
                        String jobName,
                        String jobGroup)

Create a Trigger with the given name, and group.

**Parameters:**
- group - if null, Scheduler.DEFAULT_GROUP will be used.

**Throws:**
- IllegalArgumentException - if name is null or empty, or the group is an empty string.

### Method Detail

**getName**

public String getName()

Get the name of this Trigger.

**setName**

public void setName(String name)

Set the name of this Trigger.

**Throws:**
- IllegalArgumentException - if name is null or empty.

**getGroup**

public String getGroup()

Get the group of this Trigger.
**setGroup**

```java
public void setGroup(String group)
```

Set the name of this Trigger.

**Parameters:**
- `group` - if null, Scheduler.DEFAULT_GROUP will be used.

**Throws:**
- `IllegalArgumentException` - if group is an empty string.

**setKey**

```java
public void setKey(TriggerKey key)
```

**Specified by:**
- `setKey` in interface `org.quartz.spi.MutableTrigger`

**getJobName**

```java
public String getJobName()
```

Get the name of the associated JobDetail.

**setJobName**

```java
public void setJobName(String jobName)
```

Set the name of the associated JobDetail.

**Throws:**
- `IllegalArgumentException` - if jobName is null or empty.

**getJobGroup**
public String getJobGroup()

Get the name of the associated JobDetail's group.

---

**setJobGroup**

public void setJobGroup(String jobGroup)

Set the name of the associated JobDetail's group.

**Parameters:**
jobGroup - if null, Scheduler.DEFAULT_GROUP will be used.

**Throws:**
IllegalArgumentException - if group is an empty string.

---

**setJobKey**

public void setJobKey(JobKey key)

**Specified by:**
setJobKey in interface org.quartz.spi.MutableTrigger

---

**getFullName**

public String getFullName()

Returns the 'full name' of the Trigger in the format "group.name".

---

**getKey**

public TriggerKey getKey()

**Specified by:**
getKey in interface Trigger
**getJobKey**

public `JobKey` getJobKey()

*Specified by:*
`getJobKey` in interface `Trigger`

**getFullJobName**

public `String` getFullJobName()

Returns the 'full name' of the Job that the Trigger points to, in the format "group.name".

**getDescription**

public `String` getDescription()

Return the description given to the Trigger instance by its creator (if any).

*Specified by:*
`getDescription` in interface `Trigger`

*Returns:*
null if no description was set.

**setDescription**

public void `setDescription`( `String` description)

Set a description for the Trigger instance - may be useful for remembering/displaying the purpose of the trigger, though the description has no meaning to Quartz.

*Specified by:*
`setDescription` in interface `org.quartz.spi.MutableTrigger`
**setCalendarName**

public void setCalendarName(String calendarName)

Associate the calendar with the given name with this Trigger.

**Specified by:**
setCalendarName in interface org.quartz.spi.MutableTrigger

**Parameters:**
calendarName - use null to dis-associate a Calendar.

---

**getCalendarName**

public String getCalendarName()

Get the name of the calendar associated with this Trigger.

**Specified by:**
getCalendarName in interface Trigger

**Returns:**
null if there is no associated Calendar.

---

**getJobDataMap**

public JobDataMap getJobDataMap()

Get the JobDataMap that is associated with the Trigger.

Changes made to this map during job execution are not re-persisted, and in fact typically result in an IllegalStateException.

**Specified by:**
getJobDataMap in interface Trigger

---

**setJobDataMap**

public void setJobDataMap(JobDataMap jobDataMap)
Set the JobDataMap to be associated with the Trigger.

**Specified by:**
*setJobDataMap in interface org.quartz.spi.MutableTrigger*

---

**getPriority**

```java
public int getPriority()
```

The priority of a Trigger acts as a tiebreaker such that if two Triggers have the same scheduled fire time, then the one with the higher priority will get first access to a worker thread.

If not explicitly set, the default value is 5.

**Specified by:**
*getPriority in interface Trigger*

**See Also:**
*Trigger.DEFAULT_PRIORITY*

---

**setPriority**

```java
public void setPriority(int priority)
```

The priority of a Trigger acts as a tie breaker such that if two Triggers have the same scheduled fire time, then Quartz will do its best to give the one with the higher priority first access to a worker thread.

If not explicitly set, the default value is 5.

**Specified by:**
*setPriority in interface org.quartz.spi.MutableTrigger*

**See Also:**
*Trigger.DEFAULT_PRIORITY*

---

**triggered**
public abstract void triggered(Calendar calendar)

This method should not be used by the Quartz client.

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).

Specified by:
triggered in interface org.quartz.spi.OperableTrigger
See Also:
executionComplete(JobExecutionContext, JobExecutionException)

computeFirstFireTime

public abstract Date computeFirstFireTime(Calendar calendar)

This method should not be used by the Quartz client.

Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.

After this method has been called, getNextFireTime() should return a valid answer.

Specified by:
computeFirstFireTime in interface org.quartz.spi.OperableTrigger

Returns:
the first time at which the Trigger will be fired by the scheduler, which is also the same value getNextFireTime() will return (until after the first firing of the Trigger).

executionComplete

public Trigger.CompletedExecutionInstruction executionComplete(JobEx
This method should not be used by the Quartz client.

Called after the Scheduler has executed the JobDetail associated with the Trigger in order to get the final instruction code from the trigger.

**Specified by:**
- `executionComplete` in interface `org.quartz.spi.OperableTrigger`

**Parameters:**
- `context` - is the JobExecutionContext that was used by the Job's execute(xx) method.
- `result` - is the JobExecutionException thrown by the Job, if any (may be null).

**Returns:**
- one of the CompletedExecutionInstruction constants.

**See Also:**
- `Trigger.CompletedExecutionInstruction`, `triggered(Calendar)`

---

**mayFireAgain**

```java
public abstract boolean mayFireAgain()
```

Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

If the returned value is false then the Scheduler may remove the Trigger from the JobStore.

**Specified by:**
- `mayFireAgain` in interface `Trigger`

---

**getStartTime**

```java
public abstract Date getStartTime()
```

Get the time at which the Trigger should occur.
specifed by:  
**getStartTime** in interface **Trigger**

---

**setStartTime**

```java
public abstract void setStartTime(Date startTime)
```

The time at which the trigger's scheduling should start. May or may not be 
the first actual fire time of the trigger, depending upon the type of trigger 
and the settings of the other properties of the trigger. However the first 
actual first time will not be before this date.

Setting a value in the past may cause a new trigger to compute a first fire 
time that is in the past, which may cause an immediate misfire of the trigger.

**Specified by:**  
**setStartTime** in interface org.quartz.spi.MutableTrigger

---

**setEndTime**

```java
public abstract void setEndTime(Date endTime)
```

Set the time at which the Trigger should quit repeating - regardless of any 
remaining repeats (based on the trigger's particular repeat settings).

**Specified by:**  
**setEndTime** in interface org.quartz.spi.MutableTrigger

**See Also:**  
TriggerUtils#computeEndTimeToAllowParticularNumberOfFirings(Calendar, int)

---

**getEndTime**

```java
public abstract Date getEndTime()
```

Get the time at which the Trigger should quit repeating - regardless of any
remaining repeats (based on the trigger's particular repeat settings).

**Specified by:**  
`getEndTime` in interface `Trigger`  
**See Also:**  
`getFinalFireTime()`

---

### getNextFireTime

```java
public abstract Date getNextFireTime()
```

Returns the next time at which the `Trigger` is scheduled to fire. If the trigger will not fire again, `null` will be returned. Note that the time returned can possibly be in the past, if the time that was computed for the trigger to next fire has already arrived, but the scheduler has not yet been able to fire the trigger (which would likely be due to lack of resources e.g. threads).

The value returned is not guaranteed to be valid until after the `Trigger` has been added to the scheduler.

**Specified by:**  
`getNextFireTime` in interface `Trigger`  
**See Also:**  
`TriggerUtils#computeFireTimesBetween(AbstractTrigger, Calendar, Date, Date)`

---

### getPreviousFireTime

```java
public abstract Date getPreviousFireTime()
```

Returns the previous time at which the `Trigger` fired. If the trigger has not yet fired, `null` will be returned.

**Specified by:**  
`getPreviousFireTime` in interface `Trigger`

---

### getFireTimeAfter


public abstract Date getFireTimeAfter(Date afterTime)

    Returns the next time at which the Trigger will fire, after the given time. If the trigger will not fire after the given time, null will be returned.

    Specified by:
        getFireTimeAfter in interface Trigger

getFinalFireTime

public abstract Date getFinalFireTime()

    Returns the last time at which the Trigger will fire, if the Trigger will repeat indefinitely, null will be returned.

    Note that the return time *may* be in the past.

    Specified by:
        getFinalFireTime in interface Trigger

setMisfireInstruction

public void setMisfireInstruction(int misfireInstruction)

    Set the instruction the Scheduler should be given for handling misfire situations for this Trigger- the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be passed to this method.

    If not explicitly set, the default value is MISFIRE_INSTRUCTION_SMART_POLICY.

    Specified by:
        setMisfireInstruction in interface org.quartz.spi.MutableTrigger

See Also:
        Trigger.MISFIRE_INSTRUCTION_SMART_POLICY,
        updateAfterMisfire(Calendar), SimpleTrigger, CronTrigger
**validateMisfireInstruction**

protected abstract boolean validateMisfireInstruction(int misfireInstruction)

**getMisfireInstruction**

public int getMisfireInstruction()

Get the instruction the Scheduler should be given for handling misfire situations for this Trigger - the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be passed to this method.

If not explicitly set, the default value is MISFIRE_INSTRUCTION_SMART_POLICY.

**Specified by:**
getMisfireInstruction in interface Trigger

**See Also:**
Trigger.MISFIRE_INSTRUCTION_SMART_POLICY, updateAfterMisfire(Calendar), SimpleTrigger, CronTrigger

**updateAfterMisfire**

public abstract void updateAfterMisfire(Calendar cal)

This method should not be used by the Quartz client.

To be implemented by the concrete classes that extend this class.

The implementation should update the Trigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the Trigger was created.

**Specified by:**
updateAfterMisfire in interface org.quartz.spi.OperableTrigger
**updateWithNewCalendar**

public abstract void **updateWithNewCalendar**(Calendar cal, long misfireThreshold)

This method should not be used by the Quartz client.

To be implemented by the concrete class.

The implementation should update the Trigger's state based on the given new version of the associated Calendar (the state should be updated so that it's next fire time is appropriate given the Calendar's new settings).

**Specified by:**
  updateWithNewCalendar in interface org.quartz.spi.OperableTrigger

**Parameters:**
  cal -

---

**validate**

public void **validate**()
  throws SchedulerException

Validates whether the properties of the JobDetail are valid for submission into a Scheduler.

**Specified by:**
  validate in interface org.quartz.spi.OperableTrigger

**Throws:**
  IllegalStateException - if a required property (such as Name, Group, Class) is not set.
  SchedulerException
This method should not be used by the Quartz client.

Usable by JobStore implementations, in order to facilitate 'recognizing' instances of fired Trigger s as their jobs complete execution.

Specified by:
    setFireInstanceId in interface org.quartz.spi.OperableTrigger

---

getFireInstanceId

public String getFireInstanceId()

This method should not be used by the Quartz client.

Specified by:
    getFireInstanceId in interface org.quartz.spi.OperableTrigger

---

toString

public String toString()

Return a simple string representation of this object.

Overrides:
    toString in class Object

---

compareTo

public int compareTo(Trigger other)

Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e. alphabetical) order of their keys.

Specified by:
    compareTo in interface Comparable<Trigger>

Specified by:
compareTo in interface Trigger

equals

public boolean equals(Object o)

Trigger equality is based upon the equality of the TriggerKey.

Specified by:
   equals in interface Trigger
Overrides:
   equals in class Object
Returns:
   true if the key of this Trigger equals that of the given Trigger.

hashCode

public int hashCode()

Overrides:
   hashCode in class Object

close

public Object clone()

Specified by:
   clone in interface org.quartz.spi.MutableTrigger
Overrides:
   clone in class Object

getTriggerBuilder

public TriggerBuilder<T> getTriggerBuilder()
Description copied from interface: Trigger
Get a TriggerBuilder that is configured to produce a Trigger identical to this one.

Specified by:
   getTriggerBuilder in interface Trigger
See Also:
   Trigger.getScheduleBuilder()

getScheduleBuilder

public abstract ScheduleBuilder&lt;T&gt; getScheduleBuilder()

Description copied from interface: Trigger
Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

Specified by:
   getScheduleBuilder in interface Trigger
See Also:
   Trigger.getTriggerBuilder()
public class CalendarIntervalTriggerImpl

extends AbstractTrigger
implements CalendarIntervalTrigger, CoreTrigger

A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.

The trigger will fire every N (see setRepeatInterval(int)) units of calendar time (see #setRepeatIntervalUnit(IntervalUnit)) as specified in the trigger's definition. This trigger can achieve schedules that are not possible with SimpleTrigger (e.g. because months are not a fixed number of seconds) or CronTrigger (e.g. because "every 5 months" is not an even divisor of 12).

If you use an interval unit of MONTH then care should be taken when setting a startTime value that is on a day near the end of the month. For example, if you choose a start time that occurs on January 31st, and have a trigger with unit MONTH and interval 1, then the next fire time will be February 28th, and the next time after that will be March 28th - and essentially each subsequent firing will occur on the 28th of the month, even if a 31st day exists. If you want a trigger that always fires on the last day of the month - regardless of the number of days in the month, you should use CronTrigger.

Since:
1.7

Author:
James House
See Also:
- Trigger, CronTrigger, SimpleTrigger, NthIncludedDayTrigger,
- TriggerUtils, Serialized Form

### Nested Class Summary

Nested classes/interfaces inherited from interface org.quartz.Trigger:
- Trigger.CompletedExecutionInstruction, Trigger.TriggerState,
- Trigger.TriggerTimeComparator

### Field Summary

Fields inherited from interface org.quartz.CalendarIntervalTrigger:
- MISFIRE_INSTRUCTION_DO NOTHING, MISFIRE_INSTRUCTION_FIRE_ONCE_NOW

Fields inherited from interface org.quartz.Trigger:
- DEFAULT_PRIORITY, MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY,
- MISFIRE_INSTRUCTION_SMART_POLICY

### Constructor Summary

**CalendarIntervalTriggerImpl()**

Create a DateIntervalTrigger with no settings.

**CalendarIntervalTriggerImpl(String name,**
**DateBuilder.IntervalUnit intervalUnit,**
**int repeatInterval)**

Create a DateIntervalTrigger that will occur immediately, and repeat at the the given interval.

**CalendarIntervalTriggerImpl(String name,**
**Date startTime,**
**Date endTime,**
**DateBuilder.IntervalUnit intervalUnit,**
**int repeatInterval)**

Create a DateIntervalTrigger that will occur at the given time, and repeat at the the given interval until the given end time.

**CalendarIntervalTriggerImpl(String name,**
**String group,**
**DateBuilder.IntervalUnit intervalUnit,**
**int repeatInterval)**

Create a DateIntervalTrigger that will occur immediately, and repeat
at the given interval.

**CalendarIntervalTriggerImpl(String name, String group, Date startTime, Date endTime, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)**

Create a DateIntervalTrigger that will occur at the given time, and repeat at the given interval until the given end time.

**CalendarIntervalTriggerImpl(String name, String group, String jobName, String jobGroup, Date startTime, Date endTime, DateBuilder.IntervalUnit intervalUnit, int repeatInterval)**

Create a DateIntervalTrigger that will occur at the given time, fire the identified Job and repeat at the given interval until the given end time.

### Method Summary

<table>
<thead>
<tr>
<th>Date</th>
<th><strong>computeFirstFireTime(Calendar calendar)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Called by the scheduler at the time a the scheduler, in order to have the Trigger based on any associated calendar.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th><strong>getEndTime()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the time at which the DateIntervalTrigger repeating.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th><strong>getFinalFireTime()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the final time at which the DateIntervalTrigger fire, if there is no end time set, null will be</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th><strong>getFireTimeAfter(Date afterTime)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the next time at which the DateIntervalTrigger fire, after the given time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected Date</th>
<th><strong>getFireTimeAfter(Date afterTime, boolean ignoreEndTime)</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th><strong>getNextFireTime()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the next time at which the DateIntervalTrigger fire.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th><strong>getPreviousFireTime()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the previous time at which the DateIntervalTrigger fired.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>getRepeatInterval()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the time interval that will be Added to the DateIntervalTrigger’s fire time (in the set repeat interval unit).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th><strong>getRepeatInterval()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the time interval that will be Added to the DateIntervalTrigger’s fire time (in the set repeat interval unit).</td>
</tr>
</tbody>
</table>
getRepeatIntervalUnit()  Get the interval unit - the time unit on with the interval applies.

getScheduleBuilder()  Get a ScheduleBuilder that is configured identical to this trigger's schedule.

getStartTime()  Get the time at which the DateIntervalTrigger fired.

getTimesTriggered()  Get the number of times the DateIntervalTrigger fired.

hasAdditionalProperties()  Determines whether or not the DateIntervalTrigger again.

setEndTime(Date endTime)  Set the time at which the DateIntervalTrigger repeating (and be automatically deleted).

setNextFireTime(Date nextFireTime)  Set the next time at which the DateIntervalTrigger fired.

setPreviousFireTime(Date previousFireTime)  Set the previous time at which the DateIntervalTrigger fired.

setRepeatInterval(int repeatInterval)  Set the time interval that will be added to a DateIntervalTrigger's fire time (in the set repeat interval unit) in order to calculate the time of the next trigger repeat.

setRepeatIntervalUnit(DateBuilder.IntervalUnit)  Set the interval unit - the time unit on with the interval applies.

setStartTime(Date startTime)  Set the time at which the DateIntervalTrigger fired.

setTimesTriggered(int timesTriggered)  Set the number of times the DateIntervalTrigger fired.

triggered(Calendar calendar)
Called when the Scheduler has decided to execute the associated Job, in order to give the update itself for its next triggering (if any).

```java
void updateAfterMisfire(Calendar cal)
```
Updates the DateIntervalTrigger's MISFIRE_INSTRUCTION_XXX that was selected when the DateIntervalTrigger was created.

```java
void updateWithNewCalendar(Calendar calendar, long misfireThreshold)
```
This method should not be used by the Quartz client.

```java
void validate()
```
Validates whether the properties of the submission into a Scheduler.

```java
protected boolean validateMisfireInstruction(int misfireInstruction)
```

Methods inherited from class org.quartz.impl.triggers.AbstractTrigger
- clone, compareTo, equals, executionComplete, getCalendarName, getDescription, getFireInstanceId, getFullJobName, getFullName, getGroup, getJobDataMap, getJobGroup, getJobKey, getJobName, getKey, getMisfireInstruction, getName, getPriority, getTriggerBuilder, hashCode, setCalendarName, setDescription, setFireInstanceId, setGroup, setJobDataMap, setJobGroup, setJobKey, setJobName, setKey, setMisfireInstruction, setName, setPriority, toString

Methods inherited from class java.lang.Object
- finalize, getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface org.quartz.CalendarIntervalTrigger
- getTriggerBuilder

Methods inherited from interface org.quartz.Trigger
- compareTo, equals, getCalendarName, getDescription, getJobDataMap, getJobKey, getKey, getMisfireInstruction, getPriority
Constructor Detail

CalendarIntervalTriggerImpl

```java
public CalendarIntervalTriggerImpl()
```

Create a DateIntervalTrigger with no settings.

---

CalendarIntervalTriggerImpl

```java
public CalendarIntervalTriggerImpl(String name,
                                   DateBuilder.IntervalUnit intervalUnit,
                                   int repeatInterval)
```

Create a DateIntervalTrigger that will occur immediately, and repeat at the given interval.

---

CalendarIntervalTriggerImpl

```java
public CalendarIntervalTriggerImpl(String name,
                                   String group,
                                   DateBuilder.IntervalUnit intervalUnit,
                                   int repeatInterval)
```

Create a DateIntervalTrigger that will occur immediately, and repeat at the given interval.

---

CalendarIntervalTriggerImpl

```java
public CalendarIntervalTriggerImpl(String name,
                                   Date startTime,
                                   Date endTime,
                                   DateBuilder.IntervalUnit intervalUnit,
                                   int repeatInterval)
```

Create a DateIntervalTrigger that will occur at the given time, and repeat at the given interval until the given end time.
Parameters:

startTime - A Date set to the time for the Trigger to fire.
endTime - A Date set to the time for the Trigger to quit repeat firing.
intervalUnit - The repeat interval unit (minutes, days, months, etc).
repeatInterval - The number of milliseconds to pause between the repeat firing.

CalendarIntervalTriggerImpl

public CalendarIntervalTriggerImpl(Name name,
String group,
Date startTime,
Date endTime,
DateBuilder.IntervalUnit intervalUnit,
int repeatInterval)

Create a DateIntervalTrigger that will occur at the given time, and repeat at the given interval until the given end time.

Parameters:

startTime - A Date set to the time for the Trigger to fire.
endTime - A Date set to the time for the Trigger to quit repeat firing.
intervalUnit - The repeat interval unit (minutes, days, months, etc).
repeatInterval - The number of milliseconds to pause between the repeat firing.

CalendarIntervalTriggerImpl

public CalendarIntervalTriggerImpl(Name name,
String group,
String jobName,
String jobGroup,
Date startTime,
Date endTime,
DateBuilder.IntervalUnit intervalUnit,
int repeatInterval)

Create a DateIntervalTrigger that will occur at the given time, fire the identified Job and repeat at the given interval until the given end time.
Parameters:
startTime - A Date set to the time for the Trigger to fire.
endTime - A Date set to the time for the Trigger to quit repeat firing.
intervalUnit - The repeat interval unit (minutes, days, months, etc).
repeatInterval - The number of milliseconds to pause between the repeat firing.

Method Detail

getStartTime

public Date getStartTime()

Get the time at which the DateIntervalTrigger should occur.

Specified by:
getStartTime in interface Trigger
Specified by:
getStartTime in class AbstractTrigger

setStartTime

public void setStartTime(Date startTime)

Set the time at which the DateIntervalTrigger should occur.

Specified by:
setStartTime in interface org.quartz.spi.MutableTrigger
Specified by:
setStartTime in class AbstractTrigger

Throws:
IllegalArgumentException - if startTime is null.

getEndTime

public Date getEndTime()
Get the time at which the `DateIntervalTrigger` should quit repeating.

**Specified by:**
- `getEndTime` in interface `Trigger`
- `getEndTime` in class `AbstractTrigger`

**See Also:**
- `getFinalFireTime()`

---

**setEndTime**

```java
public void setEndTime(Date endTime)
```

Set the time at which the `DateIntervalTrigger` should quit repeating (and be automatically deleted).

**Specified by:**
- `setEndTime` in interface `org.quartz.spi.MutableTrigger`
- `setEndTime` in class `AbstractTrigger`

**Throws:**
- `IllegalArgumentException` - if `endTime` is before start time.

**See Also:**
- `TriggerUtils#computeEndTimeToAllowParticularNumberOfFirings(Calendar, int)`

---

**getRepeatIntervalUnit**

```java
public DateBuilder.IntervalUnit getRepeatIntervalUnit()
```

**Description copied from interface:** `CalendarIntervalTrigger`

Get the interval unit - the time unit on with the interval applies.

**Specified by:**
- `getRepeatIntervalUnit` in interface `CalendarIntervalTrigger`
**setRepeatIntervalUnit**

```java
public void setRepeatIntervalUnit(DateBuilder.IntervalUnit intervalUnit)
```

Set the interval unit - the time unit on with the interval applies.

---

**getRepeatInterval**

```java
public int getRepeatInterval()
```

*Description copied from interface: CalendarIntervalTrigger*

Get the the time interval that will be added to the DateIntervalTrigger's fire time (in the set repeat interval unit) in order to calculate the time of the next trigger repeat.

*Specified by:*

```java
getRepeatInterval in interface CalendarIntervalTrigger
```

---

**setRepeatInterval**

```java
public void setRepeatInterval(int repeatInterval)
```

set the the time interval that will be added to the DateIntervalTrigger's fire time (in the set repeat interval unit) in order to calculate the time of the next trigger repeat.

*Throws:*

```java
IllegalArgumentException - if repeatInterval is < 1
```

---

**getTimesTriggered**

```java
public int getTimesTriggered()
```

*Description copied from interface: CalendarIntervalTrigger*

Get the number of times the DateIntervalTrigger has already fired.
**setTimesTriggered**

```java
public void setTimesTriggered(int timesTriggered)
```

Set the number of times the DateIntervalTrigger has already fired.

---

**validateMisfireInstruction**

```java
protected boolean validateMisfireInstruction(int misfireInstruction)
```

Specified by: `validateMisfireInstruction` in class `AbstractTrigger`

---

**updateAfterMisfire**

```java
public void updateAfterMisfire(Calendar cal)
```

Updates the DateIntervalTrigger's state based on the
MISFIRE_INSTRUCTION_XXX that was selected when the
DateIntervalTrigger was created.

If the misfire instruction is set to
MISFIRE_INSTRUCTION_SMART_POLICY, then the following scheme
will be used:

- The instruction will be interpreted as
  MISFIRE_INSTRUCTION_FIRE_ONCE_NOW

Specified by: `updateAfterMisfire` in interface `org.quartz.spi.OperableTrigger`

Specified by: `updateAfterMisfire` in class `AbstractTrigger`
triggered

```java
public void triggered(Calendar calendar)
```

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).

**Specified by:**
- triggered in interface org.quartz.spi.OperableTrigger

**Specified by:**
- triggered in class AbstractTrigger

**See Also:**
- AbstractTrigger.executionComplete(JobExecutionContext, JobExecutionException)

---

updateWithNewCalendar

```java
public void updateWithNewCalendar(Calendar calendar, long misfireThreshold)
```

**Description copied from class:** AbstractTrigger

This method should not be used by the Quartz client.

To be implemented by the concrete class.

The implementation should update the Trigger's state based on the given new version of the associated Calendar (the state should be updated so that it's next fire time is appropriate given the Calendar's new settings).

**Specified by:**
- updateWithNewCalendar in interface org.quartz.spi.OperableTrigger

**Specified by:**
- updateWithNewCalendar in class AbstractTrigger

**See Also:**
- org.quartz.Trigger#updateWithNewCalendar(org.quartz.Calendar, long)
**computeFirstFireTime**

public Date computeFirstFireTime(Calendar calendar)

Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.

After this method has been called, getNextFireTime() should return a valid answer.

**Specified by:**
- computeFirstFireTime in interface org.quartz.spi.OperableTrigger

**Specified by:**
- computeFirstFireTime in class AbstractTrigger

**Returns:**
the first time at which the Trigger will be fired by the scheduler, which is also the same value getNextFireTime() will return (until after the first firing of the Trigger).

---

**getNextFireTime**

public Date getNextFireTime()

Returns the next time at which the Trigger is scheduled to fire. If the trigger will not fire again, null will be returned. Note that the time returned can possibly be in the past, if the time that was computed for the trigger to next fire has already arrived, but the scheduler has not yet been able to fire the trigger (which would likely be due to lack of resources e.g. threads).

The value returned is not guaranteed to be valid until after the Trigger has been added to the scheduler.

**Specified by:**
- getNextFireTime in interface Trigger

**Specified by:**
- getNextFireTime in class AbstractTrigger

**See Also:**
getPreviousFireTime

public Date getPreviousFireTime()

Returns the previous time at which the DateIntervalTrigger fired. If the trigger has not yet fired, null will be returned.

Specified by:
   getPreviousFireTime in interface Trigger
Specified by:
   getPreviousFireTime in class AbstractTrigger

setNextFireTime

public void setNextFireTime(Date nextFireTime)

Set the next time at which the DateIntervalTrigger should fire.

This method should not be invoked by client code.

Specified by:
   setNextFireTime in interface org.quartz.spi.OperableTrigger

setPreviousFireTime

public void setPreviousFireTime(Date previousFireTime)

Set the previous time at which the DateIntervalTrigger fired.

This method should not be invoked by client code.

Specified by:
   setPreviousFireTime in interface org.quartz.spi.OperableTrigger
getFireTimeAfter

public Date getFireTimeAfter(Date afterTime)

Returns the next time at which the DateIntervalTrigger will fire, after the given time. If the trigger will not fire after the given time, null will be returned.

Specified by:
   getFireTimeAfter in interface Trigger
Specified by:
   getFireTimeAfter in class AbstractTrigger

getFireTimeAfter

protected Date getFireTimeAfter(Date afterTime, boolean ignoreEndTime)

getFinalFireTime

public Date getFinalFireTime()

Returns the final time at which the DateIntervalTrigger will fire, if there is no end time set, null will be returned.

Note that the return time may be in the past.

Specified by:
   getFinalFireTime in interface Trigger
Specified by:
   getFinalFireTime in class AbstractTrigger

mayFireAgain

public boolean mayFireAgain()
Determines whether or not the `DateIntervalTrigger` will occur again.

**Specified by:**
- `mayFireAgain` in interface `Trigger`

**Specified by:**
- `mayFireAgain` in class `AbstractTrigger`

---

**validate**

```java
public void validate()
    throws SchedulerException
```

Validates whether the properties of the `JobDetail` are valid for submission into a Scheduler.

**Specified by:**
- `validate` in interface `org.quartz.spi.OperableTrigger`

**Overrides:**
- `validate` in class `AbstractTrigger`

**Throws:**
- `IllegalStateException` - if a required property (such as Name, Group, Class) is not set.
- `SchedulerException`

---

**getScheduleBuilder**

```java
public ScheduleBuilder<CalendarIntervalTrigger> getScheduleBuilder()
```

Get a `ScheduleBuilder` that is configured to produce a schedule identical to this trigger's schedule.

**Specified by:**
- `getScheduleBuilder` in interface `Trigger`

**Specified by:**
- `getScheduleBuilder` in class `AbstractTrigger`

**See Also:**
- `AbstractTrigger.getTriggerBuilder()`
hasAdditionalProperties

public boolean hasAdditionalProperties()

Specified by:
hasAdditionalProperties in interface CoreTrigger
**Interface CoreTrigger**

**All Superinterfaces:**
[Cloneable], [Comparable<Trigger>], [Serializable], [Trigger]

**All Known Implementing Classes:**
[CalendarIntervalTriggerImpl], [CronTriggerImpl], [SimpleTriggerImpl]

```java
public interface CoreTrigger
extends Trigger
```

internal interface preserved for backward compatibility

---

**Nested Class Summary**

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from interface org.quartz.Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger.CompletedExecutionInstruction, Trigger.TriggerState,</td>
</tr>
<tr>
<td>Trigger.TriggerTimeComparator</td>
</tr>
</tbody>
</table>

**Field Summary**

<table>
<thead>
<tr>
<th>Fields inherited from interface org.quartz.Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_PRIORITY, MISFIRE INSTRUCTION_IGNORE_MISFIRE_POLICY,</td>
</tr>
<tr>
<td>MISFIRE_INSTRUCTION_SMART_POLICY, serialVersionUID</td>
</tr>
</tbody>
</table>

**Method Summary**

```java
boolean hasAdditionalProperties()
```

**Methods inherited from interface org.quartz.Trigger**

```java
compareTo, equals, getCalendarName, getDescription, getEndTime,
```
Method Detail

hasAdditionalProperties

boolean hasAdditionalProperties()
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Class CronTriggerImpl

java.lang.Object
   ▼ org.quartz.impl.triggers.AbstractTrigger<CronTrigger>
   ▼ org.quartz.impl.triggers.CronTriggerImpl

All Implemented Interfaces:
   Serializable, Cloneable, Comparable<Trigger>, CronTrigger, CoreTrigger,
   org.quartz.spi.MutableTrigger, org.quartz.spi.OperableTrigger, Trigger

public class CronTriggerImpl

extends AbstractTrigger<CronTrigger>
implements CronTrigger, CoreTrigger

A concrete Trigger that is used to fire a JobDetail at given moments in time,
defined with Unix 'cron-like' definitions.

Author:
   Sharada Jambula, James House, Contributions from Mads Henderson

See Also:
  Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from interface org.quartz.Trigger
   Trigger.CompletedExecutionInstruction, Trigger.TriggerState,
   Trigger.TriggerTimeComparator

Field Summary

protected static int YEAR_TO_GIVEUP_SCHEDULING_AT

Fields inherited from interface org.quartz.CronTrigger
Fields inherited from interface org.quartz.Trigger
DEFAULT_PRIORITY, MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY, MISFIRE_INSTRUCTION_SMART_POLICY

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CronTriggerImpl()</td>
<td>Create a CronTrigger with no settings.</td>
</tr>
<tr>
<td>CronTriggerImpl(String name)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>CronTriggerImpl(String name, String group)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>CronTriggerImpl(String name, String group, String cronExpression)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>CronTriggerImpl(String name, String group, String jobName, String jobGroup)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>CronTriggerImpl(String name, String group, String jobName, String jobGroup, Date startTime, Date endTime, String cronExpression)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>CronTriggerImpl(String name, String group, String jobName, String jobGroup, Date startTime, Date endTime, String cronExpression, TimeZone timeZone)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>CronTriggerImpl(String name, String group, String jobName, String jobGroup, String cronExpression)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>CronTriggerImpl(String name, String group, String jobName, String jobGroup, String cronExpression, TimeZone timeZone)</td>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object clone()</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><code>computeFirstFireTime(Calendar calendar)</code></td>
<td>Called by the scheduler at the time a Trigger is added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.</td>
</tr>
<tr>
<td><code>getCronExpression()</code></td>
<td></td>
</tr>
<tr>
<td><code>getEndTime()</code></td>
<td>Get the time at which the CronTrigger should repeat - even if repeatCount isn't yet satisfied.</td>
</tr>
<tr>
<td><code>getExpressionSummary()</code></td>
<td></td>
</tr>
<tr>
<td><code>getFinalFireTime()</code></td>
<td>NOT YET IMPLEMENTED: Returns the final time at which the CronTrigger will fire.</td>
</tr>
<tr>
<td><code>getFireTimeAfter(Date afterTime)</code></td>
<td>Returns the next time at which the CronTrigger will fire, after the given time.</td>
</tr>
<tr>
<td><code>getNextFireTime()</code></td>
<td>Returns the next time at which the Trigger is scheduled to fire.</td>
</tr>
<tr>
<td><code>getPreviousFireTime()</code></td>
<td>Returns the previous time at which the CronTrigger fired.</td>
</tr>
<tr>
<td><code>getScheduleBuilder()</code></td>
<td>Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.</td>
</tr>
<tr>
<td><code>startTime()</code></td>
<td>Get the time at which the CronTrigger should occur.</td>
</tr>
<tr>
<td><code>getTimeAfter(Date afterTime)</code></td>
<td>protected Returns the time after the given time that this CronTrigger will fire.</td>
</tr>
<tr>
<td><code>getTimeBefore(Date endTime)</code></td>
<td>protected NOT YET IMPLEMENTED: Returns the time before the given time that this CronTrigger will fire.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>getTimeZone()</strong></td>
<td>Returns the time zone for which the cronExpression of this CronTrigger will be resolved.</td>
</tr>
<tr>
<td><strong>hasAdditionalProperties()</strong></td>
<td>Used by extensions of CronTrigger to imply that there are additional properties, specifically so that extensions can choose whether to be stored as a serial blob, or as a flattened CronTrigger table.</td>
</tr>
<tr>
<td><strong>mayFireAgain()</strong></td>
<td>Determines whether or not the CronTrigger will occur again.</td>
</tr>
<tr>
<td><strong>setCronExpression(CronExpression cronExpression)</strong></td>
<td>Set the CronExpression to the given one.</td>
</tr>
<tr>
<td><strong>setCronExpression(String cronExpression)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>setEndTime(Date endTime)</strong></td>
<td>Set the time at which the Trigger should quit repeating - regardless of any remaining repeats (based the trigger's particular repeat settings).</td>
</tr>
<tr>
<td><strong>setNextFireTime(Date nextFireTime)</strong></td>
<td>Sets the next time at which the CronTrigger will fire.</td>
</tr>
<tr>
<td><strong>setPreviousFireTime(Date previousFireTime)</strong></td>
<td>Set the previous time at which the CronTrigger fired.</td>
</tr>
<tr>
<td><strong>setStartTime(Date startTime)</strong></td>
<td>The time at which the trigger's scheduling should start.</td>
</tr>
<tr>
<td><strong>setTimeZone(TimeZone timeZone)</strong></td>
<td>Sets the time zone for which the cronExpression of this CronTrigger will be resolved.</td>
</tr>
<tr>
<td><strong>triggered(Calendar calendar)</strong></td>
<td>Called when the Scheduler has decided to 'fire' trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).</td>
</tr>
</tbody>
</table>
void updateAfterMisfire(Calendar cal)

Updates the CronTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the CronTrigger was created.

void updateWithNewCalendar(Calendar calendar, long misfireThreshold)

This method should not be used by the Quartz client.

protected boolean validateMisfireInstruction(int misfireInstruction)

boolean willFireOn(Calendar test)

Determines whether the date and (optionally) time of the given Calendar instance falls on a scheduled fire-time of this trigger.

boolean willFireOn(Calendar test, boolean dayOnly)

Determines whether the date and (optionally) time of the given Calendar instance falls on a scheduled fire-time of this trigger.

Methods inherited from class org.quartz.impl.triggers.AbstractTrigger
compareTo, equals, executionComplete, getCalendarName, getDescription, getFireInstanceId, getFullJobName, getFullName, getGroup, getJobDataMap, getJobGroup, getJobKey, getJobName, getFireInstanceId, getJobKey, getMisfireInstruction, getName, getPriority, getTriggerBuilder, hashCode, setCalendarName, setDescription, setFireInstanceId, setGroup, setJobDataMap, setJobGroup, setJobKey, setJobName, setKey, setMisfireInstruction, setName, setPriority, toString, validate

Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface org.quartz.CronTrigger
getTriggerBuilder

Methods inherited from interface org.quartz.Trigger
compareTo, equals, getCalendarName, getDescription, getJobDataMap, getJobKey, getKey, getMisfireInstruction, getPriority
Field Detail

YEAR_TO_GIVEUP_SCHEDULING_AT

protected static final int YEAR_TO_GIVEUP_SCHEDULING_AT

Constructor Detail

CronTriggerImpl

public CronTriggerImpl()

Create a CronTrigger with no settings.

The start-time will also be set to the current time, and the time zone will be set to the system's default time zone.

CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name)

Deprecated. use a TriggerBuilder instead

Create a CronTrigger with the given name and default group.

The start-time will also be set to the current time, and the time zone will be set to the system's default time zone.

CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name, String group)
Deprecated. use a TriggerBuilder instead

Create a CronTrigger with the given name and group.

The start-time will also be set to the current time, and the time zone will be set the the system's default time zone.

CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name,
                      String group,
                      String cronExpression)
                      throws ParseException

Deprecated. use a TriggerBuilder instead

Create a CronTrigger with the given name, group and expression.

The start-time will also be set to the current time, and the time zone will be set the the system's default time zone.

Throws:
ParseException

CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name,
                      String group,
                      String jobName,
                      String jobGroup)

Deprecated. use a TriggerBuilder instead

Create a CronTrigger with the given name and group, and associated with the identified JobDetail.

The start-time will also be set to the current time, and the time zone will be set the the system's default time zone.
CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name,
                        String group,
                        String jobName,
                        String jobGroup,
                        String cronExpression)
    throws ParseException

Deprecated. use a TriggerBuilder instead

Create a CronTrigger with the given name and group, associated with the identified JobDetail, and with the given "cron" expression. The start-time will also be set to the current time, and the time zone will be set the the system's default time zone.

Throws:
    ParseException

CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name,
                        String group,
                        String jobName,
                        String jobGroup,
                        String cronExpression,
                        TimeZone timeZone)
    throws ParseException

Deprecated. use a TriggerBuilder instead

Create a CronTrigger with the given name and group, associated with the identified JobDetail, and with the given "cron" expression resolved with respect to the TimeZone.

Throws:
    ParseException
CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name, String group, String jobName, String jobGroup, Date startTime, Date endTime, String cronExpression)
throws ParseException

Deprecated. use a TriggerBuilder instead

Create a CronTrigger that will occur at the given time, until the given end time.

If null, the start-time will also be set to the current time, the time zone will be set the system's default.

Parameters:
startTime - A Date set to the time for the Trigger to fire.
endTime - A Date set to the time for the Trigger to quit repeat firing.

Throws:
ParseException

CronTriggerImpl

@Deprecated
public CronTriggerImpl(String name, String group, String jobName, String jobGroup, Date startTime, Date endTime, String cronExpression, TimeZone timeZone)
throws ParseException

Deprecated. use a TriggerBuilder instead
Create a `CronTrigger` with fire time dictated by the `cronExpression` resolved with respect to the specified `timeZone` occurring from the `startTime` until the given `endTime`.

If null, the start-time will also be set to the current time. If null, the time zone will be set to the system's default.

**Parameters:**
- `name` - of the Trigger
- `group` - of the Trigger
- `jobName` - name of the JobDetail executed on firetime
- `jobGroup` - group of the JobDetail executed on firetime
- `startTime` - A Date set to the earliest time for the Trigger to start firing.
- `endTime` - A Date set to the time for the Trigger to quit repeat firing.
- `cronExpression` - A cron expression dictating the firing sequence of the Trigger
- `timeZone` - Specifies for which time zone the `cronExpression` should be interpreted, i.e. the expression `0 0 10 * * ?`, is resolved to 10:00 am in this time zone.

**Throws:**
- `ParseException` - if the `cronExpression` is invalid.

### Method Detail

#### clone

```java
public Object clone()
```

**Specified by:**
- `clone` in interface `org.quartz.spi.MutableTrigger`

**Overrides:**
- `clone` in class `AbstractTrigger<CronTrigger>`

#### setCronExpression

```java
public void setCronExpression(String cronExpression)
```

throws `ParseException`
Throws:  
ParseException

getCronExpression

public String getCronExpression()

Specified by:  
getCronExpression in interface CronTrigger

setCronExpression

public void setCronExpression(CronExpression cronExpression)

Set the CronExpression to the given one. The TimeZone on the passed-in CronExpression over-rides any that was already set on the Trigger.

Parameters:

cronExpression -

getStartime

public Date getStartime()

Get the time at which the CronTrigger should occur.

Specified by:  
getStartime in interface Trigger
Specified by:  
getStartime in class AbstractTrigger<CronTrigger>

setStartTime

public void setStartTime(Date startTime)
Description copied from class: AbstractTrigger

The time at which the trigger's scheduling should start. May or may not be the first actual fire time of the trigger, depending upon the type of trigger and the settings of the other properties of the trigger. However the first actual first time will not be before this date.

Setting a value in the past may cause a new trigger to compute a first fire time that is in the past, which may cause an immediate misfire of the trigger.

Specified by:
setStartTime in interface org.quartz.spi.MutableTrigger

Specified by:
setStartTime in class AbstractTrigger<CronTrigger>

---

getEndTime

public Date getEndTime()

Get the time at which the CronTrigger should quit repeating - even if repeastCount isn't yet satisfied.

Specified by:
getEndTime in interface Trigger

Specified by:
getEndTime in class AbstractTrigger<CronTrigger>

See Also:
getFinalFireTime()

---

setEndTime

public void setEndTime(Date endTime)

Description copied from class: AbstractTrigger

Set the time at which the Trigger should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).
Specified by:
setEndTime in interface org.quartz.spi.MutableTrigger
Specified by:
setEndTime in class AbstractTrigger<CronTrigger>
See Also:
TriggerUtils#computeEndTimeToAllowParticularNumberOfFirings(Calendar, int)

getNextFireTime

public Date getNextFireTime()

Returns the next time at which the Trigger is scheduled to fire. If the trigger will not fire again, null will be returned. Note that the time returned can possibly be in the past, if the time that was computed for the trigger to next fire has already arrived, but the scheduler has not yet been able to fire the trigger (which would likely be due to lack of resources e.g. threads).

The value returned is not guaranteed to be valid until after the Trigger has been added to the scheduler.

Specified by:
getNextFireTime in interface Trigger
Specified by:
getNextFireTime in class AbstractTrigger<CronTrigger>
See Also:
TriggerUtils#computeFireTimesBetween(Trigger, org.quartz.Calendar, Date, Date)

getPreviousFireTime

public Date getPreviousFireTime()

Returns the previous time at which the CronTrigger fired. If the trigger has not yet fired, null will be returned.

Specified by:
getPreviousFireTime in interface Trigger
Specified by:
   getPreviousFireTime in class AbstractTrigger<CronTrigger>

---

**setNextFireTime**

```
public void setNextFireTime(Date nextFireTime)
```

Sets the next time at which the CronTrigger will fire. **This method should not be invoked by client code.**

Specified by:
   setNextFireTime in interface org.quartz.spi.OperableTrigger

---

**setPreviousFireTime**

```
public void setPreviousFireTime(Date previousFireTime)
```

Set the previous time at which the CronTrigger fired.

**This method should not be invoked by client code.**

Specified by:
   setPreviousFireTime in interface org.quartz.spi.OperableTrigger

---

**getTimeZone**

```
public TimeZone getTimeZone()
```

Description copied from interface: `CronTrigger`

Returns the time zone for which the cronExpression of this CronTrigger will be resolved.

Specified by:
   getTimeZone in interface CronTrigger
**setTimeZone**

```java
public void setTimeZone(TimeZone timeZone)
```

Sets the time zone for which the cronExpression of this CronTrigger will be resolved.

If `setCronExpression(CronExpression)` is called after this method, the TimeZon setting on the CronExpression will "win". However if `setCronExpression(String)` is called after this method, the time zone applied by this method will remain in effect, since the String cron expression does not carry a time zone!

**getFireTimeAfter**

```java
public Date getFireTimeAfter(Date afterTime)
```

Returns the next time at which the CronTrigger will fire, after the given time. If the trigger will not fire after the given time, `null` will be returned.

Note that the date returned is NOT validated against the related `org.quartz.Calendar` (if any)

**Specified by:**

`getFireTimeAfter` in interface `Trigger`

**Specified by:**

`getFireTimeAfter` in class `AbstractTrigger<CronTrigger>`

**getFinalFireTime**

```java
public Date getFinalFireTime()
```

NOT YET IMPLEMENTED: Returns the final time at which the CronTrigger will fire.

Note that the return time *may* be in the past. and the date returned is not validated against org.quartz.calendar
Specified by:
- `getFinalFireTime` in interface `Trigger`

Specified by:
- `getFinalFireTime` in class `AbstractTrigger<CronTrigger>`

---

**mayFireAgain**

```java
public boolean mayFireAgain()
```

Determines whether or not the `CronTrigger` will occur again.

Specified by:
- `mayFireAgain` in interface `Trigger`

Specified by:
- `mayFireAgain` in class `AbstractTrigger<CronTrigger>`

---

**validateMisfireInstruction**

```java
protected boolean validateMisfireInstruction(int misfireInstruction)
```

Specified by:
- `validateMisfireInstruction` in class `AbstractTrigger<CronTrigger>`

---

**updateAfterMisfire**

```java
public void updateAfterMisfire(Calendar cal)
```

Updates the `CronTrigger`'s state based on the `MISFIRE_INSTRUCTION_XXX` that was selected when the `CronTrigger` was created.

If the misfire instruction is set to `MISFIRE_INSTRUCTION_SMART_POLICY`, then the following scheme will be used:

- The instruction will be interpreted as
MISFIRE_INSTRUCTION_FIRE_ONCE_NOW

Specified by:
updateAfterMisfire in interface org.quartz.spi.OperableTrigger

Specified by:
updateAfterMisfire in class AbstractTrigger<CronTrigger>

willFireOn

public boolean willFireOn(Calendar test)

Determines whether the date and (optionally) time of the given Calendar instance falls on a scheduled fire-time of this trigger.

Equivalent to calling willFireOn(cal, false).

Parameters:
test - the date to compare

See Also:
willFireOn(Calendar, boolean)

willFireOn

public boolean willFireOn(Calendar test, boolean dayOnly)

Determines whether the date and (optionally) time of the given Calendar instance falls on a scheduled fire-time of this trigger.

Note that the value returned is NOT validated against the related org.quartz.Calendar (if any)

Parameters:
test - the date to compare
dayOnly - if set to true, the method will only determine if the trigger will fire during the day represented by the given Calendar (hours, minutes and seconds will be ignored).

See Also:
willFireOn(Calendar)

triggered

```java
public void triggered(Calendar calendar)
```

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).

**Specified by:**
- triggered in interface org.quartz.spi.OperableTrigger
- **Specified by:**
  - triggered in class AbstractTrigger<CronTrigger>

**See Also:**
- AbstractTrigger.executionComplete(JobExecutionContext, JobExecutionException)

updateWithNewCalendar

```java
public void updateWithNewCalendar(Calendar calendar, long misfireThreshold)
```

**Description copied from class: AbstractTrigger**

This method should not be used by the Quartz client.

To be implemented by the concrete class.

The implementation should update the Trigger's state based on the given new version of the associated Calendar (the state should be updated so that it's next fire time is appropriate given the Calendar's new settings).

**Specified by:**
- updateWithNewCalendar in interface org.quartz.spi.OperableTrigger
- **Specified by:**
  - updateWithNewCalendar in class AbstractTrigger<CronTrigger>
See Also:
org.quartz.Trigger#updateWithNewCalendar(org.quartz.Calendar long)

computeFirstFireTime

public Date computeFirstFireTime(Calendar calendar)

Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.

After this method has been called, getNextFireTime() should return a valid answer.

Specified by:
computeFirstFireTime in interface org.quartz.spi.OperableTrigger

Specified by:
computeFirstFireTime in class AbstractTrigger<CronTrigger>

Returns:
the first time at which the Trigger will be fired by the scheduler, which is also the same value getNextFireTime() will return (until after the first firing of the Trigger).

getExpressionSummary

public String getExpressionSummary()

Specified by:
getExpressionSummary in interface CronTrigger

hasAdditionalProperties

public boolean hasAdditionalProperties()

Used by extensions of CronTrigger to imply that there are additional
properties, specifically so that extensions can choose whether to be stored as a serialized blob, or as a flattened CronTrigger table.

**Specified by:**

`hasAdditionalProperties` in interface `CoreTrigger`

---

**getScheduleBuilder**

```java
public ScheduleBuilder<CronTrigger> getScheduleBuilder()
```

Get a `ScheduleBuilder` that is configured to produce a schedule identical to this trigger's schedule.

**Specified by:**

`getScheduleBuilder` in interface `Trigger`

**Specified by:**

`getScheduleBuilder` in class `AbstractTrigger<CronTrigger>`

**See Also:**

`AbstractTrigger.getTriggerBuilder()`

---

**getTimeAfter**

```java
protected Date getTimeAfter(Date afterTime)
```

---

**getTimeBefore**

```java
protected Date getTimeBefore(Date endTime)
```

**NOT YET IMPLEMENTED:** Returns the time before the given time that this CronTrigger will fire.
org.quartz.impl.triggers

Interfaces  CoreTrigger

Classes  
AbstractTrigger
CalendarIntervalTriggerImpl
CronTriggerImpl
SimpleTriggerImpl
Package org.quartz.impl.triggers

This package contains Trigger implementations that ship with Quartz.

See:   Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreTrigger</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractTrigger&lt;T extends Trigger&gt;</td>
</tr>
<tr>
<td>CalendarIntervalTriggerImpl</td>
</tr>
<tr>
<td>CronTriggerImpl</td>
</tr>
<tr>
<td>SimpleTriggerImpl</td>
</tr>
</tbody>
</table>
Package org.quartz.impl.triggers Description

This package contains Trigger implementations that ship with Quartz. End-users should not directly use these classes, but rather use the builders and interfaces found in the main org.quartz package.

See the Quartz project for more information.
Hierarchy For Package org.quartz.impl.triggers

Package Hierarchies:
  All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.impl.triggers.**AbstractTrigger<T>** (implements org.quartz.spi.OperableTrigger)
    - org.quartz.impl.triggers.**CalendarIntervalTriggerImpl** (implements org.quartz.**CalendarIntervalTrigger**, org.quartz.impl.triggers.**CoreTrigger**)
    - org.quartz.impl.triggers.**CronTriggerImpl** (implements org.quartz.impl.triggers.**CoreTrigger**, org.quartz.**CronTrigger**)
    - org.quartz.impl.triggers.**SimpleTriggerImpl** (implements org.quartz.impl.triggers.**CoreTrigger**, org.quartz.**SimpleTrigger**)


Interface Hierarchy

- java.lang.**Cloneable**
  - org.quartz.**Trigger** (also extends java.lang.**Comparable**<T>, java.io.**Serializable**)
    - org.quartz.impl.triggers.**CoreTrigger**
- java.lang.**Comparable**<T>
  - org.quartz.**Trigger** (also extends java.lang.**Cloneable**, java.io.**Serializable**)
    - org.quartz.impl.triggers.**CoreTrigger**
- java.io.**Serializable**
  - org.quartz.**Trigger** (also extends java.lang.**Cloneable**, java.lang.**Comparable**<T>)
    - org.quartz.impl.triggers.**CoreTrigger**

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.impl.triggers

Packages that use org.quartz.impl.triggers

<table>
<thead>
<tr>
<th>Package</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
</tbody>
</table>

Classes in org.quartz.impl.triggers used by org.quartz.impl.triggers

<table>
<thead>
<tr>
<th>Class</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractTrigger</td>
<td>The base abstract class to be extended by all Triggers.</td>
</tr>
<tr>
<td>CoreTrigger</td>
<td>internal interface preserved for backward compatibility</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Class SimpleTriggerImpl

java.lang.Object
   ↓ org.quartz.impl.triggers.AbstractTrigger<SimpleTrigger>
   ↑ org.quartz.impl.triggers.SimpleTriggerImpl

All Implemented Interfaces:
   Serializable, Cloneable, Comparable<Trigger>, CoreTrigger,
   SimpleTrigger, org.quartz.spi.MutableTrigger,
   org.quartz.spi.OperableTrigger, Trigger

public class SimpleTriggerImpl

extends AbstractTrigger<SimpleTrigger>

implements SimpleTrigger, CoreTrigger

A concrete Trigger that is used to fire a JobDetail at a given moment in time,
and optionally repeated at a specified interval.

Author:
   James House, contributions by Lieven Govaerts of Ebitec Nv, Belgium.

See Also:
   Trigger, CronTrigger, TriggerUtils, Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from interface org.quartz.Trigger

Trigger.CompletedExecutionInstruction, Trigger.TriggerState,
Trigger.TriggerTimeComparator

Field Summary

Fields inherited from interface org.quartz.SimpleTrigger

MISFIRE_INSTRUCTION_FIRE_NOW,
Fields inherited from interface org.quartz.Trigger
DEFAULT_PRIORITY, MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY, MISFIRE_INSTRUCTION_SMART_POLICY

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SimpleTriggerImpl()</strong></td>
</tr>
<tr>
<td>Create a SimpleTrigger with no settings.</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, Date startTime)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, Date startTime, Date endTime, int repeatCount, long repeatInterval)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, int repeatCount, long repeatInterval)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, String group)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, String group, Date startTime)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, String group, Date startTime, Date endTime, int repeatCount, long repeatInterval)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, String group, int repeatCount, long repeatInterval)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td><strong>SimpleTriggerImpl(String name, String group, String jobName, String jobGroup, Date startTime, Date endTime, int repeatCount, long repeatInterval)</strong></td>
</tr>
<tr>
<td>Deprecated. use a TriggerBuilder instead</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><code>computeFirstFireTime(Calendar calendar)</code></td>
</tr>
<tr>
<td><code>computeNumTimesFiredBetween(Date start, Date end)</code></td>
</tr>
<tr>
<td><code>getEndTime()</code></td>
</tr>
<tr>
<td><code>getFinalFireTime()</code></td>
</tr>
<tr>
<td><code>getFireTimeAfter(Date afterTime)</code></td>
</tr>
<tr>
<td><code>getFireTimeBefore(Date end)</code></td>
</tr>
<tr>
<td><code>getNextFireTime()</code></td>
</tr>
<tr>
<td><code>getPreviousFireTime()</code></td>
</tr>
<tr>
<td><code>getRepeatCount()</code></td>
</tr>
<tr>
<td><code>getRepeatInterval()</code></td>
</tr>
<tr>
<td><code>getScheduleBuilder()</code></td>
</tr>
</tbody>
</table>
Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

Date getStartTime()
Get the time at which the SimpleTrigger should occur.

int getTimesTriggered()
Get the number of times the SimpleTrigger has already fired.

boolean hasAdditionalProperties()
Used by extensions of SimpleTrigger to imply there are additional properties, specifically so that extensions can choose whether to be stored as a serialized blob, or as a flattened SimpleTrigger table.

boolean mayFireAgain()
Determines whether or not the SimpleTrigger should occur again.

void setEndTime(Date endTime)
Set the time at which the SimpleTrigger should quit repeating (and be automatically deleted).

void setNextFireTime(Date nextFireTime)
Set the next time at which the SimpleTrigger should fire.

void setPreviousFireTime(Date previousFireTime)
Set the previous time at which the SimpleTrigger fired.

void setRepeatCount(int repeatCount)
Set the number of time the SimpleTrigger should repeat, after which it will be automatically deleted.

void setRepeatInterval(long repeatInterval)
Set the time interval (in milliseconds) at which the SimpleTrigger should repeat.

void setStartTime(Date startTime)
Set the time at which the SimpleTrigger should occur.

void setTimesTriggered(int timesTriggered)
Set the number of times the SimpleTrigger has
already fired.

void triggered(Calendar calendar)

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the trigger a chance to update itself for its next triggering (if any).

void updateAfterMisfire(Calendar cal)

Updates the SimpleTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the SimpleTrigger was created.

void updateWithNewCalendar(Calendar calendar, long misfireThreshold)

This method should not be used by the Quartz client.

void validate()

Validates whether the properties of the JobDetail are valid for submission into a Scheduler.

protected boolean validateMisfireInstruction(int misfireInstruction)

Methods inherited from class org.quartz.impl.triggers.AbstractTrigger

clone, compareTo, equals, executionComplete, getCalendarName, getDescription, getFireInstanceId, getFullJobName, getFullName, getGroup, getJobDataMap, getJobGroup, getJobKey, getJobName, getMisfireInstruction, getName, getPriority, getTriggerBuilder, hashCode, setCalendarName, setDescription, setFireInstanceId, setGroup, setJobDataMap, setJobGroup, setJobKey, setJobName, setMisfireInstruction, setName, setPriority, toString

Methods inherited from class java.lang.Object

finalize, getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface org.quartz.SimpleTrigger

getTriggerBuilder

Methods inherited from interface org.quartz.Trigger

compareTo, equals, getCalendarName, getDescription, getJobDataMap,
SimpleTriggerImpl

public SimpleTriggerImpl()

Create a SimpleTrigger with no settings.

SimpleTriggerImpl

@Deprecated
public SimpleTriggerImpl(String name)

Deprecated. use a TriggerBuilder instead

Create a SimpleTrigger that will occur immediately, and not repeat.

SimpleTriggerImpl

@Deprecated
public SimpleTriggerImpl(String name, String group)

Deprecated. use a TriggerBuilder instead

Create a SimpleTrigger that will occur immediately, and not repeat.

SimpleTriggerImpl

@Deprecated
public SimpleTriggerImpl(String name, int repeatCount, long repeatInterval)
**Deprecated.** use a TriggerBuilder instead

Create a SimpleTrigger that will occur immediately, and repeat at the given interval the given number of times.

---

**SimpleTriggerImpl**

```java
@Deprecated
class SimpleTriggerImpl {
    public SimpleTriggerImpl(String name,
                               String group,
                               int repeatCount,
                               long repeatInterval)
}
```

**Deprecated.** use a TriggerBuilder instead

Create a SimpleTrigger that will occur immediately, and repeat at the given interval the given number of times.

---

**SimpleTriggerImpl**

```java
@Deprecated
class SimpleTriggerImpl {
    public SimpleTriggerImpl(String name,
                               Date startTime)
}
```

**Deprecated.** use a TriggerBuilder instead

Create a SimpleTrigger that will occur at the given time, and not repeat.

---

**SimpleTriggerImpl**

```java
@Deprecated
class SimpleTriggerImpl {
    public SimpleTriggerImpl(String name,
                               String group,
                               Date startTime)
}
```

**Deprecated.** use a TriggerBuilder instead

Create a SimpleTrigger that will occur at the given time, and not repeat.
SimpleTriggerImpl

@Deprecated
public SimpleTriggerImpl(String name, Date startTime, Date endTime, int repeatCount, long repeatInterval)

Deprecated. use a TriggerBuilder instead

Create a SimpleTrigger that will occur at the given time, and repeat at the given interval the given number of times, or until the given end time.

Parameters:
- startTime - A Date set to the time for the Trigger to fire.
- endTime - A Date set to the time for the Trigger to quit repeat firing.
- repeatCount - The number of times for the Trigger to repeat firing, use SimpleTrigger.REPEAT_INDEFINITELY for unlimited times.
- repeatInterval - The number of milliseconds to pause between the repeat firing.

SimpleTriggerImpl

@Deprecated
public SimpleTriggerImpl(String name, String group, Date startTime, Date endTime, int repeatCount, long repeatInterval)

Deprecated. use a TriggerBuilder instead

Create a SimpleTrigger that will occur at the given time, and repeat at the given interval the given number of times, or until the given end time.

Parameters:
- startTime - A Date set to the time for the Trigger to fire.
- endTime - A Date set to the time for the Trigger to quit repeat firing.
- repeatCount - The number of times for the Trigger to repeat firing,
use `SimpleTrigger.REPEAT_INDEFINITELY` for unlimited times.
repeatInterval - The number of milliseconds to pause between the repeat firing.

---

**SimpleTriggerImpl**

```java
@Deprecated
class SimpleTriggerImpl {
    @Deprecated
    public SimpleTriggerImpl(String name,
                              String group,
                              String jobName,
                              String jobGroup,
                              Date startTime,
                              Date endTime,
                              int repeatCount,
                              long repeatInterval)
```

**Deprecated. use a TriggerBuilder instead**

Create a SimpleTrigger that will occur at the given time, fire the identified Job and repeat at the given interval the given number of times, or until the given end time.

**Parameters:**
- `startTime` - A Date set to the time for the Trigger to fire.
- `endTime` - A Date set to the time for the Trigger to quit repeat firing.
- `repeatCount` - The number of times for the Trigger to repeat firing,
  use `SimpleTrigger.REPEAT_INDEFINITELY` for unlimited times.
- `repeatInterval` - The number of milliseconds to pause between the repeat firing.

---

**Method Detail**

**getStartTime**

```java
public Date getStartTime()
```

Get the time at which the SimpleTrigger should occur.

**Specified by:**
**getStartTime** in interface *Trigger*
Specified by:
**getStartTime** in class *AbstractTrigger*<SimpleTrigger>*

---

**setStartTime**

```java
public void setStartTime(Date startTime)
```

Set the time at which the SimpleTrigger should occur.

Specified by:
**setStartTime** in interface org.quartz.spi.MutableTrigger
Specified by:
**setStartTime** in class *AbstractTrigger*<SimpleTrigger>*

Throws:
**IllegalArgumentException** - if startTime is null.

---

**getEndTime**

```java
public Date getEndTime()
```

Get the time at which the SimpleTrigger should quit repeating - even if repeatCount isn't yet satisfied.

Specified by:
**getEndTime** in interface *Trigger*
Specified by:
**getEndTime** in class *AbstractTrigger*<SimpleTrigger>*

See Also:
**getFinalFireTime()**

---

**setEndTime**

```java
public void setEndTime(Date endTime)
```

Set the time at which the SimpleTrigger should quit repeating (and be
automatically deleted).

**Specified by:**
`setEndTime` in interface `org.quartz.spi.MutableTrigger`

**Specified by:**
`setEndTime` in class `AbstractTrigger<SimpleTrigger>`

**Throws:**
`IllegalArgumentException` - if `endTime` is before `startTime`.

**See Also:**
`TriggerUtils#computeEndTimeToAllowParticularNumberOfFirings(Calendar, int)`

---

### getRepeatCount

**public int** `getRepeatCount()`

**Description copied from interface:** `SimpleTrigger`

Get the the number of times the `SimpleTrigger` should repeat, after which it will be automatically deleted.

**Specified by:**
`getRepeatCount` in interface `SimpleTrigger`

**See Also:**
`SimpleTrigger.REPEAT_INDEFINITELY`

---

### setRepeatCount

**public void** `setRepeatCount(int repeatCount)`

Set the the number of time the `SimpleTrigger` should repeat, after which it will be automatically deleted.

**Throws:**
`IllegalArgumentException` - if `repeatCount` is < 0

**See Also:**
`SimpleTrigger.REPEAT_INDEFINITELY`
getRepeatInterval

public long getRepeatInterval()

**Description copied from interface:** SimpleTrigger

Get the the time interval (in milliseconds) at which the SimpleTrigger should repeat.

**Specified by:**
getRepeatInterval in interface SimpleTrigger

---

setRepeatInterval

public void setRepeatInterval(long repeatInterval)

Set the the time interval (in milliseconds) at which the SimpleTrigger should repeat.

**Throws:**
IllegallegalArgumentException - if repeatInterval is <= 0

---

getTimesTriggered

public int getTimesTriggered()

Get the number of times the SimpleTrigger has already fired.

**Specified by:**
getTimesTriggered in interface SimpleTrigger

---

setTimesTriggered

public void setTimesTriggered(int timesTriggered)

Set the number of times the SimpleTrigger has already fired.
validateMisfireInstruction

protected boolean validateMisfireInstruction(int misfireInstruction)

Specified by:

validateMisfireInstruction in class AbstractTrigger<SimpleTrigger>

updateAfterMisfire

public void updateAfterMisfire(Calendar cal)

Updates the SimpleTrigger's state based on the MISFIRE_INSTRUCTION_XXX that was selected when the SimpleTrigger was created.

If the misfire instruction is set to MISFIRE_INSTRUCTION_SMART_POLICY, then the following scheme will be used:

- If the Repeat Count is 0, then the instruction will be interpreted as MISFIRE_INSTRUCTION_FIRE_NOW.
- If the Repeat Count is REPEAT_INDEFINITELY, then the instruction will be interpreted as MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT.

WARNING: using MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT with a trigger that has a non-null end-time may cause the trigger to never fire again if the end-time arrived during the misfire time span.

- If the Repeat Count is > 0, then the instruction will be interpreted as MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT.

If the misfire instruction is set to Trigger.MISFIRE_INSTRUCTION_SKIP_TO_NEXT_FIRE_AFTER_CURRENT_DATE then the behavior will be identical to

Specified by:

updateAfterMisfire in interface org.quartz.spi.OperableTrigger
updateAfterMisfire in class AbstractTrigger&lt;SimpleTrigger&gt;

triggered

public void triggered(Calendar calendar)

Called when the Scheduler has decided to 'fire' the trigger (execute the associated Job), in order to give the Trigger a chance to update itself for its next triggering (if any).

Specified by:
   triggered in interface org.quartz.spi.OperableTrigger

Specified by:
   triggered in class AbstractTrigger&lt;SimpleTrigger&gt;

See Also:
   AbstractTrigger.executionComplete(JobExecutionContext, JobExecutionException)

updateWithNewCalendar

public void updateWithNewCalendar(Calendar calendar, long misfireThreshold)

Description copied from class: AbstractTrigger

This method should not be used by the Quartz client.

To be implemented by the concrete class.

The implementation should update the Trigger's state based on the given new version of the associated Calendar (the state should be updated so that it's next fire time is appropriate given the Calendar's new settings).

Specified by:
   updateWithNewCalendar in interface org.quartz.spi.OperableTrigger

Specified by:
   updateWithNewCalendar in class AbstractTrigger&lt;SimpleTrigger&gt;
computeFirstFireTime

```java
public Date computeFirstFireTime(Calendar calendar)
```

Called by the scheduler at the time a Trigger is first added to the scheduler, in order to have the Trigger compute its first fire time, based on any associated calendar.

After this method has been called, getNextFireTime() should return a valid answer.

Specified by:
- computeFirstFireTime in interface org.quartz.spi.OperableTrigger
- computeFirstFireTime in class AbstractTrigger<SimpleTrigger>

Returns:
- the first time at which the Trigger will be fired by the scheduler, which is also the same value getNextFireTime() will return (until after the first firing of the Trigger).

getNextFireTime

```java
public Date getNextFireTime()
```

Returns the next time at which the Trigger is scheduled to fire. If the trigger will not fire again, null will be returned. Note that the time returned can possibly be in the past, if the time that was computed for the trigger to next fire has already arrived, but the scheduler has not yet been able to fire the trigger (which would likely be due to lack of resources e.g. threads).

The value returned is not guaranteed to be valid until after the Trigger has been added to the scheduler.
Specified by:
    getNextFireTime in interface Trigger
Specified by:
    getNextFireTime in class AbstractTrigger<SimpleTrigger>
See Also:
    TriggerUtils#computeFireTimesBetween(Trigger, Calendar, Date, Date)

getPreviousFireTime

public Date getPreviousFireTime()

    Returns the previous time at which the SimpleTrigger fired. If the trigger has not yet fired, null will be returned.

Specified by:
    getPreviousFireTime in interface Trigger
Specified by:
    getPreviousFireTime in class AbstractTrigger<SimpleTrigger>

setNextFireTime

public void setNextFireTime(Date nextFireTime)

    Set the next time at which the SimpleTrigger should fire.

    This method should not be invoked by client code.

Specified by:
    setNextFireTime in interface org.quartz.spi.OperableTrigger

setPreviousFireTime

public void setPreviousFireTime(Date previousFireTime)

    Set the previous time at which the SimpleTrigger fired.
This method should not be invoked by client code.

Specified by:
    setPreviousFireTime in interface org.quartz.spi.OperableTrigger

getFireTimeAfter

public Date getFireTimeAfter(Date afterTime)

    Returns the next time at which the SimpleTrigger will fire, after the given time. If the trigger will not fire after the given time, null will be returned.

Specified by:
    getFireTimeAfter in interface Trigger
Specified by:
    getFireTimeAfter in class AbstractTrigger<SimpleTrigger>

getFireTimeBefore

public Date getFireTimeBefore(Date end)

    Returns the last time at which the SimpleTrigger will fire, before the given time. If the trigger will not fire before the given time, null will be returned.

computeNumTimesFiredBetween

public int computeNumTimesFiredBetween(Date start, Date end)

getFinalFireTime

public Date getFinalFireTime()

    Returns the final time at which the SimpleTrigger will fire, if repeatCount is REPEAT_INDEFINITELY, null will be returned.
Note that the return time may be in the past.

Specified by:
getFinalFireTime in interface Trigger
Specified by:
getFinalFireTime in class AbstractTrigger<SimpleTrigger>

---

mayFireAgain

public boolean mayFireAgain()

Determines whether or not the SimpleTrigger will occur again.

Specified by:
mayFireAgain in interface Trigger
Specified by:
mayFireAgain in class AbstractTrigger<SimpleTrigger>

---

validate

public void validate()
throws SchedulerException

Validates whether the properties of the JobDetail are valid for submission into a Scheduler.

Specified by:
validate in interface org.quartz.spi.OperableTrigger
Overrides:
validate in class AbstractTrigger<SimpleTrigger>
Throws:
IllegalStateException - if a required property (such as Name, Group, Class) is not set.
SchedulerException

---

hasAdditionalProperties
public boolean hasAdditionalProperties()

Used by extensions of SimpleTrigger to imply that there are additional properties, specifically so that extensions can choose whether to be stored as a serialized blob, or as a flattened SimpleTrigger table.

Specified by:
    hasAdditionalProperties in interface CoreTrigger

getScheduleBuilder

public ScheduleBuilder<SimpleTrigger> getScheduleBuilder()

Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

Specified by:
    getScheduleBuilder in interface Trigger
Specified by:
    getScheduleBuilder in class AbstractTrigger<SimpleTrigger>
See Also:
    AbstractTrigger.getTriggerBuilder()
Uses of Class
org.quartz.impl.triggers.AbstractTrigger

Packages that use AbstractTrigger

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
</tbody>
</table>

Uses of AbstractTrigger in org.quartz.impl.triggers

Subclasses of AbstractTrigger in org.quartz.impl.triggers

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalendarIntervalTriggerImpl</td>
<td>A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.</td>
</tr>
<tr>
<td>CronTriggerImpl</td>
<td>A concrete Trigger that is used to fire a JobDetail at given moments in time, defined with Unix 'cron-like' definitions.</td>
</tr>
<tr>
<td>SimpleTriggerImpl</td>
<td>A concrete Trigger that is used to fire a JobDetail at a given moment in time, and optionally repeated at a specified interval.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.triggers.CalendarIntervalTriggerImpl

No usage of org.quartz.impl.triggers.CalendarIntervalTriggerImpl

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.impl.triggers.CoreTrigger

Packages that use **CoreTrigger**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.impl.triggers</td>
<td>This package contains Trigger implementations that ship with Quartz.</td>
</tr>
</tbody>
</table>

Uses of **CoreTrigger** in **org.quartz.impl.triggers**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalendarIntervalTriggerImpl</td>
<td>A concrete Trigger that is used to fire a JobDetail based upon repeating calendar time intervals.</td>
</tr>
<tr>
<td>CronTriggerImpl</td>
<td>A concrete Trigger that is used to fire a JobDetail at given moments in time, defined with Unix 'cron-like' definitions.</td>
</tr>
<tr>
<td>SimpleTriggerImpl</td>
<td>A concrete Trigger that is used to fire a JobDetail at a given moment in time, and optionally repeated at a specified interval.</td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.impl.triggers.CronTriggerImpl

No usage of org.quartz.impl.triggers.CronTriggerImpl

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.impl.triggers.SimpleTriggerImpl

No usage of org.quartz.impl.triggers.SimpleTriggerImpl

Copyright 2001-2011, Terracotta, Inc.
org.quartz.jobs Class DirectoryScanJob

java.lang.Object
   └ org.quartz.jobs.DirectoryScanJob

All Implemented Interfaces:
   Job

@DisallowConcurrentExecution
@PersistJobDataAfterExecution
public class DirectoryScanJob

extends Object
implements Job

Inspects a directory and compares whether any files' "last modified dates" have changed since the last time it was inspected. If one or more files have been updated (or created), the job invokes a "call-back" method on an identified DirectoryScanListener that can be found in the SchedulerContext.

Author:  
   pl47ypus, jhouse

See Also:  
   DirectoryScanListener, SchedulerContext

---

Field Summary

<table>
<thead>
<tr>
<th>static String DIRECTORY_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDataMap key with which to specify the directory to be monitored - an absolute path is recommended.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String DIRECTORY_SCAN_LISTENER_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDataMap key with which to specify the DirectoryScanListener to be notified when the directory contents change.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String MINIMUM_UPDATE_AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobDataMap key with which to specify a long value that represents the minimum number of milliseconds that must have elapsed since the last directory scan.</td>
</tr>
</tbody>
</table>
past since the file's last modified time in order to consider the file new/altered.

### Constructor Summary

**DirectoryScanJob()**

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>execute(JobExecutionContext context)</code></td>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
<tr>
<td>protected</td>
<td><code>File[] getUpdatedOrNewFiles(String dirName, long lastDate, long maxAgeDate)</code></td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.*Object*

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Field Detail

**DIRECTORY_NAME**

- public static final `String DIRECTORY_NAME`

  JobDataMap key with which to specify the directory to be monitored - an absolute path is recommended.

  **See Also:**

  Constant Field Values

**DIRECTORY_SCAN_LISTENER_NAME**
public static final String DIRECTORY_SCAN_LISTENER_NAME

JobDataMap key with which to specify the DirectoryScanListener to be notified when the directory contents change.

See Also: Constant Field Values

MINIMUM_UPDATE_AGE

public static final String MINIMUM_UPDATE_AGE

JobDataMap key with which to specify a long value that represents the minimum number of milliseconds that must have past since the file's last modified time in order to consider the file new/altered. This is necessary because another process may still be in the middle of writing to the file when the scan occurs, and the file may therefore not yet be ready for processing.

If this parameter is not specified, a default value of 5000 (five seconds) will be used.

See Also: Constant Field Values

Constructor Detail

DirectoryScanJob

public DirectoryScanJob()

Method Detail

execute

public void execute(JobExecutionContext context) throws JobExecutionException
Description copied from interface: Job

Called by the Scheduler when a Trigger fires that is associated with the Job.

The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
   execute in interface Job

Throws:
   JobExecutionException - if there is an exception while executing the job.

See Also:
   Job.execute(org.quartz.JobExecutionContext)

getUpdatedOrNewFiles

protected File[] getUpdatedOrNewFiles(String dirName, long lastDate, long maxAgeDate)
public interface DirectoryScanListener

Interface for objects wishing to receive a 'call-back' from a DirectoryScanJob.

Instances should be stored in the SchedulerContext such that the DirectoryScanJob can find it.

Author:  
  jhouse

See Also:  
  DirectoryScanJob, SchedulerContext

---

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>filesUpdatedOrAdded(File[] updatedFiles)</td>
</tr>
</tbody>
</table>

---

Method Detail

filesUpdatedOrAdded

void filesUpdatedOrAdded(File[] updatedFiles)

Parameters:
updatedFiles - The set of files that were updated/added since the last scan of the directory

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.jobs Class FileScanJob

java.lang.Object
   └ org.quartz.jobs.FileScanJob

All Implemented Interfaces:
   Job

@DisallowConcurrentExecution
@PersistJobDataAfterExecution
public class FileScanJob

extends Object
implements Job

Inspects a file and compares whether its "last modified date" has changed since
the last time it was inspected. If the file has been updated, the job invokes a
"call-back" method on an identified FileScanListener that can be found in the
SchedulerContext.

Author:
   jhouse, pl47ypus

See Also:
   FileScanListener

Field Summary

| static String | FILE_NAME | JobDataMap key with which to specify the name of the file to monitor. |
| static String | FILE_SCAN_LISTENER_NAME | JobDataMap key with which to specify the FileScanListener to be notified when the file contents change. |
| static String | MINIMUM_UPDATE_AGE | JobDataMap key with which to specify a long value that represents the minimum number of milliseconds that must have past since the file's last modified time in order to consider the file... |
Constructor Summary

FileScanJob()

Method Summary

void execute(JobExecutionContext context)

  Called by the Scheduler when a Trigger fires that is associated with the Job.

protected long getLastModifiedDate(String fileName)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

FILE_NAME

public static final String FILE_NAME

  JobDataMap key with which to specify the name of the file to monitor.

See Also:

  Constant Field Values

FILE_SCAN_LISTENER_NAME

public static final String FILE_SCAN_LISTENER_NAME

  JobDataMap key with which to specify the FileScanListener to be notified
when the file contents change.

See Also:
Constant Field Values

MINIMUM_UPDATE_AGE

public static final String MINIMUM_UPDATE_AGE

JobDataMap key with which to specify a long value that represents the minimum number of milliseconds that must have past since the file's last modified time in order to consider the file new/altered. This is necessary because another process may still be in the middle of writing to the file when the scan occurs, and the file may therefore not yet be ready for processing.

If this parameter is not specified, a default value of 5000 (five seconds) will be used.

See Also:
Constant Field Values

Constructor Detail

FileScanJob

public FileScanJob()

Method Detail

execute

public void execute(JobExecutionContext context)
  throws JobExecutionException

  Description copied from interface: Job
Called by the Scheduler when a Trigger fires that is associated with the Job.

The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
   execute in interface Job

Throws:
   JobExecutionException - if there is an exception while executing the job.

See Also:
   Job.execute(org.quartz.JobExecutionContext)

getLastModifiedDate

protected long getLastModifiedDate(String fileName)
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td>NO FRAME</td>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Interface FileScanListener

All Known Implementing Classes:

XMLSchedulingDataProcessorPlugin

public interface FileScanListener

Interface for objects wishing to receive a 'call-back' from a FileScanJob.

Author:
jhouse

See Also:
FileScanJob

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>void fileUpdated(String fileName)</td>
<td></td>
</tr>
</tbody>
</table>

Method Detail

fileUpdated

void fileUpdated(String fileName)
org.quartz.jobs **Class NativeJob**

```java
public class NativeJob
    extends Object
    implements Job
```

Built in job for executing native executables in a separate process.

```java
JobDetail job = new JobDetail("dumbJob", null, org.quartz.jobs.NativeJob.class);
job.getJobDataMap().put(org.quartz.jobs.NativeJob.PROP_COMMAND, "echo "hi" >> foobar.txt";
Trigger trigger = TriggerUtils.makeSecondlyTrigger(5);
trigger.setName("dumbTrigger");
sched.scheduleJob(job, trigger);
```

If PROP_WAIT_FOR_PROCESS is true, then the Integer exit value of the process will be saved as the job execution result in the JobExecutionContext.

**Author:**
Matthew Payne, James House, Steinar Overbeck Cook

**See Also:**
`PROP_COMMAND`, `PROP_PARAMETERS`, `PROP_WAIT_FOR_PROCESS`, `PROP_CONSUME_STREAMS`

---

**Field Summary**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String</code></td>
<td>PROP_COMMAND</td>
<td>Required parameter that specifies the name of the command (executable) to be ran.</td>
</tr>
<tr>
<td><code>String</code></td>
<td>PROP_CONSUME_STREAMS</td>
<td>Optional parameter (value should be 'true' or 'false') that</td>
</tr>
</tbody>
</table>
specifies whether the spawned process's stdout and stderr streams should be consumed.

<table>
<thead>
<tr>
<th>static String PROP_PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional parameter that specifies the parameters to be passed to the executed command.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String PROP_WAIT_FOR_PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional parameter (value should be 'true' or 'false') that specifies whether the job should wait for the execution of the native process to complete before it completes.</td>
</tr>
</tbody>
</table>

**Constructor Summary**

<table>
<thead>
<tr>
<th>NativeJob()</th>
</tr>
</thead>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>void execute(JobExecutionContext context)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected org.slf4j.Logger getLog()</th>
</tr>
</thead>
</table>

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Field Detail**

**PROP_COMMAND**

public static final String PROP_COMMAND

Required parameter that specifies the name of the command (executable) to be ran.
See Also:
Constant Field Values

PROP_PARAMETERS

public static final String PROP_PARAMETERS

Optional parameter that specifies the parameters to be passed to the executed command.

See Also:
Constant Field Values

PROP_WAIT_FOR_PROCESS

public static final String PROP_WAIT_FOR_PROCESS

Optional parameter (value should be 'true' or 'false') that specifies whether the job should wait for the execution of the native process to complete before it completes.

Defaults to true.

See Also:
Constant Field Values

PROP_CONSUME_STREAMS

public static final String PROP_CONSUME_STREAMS

Optional parameter (value should be 'true' or 'false') that specifies whether the spawned process's stdout and stderr streams should be consumed. If the process creates output, it is possible that it might 'hang' if the streams are not consumed.

Defaults to false.
NativeJob

public NativeJob()

execute

public void execute(JobExecutionContext context)
throws JobExecutionException

Description copied from interface: Job

Called by the Scheduler when a Trigger fires that is associated with the Job.

The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
execute in interface Job

Throws:
JobExecutionException - if there is an exception while executing the job.

getLog

protected org.slf4j.Logger getLog()
Class NoOpJob

All Implemented Interfaces:
   Job

public class NoOpJob
   extends Object
   implements Job

An implementation of Job, that does absolutely nothing - useful for system
which only wish to use TriggerListeners and JobListeners, rather than
writing Jobs that perform work.

Author:
   James House

Constructor Summary

| NoOpJob() |

Method Summary

| void execute(JobExecutionContext context) |
| Do nothing. |

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
NoOpJob

public NoOpJob()

execute

public void execute(JobExecutionContext context) throws JobExecutionException

Do nothing.

Specified by:
execute in interface Job

Throws:
JobExecutionException - if there is an exception while executing the job.
org.quartz.jobs

Interfaces
DirectoryScanListener
FileScanListener

Classes
DirectoryScanJob
FileScanJob
NativeJob
NoOpJob
## Package org.quartz.jobs

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DirectoryScanListener</code></td>
<td>Interface for objects wishing to receive a 'call-back' from a <code>DirectoryScanJob</code>.</td>
</tr>
<tr>
<td><code>FileScanListener</code></td>
<td>Interface for objects wishing to receive a 'call-back' from a <code>FileScanJob</code>.</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DirectoryScanJob</code></td>
<td>Inspects a directory and compares whether any files' &quot;last modified dates&quot; have changed since the last time it was inspected.</td>
</tr>
<tr>
<td><code>FileScanJob</code></td>
<td>Inspects a file and compares whether its &quot;last modified date&quot; has changed since the last time it was inspected.</td>
</tr>
<tr>
<td><code>NativeJob</code></td>
<td>Built in job for executing native executables in a separate process.</td>
</tr>
<tr>
<td><code>NoOpJob</code></td>
<td>An implementation of Job, that does absolutely nothing - useful for system which only wish to use <code>TriggerListeners</code> and <code>JobListeners</code>, rather than writing Jobs that perform work.</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.jobs

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.jobs.**DirectoryScanJob** (implements org.quartz.**Job**)
  - org.quartz.jobs.**FileScanJob** (implements org.quartz.**Job**)
  - org.quartz.jobs.**NativeJob** (implements org.quartz.**Job**)
  - org.quartz.jobs.**NoOpJob** (implements org.quartz.**Job**)

Interface Hierarchy

- org.quartz.jobs.DirectoryScanListener
- org.quartz.jobs.FileScanListener

Copyright 2001-2011, Terracotta, Inc.
Uses of Package org.quartz.jobs

Packages that use org.quartz.jobs

org.quartz.plugins.xml

Classes in org.quartz.jobs used by org.quartz.plugins.xml

FileScanListener

Interface for objects wishing to receive a 'call-back' from a FileScanJob.

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.DirectoryScanJob

No usage of org.quartz.jobs.DirectoryScanJob

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.jobs.DirectoryScanListener

No usage of org.quartz.jobs.DirectoryScanListener

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.FileScanJob

No usage of org.quartz.jobs.FileScanJob

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface

org.quartz.jobs.FileScanListener

Packages that use FileScanListener

org.quartz.plugins.xml

Uses of FileScanListener in org.quartz.plugins.xml

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLSchedulingDataProcessorPlugin</td>
</tr>
</tbody>
</table>

This plugin loads XML file(s) to add jobs and schedule them with triggers as the scheduler is initialized, and can optionally periodically scan the file for changes.

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.NativeJob

No usage of org.quartz.jobs.NativeJob

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.NoOpJob

No usage of org.quartz.jobs.NoOpJob

Copyright 2001-2011, Terracotta, Inc.
Class EJBInvokerJob

All Implemented Interfaces:
   Job

public class EJBInvokerJob
extends Object
implements Job

A Job that invokes a method on an EJB.

Expects the properties corresponding to the following keys to be in the JobDataMap when it executes:

- EJB_JNDI_NAME_KEY - the JNDI name (location) of the EJB's home interface.
- EJB_METHOD_KEY - the name of the method to invoke on the EJB.
- EJB_ARGS_KEY - an Object[] of the args to pass to the method (optional, if left out, there are no arguments).
- EJB_ARG_TYPES_KEY - an Class[] of the types of the args to pass to the method (optional, if left out, the types will be derived by calling getClass() on each of the arguments).

The following keys can also be used at need:

- INITIAL_CONTEXT_FACTORY - the context factory used to build the context.
- PROVIDER_URL - the name of the environment property for specifying configuration information for the service provider to use.

The result of the EJB method invocation will be available to Job/TriggerListeners via JobExecutionContext.getResult().

Author:
   Andrew Collins, James House, Joel Shellman, Chris Bonham
### Field Summary

<table>
<thead>
<tr>
<th>Static String</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDENTIALS</td>
<td></td>
</tr>
<tr>
<td>EJB_ARG_TYPES_KEY</td>
<td></td>
</tr>
<tr>
<td>EJB_ARGS_KEY</td>
<td></td>
</tr>
<tr>
<td>EJB_JNDI_NAME_KEY</td>
<td></td>
</tr>
<tr>
<td>EJB_METHOD_KEY</td>
<td></td>
</tr>
<tr>
<td>INITIAL_CONTEXT_FACTORY</td>
<td></td>
</tr>
<tr>
<td>PRINCIPAL</td>
<td></td>
</tr>
<tr>
<td>PROVIDER_URL</td>
<td></td>
</tr>
</tbody>
</table>

### Constructor Summary

**EJBInvokerJob()**

### Method Summary

**void execute(JobExecutionContext context)**

Called by the Scheduler when a Trigger fires that is associated with the Job.

### Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Field Detail

EJB_JNDI_NAME_KEY

public static final String EJB_JNDI_NAME_KEY

See Also:
   Constant Field Values

EJB_METHOD_KEY

public static final String EJB_METHOD_KEY

See Also:
   Constant Field Values

EJB_ARG_TYPES_KEY

public static final String EJB_ARG_TYPES_KEY

See Also:
   Constant Field Values

EJB_ARGS_KEY

public static final String EJB_ARGS_KEY

See Also:
   Constant Field Values

INITIAL_CONTEXT_FACTORY

public static final String INITIAL_CONTEXT_FACTORY

See Also:
Constant Field Values

PROVIDER_URL

public static final String PROVIDER_URL

See Also:
Constant Field Values

PRINCIPAL

public static final String PRINCIPAL

See Also:
Constant Field Values

CREDENTIALS

public static final String CREDENTIALS

See Also:
Constant Field Values

Constructor Detail

EJBInvokerJob

public EJBInvokerJob()

Method Detail

execute

public void execute(JobExecutionContext context)
  throws JobExecutionException
Description copied from interface: Job

Called by the Scheduler when a Trigger fires that is associated with the Job.

The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
   execute in interface Job

Throws:
   JobExecutionException - if there is an exception while executing the job.

Copyright 2001-2011, Terracotta, Inc.
org.quartz.jobs.ee.ejb Classes EJBInvokerJob
Package org.quartz.jobs.ee.ejb

Class Summary

EJBInvokerJob | A Job that invokes a method on an EJB.

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.jobs.ee.ejb

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.jobs.ee.ejb.**EJBInvokerJob** (implements org.quartz.**Job**)

Copyright 2001-2011, **Terracotta, Inc.**
Uses of Package
org.quartz.jobs.ee.ejb

No usage of org.quartz.jobs.ee.ejb

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.ee.ejb.EJBInvokerJob

No usage of org.quartz.jobs.ee.ejb.EJBInvokerJob

Copyright 2001-2011, Terracotta, Inc.
public class JmsHelper
extends Object

Utility class that aids in the processing of JMS based jobs and sending of javax.jms.Message

Author:
Weston M. Price

Field Summary

<table>
<thead>
<tr>
<th>Static Type</th>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>CREDENTIALS</td>
</tr>
<tr>
<td>String</td>
<td>INITIAL_CONTEXT_FACTORY</td>
</tr>
<tr>
<td>String</td>
<td>JMS_ACK_MODE</td>
</tr>
<tr>
<td>String</td>
<td>JMS_CONNECTION_FACTORY_JNDI</td>
</tr>
<tr>
<td>String</td>
<td>JMS_DESTINATION_JNDI</td>
</tr>
<tr>
<td>String</td>
<td>JMS_MSG_FACTORY_CLASS_NAME</td>
</tr>
<tr>
<td>String</td>
<td>JMS_PASSWORD</td>
</tr>
<tr>
<td>String</td>
<td>JMS_USE_TXN</td>
</tr>
<tr>
<td>String</td>
<td>JMS_USER</td>
</tr>
</tbody>
</table>
**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void closeResource(Object resource)</code></td>
<td>Closes a resource that has a close() method.</td>
</tr>
<tr>
<td><code>static InitialContext getInitialContext(JobDataMap jobDataMap)</code></td>
<td></td>
</tr>
<tr>
<td><code>static JmsMessageFactory getMessageFactory(String factoryName)</code></td>
<td>Creates the JmsMessageFactory</td>
</tr>
<tr>
<td><code>static boolean isDestinationSecure(JobDataMap jobDataMap)</code></td>
<td></td>
</tr>
<tr>
<td><code>static boolean useTransaction(JobDataMap jobDataMap)</code></td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Field Detail**

**INITIAL_CONTEXT_FACTORY**

```java
public static final String INITIAL_CONTEXT_FACTORY
```

**See Also:**

- Constant Field Values

**PROVIDER_URL**

```java
public static final String PROVIDER_URL
```
See Also:
Constant Field Values

PRINCIPAL

public static final String PRINCIPAL

See Also:
Constant Field Values

CREDENTIALS

public static final String CREDENTIALS

See Also:
Constant Field Values

JMS_CONNECTION_FACTORY_JNDI

public static final String JMS_CONNECTION_FACTORY_JNDI

See Also:
Constant Field Values

JMS_DESTINATION_JNDI

public static final String JMS_DESTINATION_JNDI

See Also:
Constant Field Values

JMS_USER

public static final String JMS_USER
See Also:
Constant Field Values

---

**JMS_PASSWORD**

public static final `String` JMS_PASSWORD

See Also:
Constant Field Values

---

**JMS_ACK_MODE**

public static final `String` JMS_ACK_MODE

See Also:
Constant Field Values

---

**JMS_USE_TXN**

public static final `String` JMS_USE_TXN

See Also:
Constant Field Values

---

**JMS_MSG_FACTORY_CLASS_NAME**

public static final `String` JMS_MSG_FACTORY_CLASS_NAME

See Also:
Constant Field Values

---

**Method Detail**

getInitialContext
public static InitialContext getInitialContext(JobDataMap jobDataMap)
throws NamingException

Throws:
   NamingException

isDestinationSecure

public static boolean isDestinationSecure(JobDataMap jobDataMap)

closeResource

public static void closeResource(Object resource)

   Closes a resource that has a close() method.

Parameters:
   resource - the resource to close.

useTransaction

public static boolean useTransaction(JobDataMap jobDataMap)

getMessageFactory

public static JmsMessageFactory getMessageFactory(String factoryName)
throws JmsJobException

   Creates the JmsMessageFactory

Parameters:
   factoryName -

Returns:

Throws:
   JmsJobException
Copyright 2001-2011, Terracotta, Inc.
Class JmsJobException

java.lang.Object
  ↓ java.lang.Throwable
    ↓ java.lang.Exception
      ↓ org.quartz.SchedulerException
        ↓ org.quartz.jobs.ee.jms.JmsJobException

All Implemented Interfaces:
  Serializable

public class JmsJobException
extends SchedulerException

The JmsJobException is used to indicate an error during sending of a javax.jms.Message.

Author:
  Weston M. Price

See Also:
  Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>JmsJobException()</td>
<td></td>
</tr>
<tr>
<td>JmsJobException(String message)</td>
<td></td>
</tr>
<tr>
<td>JmsJobException(String message, Throwable cause)</td>
<td></td>
</tr>
<tr>
<td>JmsJobException(Throwable cause)</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary


Methods inherited from class org.quartz.SchedulerException
getUnderlyingException, toString

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

JmsJobException

public JmsJobException()  

JmsJobException

public JmsJobException(String message)  

JmsJobException

public JmsJobException(Throwable cause)  

JmsJobException

public JmsJobException(String message, Throwable cause)
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public interface JmsMessageFactory

The JmsMessageFactory interface allows for the creation of javax.jms.Message. This interface is used in constructing a javax.jms.Message that is to be sent upon execution of a JMS enabled job.

Author:
    Weston M. Price

See Also:
    SendDestinationMessageJob, SendQueueMessageJob, SendTopicMessageJob

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>

**Method Detail**

createMessage

javax.jms.Message createMessage(JobDataMap jobDataMap, javax.jms.Session session)

Creates a javax.jms.Message.

**Parameters:**
- jobDataMap - the JobDataMap
- session - the javax.jms.Session

**Returns:**
the javax.jms.Message
org.quartz.jobs.ee.jms

Interfaces  JmsMessageFactory

Classes  
  JmsHelper  
  SendDestinationMessageJob  
  SendQueueMessageJob  
  SendTopicMessageJob

Exceptions  
  JmsJobException
Interface Summary

| JmsMessageFactory | The JmsMessageFactory interface allows for the creation of a javax.jms.Message. |

Class Summary

<table>
<thead>
<tr>
<th>JmsHelper</th>
<th>Utility class that aids in the processing of JMS based jobs and sending of javax.jms.Message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendDestinationMessageJob</td>
<td>A Job that sends a javax.jms.Message to a javax.jms.Destination.</td>
</tr>
<tr>
<td>SendQueueMessageJob</td>
<td>A Job that sends a javax.jms.Message to a javax.jms.Queue.</td>
</tr>
<tr>
<td>SendTopicMessageJob</td>
<td>A Job that sends a javax.jms.Message to a javax.jms.Topic.</td>
</tr>
</tbody>
</table>

Exception Summary

| JmsJobException | The JmsJobException is used to indicate an error during sending of a javax.jms.Message. |

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.jobs.ee.jms

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.jobs.ee.jms.**JmsHelper**
  - org.quartz.jobs.ee.jms.**SendDestinationMessageJob** (implements org.quartz.**Job**)
  - org.quartz.jobs.ee.jms.**SendQueueMessageJob** (implements org.quartz.**Job**)
  - org.quartz.jobs.ee.jms.**SendTopicMessageJob** (implements org.quartz.**Job**)
- java.lang.**Throwable** (implements java.io.**Serializable**)
  - java.lang.**Exception**
    - org.quartz.**SchedulerException**
      - org.quartz.jobs.ee.jms.**JmsJobException**
## Interface Hierarchy

- org.quartz.jobs.ee.jms.**JmsMessageFactory**

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.jobs.ee.jms

Packages that use org.quartz.jobs.ee.jms

org.quartz.jobs.ee.jms

Classes in org.quartz.jobs.ee.jms used by org.quartz.jobs.ee.jms

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JmsJobException</td>
<td>The JmsJobException is used to indicate an error during sending of a javax.jms.Message.</td>
</tr>
<tr>
<td>JmsMessageFactory</td>
<td>The JmsMessageFactory interface allows for the creation of javax.jms.Message.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
org.quartz.jobs.ee.jms Class SendDestinationMessageJob

java.lang.Object
   | org.quartz.jobs.ee.jms.SendDestinationMessageJob

All Implemented Interfaces:
   Job

_____________________________________________________

public final class SendDestinationMessageJob

extends Object
implements Job

A Job that sends a javax.jms.Message to a javax.jms.Destination. Note, this class can only be used in a JMS 1.1 compliant environment.

The following properties are expected to be provided in the JobDataMap:

- JMS_CONNECTION_FACTORY_JNDI - The JNDI name of the JMS Connection Factory.
- JMS_DESTINATION_JNDI - The JNDI name of the JMS destination.
- JMS_USE_TXN - Whether or not to use a transacted javax.jms.Session.
- JMS_ACK_MODE - The acknowledgement mode for the javax.jms.Session.
- JMS_MSG_FACTORY_CLASS_NAME - The implementation class name for the JmsMessageFactory.

The following properties are optional

- JMS_USER - The JMS user for secure destinations.
- JMS_PASSWORD - The JMS password for secure destinations.

The following properties can be used for JNDI support:

- INITIAL_CONTEXT_FACTORY - The java.naming.factory.initial setting for JNDI.
- PROVIDER_URL - The java.naming.provider.url for JNDI.

Author:
Fernando Ribeiro, Weston M. Price, Frank Van Uffelen

See Also:
JmsMessageFactory

---

## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendDestinationMessageJob()</td>
<td></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute(JobExecutionContext jobCtx)</td>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
</tbody>
</table>

## Constructor Detail

### SendDestinationMessageJob

```java
public SendDestinationMessageJob()
```

## Method Detail

### execute

```java
public void execute(JobExecutionContext jobCtx)
throws JobExecutionException
```

**Description copied from interface: Job**

Called by the Scheduler when a Trigger fires that is associated with the
The implementation may wish to set a `result` object on the `JobExecutionContext` before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

**Specified by:**
- `execute` in interface `Job`

**Throws:**
- `JobExecutionException` - if there is an exception while executing the job.
public class SendQueueMessageJob

extends Object
implements Job

A Job that sends a javax.jms.Message to a javax.jms.Queue

The following properties are expected to be provided in the JobDataMap:

- JMS_CONNECTION_FACTORY_JNDI - The JNDI name of the JMS Connection Factory.
- JMS_DESTINATION_JNDI - The JNDI name of the JMS destination.
- JMS_USE_TXN - Whether or not to use a transacted javax.jms.Session.
- JMS_ACK_MODE - The acknowledgement mode for the javax.jms.Session.
- JMS_MSG_FACTORY_CLASS_NAME - The implementation class name for the JmsMessageFactory.

The following properties are optional

- JMS_USER - The JMS user for secure destinations.
- JMS_PASSWORD - The JMS password for secure destinations.

The following properties can be used for JNDI support:

- INITIAL_CONTEXT_FACTORY - The java.naming.factory.initial setting for JNDI.
- PROVIDER_URL - The java.naming.provider.url for JNDI.

Author:
Weston M. Price (little fixes v. in 1.6.0 by Toni Alatalo)
See Also:

JmsMessageFactory

## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendQueueMessageJob()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute(JobExecutionContext context)</td>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
</tbody>
</table>

## Constructor Detail

### SendQueueMessageJob

```java
public SendQueueMessageJob()
```

## Method Detail

### execute

```java
public void execute(JobExecutionContext context) throws JobExecutionException
```

Description copied from interface: Job

Called by the Scheduler when a Trigger fires that is associated with the Job.
The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
   execute in interface Job

Throws:
   JobExecutionException - if there is an exception while executing the job.

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS   NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES   NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
Class SendTopicMessageJob

public final class SendTopicMessageJob
extends Object
implements Job

A Job that sends a javax.jms.Message to a javax.jms.Topic.

The following properties are expected to be provided in the JobDataMap:

- JMS_CONNECTION_FACTORY_JNDI - The JNDI name of the JMS Connection Factory.
- JMS_DESTINATION_JNDI - The JNDI name of the JMS destination.
- JMS_USE_TXN - Whether or not to use a transacted javax.jms.Session.
- JMS_ACK_MODE - The acknowledgment mode for the javax.jms.Session.
- JMS_MSG_FACTORY_CLASS_NAME - The implementation class name for the JmsMessageFactory.

The following properties are optional

- JMS_USER - The JMS user for secure destinations.
- JMS_PASSWORD - The JMS password for secure destinations.

The following properties can be used for JNDI support:

- INITIAL_CONTEXT_FACTORY - The java.naming.factory.initial setting for JNDI.
- PROVIDER_URL - The java.naming.provider.url for JNDI.

Author:
Fernando Ribeiro, Weston M. Price
See Also:

JmsMessageFactory

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>SendTopicMessageJob()</th>
</tr>
</thead>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>execute(JobExecutionContext jobCtx)</td>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object:

class, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

**SendTopicMessageJob**

```java
class SendTopicMessageJob

public SendTopicMessageJob()
```

### Method Detail

**execute**

```java
public void execute(JobExecutionContext jobCtx)
```

Throws: JobExecutionException

Description copied from interface: Job

Called by the Scheduler when a Trigger fires that is associated with the Job.
The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
   execute in interface Job

Throws:
   JobExecutionException - if there is an exception while executing the job.
Uses of Class
org.quartz.jobs.ee.jms.JmsHelper

No usage of org.quartz.jobs.ee.jms.JmsHelper

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.jobs.ee.jms.JmsJobException**

## Packages that use *JmsJobException*

<table>
<thead>
<tr>
<th>Package</th>
<th><code>org.quartz.jobs.ee.jms</code></th>
</tr>
</thead>
</table>

## Uses of *JmsJobException* in *org.quartz.jobs.ee.jms*

## Methods in *org.quartz.jobs.ee.jms* that throw *JmsJobException*

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>JmsMessageFactory</code> JmsHelper.getMessageFactory(String factoryName)</td>
<td>Creates the JmsMessageFactory</td>
</tr>
</tbody>
</table>
Packages that use `JmsMessageFactory`

**org.quartz.jobs.ee.jms**

Uses of `JmsMessageFactory` in `org.quartz.jobs.ee.jms`

Methods in `org.quartz.jobs.ee.jms` that return `JmsMessageFactory`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>JmsMessageFactory</code> JmsHelper.getMessageFactory(String factoryName)`</td>
<td>Creates the JmsMessageFactory</td>
</tr>
</tbody>
</table>

Uses of Class

org.quartz.jobs.ee.jms.SendDestinationMessageJob

No usage of org.quartz.jobs.ee.jms.SendDestinationMessageJob

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.ee.jms.SendQueueMessageJob

No usage of org.quartz.jobs.ee.jms.SendQueueMessageJob

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.ee.jms.SendTopicMessageJob

No usage of org.quartz.jobs.ee.jms.SendTopicMessageJob

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>DETAIL: FIELD</td>
</tr>
</tbody>
</table>
public class JMXInvokerJob

extends Object
implements Job

Generic JMX invoker Job. It supports any number or type of parameters to the JMX bean.

The required parameters are as follows (case doesn't matter):

**JMX_OBJECTNAME**
This is the fully qualified name of the object (ie in JBoss to lookup the log4j jmx bean you would specify "jboss.system:type=Log4jService,service=Logging"

**JMX_METHOD**
This is the method to invoke on the specified JMX Bean. (ie in JBoss to change the log level you would specify "setLoggerLevel"

**JMX_PARAMDEFS**
This is a definition of the parameters to be passed to the specified method and their corresponding java types. Each parameter definition is comma seperated and has the following parts: . Type is the java type for the parameter. The following types are supported:

i - is for int

l - is for long

f - is for float

d - is for double
s - is for String

b - is for boolean

For ilfdb use lower for native type and upper for object wrapper. The name portion of the definition is the name of the parameter holding the string value. (ie s:fname,s:lname would require 2 parameters of the name fname and lname and would be passed in that order to the method.

**Author:**

James Nelson (jmn@provident-solutions.com) -- Provident Solutions LLC

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JMXInvokerJob()</strong></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void execute(JobExecutionContext context)</strong></td>
<td>Called by the Scheduler when a Trigger fires that is associated with the Job.</td>
</tr>
<tr>
<td><strong>protected org.slf4j.Logger getLog()</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`
execute

public void execute(JobExecutionContext context) throws JobExecutionException

Description copied from interface: Job

Called by the Scheduler when a Trigger fires that is associated with the Job.

The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
execute in interface Job

Throws:
JobExecutionException - if there is an exception while executing the job.

getLog

protected org.slf4j.Logger getLog()
org.quartz.jobs.ee.jmx Classes JMXInvokerJob
Package org.quartz.jobs.ee.jmx

Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMXInvokerJob</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.jobs.ee.jmx

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
- org.quartz.jobs.ee.jmx.**JMXInvokerJob** (implements org.quartz.**Job**)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.jobs.ee.jmx

No usage of org.quartz.jobs.ee.jmx

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.jobs.ee.jmx.JMXInvokerJob

No usage of org.quartz.jobs.ee.jmx.JMXInvokerJob

Copyright 2001-2011, Terracotta, Inc.
org.quartz.jobs.ee.mail Classes SendMailJob
SendMailJob.MailInfo
### Package org.quartz.jobs.ee.mail

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SendMailJob</strong></td>
</tr>
<tr>
<td>A Job which sends an e-mail with the configured</td>
</tr>
<tr>
<td>content to the configured recipient.</td>
</tr>
<tr>
<td><strong>SendMailJob.MailInfo</strong></td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Hierarchy For Package org.quartz.jobs.ee.mail

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.jobs.ee.mail.**SendMailJob** (implements org.quartz.**Job**)
  - org.quartz.jobs.ee.mail.**SendMailJob.MailInfo**
# Uses of Package
/org.quartz.jobs.ee.mail

## Packages that use org.quartz.jobs.ee.mail

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.jobs.ee.mail</td>
</tr>
</tbody>
</table>

## Classes in org.quartz.jobs.ee.mail used by org.quartz.jobs.ee.mail

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendMailJob.MailInfo</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.jobs.ee.mail  Class SendMailJob

java.lang.Object  
  └ org.quartz.jobs.ee.mail.SendMailJob

All Implemented Interfaces:
  Job

public class SendMailJob
extends Object
implements Job

A Job which sends an e-mail with the configured content to the configured recipient.

Author:
  James House

Nested Class Summary

| protected static class | SendMailJob.MailInfo |

Field Summary

<table>
<thead>
<tr>
<th>static String</th>
<th>PROP_CC_RECIPIENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>The e-mail address to cc the mail to.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String</th>
<th>PROP_CONTENT_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The message content type.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String</th>
<th>PROP_MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The e-mail message body.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String</th>
<th>PROP_RECIPIENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>The e-mail address to send the mail to.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String</th>
<th>PROP_REPLY_TO</th>
</tr>
</thead>
</table>
The e-mail address the message should say to reply to.

<table>
<thead>
<tr>
<th>static String PROP_SENDER</th>
<th>The e-mail address to claim the mail is from.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String PROP_SMTP_HOST</td>
<td>The host name of the smtp server.</td>
</tr>
<tr>
<td>static String PROP_SUBJECT</td>
<td>The subject to place on the e-mail.</td>
</tr>
</tbody>
</table>

## Constructor Summary

SendMailJob()
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

PROP_SMTP_HOST

public static final String PROP_SMTP_HOST

The host name of the smtp server. REQUIRED.

See Also:
Constant Field Values

PROP_RECIPIENT

public static final String PROP_RECIPIENT

The e-mail address to send the mail to. REQUIRED.

See Also:
Constant Field Values

PROP_CC_RECIPIENT

public static final String PROP_CC_RECIPIENT

The e-mail address to cc the mail to. Optional.

See Also:
Constant Field Values
**PROP_SENDER**

public static final String PROP_SENDER

The e-mail address to claim the mail is from. REQUIRED.

See Also:  
Constant Field Values

---

**PROP_REPLY_TO**

public static final String PROP_REPLY_TO

The e-mail address the message should say to reply to. Optional.

See Also:  
Constant Field Values

---

**PROP_SUBJECT**

public static final String PROP_SUBJECT

The subject to place on the e-mail. REQUIRED.

See Also:  
Constant Field Values

---

**PROP_MESSAGE**

public static final String PROP_MESSAGE

The e-mail message body. REQUIRED.

See Also:  
Constant Field Values
PROP_CONTENT_TYPE

public static final String PROP_CONTENT_TYPE

The message content type. For example, "text/html". Optional.

See Also:

Constant Field Values

Constructor Detail

SendMailJob

public SendMailJob()

Method Detail

execute

public void execute(JobExecutionContext context)
    throws JobExecutionException

Description copied from interface: Job

Called by the Scheduler when a Trigger fires that is associated with the Job.

The implementation may wish to set a result object on the JobExecutionContext before this method exits. The result itself is meaningless to Quartz, but may be informative to JobListeners or TriggerListeners that are watching the job's execution.

Specified by:
    execute in interface Job

Throws:
    JobExecutionException - if there is an exception while executing the job.

See Also:
getLog
protected org.slf4j.Logger getLog()

prepareMimeMessage
protected javax.mail.internet.MimeMessage prepareMimeMessage(SendMailJob.MailInfo) throws javax.mail.MessagingException

Throws:
javax.mail.MessagingException

setMimeMessageContent
protected void setMimeMessageContent(javax.mail.internet.MimeMessage SendMailJob.MailInfo mailInfo) throws javax.mail.MessagingException

Throws:
javax.mail.MessagingException

getMailSession
protected javax.mail.Session getMailSession(SendMailJob.MailInfo mailInfo) throws javax.mail.MessagingException

Throws:
javax.mail.MessagingException

createMailInfo
protected SendMailJob.MailInfo createMailInfo()
protected SendMailJob.MailInfo populateMailInfo(JobDataMap data, SendMailJob.MailInfo)

protected String getRequiredParm(JobDataMap data, String property, String constantName)

protected String getOptionalParm(JobDataMap data, String property)

Copyright 2001-2011, Terracotta, Inc.
org.quartz.jobs.ee.mail  Class SendMailJob.MailInfo

java.lang.Object
  └ org.quartz.jobs.ee.mail.SendMailJob.MailInfo

Enclosing class:  
  SendMailJob

protected static class SendMailJob.MailInfo

extends Object

---

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected SendMailJob.MailInfo()</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getCC()</td>
</tr>
<tr>
<td>String getContentType()</td>
</tr>
<tr>
<td>String getFrom()</td>
</tr>
<tr>
<td>String getMessage()</td>
</tr>
<tr>
<td>String getReplyTo()</td>
</tr>
<tr>
<td>String getSmtpHost()</td>
</tr>
<tr>
<td>String getSubject()</td>
</tr>
<tr>
<td>String getTo()</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>void setCc(String cc)</td>
</tr>
<tr>
<td>void setContentType(String contentType)</td>
</tr>
<tr>
<td>void setFrom(String from)</td>
</tr>
<tr>
<td>void setMessage(String message)</td>
</tr>
<tr>
<td>void setReplyTo(String replyTo)</td>
</tr>
<tr>
<td>void setSmtpHost(String smtpHost)</td>
</tr>
<tr>
<td>void setSubject(String subject)</td>
</tr>
<tr>
<td>void setTo(String to)</td>
</tr>
<tr>
<td>String toString()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object:
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait

Constructor Detail

SendMailJob.MailInfo

protected SendMailJob.MailInfo()

Method Detail

toString
public String toString()

    Overrides:
    toString in class Object

getCc

public String getCc()

setCc

public void setCc(String cc)

getContentType

public String getContentType()

setContentType

public void setContentType(String contentType)

getFrom

public String getFrom()

setFrom

public void setFrom(String from)

getMessage

public String getMessage()
**setMessage**

public void setMessage(String message)

**getReplyTo**

public String getReplyTo()

**setReplyTo**

public void setReplyTo(String replyTo)

**getSmtpHost**

public String getSmtpHost()

**setSmtpHost**

public void setSmtpHost(String smtpHost)

**getSubject**

public String getSubject()

**setSubject**

public void setSubject(String subject)

**getTo**

public String getTo()
public void setTo(String to)
Uses of Class
org.quartz.jobs.ee.mail.SendMailJob

No usage of org.quartz.jobs.ee.mail.SendMailJob

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.jobs.ee.mail.SendMailJob.MailInfo**

## Packages that use **SendMailJob.MailInfo**

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.jobs.ee.mail</td>
</tr>
</tbody>
</table>

## Uses of **SendMailJob.MailInfo** in **org.quartz.jobs.ee.mail**

## Methods in **org.quartz.jobs.ee.mail** that return **SendMailJob.MailInfo**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected <strong>SendMailJob</strong>.createMailInfo()</td>
</tr>
<tr>
<td>protected <strong>SendMailJob</strong>.populateMailInfo(JobDataMap data, SendMailJob.MailInfo mailInfo)</td>
</tr>
</tbody>
</table>

## Methods in **org.quartz.jobs.ee.mail** with parameters of type **SendMailJob.MailInfo**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected javax.mail.Session <strong>SendMailJob</strong>.getMailSession(SendMailJob.MailInfo mailInfo)</td>
</tr>
<tr>
<td>protected <strong>SendMailJob</strong>.populateMailInfo(JobDataMap data, SendMailJob.MailInfo mailInfo)</td>
</tr>
<tr>
<td>protected javax.mail.internet.MimeMessage <strong>SendMailJob</strong>.prepareMimeMessage(SendMailJob.MailInfo mailInfo)</td>
</tr>
<tr>
<td>protected void <strong>SendMailJob</strong>.setMimeMessageContent(javax.mail.internet.MimeMessage mimeMessage, SendMailJob.MailInfo mailInfo)</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
public class BroadcastJobListener

extends Object
implements JobListener

Holds a List of references to JobListener instances and broadcasts all events to them (in order).

The broadcasting behavior of this listener to delegate listeners may be more convenient than registering all of the listeners directly with the Scheduler, and provides the flexibility of easily changing which listeners get notified.

Author:  
James House (jhouse AT revolition DOT net)

See Also:  
addListener(org.quartz.JobListener), 
removeListener(org.quartz.JobListener), removeListener(String)

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>BroadcastJobListener(String name)</td>
</tr>
<tr>
<td>Construct an instance with the given name.</td>
</tr>
<tr>
<td>BroadcastJobListener(String name, List&lt;JobListener&gt; listeners)</td>
</tr>
<tr>
<td>Construct an instance with the given name, and List of listeners.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addListener(JobListener listener)</td>
</tr>
</tbody>
</table>
### List<JobListener>

#### `getListeners()`

Get the list of JobListeners.

#### `getName()`

Get the name of the JobListener.

### `JobListener`

#### `jobExecutionVetoed(JobExecutionContext context)`

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed its execution.

#### `jobToBeExecuted(JobExecutionContext context)`

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

#### `jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)`

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(XX) method has been called.

#### `removeListener(JobListener listener)`

Remove a JobListener.

#### `removeListener(String listenerName)`

Remove a JobListener by name.

---

**Methods inherited from class java.lang.Object**

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
Parameters:
  
  name - the name of this instance

---

**BroadcastJobListener**

public BroadcastJobListener(String name, 
                           List&lt;JobListener&gt; listeners)

Construct an instance with the given name, and List of listeners.

**Parameters:**
  
  name - the name of this instance
  
  listeners - the initial List of JobListeners to broadcast to.

---

**Method Detail**

**getName**

public String getName()

  **Description copied from interface: JobListener**

  Get the name of the JobListener.

  **Specified by:**
  
  getName in interface JobListener
removeListener
public boolean removeListener(String listenerName)

getListeners
public List<JobListener> getListeners()

jobToBeExecuted
public void jobToBeExecuted(JobExecutionContext context)

Description copied from interface: JobListener

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

This method will not be invoked if the execution of the Job was vetoed by a TriggerListener.

Specified by:
    jobToBeExecuted in interface JobListener
See Also:
    JobListener.jobExecutionVetoed(JobExecutionContext)

jobExecutionVetoed
public void jobExecutionVetoed(JobExecutionContext context)

Description copied from interface: JobListener

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

Specified by:
    jobExecutionVetoed in interface JobListener
jobWasExecuted

public void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)

Description copied from interface: JobListener

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

Specified by:

jobWasExecuted in interface JobListener
Class BroadcastSchedulerListener

public class BroadcastSchedulerListener
    extends Object
    implements SchedulerListener

Holds a List of references to SchedulerListener instances and broadcasts all events to them (in order).

This may be more convenient than registering all of the listeners directly with the Scheduler, and provides the flexibility of easily changing which listeners get notified.

Author:
    James House (jhouse AT revolition DOT net)

See Also:
    addListener(org.quartz.SchedulerListener),
    removeListener(org.quartz.SchedulerListener)

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>BroadcastSchedulerListener()</td>
</tr>
<tr>
<td>BroadcastSchedulerListener(List listeners)</td>
</tr>
<tr>
<td>Construct an instance with the given List of listeners.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addListener(SchedulerListener listener)</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>----------------------------------------------</td>
</tr>
<tr>
<td><code>List&lt;SchedulerListener&gt;</code> <code>getListeners()</code></td>
</tr>
<tr>
<td><code>jobAdded(JobDetail jobDetail)</code></td>
</tr>
<tr>
<td><code>jobDeleted(JobKey jobKey)</code></td>
</tr>
<tr>
<td><code>jobPaused(JobKey key)</code></td>
</tr>
<tr>
<td><code>jobResumed(JobKey key)</code></td>
</tr>
<tr>
<td><code>jobScheduled(Trigger trigger)</code></td>
</tr>
<tr>
<td><code>jobsPaused(String jobGroup)</code></td>
</tr>
<tr>
<td><code>jobsResumed(String jobGroup)</code></td>
</tr>
<tr>
<td><code>jobUnscheduled(TriggerKey triggerKey)</code></td>
</tr>
<tr>
<td><code>boolean removeListener(SchedulerListener listener)</code></td>
</tr>
<tr>
<td><code>schedulerError(String msg, SchedulerException cause)</code></td>
</tr>
<tr>
<td><code>void schedulerInStandbyMode()</code></td>
</tr>
</tbody>
</table>
that it has move to standby mode.

```java
void schedulerShutdown()
    Called by the Scheduler to inform the listener that it has shutdown.

void schedulerShuttingdown()
    Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

void schedulerStarted()
    Called by the Scheduler to inform the listener that it has started.

void schedulingDataCleared()
    Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.

void triggerFinalized(Trigger trigger)
    Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.

void triggerPaused(TriggerKey key)
    Called by the Scheduler when a Trigger has been paused.

void triggerResumed(TriggerKey key)
    Called by the Scheduler when a Trigger has been un-paused.

void triggersPaused(String triggerGroup)
    Called by the Scheduler when a group of Triggers has been paused.

void triggersResumed(String triggerGroup)
    Called by the Scheduler when a group of Triggers has been un-paused.
```

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

**Constructor Detail**
public BroadcastSchedulerListener()

public BroadcastSchedulerListener(List listeners)

Construct an instance with the given List of listeners.

Parameters:
- listeners - the initial List of SchedulerListeners to broadcast to.

Method Detail

addListener

public void addListener(SchedulerListener listener)

removeListener

public boolean removeListener(SchedulerListener listener)

getListeners

public List<SchedulerListener> getListeners()

jobAdded

public void jobAdded(JobDetail jobDetail)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been added.
Specified by:  
jobAdded in interface SchedulerListener

---

**jobDeleted**

public void **jobDeleted**(JobKey jobKey)

- **Description copied from interface:** SchedulerListener
- Called by the Scheduler when a JobDetail has been deleted.
- **Specified by:**
  jobDeleted in interface SchedulerListener

---

**jobScheduled**

public void **jobScheduled**(Trigger trigger)

- **Description copied from interface:** SchedulerListener
- Called by the Scheduler when a JobDetail is scheduled.
- **Specified by:**
  jobScheduled in interface SchedulerListener

---

**jobUnscheduled**

public void **jobUnscheduled**(TriggerKey triggerKey)

- **Description copied from interface:** SchedulerListener
- Called by the Scheduler when a JobDetail is unscheduled.
- **Specified by:**
  jobUnscheduled in interface SchedulerListener

**See Also:**
- SchedulerListener.schedulingDataCleared()
triggerFinalized

public void triggerFinalized(Trigger trigger)

Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.

Specified by:
   triggerFinalized in interface SchedulerListener

triggerPaused

public void triggerPaused(TriggerKey key)

Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has been paused.

Specified by:
   triggerPaused in interface SchedulerListener

triggersPaused

public void triggersPaused(String triggerGroup)

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of Triggers has been paused.

If all groups were paused then triggerGroup will be null

Specified by:
   triggersPaused in interface SchedulerListener
Parameters:
   triggerGroup - the paused group, or null if all were paused
triggerResumed

public void triggerResumed(TriggerKey key)

Description copied from interface: SchedulerListener
Called by the Scheduler when a Trigger has been un-paused.

Specified by:
    triggerResumed in interface SchedulerListener

triggersResumed

public void triggersResumed(String triggerGroup)

Description copied from interface: SchedulerListener
Called by the Scheduler when a group of Triggers has been un-paused.

Specified by:
    triggersResumed in interface SchedulerListener

schedulingDataCleared

public void schedulingDataCleared()

Description copied from interface: SchedulerListener
Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.

Specified by:
    schedulingDataCleared in interface SchedulerListener

jobPaused

public void jobPaused(JobKey key)
SchedulerListener

Called by the Scheduler when a JobDetail has been paused.

Specified by:

jobPaused in interface SchedulerListener

jobsPaused

public void jobsPaused(String jobGroup)

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of JobDetails has been paused.

Specified by:

jobsPaused in interface SchedulerListener

Parameters:

jobGroup - the paused group, or null if all were paused

jobResumed

public void jobResumed(JobKey key)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been un-paused.

Specified by:

jobResumed in interface SchedulerListener

jobsResumed

public void jobsResumed(String jobGroup)

Description copied from interface: SchedulerListener
Called by the Scheduler when a group of JobDetails has been un-paused.

**Specified by:**

`jobsResumed` in interface `SchedulerListener`

---

**schedulerError**

```java
class SchedulerException
{
    public void schedulerError(String msg, SchedulerException cause)
}
```

**Description copied from interface:** `SchedulerListener`

Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

The `getErrorCode()` method of the given SchedulerException can be used to determine more specific information about the type of error that was encountered.

**Specified by:**

`schedulerError` in interface `SchedulerListener`

---

**schedulerStarted**

```java
public void schedulerStarted()
```

**Description copied from interface:** `SchedulerListener`

Called by the Scheduler to inform the listener that it has started.

**Specified by:**

`schedulerStarted` in interface `SchedulerListener`

---

**schedulerInStandbyMode**

```java
public void schedulerInStandbyMode()
```
Description copied from interface: `SchedulerListener`

Called by the Scheduler to inform the listener that it has move to standby mode.

Specified by:

`schedulerInStandbyMode` in interface `SchedulerListener`

---

`schedulerShutdown`

```java
public void schedulerShutdown()
```

Description copied from interface: `SchedulerListener`

Called by the Scheduler to inform the listener that it has shutdown.

Specified by:

`schedulerShutdown` in interface `SchedulerListener`

---

`schedulerShuttingdown`

```java
public void schedulerShuttingdown()
```

Description copied from interface: `SchedulerListener`

Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

Specified by:

`schedulerShuttingdown` in interface `SchedulerListener`

---

Copyright 2001-2011, Terrain, Inc.
Class BroadcastTriggerListener

public class BroadcastTriggerListener extends Object implements TriggerListener

Holds a List of references to TriggerListener instances and broadcasts all events to them (in order).

The broadcasting behavior of this listener to delegate listeners may be more convenient than registering all of the listeners directly with the Scheduler, and provides the flexibility of easily changing which listeners get notified.

Author:
James House (jhouse AT revolition DOT net)

See Also:
addListener(org.quartz.TriggerListener),
removeListener(org.quartz.TriggerListener),
removeListener(String)

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BroadcastTriggerListener(String name)</td>
<td>Construct an instance with the given name.</td>
</tr>
<tr>
<td>BroadcastTriggerListener(String name, List listeners)</td>
<td>Construct an instance with the given name, and List of listeners.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addListener(TriggerListener listener)</td>
<td></td>
</tr>
</tbody>
</table>
List<TriggerListener> getListeners()

String getName()
   Get the name of the TriggerListener.

boolean removeListener(String listenerName)

boolean removeListener(TriggerListener listener)

void triggerComplete(Trigger trigger, JobExecutionContext<Trigger.CompletedExecutionInstruction triggerInstructionCode)
   Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered(xx) method was called.

void triggerFired(Trigger trigger, JobExecutionContext context)
   Called by the Scheduler when a Trigger has fired, and its associated JobDetail is about to be executed.

void triggerMisfired(Trigger trigger)
   Called by the Scheduler when a Trigger has misfired.

boolean vetoJobExecution(Trigger trigger, JobExecutionContext context)
   Called by the Scheduler when a Trigger has fired, and its associated JobDetail is about to be executed.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

BroadcastTriggerListener

public BroadcastTriggerListener(String name)

   Construct an instance with the given name. (Remember to add some
delegate listeners!)

**Parameters:**
- `name` - the name of this instance

---

**BroadcastTriggerListener**

```java
public BroadcastTriggerListener(String name,
                                 List listeners)
```

Construct an instance with the given name, and List of listeners.

**Parameters:**
- `name` - the name of this instance
- `listeners` - the initial List of TriggerListeners to broadcast to.

---

**Method Detail**

### getName

```java
public String getName()
```

**Description copied from interface: TriggerListener**

Get the name of the TriggerListener.

**Specified by:**
- ```java
    getName in interface TriggerListener
  ```

### addListener

```java
public void addListener(TriggerListener listener)
```

### removeListener

```java
public boolean removeListener(TriggerListener listener)
```
removeListener

public boolean removeListener(String listenerName)

getListeners

public List<TriggerListener> getListeners()

triggerFired

public void triggerFired(Trigger trigger,
                              JobExecutionContext context)

   Description copied from interface: TriggerListener

   Called by the Scheduler when a Trigger has fired, and it's associated
   JobDetail is about to be executed.

   It is called before the vetoJobExecution(..) method of this interface.

   Specified by:
   triggerFired in interface TriggerListener

   Parameters:
   trigger - The Trigger that has fired.
   context - The JobExecutionContext that will be passed to the
              Job'sexecute(xx) method.

vetoJobExecution

public boolean vetoJobExecution(Trigger trigger,
                                  JobExecutionContext context)

   Description copied from interface: TriggerListener

   Called by the Scheduler when a Trigger has fired, and it's associated
   JobDetail is about to be executed. If the implementation vetos the
execution (via returning true), the job's execute method will not be called.

It is called after the triggerFired(...) method of this interface.

Specified by:
  vetoJobExecution in interface TriggerListener

Parameters:
  trigger - The Trigger that has fired.
  context - The JobExecutionContext that will be passed to the Job's execute method.

---

triggerMisfired

public void triggerMisfired(Trigger trigger)

Description copied from interface: TriggerListener

Called by the Scheduler when a Trigger has misfired.

Consideration should be given to how much time is spent in this method, as it will affect all triggers that are misfiring. If you have lots of triggers misfiring at once, it could be an issue if this method does a lot.

Specified by: triggerMisfired in interface TriggerListener

Parameters:
  trigger - The Trigger that has misfired.

---

triggerComplete

public void triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)

Description copied from interface: TriggerListener

Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered method has been
called.

Specified by:

triggerComplete in interface TriggerListener

Parameters:

trigger - The Trigger that was fired.
context - The JobExecutionContext that was passed to the Job's execute(XX) method.
triggerInstructionCode - the result of the call on the Trigger's triggered(XX) method.
org.quartz.listeners  Class JobChainingJobListener

java.lang.Object  
   org.quartz.listeners.JobListenerSupport
   org.quartz.listeners.JobChainingJobListener

All Implemented Interfaces:
   JobListener

public class JobChainingJobListener
   extends JobListenerSupport

Keeps a collection of mappings of which Job to trigger after the completion of a given job. If this listener is notified of a job completing that has a mapping, then it will then attempt to trigger the follow-up job. This achieves "job chaining", or a "poor man's workflow".

Generally an instance of this listener would be registered as a global job listener, rather than being registered directly to a given job.

If for some reason there is a failure creating the trigger for the follow-up job (which would generally only be caused by a rare serious failure in the system, or the non-existence of the follow-up job), an error message is logged, but no other action is taken. If you need more rigorous handling of the error, consider scheduling the triggering of the flow-up job within your job itself.

Author:
   James House (jhouse AT revolition DOT net)

---

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobChainingJobListener(String name)</td>
</tr>
<tr>
<td>Construct an instance with the given name.</td>
</tr>
</tbody>
</table>

---

| Method Summary |
### addJobChainLink

```java
void addJobChainLink(JobKey firstJob, JobKey secondJob)
```

Add a chain mapping - when the Job identified by the first key completes the job identified by the second key will be triggered.

### getName

```java
String getName()
```

Get the name of the JobListener.

### jobWasExecuted

```java
void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)
```

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

### Constructor Detail

#### JobChainingJobListener

```java
public JobChainingJobListener(String name)
```

Construct an instance with the given name.

**Parameters:**

- `name` - the name of this instance

### Method Detail

#### getName

```java
public String getName()
```
Description copied from interface: JobListener

Get the name of the JobListener.

addJobChainLink

```java
public void addJobChainLink(JobKey firstJob, JobKey secondJob)
```

Add a chain mapping - when the Job identified by the first key completes the job identified by the second key will be triggered.

**Parameters:**
- firstJob - a JobKey with the name and group of the first job
- secondJob - a JobKey with the name and group of the follow-up job

jobWasExecuted

```java
public void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)
```

Description copied from interface: JobListener

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

**Specified by:**
- jobWasExecuted in interface JobListener

**Overrides:**
- jobWasExecuted in class JobListenerSupport

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.listeners Class JobListenerSupport

java.lang.Object
   └ org.quartz.listeners.JobListenerSupport

All Implemented Interfaces:
   JobListener

Direct Known Subclasses:
   JobChainingJobListener

public abstract class JobListenerSupport
    extends Object
    implements JobListener

A helpful abstract base class for implementors of JobListener.

The methods in this class are empty so you only need to override the subset for
the JobListener events you care about.

You are required to implement JobListener.getName() to return the unique
name of your JobListener.

See Also:
   JobListener

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobListenerSupport()</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getLog()</td>
<td>org.slf4j.Logger</td>
<td>Get the Logger for this class's category.</td>
</tr>
<tr>
<td>jobExecutionVetoed(JobExecutionContext context)</td>
<td>void</td>
<td></td>
</tr>
</tbody>
</table>
Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed its execution.

```java
void jobToBeExecuted(JobExecutionContext context)
```

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

```java
void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)
```

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

### Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Methods inherited from interface org.quartz.JobListener

- `getName`

### Constructor Detail

#### JobListenerSupport

```java
public JobListenerSupport()
```

### Method Detail

#### getLog

```java
protected org.slf4j.Logger getLog()
```

Get the Logger for this class's category. This should be used by subclasses for logging.
jobToBeExecuted

public void jobToBeExecuted(JobExecutionContext context)

Description copied from interface: JobListener

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

This method will not be invoked if the execution of the Job was vetoed by a TriggerListener.

Specified by:
    jobToBeExecuted in interface JobListener
See Also:
    JobListener.jobExecutionVetoed(JobExecutionContext)

jobExecutionVetoed

public void jobExecutionVetoed(JobExecutionContext context)

Description copied from interface: JobListener

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

Specified by:
    jobExecutionVetoed in interface JobListener
See Also:
    JobListener.jobToBeExecuted(JobExecutionContext)

jobWasExecuted

public void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)

Description copied from interface: JobListener
Called by the Scheduler after a JobDetail has been executed, and be for
the associated Trigger's triggered(xx) method has been called.

Specified by:

jobWasExecuted in interface JobListener
org.quartz.listeners Classes  BroadcastJobListener
BroadcastSchedulerListener  BroadcastTriggerListener
JobChainingJobListener  JobListenerSupport
SchedulerListenerSupport  TriggerListenerSupport
## Package org.quartz.listeners

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BroadcastJobListener</strong></td>
</tr>
<tr>
<td><strong>BroadcastSchedulerListener</strong></td>
</tr>
<tr>
<td><strong>BroadcastTriggerListener</strong></td>
</tr>
<tr>
<td><strong>JobChainingJobListener</strong></td>
</tr>
<tr>
<td><strong>JobListenerSupport</strong></td>
</tr>
<tr>
<td><strong>SchedulerListenerSupport</strong></td>
</tr>
<tr>
<td><strong>TriggerListenerSupport</strong></td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.listeners

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.listeners.**BroadcastJobListener** (implements org.quartz.**JobListener**)
  - org.quartz.listeners.**BroadcastSchedulerListener** (implements org.quartz.**SchedulerListener**)
  - org.quartz.listeners.**BroadcastTriggerListener** (implements org.quartz.**TriggerListener**)
  - org.quartz.listeners.**JobListenerSupport** (implements org.quartz.**JobListener**)
    - org.quartz.listeners.**JobChainingJobListener**
  - org.quartz.listeners.**SchedulerListenerSupport** (implements org.quartz.**SchedulerListener**)
  - org.quartz.listeners.**TriggerListenerSupport** (implements org.quartz.**TriggerListener**)
# Uses of Package org.quartz.listeners

## Packages that use org.quartz.listeners

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.core</td>
<td>Contains the core classes and interfaces for the Quartz job scheduler.</td>
</tr>
<tr>
<td>org.quartz.ee.jta</td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
<tr>
<td>org.quartz.impl</td>
<td></td>
</tr>
<tr>
<td>org.quartz.listeners</td>
<td></td>
</tr>
</tbody>
</table>

## Classes in org.quartz.listeners used by org.quartz.core

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerListenerSupport</td>
<td>A helpful abstract base class for implementors of SchedulerListener.</td>
</tr>
</tbody>
</table>

## Classes in org.quartz.listeners used by org.quartz.ee.jta

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerListenerSupport</td>
<td>A helpful abstract base class for implementors of SchedulerListener.</td>
</tr>
</tbody>
</table>

## Classes in org.quartz.listeners used by org.quartz.impl

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerListenerSupport</td>
<td>A helpful abstract base class for implementors of SchedulerListener.</td>
</tr>
</tbody>
</table>
Classes in `org.quartz.listeners` used by `org.quartz.listeners`:

JobListenerSupport

A helpful abstract base class for implementors of JobListener.
public abstract class SchedulerListenerSupport
extends Object
implements SchedulerListener

A helpful abstract base class for implementors of SchedulerListener.

The methods in this class are empty so you only need to override the subset for
the SchedulerListener events you care about.

See Also:
SchedulerListener

Constructor Summary

SchedulerListenerSupport()

Method Summary

protected org.slf4j.Logger getLog()
Get the Logger for this class's category.

void jobAdded(JobDetail jobDetail)
Called by the Scheduler when a JobDetail has been added.
void **jobDeleted** (**JobKey** jobKey)
   Called by the Scheduler when a JobDetail has been deleted.

void **jobPaused** (**JobKey** jobKey)
   Called by the Scheduler when a JobDetail has been paused.

void **jobResumed** (**JobKey** jobKey)
   Called by the Scheduler when a JobDetail has been un-paused.

void **jobScheduled** (**Trigger** trigger)
   Called by the Scheduler when a JobDetail is scheduled.

void **jobsPaused** (**String** jobGroup)
   Called by the Scheduler when a group of JobDetails has been paused.

void **jobsResumed** (**String** jobGroup)
   Called by the Scheduler when a group of JobDetails has been un-paused.

void **jobUnscheduled** (**TriggerKey** triggerKey)
   Called by the Scheduler when a JobDetail is unscheduled.

void **schedulerError** (**String** msg, **SchedulerException** cause)
   Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

void **schedulerInStandbyMode**()
   Called by the Scheduler to inform the listener that it has move to standby mode.

void **schedulerShutdown**()
   Called by the Scheduler to inform the listener that it has shutdown.

void **schedulerShuttingdown**()
   Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

void **schedulerStarted**()
Called by the Scheduler to inform the listener that it has started.

```java
void schedulingDataCleared()
    Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.
```

```java
void triggerFinalized(Trigger trigger)
    Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.
```

```java
void triggerPaused(TriggerKey triggerKey)
    Called by the Scheduler when a Trigger has been paused.
```

```java
void triggerResumed(TriggerKey triggerKey)
    Called by the Scheduler when a Trigger has been un-paused.
```

```java
void triggersPaused(String triggerGroup)
    Called by the Scheduler when a group of Triggers has been paused.
```

```java
void triggersResumed(String triggerGroup)
    Called by the Scheduler when a group of Triggers has been un-paused.
```

Methods inherited from class java.lang.**Object**
clonetoString, finalizetoString, notify, notifyAll, wait, wait,wa

**Constructor Detail**

**SchedulerListenerSupport**

public **SchedulerListenerSupport()**

**Method Detail**

**getLog**
protected org.slf4j.Logger getLog()

Get the Logger for this class's category. This should be used by subclasses for logging.

---

**jobAdded**

public void jobAdded(JobDetail jobDetail)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been added.

Specified by:

  jobAdded in interface SchedulerListener

---

**jobDeleted**

public void jobDeleted(JobKey jobKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been deleted.

Specified by:

  jobDeleted in interface SchedulerListener

---

**jobPaused**

public void jobPaused(JobKey jobKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been paused.

Specified by:

  jobPaused in interface SchedulerListener
jobResumed

public void jobResumed(JobKey jobKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail has been un-paused.

Specified by:
  jobResumed in interface SchedulerListener

jobScheduled

public void jobScheduled(Trigger trigger)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail is scheduled.

Specified by:
  jobScheduled in interface SchedulerListener

jobsPaused

public void jobsPaused(String jobGroup)

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of JobDetails has been paused.

Specified by:
  jobsPaused in interface SchedulerListener

Parameters:
  jobGroup - the paused group, or null if all were paused
jobsResumed

public void jobsResumed(String jobGroup)

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of JobDetails has been un-paused.

Specified by:
      jobsResumed in interface SchedulerListener

jobUnscheduled

public void jobUnscheduled(TriggerKey triggerKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a JobDetail is unscheduled.

Specified by:
      jobUnscheduled in interface SchedulerListener

See Also:
      SchedulerListener.schedulingDataCleared()

schedulerError

public void schedulerError(String msg,
                                   SchedulerException cause)

Description copied from interface: SchedulerListener

Called by the Scheduler when a serious error has occurred within the scheduler - such as repeated failures in the JobStore, or the inability to instantiate a Job instance when its Trigger has fired.

The getErrorCode() method of the given SchedulerException can be used to determine more specific information about the type of error that was encountered.
Specified by:  
schedulerError in interface SchedulerListener

schedulerInStandbyMode

public void schedulerInStandbyMode() 

Description copied from interface: SchedulerListener

Called by the Scheduler to inform the listener that it has move to standby mode.

Specified by:  
schedulerInStandbyMode in interface SchedulerListener

schedulerShutdown

public void schedulerShutdown() 

Description copied from interface: SchedulerListener

Called by the Scheduler to inform the listener that it has shutdown.

Specified by:  
schedulerShutdown in interface SchedulerListener

schedulerShuttingdown

public void schedulerShuttingdown() 

Description copied from interface: SchedulerListener

Called by the Scheduler to inform the listener that it has begun the shutdown sequence.

Specified by:  
schedulerShuttingdown in interface SchedulerListener
schedulerStarted

public void schedulerStarted()

Description copied from interface: SchedulerListener

Called by the Scheduler to inform the listener that it has started.

Specified by:
    schedulerStarted in interface SchedulerListener

triggerFinalized

public void triggerFinalized(Trigger trigger)

Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has reached the condition in which it will never fire again.

Specified by:
    triggerFinalized in interface SchedulerListener

triggerPaused

public void triggerPaused(TriggerKey triggerKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has been paused.

Specified by:
    triggerPaused in interface SchedulerListener

triggerResumed
public void triggerResumed(TriggerKey triggerKey)

Description copied from interface: SchedulerListener

Called by the Scheduler when a Trigger has been un-paused.

Specified by:

   triggerResumed in interface SchedulerListener

------------------------

triggersPaused

public void triggersPaused(String triggerGroup)

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of Triggers has been paused.

If all groups were paused then triggerGroup will be null

Specified by:

   triggersPaused in interface SchedulerListener

Parameters:

   triggerGroup - the paused group, or null if all were paused

------------------------

triggersResumed

public void triggersResumed(String triggerGroup)

Description copied from interface: SchedulerListener

Called by the Scheduler when a group of Triggers has been un-paused.

Specified by:

   triggersResumed in interface SchedulerListener

------------------------

schedulingDataCleared
public void schedulingDataCleared()

Description copied from interface: SchedulerListener
Called by the Scheduler to inform the listener that all jobs, triggers and calendars were deleted.

Specified by:
    schedulingDataCleared in interface SchedulerListener
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAME</td>
<td>NO FRAME</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
public abstract class TriggerListenerSupport

extends Object
implements TriggerListener

A helpful abstract base class for implementors of TriggerListener.

The methods in this class are empty so you only need to override the subset for the TriggerListener events you care about.

You are required to implement TriggerListener.getName() to return the unique name of your TriggerListener.

See Also:
    TriggerListener

Constructor Summary

| TriggerListenerSupport() |

Method Summary

| protected org.slf4j.Logger getLog() |
| Get the Logger for this class's category. |

| void triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode) |
| Called by the Scheduler when a Trigger has fired, it's as if JobDetail has been executed, and it's triggered(xx) method has been called. |
void triggerFired(Trigger trigger, JobExecutionContext context)

Called by the Scheduler when a Trigger has fired, and it associated JobDetail is about to be executed.

void triggerMisfired(Trigger trigger)

Called by the Scheduler when a Trigger has misfired.

boolean vetoJobExecution(Trigger trigger, JobExecutionContext context)

Called by the Scheduler when a Trigger has fired, and it associated JobDetail is about to be executed.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface org.quartz.TriggerListener

generate

Constructor Detail

TriggerListenerSupport

public TriggerListenerSupport()

Method Detail

getLog

protected org.slf4j.Logger getLog()

Get the Logger for this class's category. This should be used by subclasses for logging.

triggerFired
public void triggerFired(Trigger trigger, JobExecutionContext context)

Description copied from interface: TriggerListener

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.

It is called before the vetoJobExecution(..) method of this interface.

Specified by:
    triggerFired in interface TriggerListener

Parameters:
    trigger - The Trigger that has fired.
    context - The JobExecutionContext that will be passed to the Job's execute(xx) method.

---

vetoJobExecution

public boolean vetoJobExecution(Trigger trigger, JobExecutionContext context)

Description copied from interface: TriggerListener

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed. If the implementation vetos the execution (via returning true), the job's execute method will not be called.

It is called after the triggerFired(..) method of this interface.

Specified by:
    vetoJobExecution in interface TriggerListener

Parameters:
    trigger - The Trigger that has fired.
    context - The JobExecutionContext that will be passed to the Job's execute(xx) method.

---

triggerMisfired
public void triggerMisfired(Trigger trigger)

Description copied from interface: TriggerListener

Called by the Scheduler when a Trigger has misfired.

Consideration should be given to how much time is spent in this method, as it will affect all triggers that are misfiring. If you have lots of triggers misfiring at once, it could be an issue if this method does a lot.

Specified by:
    triggerMisfired in interface TriggerListener

Parameters:
    trigger - The Trigger that has misfired.

triggerComplete

public void triggerComplete(Trigger trigger,
                           JobExecutionContext context,
                           Trigger.CompletedExecutionInstruction triggerInstructionCode)

Description copied from interface: TriggerListener

Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered(xx) method has been called.

Specified by:
    triggerComplete in interface TriggerListener

Parameters:
    trigger - The Trigger that was fired.
    context - The JobExecutionContext that was passed to the Job's execute(xx) method.
    triggerInstructionCode - the result of the call on the Trigger's triggered(xx) method.
Uses of Class
org.quartz.listeners.BroadcastJobListener

No usage of org.quartz.listeners.BroadcastJobListener

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.listeners.BroadcastSchedulerListener

No usage of org.quartz.listeners.BroadcastSchedulerListener

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.listeners.BroadcastTriggerListener

No usage of org.quartz.listeners.BroadcastTriggerListener

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.listeners.JobChainingJobListener

No usage of org.quartz.listeners.JobChainingJobListener

Copyright 2001-2011, Terracotta, Inc.
Packages that use JobListenerSupport

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.listeners</td>
</tr>
</tbody>
</table>

Uses of JobListenerSupport in org.quartz.listeners

Subclasses of JobListenerSupport in org.quartz.listeners

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobChainingJobListener</td>
</tr>
<tr>
<td>Keeps a collection of mappings of which Job to trigger after the completion of a given job.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Packages that use `SchedulerListenerSupport`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.core</code></td>
<td>Contains the core classes and interfaces for the <code>Quartz</code> job scheduler.</td>
</tr>
<tr>
<td><code>org.quartz.ee.jta</code></td>
<td>Contains implementations of the SchedulerFactory, JobStore, ThreadPool, and other interfaces required by the <code>org.quartz.core.QuartzScheduler</code>.</td>
</tr>
<tr>
<td><code>org.quartz.impl</code></td>
<td></td>
</tr>
</tbody>
</table>

Uses of `SchedulerListenerSupport` in `org.quartz.core`

<table>
<thead>
<tr>
<th>Subclass</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JobRunShell</code></td>
<td>JobRunShell instances are responsible for providing the 'safe' environment for Jobs to run in, and for performing all of the work of executing the Job, catching ANY thrown exceptions, updating the Trigger with the Job's completion code, etc.</td>
</tr>
<tr>
<td><code>SampledStatisticsImpl</code></td>
<td></td>
</tr>
</tbody>
</table>

Uses of `SchedulerListenerSupport` in `org.quartz.ee.jta`

<table>
<thead>
<tr>
<th>Subclass</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JTAJobRunShell</code></td>
<td>An extension of JobRunShell11 that begins an XA transaction</td>
</tr>
</tbody>
</table>
Before executing the Job, and commits (or rolls-back) the transaction after execution completes.

**Uses of **SchedulerListenerSupport **in** org.quartz.impl

<table>
<thead>
<tr>
<th>Subclasses of SchedulerListenerSupport in org.quartz.impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
</tr>
<tr>
<td>QuartzServer</td>
</tr>
</tbody>
</table>

Instantiates an instance of Quartz Scheduler as a stand-alone program, if the scheduler is configured for RMI it will be made available.

Copyright 2001-2011, Terracotta, Inc.
No usage of org.quartz.listeners.TriggerListenerSupport
public class DelegatingLocalityJobDetail

extends Object
implements Cloneable, Serializable, LocalityJobDetail

Wrapping a JobDetail instance while adding the LocalityAware contract All
JobDetail method calls will be delegated to the wrapped JobDetail instance

Author:
    Alex Snaps

See Also:
    Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelegatingLocalityJobDetail(JobDetail jobDetail, NodeSpec nodeSpec)</td>
<td>Constructs a LocalityAware JobDetail, wrapping an existing JobDetail instance, with additional NodeSpec</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone()</td>
<td></td>
</tr>
<tr>
<td>getDescription()</td>
<td>Return the description given to the Job instance by its creator (if any).</td>
</tr>
<tr>
<td>getJobBuilder()</td>
<td></td>
</tr>
</tbody>
</table>
Get a [JobBuilder](#) that is configured to produce a JobDetail identical to this one.

<table>
<thead>
<tr>
<th>Class&lt;Job extends Job&gt;</th>
<th>JobClass()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the instance of Job that will be executed.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JobDataMap</th>
<th>JobDataMap()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the JobDataMap that is associated with the Job.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JobKey</th>
<th>getKey()</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NodeSpec</th>
<th>NodeSpec()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessor to the NodeSpec instance to evaluate constraints</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isConcurrentExeuctionDisallow()</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isDurable()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isPersistJobDataAfterExecution()</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>requestsRecovery()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.</td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Constructor Detail**

**DelegatingLocalityJobDetail**

```java
public DelegatingLocalityJobDetail(JobDetail jobDetail, NodeSpec nodeSpec)
```

Constructs a [LocalityAware](#) JobDetail, wrapping an existing JobDetail instance, with additional NodeSpec
Parameters:

- `jobDetail` - The `JobDetail` instance to wrap
- `nodeSpec` - The `NodeSpec` instance

Method Detail

**getKey**

```java
public JobKey getKey()
```

**Specified by:**

`getKey` in interface `JobDetail`

**getDescription**

```java
public String getDescription()
```

Return the description given to the Job instance by its creator (if any).

**Specified by:**

`getDescription` in interface `JobDetail`

**Returns:**

null if no description was set.

**getJobClass**

```java
public Class<? extends Job> getJobClass()
```

Get the instance of Job that will be executed.

**Specified by:**

`getJobClass` in interface `JobDetail`

**getJobDataMap**

```java
public JobDataMap getJobDataMap()
```
Get the JobDataMap that is associated with the Job.

**Specified by:**
getJobDataMap in interface JobDetail

---

**isDurable**

public boolean **isDurable()**

Whether or not the Job should remain stored after it is orphaned (no Triggers point to it).

If not explicitly set, the default value is false.

**Specified by:**
isDurable in interface JobDetail

**Returns:**
true if the Job should remain persisted after being orphaned.

---

**isPersistJobDataAfterExecution**

public boolean **isPersistJobDataAfterExecution()**

**Specified by:**
isPersistJobDataAfterExecution in interface JobDetail

**Returns:**
whether the associated Job class carries the PersistJobDataAfterExecution annotation.

**See Also:**
PersistJobDataAfterExecution

---

**isConcurrentExectionDisallowed**

public boolean **isConcurrentExectionDisallowed()**

**Specified by:**
isConcurrentExectionDisallowed in interface JobDetail
Returns: whether the associated Job class carries the `DisallowConcurrentExecution` annotation.

See Also: `DisallowConcurrentExecution`

---

`requestsRecovery` public boolean `requestsRecovery()`

Instructs the Scheduler whether or not the Job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.

If not explicitly set, the default value is `false`.

Specified by: `requestsRecovery` in interface `JobDetail`

See Also: `JobExecutionContext.isRecovering()`

---

`getJobBuilder` public `JobBuilder` `getJobBuilder()`

Get a `JobBuilder` that is configured to produce a JobDetail identical to this one.

Specified by: `getJobBuilder` in interface `JobDetail`

---

`getNodeSpec` public `NodeSpec` `getNodeSpec()`

Accessor to the `NodeSpec` instance to evaluate constraints

Specified by:
getNodeSpec in interface LocalityAware

Returns:

clone

public Object clone()

Specified by:
clone in interface JobDetail

Overrides:
clone in class Object
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
</tr>
</tbody>
</table>
org.quartz.locality Class DelegatingLocalityTrigger

java.lang.Object
  └ org.quartz.locality.DelegatingLocalityTrigger

All Implemented Interfaces:
  Serializable, Cloneable, Comparable<Trigger>, LocalityAware,
  LocalityTrigger, org.quartz.spi.MutableTrigger,
  org.quartz.spi.OperableTrigger, Trigger

public class DelegatingLocalityTrigger
  extends Object
  implements LocalityTrigger, org.quartz.spi.OperableTrigger, Serializable

Wrapping a Trigger instance while adding the LocalityAware contract All
Trigger method calls will be delegated to the wrapped Trigger instance

Author:
  Alex Snaps

See Also:
  Serialized Form

Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from interface org.quartz.Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger.CompletedExecutionInstruction, Trigger.TriggerState, Trigger.TriggerTimeComparator</td>
</tr>
</tbody>
</table>

Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from interface org.quartz.Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_PRIORITY, MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY, MISFIRE_INSTRUCTION_SMART_POLICY, serialVersionUID</td>
</tr>
</tbody>
</table>
## Constructor Summary

**DelegatingLocalityTrigger**(*Trigger* trigger, *NodeSpec* nodeSpec)  
Constructs a *LocalityAware* Trigger, wrapping an existing Trigger instance, with additional NodeSpec

## Method Summary

<table>
<thead>
<tr>
<th>Class/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td><strong>clone()</strong></td>
</tr>
<tr>
<td><strong>int</strong></td>
<td><strong>compareTo(Trigger other)</strong></td>
</tr>
<tr>
<td></td>
<td>Compare the next fire time of this Trigger that of another by comparing their keys, or in other words, sorts them according to the natural (i.e. Date)</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td><strong>computeFirstFireTime(Calendar calendar)</strong></td>
</tr>
<tr>
<td><strong>Trigger.CompletedExecutionInstruction</strong></td>
<td><strong>executionComplete(JobExecutionContext context, JobExecutionException result)</strong></td>
</tr>
<tr>
<td><strong>String</strong></td>
<td><strong>getCalendarName()</strong></td>
</tr>
<tr>
<td></td>
<td>Get the name of the Calendar associated with this Trigger.</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td><strong>getDescription()</strong></td>
</tr>
<tr>
<td></td>
<td>Return the description given to the Trigger instance by its creator (if any).</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td><strong>getEndTime()</strong></td>
</tr>
<tr>
<td></td>
<td>Get the time at which the Trigger should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td><strong>getFinalFireTime()</strong></td>
</tr>
<tr>
<td></td>
<td>Returns the last time at which the Trigger fire, if the Trigger will repeat indefinitely, null will be returned.</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td><strong>getFireInstanceId()</strong></td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td></td>
</tr>
</tbody>
</table>
**getFireTimeAfter** *(Date afterTime)*

Returns the next time at which the Trigger will fire, after the given time.

**JobDataMap**

**getJobDataMap()**

Get the JobDataMap that is associated with the Trigger.

**JobKey**

**getJobKey()**

**TriggerKey**

**getKey()**

**int getMisfireInstruction()**

Get the instruction the Scheduler should be given for handling misfire situations for this Trigger. The concrete Trigger type that you are using may have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be set as this property's value.

**Date getNextFireTime()**

Returns the next time at which the Trigger is scheduled to fire.

**NodeSpec**

**getNodeSpec()**

Accessor to the NodeSpec instance to evaluate constraints.

**Date getPreviousFireTime()**

Returns the previous time at which the Trigger fired.

**int getPriority()**

The priority of a Trigger acts as a tiebreaker such that if two Triggers have the same scheduled time, then the one with the higher priority will get access to a worker thread.

**ScheduleBuilder<Trigger>**

**getScheduleBuilder()**

Get a ScheduleBuilder that is configured to produce a schedule identical to this trigger's schedule.

**Date getStartTime()**

Get the time at which the Trigger should occur.

**TriggerBuilder<Trigger>**

**getTriggerBuilder()**
Get a `TriggerBuilder` that is configured to produce a Trigger identical to this one.

```java
boolean mayFireAgain()

Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

```java
void setCalendarName(String calendarName)

```java
void setDescription(String description)

```java
void setEndTime(Date endTime)

```java
void setFireInstanceId(String id)

```java
void setJobDataMap(JobDataMap jobDataMap)

```java
void setJobKey(JobKey key)

```java
void setKey(TriggerKey key)

```java
void setMisfireInstruction(int misfireInstruction)

```java
void setNextFireTime(Date nextFireTime)

```java
void setPreviousFireTime(Date previousFireTime)

```java
void setPriority(int priority)

```java
void setStartTime(Date startTime)

```java
void triggered(Calendar calendar)

```java
void updateAfterMisfire(Calendar cal)

```java
void updateWithNewCalendar(Calendar cal, long misfireThreshold)
```
void validate()

Methods inherited from class java.lang.Object
equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface org.quartz.Trigger
equals

Constructor Detail

DelegatingLocalityTrigger

public DelegatingLocalityTrigger(Trigger trigger, NodeSpec nodeSpec)

Constructs a LocalityAware Trigger, wrapping an existing Trigger instance, with additional NodeSpec

Parameters:
  trigger - The Trigger instance to wrap
  nodeSpec - The NodeSpec instance

Method Detail

getNodeSpec

public NodeSpec getNodeSpec()

Accessor to the NodeSpec instance to evaluate constraints

Specified by:
  getNodeSpec in interface LocalityAware

Returns:
**getKey**

```java
public TriggerKey getKey()
```

**Specified by:**
```java
generic Key in interface Trigger
```

**getJobKey**

```java
public JobKey getJobKey()
```

**Specified by:**
```java
generic JobKey in interface Trigger
```

**getDescription**

```java
public String getDescription()
```

Return the description given to the Trigger instance by its creator (if any).

**Specified by:**
```java
generic Description in interface Trigger
```

**Returns:**
```java
null if no description was set.
```

**getCalendarName**

```java
public String getCalendarName()
```

Get the name of the Calendar associated with this Trigger.

**Specified by:**
```java
generic CalendarName in interface Trigger
```

**Returns:**
```java
null if there is no associated Calendar.
```
**getJobDataMap**

```java
public JobDataMap getJobDataMap()
```

Get the JobDataMap that is associated with the Trigger.

Changes made to this map during job execution are not re-persisted, and in fact typically result in an `IllegalStateException`.

**Specified by:**

`getJobDataMap` in interface `Trigger`

---

**getPriority**

```java
public int getPriority()
```

The priority of a `Trigger` acts as a tiebreaker such that if two Triggers have the same scheduled fire time, then the one with the higher priority will get first access to a worker thread.

If not explicitly set, the default value is 5.

**Specified by:**

`getPriority` in interface `Trigger`

**See Also:**

`Trigger.DEFAULT_PRIORITY`

---

**mayFireAgain**

```java
public boolean mayFireAgain()
```

Used by the Scheduler to determine whether or not it is possible for this Trigger to fire again.

If the returned value is false then the Scheduler may remove the Trigger from the JobStore.
Specified by:

`mayFireAgain` in interface `Trigger`

---

**getStartime**

```java
public Date getStartTime()
```

Get the time at which the `Trigger` should occur.

Specified by:

`getStartTime` in interface `Trigger`

---

**getEndTime**

```java
public Date getEndTime()
```

Get the time at which the `Trigger` should quit repeating - regardless of any remaining repeats (based on the trigger's particular repeat settings).

Specified by:

`getEndTime` in interface `Trigger`

See Also:

`Trigger.getFinalFireTime()`

---

**getNextFireTime**

```java
public Date getNextFireTime()
```

Returns the next time at which the `Trigger` is scheduled to fire. If the trigger will not fire again, `null` will be returned. Note that the time returned can possibly be in the past, if the time that was computed for the trigger to next fire has already arrived, but the scheduler has not yet been able to fire the trigger (which would likely be due to lack of resources e.g. threads).

The value returned is not guaranteed to be valid until after the `Trigger` has been added to the scheduler.
Specified by:
getPreviousFireTime in interface Trigger

See Also:
TriggerUtils#computeFireTimesBetween(Trigger, Calendar, Date, Date)

getPreviousFireTime

public Date getPreviousFireTime()

Returns the previous time at which the Trigger fired. If the trigger has not yet fired, null will be returned.

Specified by:
getPreviousFireTime in interface Trigger

getFireTimeAfter

public Date getFireTimeAfter(Date afterTime)

Returns the next time at which the Trigger will fire, after the given time. If the trigger will not fire after the given time, null will be returned.

Specified by:
getFireTimeAfter in interface Trigger

getFinalFireTime

public Date getFinalFireTime()

Returns the last time at which the Trigger will fire, if the Trigger will repeat indefinitely, null will be returned.

Note that the return time *may* be in the past.

Specified by:
getFinalFireTime in interface Trigger
getMisfireInstruction

public int getMisfireInstruction()

Get the instruction the Scheduler should be given for handling misfire situations for this Trigger - the concrete Trigger type that you are using will have defined a set of additional MISFIRE_INSTRUCTION_XXX constants that may be set as this property's value.

If not explicitly set, the default value is MISFIRE_INSTRUCTION_SMART_POLICY.

Specified by:
getMisfireInstruction in interface Trigger
See Also:
Trigger.MISFIRE_INSTRUCTION_SMART_POLICY, 
#updateAfterMisfire(Calendar), SimpleTrigger, CronTrigger

compareTo

public int compareTo(Trigger other)

Compare the next fire time of this Trigger to that of another by comparing their keys, or in other words, sorts them according to the natural (i.e. alphabetical) order of their keys.

Specified by:
compareTo in interface Comparable<Trigger>
Specified by:
compareTo in interface Trigger

getTriggerBuilder

public TriggerBuilder<Trigger> getTriggerBuilder()

Get a TriggerBuilder that is configured to produce a Trigger identical to
this one.

**Specified by:**
`getTriggerBuilder` in interface `Trigger`

**See Also:**
`Trigger.getScheduleBuilder()`

---

**getScheduleBuilder**

```java
public ScheduleBuilder<Trigger> getScheduleBuilder()
```

Get a `ScheduleBuilder` that is configured to produce a schedule identical to this trigger's schedule.

**Specified by:**
`getScheduleBuilder` in interface `Trigger`

**See Also:**
`Trigger.getTriggerBuilder()`

---

**setKey**

```java
public void setKey(TriggerKey key)
```

**Specified by:**
`setKey` in interface `org.quartz.spi.MutableTrigger`

---

**setJobKey**

```java
public void setJobKey(JobKey key)
```

**Specified by:**
`setJobKey` in interface `org.quartz.spi.MutableTrigger`

---

**setDescription**

```java
public void setDescription(String description)
```
Specified by:
    setDescription in interface org.quartz.spi.MutableTrigger

setCalendarName

public void setCalendarName(String calendarName)

Specified by:
    setCalendarName in interface org.quartz.spi.MutableTrigger

setJobDataMap

public void setJobDataMap(JobDataMap jobDataMap)

Specified by:
    setJobDataMap in interface org.quartz.spi.MutableTrigger

setPriority

public void setPriority(int priority)

Specified by:
    setPriority in interface org.quartz.spi.MutableTrigger

setStartTime

public void setStartTime(Date startTime)

Specified by:
    setStartTime in interface org.quartz.spi.MutableTrigger

setEndTime

public void setEndTime(Date endTime)
setEndTime

Specified by:
setEndTime in interface org.quartz.spi.MutableTrigger

---

setMisfireInstruction

public void setMisfireInstruction(int misfireInstruction)

Specified by:
setMisfireInstruction in interface org.quartz.spi.MutableTrigger

---

triggered

public void triggered(Calendar calendar)

Specified by:
triggered in interface org.quartz.spi.OperableTrigger

---

computeFirstFireTime

public Date computeFirstFireTime(Calendar calendar)

Specified by:
computeFirstFireTime in interface org.quartz.spi.OperableTrigger

---

executionComplete

public Trigger.CompletedExecutionInstruction executionComplete(JobEx)

Specified by:
executionComplete in interface org.quartz.spi.OperableTrigger
**updateAfterMisfire**

```java
public void updateAfterMisfire(Calendar cal)
```

**Specified by:**
updateAfterMisfire in interface org.quartz.spi.OperableTrigger

---

**updateWithNewCalendar**

```java
public void updateWithNewCalendar(Calendar cal,
                                  long misfireThreshold)
```

**Specified by:**
updateWithNewCalendar in interface org.quartz.spi.OperableTrigger

---

**validate**

```java
public void validate()
    throws SchedulerException
```

**Specified by:**
validate in interface org.quartz.spi.OperableTrigger

**Throws:**
SchedulerException

---

**setFireInstanceId**

```java
public void setFireInstanceId(String id)
```

**Specified by:**
setFireInstanceId in interface org.quartz.spi.OperableTrigger

---

**getFireInstanceId**

```java
public String getFireInstanceId()
```
Specified by:
getFireInstanceId in interface org.quartz.spi.OperableTrigger

```java
public void setNextFireTime(Date nextFireTime)
```

Specified by:
setNextFireTime in interface org.quartz.spi.OperableTrigger

```java
public void setPreviousFireTime(Date previousFireTime)
```

Specified by:
setPreviousFireTime in interface org.quartz.spi.OperableTrigger

```java
public Object clone()
```

Specified by:
clone in interface org.quartz.spi.MutableTrigger

Overrides:
clone in class Object
org.quartz.locality  Interface LocalityAware

All Known Subinterfaces:
   LocalityJobDetail, LocalityTrigger

All Known Implementing Classes:
   DelegatingLocalityJobDetail, DelegatingLocalityTrigger

public interface LocalityAware

LocalityAware types can provide node specifications to potentially dispatch
the instance to a particular node in the cluster.

Author:
   Alex Snaps

Method Summary

<table>
<thead>
<tr>
<th>NodeSpec</th>
<th>getNodeSpec()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accessor to the NodeSpec instance to evaluate constraints</td>
</tr>
</tbody>
</table>

Method Detail

getNodeSpec

NodeSpec getNodeSpec() |
| Accessor to the NodeSpec instance to evaluate constraints |

Returns:
Class LocalityEngine

java.lang.Object
  ▼ java.lang.Throwable
    ▼ java.lang.Exception
      ▼ org.quartz.SchedulerException
        ▼ org.quartz.JobPersistenceException
          ▼ org.quartz.locality.LocalityEngine

All Implemented Interfaces:
  Serializable

public class LocalityEngine
  extends JobPersistenceException

Exception thrown when some Constraint cannot be handled

Author:
  Alex Snaps

See Also:
  Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalityEngine(String message, Constraint constraint)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint</td>
<td>getConstraint()</td>
</tr>
<tr>
<td></td>
<td>Constraint being the issue</td>
</tr>
</tbody>
</table>

Methods inherited from class org.quartz.SchedulerException

getUnderlyingException, toString
Methods inherited from class java.lang.**Throwable**

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace

Methods inherited from class java.lang.**Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

### Constructor Detail

**LocalityException**

public **LocalityException**(String message,
Constraint constraint)

### Method Detail

**getConstraint**

public **Constraint** getConstraint()

Constraint being the issue

**Returns:**
The constraint that triggered the Exception

---

Copyright 2001-2011, Terracotta, Inc.
public class **LocalityJobBuilder**

extends **Object**

Builder for **LocalityAware JobDetail** instances.

```java
localJob(
    newJob(CheckForNode1AndWaitJob.class)
    .withIdentity("testJob")
    .storeDurably(true)
    .build()
    .where(
        node()
        .is(partOfNodeGroup("group1")))
    .build();
```

**Author:**
Alex Snaps

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocalityJobBuilder</strong></td>
<td>(JobBuilder jobBuilder)</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>build()</strong></td>
<td></td>
<td>Method building an immutable LocalityAware JobDetail, wrapping the actual JobDetail instance If a JobBuilder was used at construction time, its build method is invoked</td>
</tr>
<tr>
<td>static <strong>locallyJob</strong></td>
<td>(JobBuilder jobBuilder)</td>
<td></td>
</tr>
</tbody>
</table>
LocalJob

public static LocalityJobBuilder localJob(JobDetail jobDetail)

Factory method to create builder to a wrapping LocalityAware JobDetail

Parameters:
  jobDetail - The JobDetail to wrap

Returns:
  the LocalityJobBuilder instance
public static LocalityJobBuilder localJob(JobBuilder jobBuilder)

Factory method to create builder to a wrapping LocalityAware JobDetailBuilder

Parameters:
jobBuilder - The JobBuilder to wrap

Returns:
the LocalityJobBuilder instance

where

public LocalityJobBuilder where(NodeSpecBuilder spec)

Setter to the NodeSpecBuilder that will be build at JobDetail build() time

Parameters:
spec - The builder to the node specifications

Returns:
this

build

public LocalityJobDetail build()

Method building an immutable LocalityAware JobDetail, wrapping the actual JobDetail instance If a JobBuilder was used at construction time, its build method is invoked

Returns:
the LocalityJobDetail instance
public interface LocalityJobDetail
extends JobDetail, LocalityAware

A specialized JobDetail that contains Quartz Where information

Author:
Alex Snaps

See Also:
JobDetail, LocalityAware

Method Summary

<table>
<thead>
<tr>
<th>Methods inherited from interface org.quartz.JobDetail</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, getDescription, getJobBuilder, getJobClass,</td>
</tr>
<tr>
<td>getJobDataMap, getKey, isConcurrentExectionDisallowed,</td>
</tr>
<tr>
<td>isDurable, isPersistJobDataAfterExecution, requestsRecovery</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from interface org.quartz.locality.LocalityAware</th>
</tr>
</thead>
<tbody>
<tr>
<td>getNodeSpec</td>
</tr>
</tbody>
</table>

Overview Package | Use Tree Deprecated Index Help | PREV CLASS NEXT CLASS | FRAMES NO FRAMES | SUMMARY: NESTED | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD |
Interface LocalityTrigger

All Superinterfaces:
   Cloneable, Comparable<Trigger>, LocalityAware, Serializable, Trigger

All Known Implementing Classes:
   DelegatingLocalityTrigger

public interface LocalityTrigger
   extends Trigger, LocalityAware

A specialized Trigger that contains Quartz Where information

Author:
   Alex Snaps

See Also:
   Trigger, LocalityAware

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from interface org.quartz.Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger.CompletedExecutionInstruction, Trigger.TriggerState, Trigger.TriggerTimeComparator</td>
</tr>
</tbody>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>Fields inherited from interface org.quartz.Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_PRIORITY, MISFIRE_INSTRUCTION_IGNORE_MISFIRE_POLICY, MISFIRE_INSTRUCTION_SMART_POLICY, serialVersionUID</td>
</tr>
</tbody>
</table>

### Method Summary
### Methods inherited from interface org.quartz.Trigger
- compareTo
- equals
- getCalendarName
- getDescription
- getEndTime
- getFinalFireTime
- getFireTimeAfter
- getJobDataMap
- getJobKey
- getKey
- getMisfireInstruction
- getNextFireTime
- getPreviousFireTime
- getPriority
- getScheduleBuilder
- getStartTime
- getTriggerBuilder
- mayFireAgain

### Methods inherited from interface org.quartz.locality.LocalityAware
- getNodeSpec

Copyright 2001-2011, Terracotta, Inc.
public class **LocalityTriggerBuilder** extends **Object**

Builder for **LocalityAware** Triggers, which decorate the Trigger and adds the Locality feature to it, by adding a reference to a **NodeSpec**.

Usage:

```java
localTrigger(
  newTrigger()
    .forJob("testJob")
    .withIdentity("trigger1", "group")
    .startAt(new Date(when)))
   .where(
    node()
      .is(partOfNodeGroup("group1")))
  .build();
```

**Author:**
Alex Snaps

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocalityTriggerBuilder</strong>*(Trigger trigger)*</td>
</tr>
<tr>
<td><strong>LocalityTriggerBuilder</strong>*(TriggerBuilder triggerBuilder)*</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocalityTrigger.build</strong>()</td>
</tr>
</tbody>
</table>
Method building an immutable LocalityAware Trigger, wrapping the actual Trigger instance. If a TriggerBuilder was used at construction time, its build method is invoked.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>localTrigger(Trigger trigger)</code></td>
<td>Creates a delegating LocalityTriggerBuilder based on the Trigger.</td>
</tr>
<tr>
<td><code>localTrigger(TriggerBuilder triggerBuilder)</code></td>
<td>Creates a delegating LocalityTriggerBuilder based on the TriggerBuilder.</td>
</tr>
<tr>
<td><code>where(NodeSpecBuilder specBuilder)</code></td>
<td>Setter to the NodeSpecBuilder that will be build at JobDetail build() time.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

**Constructor Detail**

**LocalityTriggerBuilder**

```java
public LocalityTriggerBuilder(TriggerBuilder triggerBuilder)
```

**LocalityTriggerBuilder**

```java
public LocalityTriggerBuilder(Trigger trigger)
```

**Method Detail**

**localTrigger**

```java
public static LocalityTriggerBuilder localTrigger(TriggerBuilder triggerBuilder)
```
Creates a delegating LocalityTriggerBuilder based on the TriggerBuilder

**Parameters:**
triggerBuilder - Wrapped builder

**Returns:**
A `LocalityTriggerBuilder` instance

---

**localTrigger**

```java
public static LocalityTriggerBuilder localTrigger(Trigger trigger)
```

Creates a delegating LocalityTriggerBuilder based on the Trigger

**Parameters:**
trigger - Wrapped trigger

**Returns:**
A `LocalityTriggerBuilder` instance

---

**where**

```java
public LocalityTriggerBuilder where(NodeSpecBuilder specBuilder)
```

Setter to the `NodeSpecBuilder` that will be build at JobDetail `build()` time

**Parameters:**
specBuilder - The builder to the node specifications

**Returns:**
this

---

**build**

```java
public LocalityTrigger build()
```

Method building an immutable LocalityAware Trigger, wrapping the actual Trigger instance If a TriggerBuilder was used at construction time, its build method is invoked
Returns:

the `LocalityJobDetail` instance
public interface NodeSpec

Encapsulates all metadata about Locality.

Author: Alex Snaps

---

**Method Summary**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;Constraint&gt;</td>
<td>getConstraints()</td>
</tr>
</tbody>
</table>

**getConstraints**

Returns a List of Constraint to be used to identify the targeted node

**Returns:**

List of Constraint

---

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
</tr>
</tbody>
</table>
public class **NodeSpecBuilder**

extends **Object**

Builder for **NodeSpec**, specifically required for the DSL builder-style API:

```java
localJob(
    newJob(CheckForNode1AndWaitJob.class)
        .withIdentity("testJob")
        .storeDurably(true)
        .build())
    .where(
        node()
            .is(partOfNodeGroup("group1")))
    .build();
```

* @author Alex Snaps

### Constructor Summary

| **NodeSpecBuilder**() |

### Method Summary

<table>
<thead>
<tr>
<th><strong>NodeSpec</strong></th>
<th><strong>build</strong>()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Method that creates the NodeSpec represented by this builder</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>NodeSpecBuilder</strong></th>
<th><strong>has</strong>(<em>Constraint</em> constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DSL method to add a constraint</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>NodeSpecBuilder</strong></th>
<th><strong>is</strong>(<em>Constraint</em> constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DSL method to add a constraint</td>
</tr>
</tbody>
</table>
### Constructor Detail

**NodeSpecBuilder**

```java
public NodeSpecBuilder()
```

### Method Detail

#### set

```java
public NodeSpecBuilder set(Constraint... constraintsToAdd)
```

#### build

```java
public NodeSpec build()
```

Method that creates the NodeSpec represented by this builder

**Returns:**

The `NodeSpec` instance

#### node

```java
public static NodeSpecBuilder node()
```
Factory method

**Returns:**
A `NodeSpecBuilder` instance

```java
public NodeSpecBuilder is(Constraint constraint)
```

DSL method to add a constraint

**Parameters:**
- constraint - The constraint to be added to the list

**Returns:**
- this

```java
public NodeSpecBuilder has(Constraint constraint)
```

DSL method to add a constraint

**Parameters:**
- constraint - The constraint to be added to the list

**Returns:**
- this

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.locality

**Interfaces**  
LocalityAware
LocalityJobDetail
LocalityTrigger
NodeSpec

**Classes**  
DelegatingLocalityJobDetail
DelegatingLocalityTrigger
LocalityJobBuilder
LocalityTriggerBuilder
NodeSpecBuilder

**Exceptions**  
LocalityException
## Package org.quartz.locality

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocalityAware</strong></td>
<td>LocalityAware types can provide node specifications to potentially dispatch the instance to a particular node in the cluster.</td>
</tr>
<tr>
<td><strong>LocalityJobDetail</strong></td>
<td>A specialized JobDetail that contains Quartz Where information</td>
</tr>
<tr>
<td><strong>LocalityTrigger</strong></td>
<td>A specialized Trigger that contains Quartz Where information</td>
</tr>
<tr>
<td><strong>NodeSpec</strong></td>
<td>Encapsulates all metadata about Locality.</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DelegatingLocalityJobDetail</strong></td>
<td>Wrapping a JobDetail instance while adding the LocalityAware contract All JobDetail method calls will be delegated to the wrapped JobDetail instance</td>
</tr>
<tr>
<td><strong>DelegatingLocalityTrigger</strong></td>
<td>Wrapping a Trigger instance while adding the LocalityAware contract Trigger method calls will be delegated to the wrapped Trigger instance</td>
</tr>
<tr>
<td><strong>LocalityJobBuilder</strong></td>
<td>Builder for LocalityAware JobDetail instances.</td>
</tr>
<tr>
<td><strong>LocalityTriggerBuilder</strong></td>
<td>Builder for LocalityAware Triggers, which decorate the Trigger and adds the Locality feature to it, by adding a reference to a NodeSpec.</td>
</tr>
<tr>
<td><strong>NodeSpecBuilder</strong></td>
<td>Builder for NodeSpec, specifically required for the DSL builder-style API:</td>
</tr>
</tbody>
</table>

### Exception Summary
Exception thrown when some `Constraint` cannot be handled.

Copyright 2001-2011, Terracotta, Inc.
Class Hierarchy

- `java.lang.Object`
  - `org.quartz.locality.LocalityJobBuilder`
  - `org.quartz.locality.LocalityTriggerBuilder`
  - `org.quartz.locality.NodeSpecBuilder`
  - `java.lang.Throwable` (implements `java.io.Serializable`)
    - `java.lang.Exception`
    - `org.quartz.SchedulerException`
    - `org.quartz.JobPersistenceException`
    - `org.quartz.locality.LocalityException`
Interface Hierarchy

- java.lang.**Cloneable**
  - org.quartz.**JobDetail** (also extends java.io.**Serializable**)
    - org.quartz.locality.**LocalityJobDetail** (also extends org.quartz.locality.**LocalityAware**)
  - org.quartz.**Trigger** (also extends java.lang.**Comparable<T>**, java.io.**Serializable**)
    - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.locality.**LocalityAware**)
- java.lang.**Comparable<T>**
  - org.quartz.**Trigger** (also extends java.lang.**Cloneable**, java.io.**Serializable**)
    - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.locality.**LocalityAware**)
- org.quartz.locality.**LocalityAware**
  - org.quartz.locality.**LocalityJobDetail** (also extends org.quartz.**JobDetail**)
  - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.**Trigger**)
- org.quartz.locality.**NodeSpec**
- java.io.**Serializable**
  - org.quartz.**JobDetail** (also extends java.lang.**Cloneable**)
    - org.quartz.locality.**LocalityJobDetail** (also extends org.quartz.locality.**LocalityAware**)
  - org.quartz.**Trigger** (also extends java.lang.**Cloneable<T>**, java.lang.**Comparable<T>**)
    - org.quartz.locality.**LocalityTrigger** (also extends org.quartz.locality.**LocalityAware**)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.locality

### Packages that use org.quartz.locality

<table>
<thead>
<tr>
<th>Package</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
<td></td>
</tr>
</tbody>
</table>

### Classes in org.quartz.locality used by org.quartz.locality

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalityAware</td>
<td>LocalityAware types can provide node specifications to potentially dispatch the instance to a particular node in the cluster.</td>
</tr>
<tr>
<td>LocalityJobBuilder</td>
<td>Builder for LocalityAware JobDetail instances.</td>
</tr>
<tr>
<td>LocalityJobDetail</td>
<td>A specialized JobDetail that contains Quartz Where information</td>
</tr>
<tr>
<td>LocalityTrigger</td>
<td>A specialized Trigger that contains Quartz Where information</td>
</tr>
<tr>
<td>LocalityTriggerBuilder</td>
<td>Builder for LocalityAware Triggers, which decorate the Trigger and adds the Locality feature to it, by adding a reference to a NodeSpec.</td>
</tr>
<tr>
<td>NodeSpec</td>
<td>Encapsulates all metadata about Locality.</td>
</tr>
<tr>
<td>NodeSpecBuilder</td>
<td>Builder for NodeSpec, specifically required for the DSL builder-style API:</td>
</tr>
</tbody>
</table>

### Classes in org.quartz.locality used by org.quartz.locality.constraint.evaluator
LocalityException
Exception thrown when some Constraint cannot be handled
No usage of org.quartz.locality.DelegatingLocalityJobDetail
Uses of Class
org.quartz.locality.DelegatingLocalityTrigger

No usage of org.quartz.locality.DelegatingLocalityTrigger
# Uses of Interface `org.quartz.locality.LocalityAware`

## Packages that use `LocalityAware`

- `org.quartz.locality`

## Uses of `LocalityAware` in `org.quartz.locality`

## Subinterfaces of `LocalityAware` in `org.quartz.locality`

<table>
<thead>
<tr>
<th>interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LocalityJobDetail</code></td>
<td>A specialized JobDetail that contains Quartz Where information</td>
</tr>
<tr>
<td><code>LocalityTrigger</code></td>
<td>A specialized Trigger that contains Quartz Where information</td>
</tr>
</tbody>
</table>

## Classes in `org.quartz.locality` that implement `LocalityAware`

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DelegatingLocalityJobDetail</code></td>
<td>Wrapping a JobDetail instance while adding the <code>LocalityAware</code> contract All <code>JobDetail</code> method calls will be delegated to the wrapped JobDetail instance</td>
</tr>
<tr>
<td><code>DelegatingLocalityTrigger</code></td>
<td>Wrapping a Trigger instance while adding the <code>LocalityAware</code> contract All <code>Trigger</code> method calls will be delegated to the wrapped Trigger instance</td>
</tr>
</tbody>
</table>

---

**Overview** | **Package** | **Class** | **Tree** | **Deprecated** | **Index** | **Help**
---|---|---|---|---|---|---
PREV | NEXT | FRAMES | NO FRAMES
Uses of Class
org.quartz.locality.LocalityException

Packages that use LocalityException
org.quartz.locality.constraint.evaluator

Uses of LocalityException in
org.quartz.locality.constraint.evaluator

Methods in org.quartz.locality.constraint.evaluator that throw LocalityException

<table>
<thead>
<tr>
<th>void NodeGroupEvaluator.verify(NodeGroupConstraint constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verifies that the node group actually exists in the current configuration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void Evaluator.verify(T constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verifies the validity of a constraint.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.locality.LocalityJobBuilder

Packages that use **LocalityJobBuilder**

<table>
<thead>
<tr>
<th>Package</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality</td>
<td></td>
</tr>
</tbody>
</table>

Uses of **LocalityJobBuilder** in **org.quartz.locality**

Methods in **org.quartz.locality** that return **LocalityJobBuilder**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static LocalityJobBuilder.<strong>localJob</strong>(JobBuilder jobBuilder)</td>
<td>Factory method to create builder to a wrapping LocalityAware JobDetailBuilder</td>
</tr>
<tr>
<td>static LocalityJobBuilder.<strong>localJob</strong>(JobDetail jobDetail)</td>
<td>Factory method to create builder to a wrapping LocalityAware JobDetail</td>
</tr>
<tr>
<td>LocalityJobBuilder.<strong>where</strong>(NodeSpecBuilder spec)</td>
<td>Setter to the NodeSpecBuilder that will be build at JobDetail build() time</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.locality.LocalityJobDetail

### Packages that use **LocalityJobDetail**

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality</td>
</tr>
</tbody>
</table>

### Uses of **LocalityJobDetail** in **org.quartz.locality**

### Classes in **org.quartz.locality** that implement **LocalityJobDetail**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelegatingLocalityJobDetail</td>
<td>Wrapping a JobDetail instance while adding the LocalitàAware contract. All JobDetail method calls will be delegated to the wrapped JobDetail instance.</td>
</tr>
</tbody>
</table>

### Methods in **org.quartz.locality** that return **LocalityJobDetail**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalityJobBuilder.build()</td>
<td>Method building an immutable LocalitàAware JobDetail, wrapping the actual JobDetail instance. If a JobBuilder was used at construction time, its build method is invoked.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
## Uses of Interface org.quartz.locality.LocalitàTrigger

### Packages that use LocalitàTrigger

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality</td>
</tr>
</tbody>
</table>

### Uses of LocalitàTrigger in org.quartz.locality

### Classes in org.quartz.locality that implement LocalitàTrigger

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelegatingLocalitàTrigger</td>
</tr>
</tbody>
</table>

Wrapping a Trigger instance while adding the LocalitàAware contract. All Trigger method calls will be delegated to the wrapped Trigger instance.

### Methods in org.quartz.locality that return LocalitàTrigger

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalitàTriggerBuilder.build()</td>
</tr>
</tbody>
</table>

Method building an immutable LocalitàAware Trigger, wrapping the actual Trigger instance. If a TriggerBuilder was used at construction time, its build method is invoked.
Uses of Class
org.quartz.locality.LocalityTriggerBuilder

<table>
<thead>
<tr>
<th>Packages that use <strong>LocalityTriggerBuilder</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality</td>
</tr>
</tbody>
</table>

| Uses of **LocalityTriggerBuilder** in **org.quartz.locality** |

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.locality</strong> that return <strong>LocalityTriggerBuilder</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>LocalityTriggerBuilder</strong></td>
</tr>
<tr>
<td><strong>LocalityTriggerBuilder</strong>.<strong>localTrigger</strong>(<em>Trigger</em> <em>trigger</em>)</td>
</tr>
<tr>
<td>Creates a delegating LocalityTriggerBuilder based on <em>trigger</em></td>
</tr>
<tr>
<td>static <strong>LocalityTriggerBuilder</strong></td>
</tr>
<tr>
<td><strong>LocalityTriggerBuilder</strong>.<strong>localTrigger</strong>(<em>TriggerBuilder</em> <em>triggerBuilder</em>)</td>
</tr>
<tr>
<td>Creates a delegating LocalityTriggerBuilder based on <em>triggerBuilder</em></td>
</tr>
<tr>
<td><strong>LocalityTriggerBuilder</strong>.<strong>where</strong>(<em>NodeSpecBuilder</em> <em>specBuilder</em>)</td>
</tr>
<tr>
<td>Setter to the <em>specBuilder</em> that will be built at JobDetail time</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

## `org.quartz.locality.NodeSpec`

### Packages that use **NodeSpec**

- `org.quartz.locality`

### Uses of **NodeSpec** in **org.quartz.locality**

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.locality</strong> that return <strong>NodeSpec</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NodeSpec</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>NodeSpec</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>NodeSpec</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>NodeSpec</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

### Constructors in **org.quartz.locality** with parameters of type **NodeSpec**

- **DelegatingLocalityJobDetail**( **JobDetail** jobDetail, **NodeSpec** nodeSpec)
  - Constructs a **LocalityAware** JobDetail, wrapping an existing JobDetail instance, with additional NodeSpec

- **DelegatingLocalityTrigger**( **Trigger** trigger, **NodeSpec** nodeSpec)
  - Constructs a **LocalityAware** Trigger, wrapping an existing Trigger instance, with additional NodeSpec
Uses of Class
org.quartz.locality.NodeSpecBuilder

Packages that use NodeSpecBuilder

org.quartz.locality

Uses of NodeSpecBuilder in org.quartz.locality

<table>
<thead>
<tr>
<th>Methods in org.quartz.locality that return NodeSpecBuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeSpecBuilder</td>
</tr>
</tbody>
</table>
| NodeSpecBuilder.
`has`(Constraint constraint)       |
| DSL method to add a constraint  |
| NodeSpecBuilder  |
| NodeSpecBuilder.
`is`(Constraint constraint)       |
| DSL method to add a constraint  |
| static NodeSpecBuilder  |
| NodeSpecBuilder.
`node`()                         |
| Factory method                |
| NodeSpecBuilder  |
| NodeSpecBuilder.
`set`(Constraint... constraintsToAdd) |

<table>
<thead>
<tr>
<th>Methods in org.quartz.locality with parameters of type NodeSpecBuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalityJobBuilder</td>
</tr>
</tbody>
</table>
| LocalityJobBuilder.
`where`(NodeSpecBuilder spec)  |
| Setter to the NodeSpecBuilder that will be build at JobLocalityJobBuilder.
`build`() time |
| LocalityTriggerBuilder  |
| LocalityTriggerBuilder.
`where`(NodeSpecBuilder spec)  |
| Setter to the NodeSpecBuilder that will be build at JobLocalityTriggerBuilder.
`build`() time |
Class CpuConstraint

All Implemented Interfaces:
  Serializable, Constraint<
  CpuConstraint.Operator, Integer>

public class CpuConstraint

extends Object

implements Constraint<
  CpuConstraint.Operator, Integer>

Cpu based constraint for best node evaluation

Author:
  Alex Snaps
See Also:
  CpuEvaluator, Serialized Form

Nested Class Summary

static class CpuConstraint.Operator

Constructor Summary

CpuConstraint(CpuConstraint.Operator op, int value)

Method Summary

static CpuConstraint coresAtLeast(int amount)
  Creates a constraint that requires the node to have a
  least an amount of core

 boolean equals(Object obj)
<table>
<thead>
<tr>
<th>CpuConstraint.Operator getOp()</th>
<th>int getValue()</th>
<th>int hashCode()</th>
</tr>
</thead>
<tbody>
<tr>
<td>static CpuConstraint loadAtMost(double amount)</td>
<td>Creates a constraint that requires the node to have at most a certain load</td>
<td></td>
</tr>
<tr>
<td>boolean matches(Integer actual)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>static CpuConstraint threadsAvailableAtLeast(int amount)</td>
<td>Creates a constraint that requires the node to have a least an amount of threads available</td>
<td></td>
</tr>
<tr>
<td>String toString()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

clone, finalize, getClass, notify, notifyAll, wait, wait, wait

**Constructor Detail**

**CpuConstraint**

public CpuConstraint(CpuConstraint.Operator op, int value)

**Parameters:**

- op -
- value -

**Method Detail**
getOp

public CpuConstraint.Operator getOp()

getvalue

public int getValue()

matches

public boolean matches(Integer actual)

threadsAvailableAtLeast

public static CpuConstraint threadsAvailableAtLeast(int amount)

Creates a constraint that requires the node to have a least an amount of threads available

Parameters:
  amount - Number of threads to be available

Returns:
  The CpuConstraint

coresAtLeast

public static CpuConstraint coresAtLeast(int amount)

Creates a constraint that requires the node to have a least an amount of core

Parameters:
  amount - Number of cores

Returns:
  The CpuConstraint
loadAtMost

public static CpuConstraint loadAtMost(double amount)

Creates a constraint that requires the node to have at most a certain load

Parameters:
    amount - The maximum load the node can currently have

Returns:
    The CpuConstraint

hashCode

public int hashCode()

Overrides:
    hashCode in class Object

equals

public boolean equals(Object obj)

Overrides:
    equals in class Object

toString

public String toString()

Overrides:
    toString in class Object
Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>ENUM CONSTANTS</td>
<td>FIELD</td>
</tr>
<tr>
<td>DETAIL: ENUM CONSTANTS</td>
<td>FIELD</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Enum Constant Summary

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>amountOfCores</code></td>
<td>Amount of cores on the node</td>
</tr>
<tr>
<td><code>availableThreads</code></td>
<td>Available threads in the scheduler's thread pool</td>
</tr>
<tr>
<td><code>avgLoad</code></td>
<td>Average load on the node</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>matches(Integer actual, Integer value)</code></td>
<td>Returns <code>true</code> if the specified <code>actual</code> matches the specified <code>value</code>.</td>
</tr>
<tr>
<td><code>valueOf(String name)</code></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td><code>values()</code></td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.**Enum**

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.**Object**

getClass, notify, notifyAll, wait, wait, wait

**Enum Constant Detail**

**availableThreads**

```java
public static final CpuConstraint.Operator availableThreads
```

Available threads in the scheduler's thread pool

**amountOfCores**

```java
public static final CpuConstraint.Operator amountOfCores
```

Amount of cores on the node

**avgLoad**

```java
public static final CpuConstraint.Operator avgLoad
```

Average load on the node

**Method Detail**

**values**

```java
public static CpuConstraint.Operator[] values()
```
Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (CpuConstraint.Operator c : CpuConstraint.Operator.values())
    System.out.println(c);
```

**Returns:**
an array containing the constants of this enum type, in the order they are declared

---

**valueOf**

```java
public static CpuConstraint.Operator valueOf(String name)
```

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

**Parameters:**
- `name` - the name of the enum constant to be returned.

**Returns:**
- the enum constant with the specified name

**Throws:**
- `IllegalArgumentException` - if this enum type has no constant with the specified name
- `NullPointerException` - if the argument is null

---

**matches**

```java
public boolean matches(Integer actual, Integer value)
```

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.locality.constraint  **Class EhcacheConstraint**

**java.lang.Object**

- **org.quartz.locality.constraint.EhcacheConstraint**

**All Implemented Interfaces:**

- **Serializable**
- **Constraint<**EhcacheConstraint.Operator, EhcacheConstraint.Value**>**

```
public class EhcacheConstraint
    extends Object
    implements Constraint<EhcacheConstraint.Operator, EhcacheConstraint.Value>
```

Constraints related the locality of Ehcache values

**Author:**

Alex Snaps

**See Also:**

EhcacheEvaluator, Serialized Form

### Nested Class Summary

<table>
<thead>
<tr>
<th>static class</th>
<th>EhcacheConstraint.Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>static class</td>
<td>EhcacheConstraint.Value</td>
</tr>
</tbody>
</table>

### Constructor Summary

**EhcacheConstraint**(EhcacheConstraint.Operator operator, EhcacheConstraint.Value value)

Constructor

The Cache needs to be alive and clustered using Terracotta in Serialization mode
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static EhcacheConstraint elements(Cache cache, Collection keys)</code></td>
<td>Creates a EhcacheConstraint for local values to a Set of keys</td>
</tr>
<tr>
<td><code>String getCacheManagerName()</code></td>
<td>Gets the CacheManager name for the Cache</td>
</tr>
<tr>
<td><code>String getCacheName()</code></td>
<td>Gets the cache name</td>
</tr>
<tr>
<td><code>Set getKeys()</code></td>
<td>Accessor</td>
</tr>
</tbody>
</table>

## Constructor Detail

**EhcacheConstraint**

```java
public EhcacheConstraint(EhcacheConstraint.Operator operator, EhcacheConstraint.Value value)
```

Constructor

The Cache needs to be alive and clustered using Terracotta in Serialization mode

## Method Detail

**elements**

```java
public static EhcacheConstraint elements(Cache cache, Collection keys)
```
Creates an `EhcacheConstraint` for local values to a Set of keys

**Parameters:**
- `cache` - The cache
- `keys` - The keys we want local

**Returns:**
The actual `EhcacheConstraint`

---

**getKeys**

```java
public Set getKeys()
```

Accessor

**Returns:**
the Set of keys we want local

---

**getCacheManagerName**

```java
public String getCacheManagerName()
```

Gets the CacheManager name for the Cache

**Returns:**
CacheManager name

---

**getCacheName**

```java
public String getCacheName()
```

Gets the cache name

**Returns:**
Cache name
**Enum EhcacheConstraint.Operator**

**java.lang.Object**
  - **java.lang.Enum**
    - **org.quartz.locality.constraint.EhcacheConstraint.Operator**

**All Implemented Interfaces:**
  - Serializable, Comparable<EhcacheConstraint.Operator>

**Enclosing class:**
  - EhcacheConstraint

---

**public static enum EhcacheConstraint.Operator**

extends Enum<EhcacheConstraint.Operator>

---

### Enum Constant Summary

<table>
<thead>
<tr>
<th>MOST_LOCAL_VALUES</th>
</tr>
</thead>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>static EhcacheConstraint.Operator</th>
<th>valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static EhcacheConstraint.Operator[]</th>
<th>values()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf
**Methods inherited from class java.lang.**Object

`getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`  

**Enum Constant Detail**

**MOST_LOCAL_VALUES**

public static final `EhcacheConstraint.Operator` MOST_LOCAL_VALUES

**Method Detail**

values

public static `EhcacheConstraint.Operator`[] values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```java
for (EhcacheConstraint.Operator c : EhcacheConstraint.Operator.values())
    System.out.println(c);
```

Returns:
an array containing the constants of this enum type, in the order they are declared

valueOf

public static `EhcacheConstraint.Operator` valueOf(String name)

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
name - the name of the enum constant to be returned.
Returns:
    the enum constant with the specified name

Throws:
    IllegalArgumentException - if this enum type has no constant with
        the specified name
    NullPointerException - if the argument is null
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAME: NO FRAME</td>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
org.quartz.locality.constraint Class EhcacheConstraint.Value

java.lang.Object
   org.quartz.locality.constraint.EhcacheConstraint.Value

All Implemented Interfaces:
   Serializable

Enclosing class:
   EhcacheConstraint

public static class EhcacheConstraint.Value
    extends Object
    implements Serializable

See Also:
   Serialized Form

Constructor Summary

EhcacheConstraint.Value(String mgrName, String cache, Collection<? > keys)

Method Summary

boolean equals(Object o)

String getCacheManagerName()

String getCacheName()

Set getKeySet()

int hashCode()
static EhcacheConstraint.Value valueOf(String stringValue)

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, toString, wait, wait

Constructor Detail

EhcacheConstraint.Value

public EhcacheConstraint.Value(String mgrName, String cache, Collection<?> keys)

Method Detail

valueOf

public static EhcacheConstraint.Value valueOf(String stringValue) throws ClassNotFoundException, InvocationTargetException, IllegalAccessException, NoSuchMethodException

Throws:
   ClassNotFoundException
   InvocationTargetException
   IllegalAccessException
   NoSuchMethodException

equals

public boolean equals(Object o)
Overrides:

equals in class Object

hashCode

public int hashCode()

Overrides:

hashCode in class Object

goingCacheManagerName

public String getCacheManagerName()

goingCacheName

public String getCacheName()

goingKeySet

public Set getKeySet()
Class MemoryConstraint

extends Object

implements Constraint<MemoryConstraint.Operator, Long>

Constraint on memory characteristics of the node to execute the Job on

Author:
Alex Snaps

See Also:
MemoryEvaluator, Serialized Form

Nested Class Summary

| static class | MemoryConstraint.Operator |
| static class | MemoryConstraint.Unit |
|              | Memory units |

Constructor Summary

MemoryConstraint(MemoryConstraint.Operator op, long value)
MemoryConstraint constructor

Method Summary

static MemoryConstraint atLeastAvailable(int amount, MemoryConstraint.Unit unit)
Creates a `MemoryConstraint` for at least an amount of memory available

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean equals(Object obj)</code></td>
<td></td>
</tr>
<tr>
<td><code>MemoryConstraint.Operator getOp()</code></td>
<td>Getter to operator</td>
</tr>
<tr>
<td><code>int hashCode()</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean matches(Long actual)</code></td>
<td></td>
</tr>
<tr>
<td><code>static MemoryConstraint orderDescending()</code></td>
<td>Returns a <code>MemoryConstraint</code> to sort the nodes by memory available</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**
`clone, finalize, getClass, notify, notifyAll, toString, wait, wait`

**Constructor Detail**

**MemoryConstraint**

`public MemoryConstraint(MemoryConstraint.Operator op, long value)`

MemoryConstraint constructor

**Parameters:**
- op - Operation to validate against
- value - Target value

**Method Detail**

`matches`
public boolean matches(Long actual)

getOp

public MemoryConstraint.Operator getOp()

   Getter to operator

   Returns:
   Operator

hashCode

public int hashCode()

   Overrides:
   hashCode in class Object

equals

public boolean equals(Object obj)

   Overrides:
   equals in class Object

atLeastAvailable

public static MemoryConstraint atLeastAvailable(int amount,
                                              MemoryConstraint.Unit

   Creates a MemoryConstraint for at least an amount of memory available

   Parameters:
   amount - minimum amount
   unit - Memory unit

   Returns:
The `MemoryConstraint` instance

```java
public static MemoryConstraint orderDescending()
```

Returns a `MemoryConstraint` to sort the nodes by memory available

**Returns:**
The `MemoryConstraint` instance
<table>
<thead>
<tr>
<th>Table Headers</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>Package</td>
</tr>
<tr>
<td>Use</td>
<td>Tree</td>
</tr>
<tr>
<td>Deprecated</td>
<td>Index</td>
</tr>
<tr>
<td>Help</td>
<td></td>
</tr>
</tbody>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | ENUM CONSTANTS | FIELD | METHOD
FRAMES        NO FRAMES
DETAIL: ENUM CONSTANTS | FIELD | METHOD
**Enum Constant Summary**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>atLeastAvailable</code></td>
<td>Operator on least available memory</td>
</tr>
<tr>
<td><code>orderDescending</code></td>
<td>Ordering operator, putting nodes with most memory available on top of matching nodes</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>valueOf(String name)</code></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td><code>values()</code></td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Enum**

- `clone`, `compareTo`, `equals`, `finalize`, `getDeclaringClass`, `hashCode`, `name`, `ordinal`, `toString`, `valueOf`
Methods inherited from class java.lang.**Object**

- `getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

---

**Enum Constant Detail**

**atLeastAvailable**

**Public static final MemoryConstraint.Operator atLeastAvailable**

Operator on least available memory

**orderDescending**

**Public static final MemoryConstraint.Operator orderDescending**

Ordering operator, putting nodes with most memory available on top of matching nodes

---

**Method Detail**

**values**

**Public static MemoryConstraint.Operator[] values()**

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```java
for (MemoryConstraint.Operator c : MemoryConstraint.Operator.values())
    System.out.println(c);
```

**Returns:**

an array containing the constants of this enum type, in the order they are declared
valueOf

public static MemoryConstraint.Operator valueOf(String name)

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
name - the name of the enum constant to be returned.

Returns:
the enum constant with the specified name

Throws:
IllegalArgumentException - if this enum type has no constant with the specified name
NullPointerException - if the argument is null
org.quartz.locality.constraint  **Enum MemoryConstraint.Unit**

**java.lang.Object**
  └ **java.lang.Enum<MemoryConstraint.Unit>**
      └ org.quartz.locality.constraint.MemoryConstraint.Unit

All Implemented Interfaces:
  Serializable, Comparable<MemoryConstraint.Unit>

Enclosing class:
  MemoryConstraint

---

class MemoryConstraint.Unit

---

public static enum MemoryConstraint.Unit
extends Enum<MemoryConstraint.Unit>

Memory units

---

**Enum Constant Summary**

<table>
<thead>
<tr>
<th>GB</th>
<th>Gigabytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>KB</td>
<td>Kilobytes</td>
</tr>
<tr>
<td>MB</td>
<td>Megabytes</td>
</tr>
</tbody>
</table>

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long toBytes(long amount)</td>
<td></td>
</tr>
<tr>
<td>static MemoryConstraint.Unit[] values()</td>
<td></td>
</tr>
</tbody>
</table>

static MemoryConstraint.Unit.valueOf(String name)
  Returns the enum constant of this type with the specified name.

static MemoryConstraint.Unit[] values()
  Returns an array containing the constants of
this enum type, in the order they are declared.

<table>
<thead>
<tr>
<th>Enum Constant Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KB</strong></td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>Kilobytes</td>
</tr>
</tbody>
</table>

| **MB**                |
| public static final  | MemoryConstraint.Unit | MB |
| Megabytes             |

| **GB**                |
| public static final  | MemoryConstraint.Unit | GB |
| Gigabytes             |

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>values</strong></td>
</tr>
<tr>
<td>public static</td>
</tr>
</tbody>
</table>
Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```java
for (MemoryConstraint.Unit c : MemoryConstraint.Unit.values())
    System.out.println(c);
```

**Returns:**
an array containing the constants of this enum type, in the order they are declared

---

**valueOf**

```java
public static MemoryConstraint.Unit valueOf(String name)
```

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

**Parameters:**
- name - the name of the enum constant to be returned.

**Returns:**
the enum constant with the specified name

**Throws:**
- `IllegalArgumentException` - if this enum type has no constant with the specified name
- `NullPointerException` - if the argument is null

---

**getBytes**

```java
public final long toBytes(long amount)
```

---

Copyright 2001-2011, Terracotta, Inc.
**Class NodeGroupConstraint**

```java
public final class NodeGroupConstraint extends Object implements Constraint<NodeGroupConstraint.Operator, String>
```

Constraint to have a Job execute on a specific node group. Node groups are defined in quartzLocality.properties

**Author:**
Alex Snaps

**See Also:**
NodeGroupEvaluator, Serialized Form

### Nested Class Summary

- **static class** `NodeGroupConstraint.Operator`

### Constructor Summary

- `NodeGroupConstraint(NodeGroupConstraint.Operator operator, String targetNodeGroup)`
- `NodeGroupConstraint(String targetNodeGroup)` Constructor

### Method Summary

- `boolean`
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals(Object o)</td>
<td></td>
</tr>
<tr>
<td>String getTargetNodeGroup()</td>
<td>Getter to the target node group</td>
</tr>
<tr>
<td>int hashCode</td>
<td></td>
</tr>
<tr>
<td>boolean matches(String actual)</td>
<td></td>
</tr>
<tr>
<td>static NodeGroupConstraint partOfNodeGroup(String targetNodeGroup)</td>
<td>Creates a NodeGroupConstraint that specifies the node group on which the Job has to be executed</td>
</tr>
</tbody>
</table>

### Constructor Detail

**NodeGroupConstraint**

```java
public NodeGroupConstraint(String targetNodeGroup)
```

**Parameters:**

- `targetNodeGroup`: the targeted node group for this constraint

---

**NodeGroupConstraint**

```java
public NodeGroupConstraint(NodeGroupConstraint.Operator operator, String targetNodeGroup)
```

---

### Method Detail


partOfNodeGroup

public static NodeGroupConstraint partOfNodeGroup(String targetNodeGroup)

Creates a NodeGroupConstraint that specifies the node group on which the Job has to be executed

Parameters:
  targetNodeGroup - The name of the target group, as specified in quartzLocality.properties

Returns:
The NodeGroupConstraint instance

------------------------

getTargetNodeGroup

public String getTargetNodeGroup()

Getter to the target node group

Returns:

------------------------

hashCode

public int hashCode()

Overrides:
  hashCode in class Object

------------------------

equals

public boolean equals(Object o)

Overrides:
  equals in class Object
public boolean matches(String actual)
**org.quartz.locality.constraint**  
**Enum**  
**NodeGroupConstraint.Operator**

**java.lang.Object**  
  ▼ **java.lang.Enum<**NodeGroupConstraint.Operator**>**  
  ▼ **org.quartz.locality.constraint.NodeGroupConstraint.Operator**

All Implemented Interfaces:
  **Serializable**, **Comparable<**NodeGroupConstraint.Operator**>**

Enclosing class:
  **NodeGroupConstraint**

---

public static enum **NodeGroupConstraint.Operator**

extends **Enum<**NodeGroupConstraint.Operator**>**

---

### Enum Constant Summary

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQUALS</td>
<td></td>
</tr>
</tbody>
</table>

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract boolean <strong>evaluate</strong>(String value, String compareTo)</td>
<td></td>
</tr>
<tr>
<td>static <strong>NodeGroupConstraint.Operator</strong> <strong>valueOf</strong>(String name)</td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static <strong>NodeGroupConstraint.Operator</strong>[] <strong>values</strong>()</td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

EQUALS

public static final NodeGroupConstraint.Operator EQUALS

Method Detail

values

public static NodeGroupConstraint.Operator[] values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

for (NodeGroupConstraint.Operator c : NodeGroupConstraint.Operator.values())
    System.out.println(c);

Returns:
an array containing the constants of this enum type, in the order they are declared

valueOf

public static NodeGroupConstraint.Operator valueOf(String name)

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this
type. (Extraneous whitespace characters are not permitted.)

**Parameters:**
- name - the name of the enum constant to be returned.

**Returns:**
- the enum constant with the specified name

**Throws:**
- `IllegalArgumentException` - if this enum type has no constant with the specified name
- `NullPointerException` - if the argument is null

**evaluate**

```java
public abstract boolean evaluate(String value, String compareTo)
```

Copyright 2001-2011, Terracotta, Inc.
Class OsConstraint

java.lang.Object
  org.quartz.locality.constraint.OsConstraint

All Implemented Interfaces:
  Serializable, Constraint<OsConstraint.Operator, OsConstraint.OS>

public class OsConstraint
  extends Object
  implements Constraint<OsConstraint.Operator, OsConstraint.OS>

Constraint to a specific operating system

Author:
  Alex Snaps

See Also:
  Serialized Form

Nested Class Summary

<table>
<thead>
<tr>
<th>static class</th>
<th>OsConstraint.Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>static class</td>
<td>OsConstraint.OS</td>
</tr>
</tbody>
</table>

Field Summary

<table>
<thead>
<tr>
<th>static OsConstraint</th>
<th>LINUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>static OsConstraint</td>
<td>OSX</td>
</tr>
<tr>
<td>static OsConstraint</td>
<td>SOLARIS</td>
</tr>
<tr>
<td>static OsConstraint</td>
<td></td>
</tr>
</tbody>
</table>
### Constructor Summary

**OsConstraint**

(Operator operator, OS targetedOs)

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td><strong>equals</strong> (Object obj)</td>
</tr>
<tr>
<td>OsConstraint.OS</td>
<td><strong>getTargetedOs</strong> ()</td>
</tr>
<tr>
<td>Getter to targeted OS</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td><strong>hashCode</strong> ()</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>matches</strong> ()</td>
</tr>
<tr>
<td>Evaluate against current node</td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td><strong>matches</strong> (OsConstraint.OS actual)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

- clone, finalize, getClass, notify, notifyAll, toString, wait, wait

### Field Detail

#### LINUX

public static final OsConstraint LINUX

#### OSX
public static final OsConstraint OSX

WINDOWS

public static final OsConstraint WINDOWS

SOLARIS

public static final OsConstraint SOLARIS

<table>
<thead>
<tr>
<th>Constructor Detail</th>
</tr>
</thead>
</table>

OsConstraint

public OsConstraint(OsConstraint.Operator operator, OsConstraint.OS targetedOs)

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
</table>

getTargetedOs

public OsConstraint.OS getTargetedOs()

Getter to targeted OS

Returns:

targeted OS

matches

public boolean matches()

Evaluate against current node

Returns:
true is OS matches

equals

public boolean equals(Object obj)

Overrides:
equals in class Object

hashCode

public int hashCode()

Overrides:
hashCode in class Object

matches

public boolean matches(OsConstraint.OS actual)

Parameters:
TODO -

Returns:
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>ENUM CONSTANTS</td>
<td>FIELD</td>
</tr>
<tr>
<td>DETAIL: ENUM CONSTANTS</td>
<td>FIELD</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
All Implemented Interfaces:
    Serializable, Comparable<OsConstraint.Operator>

Enclosing class:
    OsConstraint
**Methods inherited from class java.lang.Object**

`getClass, notify, notifyAll, wait, wait, wait`

---

**Enum Constant Detail**

**IS**

`public static final OsConstraint.Operator IS`

---

**Method Detail**

**values**

`public static OsConstraint.Operator[] values()`

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```java
for (OsConstraint.Operator c : OsConstraint.Operator.values())
    System.out.println(c);
```

**Returns:**

an array containing the constants of this enum type, in the order they are declared

---

**valueOf**

`public static OsConstraint.Operator valueOf(String name)`

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

**Parameters:**
name - the name of the enum constant to be returned.

**Returns:**
the enum constant with the specified name

**Throws:**

- [IllegalArgumentException](#) - if this enum type has no constant with the specified name
- [NullPointerException](#) - if the argument is null

---

evaluate

evaluate

```java
class OsConstraint
{
  public abstract boolean evaluate(OsConstraint.OS os,
                                  OsConstraint.OS target);
}
```

---

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | ENUM CONSTANTS | FIELD | METHOD |

FRAMES  NO FRAMES
DETAIL: ENUM CONSTANTS | FIELD | METHOD
**Enum OsConstraint.OS**

java.lang.Object  
  | java.lang.Enum<OsConstraint.OS>  
  | org.quartz.locality.constraint.OsConstraint.OS

**All Implemented Interfaces:**
Serializable, Comparable<OsConstraint.OS>

**Enclosing class:**
OsConstraint

---

```java
public static enum OsConstraint.OS
extends Enum<OsConstraint.OS>
```

---

### Enum Constant Summary

<table>
<thead>
<tr>
<th>LINUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSX</td>
</tr>
<tr>
<td>SOLARIS</td>
</tr>
<tr>
<td>WINDOWS</td>
</tr>
</tbody>
</table>

---

### Method Summary

<table>
<thead>
<tr>
<th>static OsConstraint.OS current()</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean isCurrent()</td>
</tr>
<tr>
<td>static OsConstraint.OS valueOf(String name)</td>
</tr>
</tbody>
</table>

Returns the enum constant of this type with the
static OsConstraint.OS[] values()

Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

WINDOWS

public static final OsConstraint.OS WINDOWS

OSX

public static final OsConstraint.OS OSX

LINUX

public static final OsConstraint.OS LINUX

SOLARIS

public static final OsConstraint.OS SOLARIS

Method Detail
values

public static OsConstraint.OS[] values()

    Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

    for (OsConstraint.OS c : OsConstraint.OS.values())
        System.out.println(c);

    Returns:
    an array containing the constants of this enum type, in the order they are declared

nodeValue

public static OsConstraint.OS valueOf(String name)

    Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

    Parameters:
    name - the name of the enum constant to be returned.

    Returns:
    the enum constant with the specified name

    Throws:
    IllegalArgumentException - if this enum type has no constant with the specified name
    NullPointerException - if the argument is null

isCurrent

public boolean isCurrent()
public static OsConstraint.OS current()
org.quartz.locality.constraint

Interfaces  Constraint

Classes  
CpuConstraint  
EhcacheConstraint  
EhcacheConstraint.Value  
MemoryConstraint  
NodeGroupConstraint  
OsConstraint  

Enums  
CpuConstraint.Operator  
EhcacheConstraint.Operator  
MemoryConstraint.Operator  
MemoryConstraint.Unit  
NodeGroupConstraint.Operator  
OsConstraint.Operator  
OsConstraint.OS
### Interface Summary

| Constraint<OP extends Enum,T> | A constraint about the node on which a job will be executed. A constraint always needs a matching Evaluator type present on the classpath. |

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CpuConstraint</td>
<td>Cpu based constraint for best node evaluation</td>
</tr>
<tr>
<td>EhcacheConstraint</td>
<td>Constraints related the locality of Ehcache values</td>
</tr>
<tr>
<td>EhcacheConstraint.Value</td>
<td></td>
</tr>
<tr>
<td>MemoryConstraint</td>
<td>Constraint on memory characteristics of the node to execute the Job on</td>
</tr>
<tr>
<td>NodeGroupConstraint</td>
<td>Constraint to have a Job execute on a specific node group.</td>
</tr>
<tr>
<td>OsConstraint</td>
<td>Constraint to a specific operating system</td>
</tr>
</tbody>
</table>

### Enum Summary

<table>
<thead>
<tr>
<th>Enum</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CpuConstraint.Operator</td>
<td></td>
</tr>
<tr>
<td>EhcacheConstraint.Operator</td>
<td></td>
</tr>
<tr>
<td>MemoryConstraint.Operator</td>
<td></td>
</tr>
<tr>
<td>MemoryConstraint.Unit</td>
<td>Memory units</td>
</tr>
<tr>
<td>NodeGroupConstraint.Operator</td>
<td></td>
</tr>
<tr>
<td>OsConstraint.Operator</td>
<td></td>
</tr>
<tr>
<td>OsConstraint.OS</td>
<td></td>
</tr>
</tbody>
</table>
Hierarchy For Package org.quartz.locality.constraint

Package Hierarchies:
  All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.locality.constraint.**CpuConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
  - org.quartz.locality.constraint.**EhcacheConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
  - org.quartz.locality.constraint.**EhcacheConstraint.Value** (implements java.io.**Serializable**)
  - org.quartz.locality.constraint.**MemoryConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
  - org.quartz.locality.constraint.**NodeGroupConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
  - org.quartz.locality.constraint.**OsConstraint** (implements org.quartz.locality.constraint.**Constraint**<OP,T>)
Interface Hierarchy

○ java.io.Serializable
  ○ org.quartz.locality.constraint.Constraint<OP,T>
Enum Hierarchy

- java.lang.**Object**
  - java.lang.**Enum**<E> (implements java.lang.**Comparable**<T>, java.io.**Serializable**
    - org.quartz.locality.constraint.**OsConstraint.Operator**
    - org.quartz.locality.constraint.**MemoryConstraint.Unit**
    - org.quartz.locality.constraint.**NodeGroupConstraint.Operator**
    - org.quartz.locality.constraint.**EhcacheConstraint.Operator**
    - org.quartz.locality.constraint.**CpuConstraint.Operator**
    - org.quartz.locality.constraint.**MemoryConstraint.Operator**
    - org.quartz.locality.constraint.**OsConstraint.OS**

---

**Overview**  **Package**  **Class**  **Use**  **Deprecated**  **Index**  **Help**

Copyright 2001-2011, Terracotta, Inc.
# Uses of Package

**org.quartz.locality.constraint**

### Packages that use `org.quartz.locality.constraint`

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality</td>
</tr>
<tr>
<td>org.quartz.locality.constraint</td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
</tr>
<tr>
<td>org.terracotta.modules.ehcache.store</td>
</tr>
</tbody>
</table>

### Classes in `org.quartz.locality.constraint` used by `org.quartz.locality`

- **Constraint**
  - A Constraint about the node on which a **job** will be executed
  - A Constraint always needs a matching **Evaluator** type present on the classpath

### Classes in `org.quartz.locality.constraint` used by `org.quartz.locality.constraint`

- **CpuConstraint**
  - Cpu based constraint for best node evaluation

- **EhcacheConstraint**
  - Constraints related the locality of Ehcache values

- **EhcacheConstraint.Operator**

- **EhcacheConstraint.Value**
### MemoryConstraint
Constraint on memory characteristics of the node to execute the **Job** on

#### MemoryConstraint.Operator

#### MemoryConstraint.Unit
Memory units

### NodeGroupConstraint
Constraint to have a **Job** execute on a specific node group.

#### NodeGroupConstraint.Operator

### OsConstraint
Constraint to a specific operating system

#### OsConstraint.Operator

#### OsConstraint.OS

### Classes in `org.quartz.locality.constraint` used by `org.quartz.locality.constraint.evaluator`

- **Constraint**
  A constraint about the node on which a **job** will be executed. A constraint always needs a matching **Evaluator** type present on the classpath

- **CpuConstraint**
  Cpu based constraint for best node evaluation

- **CpuConstraint.Operator**

- **MemoryConstraint**
  Constraint on memory characteristics of the node to execute the **Job** on

- **NodeGroupConstraint**
  Constraint to have a **Job** execute on a specific node group.

- **OsConstraint**
  Constraint to a specific operating system
Classes in `org.quartz.locality.constraint` used by `org.terracotta.modules.ehcache.store`  

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EhcacheConstraint</td>
<td>Constraints related the locality of Ehcache values</td>
</tr>
</tbody>
</table>
### Uses of Interface

org.quartz.locality.constraint.Constraint

#### Packages that use Constraint

- org.quartz.locality
- org.quartz.locality.constraint
- org.quartz.locality.constraint.evaluator

#### Uses of Constraint in org.quartz.locality

#### Methods in org.quartz.locality that return Constraint

<table>
<thead>
<tr>
<th>Constraint</th>
<th>LocalityException.getConstraint()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constraint being the issue</td>
</tr>
</tbody>
</table>

#### Methods in org.quartz.locality that return types with arguments of type Constraint

<table>
<thead>
<tr>
<th>List&lt;Constraint&gt;</th>
<th>NodeSpec.getConstraints()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a List of Constraint to be used to identify the targeted node</td>
</tr>
</tbody>
</table>

#### Methods in org.quartz.locality with parameters of type Constraint

<table>
<thead>
<tr>
<th>NodeSpecBuilder</th>
<th>NodeSpecBuilder.has(Constraint constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DSL method to add a constraint</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NodeSpecBuilder</th>
<th>NodeSpecBuilder.is(Constraint constraint)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DSL method to add a constraint</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NodeSpecBuilder</th>
<th>NodeSpecBuilder.set(Constraint... constraintsToAdd)</th>
</tr>
</thead>
</table>
Constructors in `org.quartz.locality` with parameters of type `Constraint`:

```
LocalityException(String message, Constraint constraint)
```

---

Uses of `Constraint` in `org.quartz.locality.constraint`:

---

Classes in `org.quartz.locality.constraint` that implement `Constraint`:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CpuConstraint</td>
<td>Cpu based constraint for best node evaluation</td>
</tr>
<tr>
<td>EhcacheConstraint</td>
<td>Constraints related the locality of Ehcache values</td>
</tr>
<tr>
<td>MemoryConstraint</td>
<td>Constraint on memory characteristics of the node to execute the Job on</td>
</tr>
<tr>
<td>NodeGroupConstraint</td>
<td>Constraint to have a Job execute on a specific node group.</td>
</tr>
<tr>
<td>OsConstraint</td>
<td>Constraint to a specific operating system</td>
</tr>
</tbody>
</table>

---

Uses of `Constraint` in `org.quartz.locality.constraint.evaluator`:

---

Classes in `org.quartz.locality.constraint.evaluator` with type parameters of type `Constraint`:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluator&lt;T extends Constraint&gt;</td>
<td>An Evaluator will match and potentially return best suited nodes for a Job to execute on.</td>
</tr>
<tr>
<td>PersistentEvaluator&lt;T extends Constraint,V&gt;</td>
<td>An particular Evaluator that will be passed a clustered ConcurrentHashMap instance to store shared data.</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.locality.constraint.CpuConstraint

Packages that use CpuConstraint

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint</td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
</tr>
</tbody>
</table>

Uses of CpuConstraint in
org.quartz.locality.constraint

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static CpuConstraint.<strong>coresAtLeast</strong>(int amount)</td>
<td>Creates a constraint that requires the node to have a least an amount of core</td>
</tr>
<tr>
<td>static CpuConstraint.<strong>loadAtMost</strong>(double amount)</td>
<td>Creates a constraint that requires the node to have at most a certain load</td>
</tr>
<tr>
<td>static CpuConstraint.<strong>threadsAvailableAtLeast</strong>(int amount)</td>
<td>Creates a constraint that requires the node to have a least an amount of threads available</td>
</tr>
</tbody>
</table>

Uses of CpuConstraint in
org.quartz.locality.constraint.evaluator

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean CpuEvaluator.<strong>matches</strong>(CpuConstraint constraint)</td>
<td></td>
</tr>
</tbody>
</table>
List\(<\text{String}\>)\quad \text{CpuEvaluator.}\text{suitedNodes}(\text{CpuConstraint}\ \text{constraint})

void\quad \text{CpuEvaluator.}\text{verify}(\text{CpuConstraint}\ \text{constraint})

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

org.quartz.locality.constraint.CpuConstraint.Operator

## Packages that use `CpuConstraint.Operator`

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint</td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
</tr>
</tbody>
</table>

## Uses of `CpuConstraint.Operator` in `org.quartz.locality.constraint`

## Methods in `org.quartz.locality.constraint` that return `CpuConstraint.Operator`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CpuConstraint.Operator getOp()</code></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td><code>static CpuConstraint.Operator[] values()</code></td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

## Constructors in `org.quartz.locality.constraint` with parameters of type `CpuConstraint.Operator`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CpuConstraint(CpuConstraint.Operator op, int value)</code></td>
<td>op, int value</td>
</tr>
</tbody>
</table>

## Uses of `CpuConstraint.Operator` in `org.quartz.locality.constraint.evaluator`
void CpuEvaluator.initialize(String nodeName,
ConcurrentMap<String, ConcurrentMap<CpuConstraint.Operator, Integer>>>
# Uses of Class

**org.quartz.locality.constraint.EhcacheConstraint**

## Packages that use **EhcacheConstraint**

<table>
<thead>
<tr>
<th>Package</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint</td>
<td></td>
</tr>
<tr>
<td>org.terracotta.modules.ehcache.store</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of **EhcacheConstraint** in

**org.quartz.locality.constraint**

## Methods in **org.quartz.locality.constraint** that return **EhcacheConstraint**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>EhcacheConstraint</strong> elements(EhcacheConstraint constraint)</td>
<td>Cache cache, Collection keys</td>
<td>Creates a EhcacheConstraint for local values to a Set of keys</td>
</tr>
</tbody>
</table>

## Uses of **EhcacheConstraint** in

**org.terracotta.modules.ehcache.store**

## Methods in **org.terracotta.modules.ehcache.store** with parameters of type **EhcacheConstraint**

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>EhcacheEvaluator.matches</td>
<td>EhcacheEvaluator.matches(EhcacheConstraint constraint)</td>
<td>Evaluates the constraint against local node.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>EhcacheEvaluator.suitedNodes</td>
<td>EhcacheEvaluator.suitedNodes(EhcacheConstraint constraint)</td>
<td>Returns a list of best suited node (instanceId) based on the constraint</td>
</tr>
<tr>
<td>void</td>
<td>EhcacheEvaluator.verify</td>
<td>EhcacheEvaluator.verify(EhcacheConstraint constraint)</td>
<td>Doesn't do anything</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.locality.constraint.EhcacheConstraint.Operator

Packages that use EhcacheConstraint.Operator
org.quartz.locality.constraint

Uses of EhcacheConstraint.Operator in
org.quartz.locality.constraint

Methods in org.quartz.locality.constraint that return EhcacheConstraint.Operator

| static EhcacheConstraint.Operator EhcacheConstraint.Operator.valueOf(String name) |
| static EhcacheConstraint.Operator[] EhcacheConstraint.Operator.values() |

Constructors in org.quartz.locality.constraint with parameters of type EhcacheConstraint.Operator

EhcacheConstraint(EhcacheConstraint.Operator operator,
EhcacheConstraint.Value value)

Constructor

The Cache needs to be alive and clustered using Terracotta in Serialization mode.
Uses of Class
org.quartz.locality.constraint.EhcacheConstraint.Value

Packages that use EhcacheConstraint.Value
org.quartz.locality.constraint

Uses of EhcacheConstraint.Value in
org.quartz.locality.constraint

Methods in org.quartz.locality.constraint that return EhcacheConstraint.Value

| static EhcacheConstraint.Value EhcacheConstraint.Value.valueOf(String stringValue) |

Constructors in org.quartz.locality.constraint with parameters of type EhcacheConstraint.Value

EhcacheConstraint(EhcacheConstraint.Operator operator, EhcacheConstraint.Value value)

Constructor

The Cache needs to be alive and clustered using Terracotta in Serialization mode

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.locality.constraint.MemoryConstraint**

## Packages that use `MemoryConstraint`

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of `MemoryConstraint` in

**org.quartz.locality.constraint**

### Methods in `org.quartz.locality.constraint` that return `MemoryConstraint`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static MemoryConstraint <code>atLeastAvailable</code>(int amount, MemoryConstraint.Unit unit)</td>
<td>Creates a <code>MemoryConstraint</code> for at least an amount of memory available</td>
</tr>
<tr>
<td>static MemoryConstraint <code>orderDescending</code>()</td>
<td>Returns a <code>MemoryConstraint</code> to sort the nodes by memory available</td>
</tr>
</tbody>
</table>

## Uses of `MemoryConstraint` in

**org.quartz.locality.constraint.evaluator**

### Methods in `org.quartz.locality.constraint.evaluator` with parameters of type `MemoryConstraint`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean <code>matches</code>(MemoryConstraint constraint)</td>
<td>Evaluates the constraint against local node.</td>
</tr>
<tr>
<td>boolean <code>matches</code>(MemoryConstraint constraint, String node)</td>
<td></td>
</tr>
</tbody>
</table>
Verifies whether the constraint passed is validating on the node

| List&lt;String&gt; MemoryEvaluator.**suitedNodes**(MemoryConstraint constraint) |
| Returns a list of best suited node (instanceId) based on the constraint |

| void MemoryEvaluator.**verify**(MemoryConstraint constraint) |
| Always true |
Uses of Class
org.quartz.locality.constraint.MemoryConstraint.Operator

Packages that use MemoryConstraint.Operator
org.quartz.locality.constraint

Uses of MemoryConstraint.Operator in org.quartz.locality.constraint

Methods in org.quartz.locality.constraint that return MemoryConstraint.Operator

<table>
<thead>
<tr>
<th>MemoryConstraint.Operator</th>
<th>MemoryConstraint.getOp()</th>
</tr>
</thead>
<tbody>
<tr>
<td>static MemoryConstraint.Operator</td>
<td>MemoryConstraint.Operator.valueOf(String)</td>
</tr>
<tr>
<td>static MemoryConstraint.Operator[]</td>
<td>MemoryConstraint.Operator.values()</td>
</tr>
</tbody>
</table>

Constructors in org.quartz.locality.constraint with parameters of type MemoryConstraint.Operator

MemoryConstraint(MemoryConstraint.Operator op, long value) MemoryConstraint constructor
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.locality.constraint.MemoryConstraint.Unit

Packages that use MemoryConstraint.Unit
org.quartz.locality.constraint

Uses of MemoryConstraint.Unit in
org.quartz.locality.constraint

Methods in org.quartz.locality.constraint that return MemoryConstraint.Unit

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static MemoryConstraint.Unit MemoryConstraint.Unit.valueOf(String name)</td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static MemoryConstraint.Unit[] MemoryConstraint.Unit.values()</td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

Methods in org.quartz.locality.constraint with parameters of type MemoryConstraint.Unit

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static MemoryConstraint MemoryConstraint.atLeastAvailable(int amount, MemoryConstraint.Unit unit)</td>
<td>Creates a MemoryConstraint for at least an amount of memory available</td>
</tr>
</tbody>
</table>

Overview Package Class Tree Deprecated Index Help
# Uses of Class

**org.quartz.locality.constraint.NodeGroupConstraint**

## Packages that use NodeGroupConstraint

<table>
<thead>
<tr>
<th>Package</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of NodeGroupConstraint in org.quartz.locality.constraint

## Methods in org.quartz.locality.constraint that return NodeGroupConstraint

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static NodeGroupConstraint partOfNodeGroup(String target)</td>
<td>Creates a NodeGroupConstraint that specifies the node group on which the Job has to be executed</td>
</tr>
</tbody>
</table>

## Uses of NodeGroupConstraint in org.quartz.locality.constraint.evaluator

## Methods in org.quartz.locality.constraint.evaluator with parameters of type NodeGroupConstraint

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean matches(NodeGroupConstraint constraint)</td>
<td>Evaluates the constraint against local node.</td>
</tr>
<tr>
<td>List&lt;String&gt; suitedNodes(NodeGroupConstraint constraint)</td>
<td>Returns a list of best suited node (instanceId) based on the constraint</td>
</tr>
<tr>
<td>void verify(NodeGroupConstraint constraint)</td>
<td>Verifies that the node group actually exists in the current configuration</td>
</tr>
</tbody>
</table>
# Uses of Class

**org.quartz.locality.constraint.NodeGroupConstraint.Operator**

## Packages that use *NodeGroupConstraint.Operator*

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint</td>
</tr>
</tbody>
</table>

## Uses of *NodeGroupConstraint.Operator* in **org.quartz.locality.constraint**

## Methods in **org.quartz.locality.constraint** that return *NodeGroupConstraint.Operator*

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td><code>NodeGroupConstraint.Operator.valueOf(String)</code></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static</td>
<td><code>NodeGroupConstraint.Operator.values()</code></td>
<td>Returns an array containing the constant enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

## Constructors in **org.quartz.locality.constraint** with parameters of type *NodeGroupConstraint.Operator***

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>NodeGroupConstraint(NodeGroupConstraint.Operator operator, String targetNodeGroup)</code></td>
<td></td>
</tr>
</tbody>
</table>
# Uses of Class

**org.quartz.locality.constraint.OsConstraint**

## Packages that use **OsConstraint**

<table>
<thead>
<tr>
<th>Package</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint</td>
<td></td>
</tr>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of **OsConstraint** in **org.quartz.locality.constraint**

## Fields in **org.quartz.locality.constraint** declared as **OsConstraint**

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>static OsConstraint</td>
<td>OsConstraint.LINUX</td>
</tr>
<tr>
<td>static OsConstraint</td>
<td>OsConstraint.OSX</td>
</tr>
<tr>
<td>static OsConstraint</td>
<td>OsConstraint.SOLARIS</td>
</tr>
<tr>
<td>static OsConstraint</td>
<td>OsConstraint.WINDOWS</td>
</tr>
</tbody>
</table>

## Uses of **OsConstraint** in **org.quartz.locality.constraint.evaluator**

## Methods in **org.quartz.locality.constraint.evaluator** with parameters of type **OsConstraint**

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>matches</td>
<td>OsEvaluator.matches(OsConstraint constraint) Evaluates the constraint against local node.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>suitedNodes</td>
<td>OsEvaluator.suitedNodes(OsConstraint constraint)</td>
</tr>
<tr>
<td>void</td>
<td>OsEvaluator.verify(OsConstraint constraint)</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verifies the validity of a constraint.</td>
<td></td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.locality.constraint.OsConstraint.Operator

Packages that use OsConstraint.Operator
org.quartz.locality.constraint

Uses of OsConstraint.Operator in
org.quartz.locality.constraint

Methods in org.quartz.locality.constraint that return OsConstraint.Operator

static OsConstraint.Operator.valueOf(String name)
    Returns the enum constant of this type with the specified name.

static OsConstraint.Operator[] values()
    Returns an array containing the constants of this enum type, in the order they are declared.

Constructors in org.quartz.locality.constraint with parameters of type OsConstraint.Operator

OsConstraint(OsConstraint.Operator operator, OsConstraint.OS targetedOs)

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

`org.quartz.locality.constraint.OsConstraint.OS`

## Packages that use `OsConstraint.OS`

- `org.quartz.locality.constraint`

## Uses of `OsConstraint.OS` in `org.quartz.locality.constraint`

### Methods in `org.quartz.locality.constraint` that return `OsConstraint.OS`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>OsConstraint.OS current()</code></td>
<td>Returns the current OS.</td>
</tr>
<tr>
<td><code>OsConstraint.OS targetedOs()</code></td>
<td>Getter for targeted OS.</td>
</tr>
<tr>
<td>static <code>OsConstraint.OS.valueOf(String name)</code></td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static <code>OsConstraint.OS[] values()</code></td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>

### Methods in `org.quartz.locality.constraint` with parameters of type `OsConstraint.OS`

- `abstract boolean OsConstraint.Operator.evaluate(OsConstraint.OS os, OsConstraint.OS target)`
- `boolean OsConstraint.matches(OsConstraint.OS actual)`
Constructors in `org.quartz.locality.constraint` with parameters of type
`OsConstraint.OS`

<table>
<thead>
<tr>
<th><code>OsConstraint</code></th>
<th><code>OsConstraint.Operator</code></th>
<th><code>OsConstraint.OS</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>operator,</td>
<td>targetedOs</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
org.quartz.locality.constraint.evaluator Class CpuEvaluator

java.lang.Object
   ↓ java.util.TimerTask
      ↓ org.quartz.locality.constraint.evaluator.EvaluatorTimerTask
         ↓ org.quartz.locality.constraint.evaluator.CpuEvaluator

All Implemented Interfaces:
   Runnable, Evaluator<CpuConstraint>,
   PersistentEvaluator<CpuConstraint, ConcurrentMap<CpuConstraint.Operator, Integer>>

public class CpuEvaluator
extends EvaluatorTimerTask
implements PersistentEvaluator<CpuConstraint, ConcurrentMap<CpuConstraint.Operator, Integer>>

Author:
   Alex Snaps

Nested Class Summary

| Nested classes/interfaces inherited from class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask.EvaluatorTimerTask.Status
| org.quartz.locality.constraint.evaluator.EvaluatorTimerTask.Status

Field Summary

| static boolean mxBeanAvgLoadMonitoring
| mxBeanAvgLoadMonitoring

Constructor Summary

| CpuEvaluator(Integer availableThreads)
| CpuEvaluator(Integer availableThreads)
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean hasAvailableThread()</td>
</tr>
<tr>
<td>boolean hasAvailableThread(String node)</td>
</tr>
<tr>
<td>void initialize(String nodeName, ConcurrentMap&lt;String, ConcurrentMap&lt;CpuConstraint.Operator&gt;&gt; nodeMap)</td>
</tr>
<tr>
<td>boolean matches(CpuConstraint constraint)</td>
</tr>
<tr>
<td>protected void monitor()</td>
</tr>
<tr>
<td>List&lt;String&gt; suitedNodes(CpuConstraint constraint)</td>
</tr>
<tr>
<td>void threadAcquired()</td>
</tr>
<tr>
<td>void threadReleased()</td>
</tr>
<tr>
<td>void verify(CpuConstraint constraint)</td>
</tr>
</tbody>
</table>

During its bootstrapping, the JobStore will call this method once passing a Clustered ConcurrentMap instance and the local instanceId.

Evaluates the constraint against local node.

Returns a list of best suited node (instanceId) based on the constraint.

Verifies the validity of a constraint.

---

### Methods inherited from class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask

- cancelMonitoring
- getInterval
- getStatus
- pauseMonitoring
- resumeMonitoring
- run
- startMonitoring

### Methods inherited from class java.util.TimerTask

- cancel
- scheduledExecutionTime

### Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait
Field Detail

mxBeanAvgLoadMonitoring

public static final boolean mxBeanAvgLoadMonitoring

Constructor Detail

CpuEvaluator

public CpuEvaluator(Integer availableThreads)

Method Detail

matches

public boolean matches(CpuConstraint constraint)

Description copied from interface: Evaluator
Evaluates the constraint against local node.

Specified by:
matches in interface Evaluator<CpuConstraint>

Parameters:
constraint - The constraint to evaluate

Returns:
true if local node is a match

See Also:
Constraint#matches(Object)

suitedNodes

public List<String> suitedNodes(CpuConstraint constraint)

Description copied from interface: Evaluator
Returns a list of best suited node (instanceId) based on the constraint
Specified by: 
**suitedNodes** in interface **Evaluator<CpuConstraint>**

**Parameters:**
- constraint - To find best suited nodes for

**Returns:**
List of best suited Quartz instanceId

---

**verify**

public void **verify**(CpuConstraint constraint)

**Description copied from interface: Evaluator**
Verifies the validity of a constraint. Throws exception should the constraint fail to validate.

**Specified by:**
**verify** in interface **Evaluator<CpuConstraint>**

**Parameters:**
- constraint - The constraint to validate

---

**initialize**

public void **initialize**(String nodeName, ConcurrentMap<String, ConcurrentMap<CpuConstraint.Operator>>)

**Description copied from interface: PersistentEvaluator**
During its bootstrapping, the JobStore will call this method once on each node, passing a Clustered ConcurrentMap instance and the local instanceId of the Scheduler

**Specified by:**
**initialize** in interface **PersistentEvaluator<CpuConstraint, ConcurrentMap<CpuConstraint.Operator>>**

**Parameters:**
- nodeName - Local instanceId
- store - the shared persistent storage for the Evaluator
threadAcquired

public void threadAcquired()

threadReleased

public void threadReleased()

hasAvailableThread

public boolean hasAvailableThread(String node)

hasAvailableThread

public boolean hasAvailableThread()

monitor

protected void monitor()

Specified by:

monitor in class EvaluatorTimerTask

Copyright 2001-2011, Terracotta, Inc.
org.quartz.locality.constraint.evaluator  Class EvaluatorTimerTask

java.lang.Object
  ↓ java.util.TimerTask
   ↓ org.quartz.locality.constraint.evaluator.EvaluatorTimerTask

All Implemented Interfaces:
  Runnable

Direct Known Subclasses:
  CpuEvaluator, MemoryEvaluator

public abstract class EvaluatorTimerTask
extends TimerTask

An evaluator that performs monitoring on a regular basis

Author:
  Alex Snaps

Nested Class Summary

static class EvaluatorTimerTask.Status

Constructor Summary

EvaluatorTimerTask()  

Method Summary

void cancelMonitoring()  
Permanently cancels the timer backing this instance.

static int
public EvaluatorTimerTask()

public void run()
Specified by:
run in interface Runnable
Specified by:
run in class TimerTask

monitor

protected abstract void monitor()

startMonitoring

public void startMonitoring()

Starts the monitoring

pauseMonitoring

public void pauseMonitoring()

Pauses the monitoring

resumeMonitoring

public void resumeMonitoring()

resumes the monitoring

cancelMonitoring

public void cancelMonitoring()

Permanently cancels the timer backing this instance.
getStatus

public EvaluatorTimerTask.Status getStatus()

getInterval

public static int getInterval()
EvaluatorTimerTask.Status

public static enum EvaluatorTimerTask.Status
extends Enum<EvaluatorTimerTask.Status>

All Implemented Interfaces:
Serializable, Comparable<EvaluatorTimerTask.Status>

Enclosing class:
EvaluatorTimerTask

Enum Constant Summary

<table>
<thead>
<tr>
<th>Constant</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAUSED</td>
<td></td>
</tr>
<tr>
<td>RUNNING</td>
<td></td>
</tr>
<tr>
<td>STOPPED</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static EvaluatorTimerTask.Status.valueOf(String name)</td>
<td>Returns the enum constant of this type with the specified name.</td>
</tr>
<tr>
<td>static EvaluatorTimerTask.Status[] values()</td>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.**Enum**

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.**Object**

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>getClass, notify, notifyAll, wait, wait, wait</td>
</tr>
</tbody>
</table>

**Enum Constant Detail**

**RUNNING**

public static final **EvaluatorTimerTask.Status** RUNNING

---

**PAUSED**

public static final **EvaluatorTimerTask.Status** PAUSED

---

**STOPPED**

public static final **EvaluatorTimerTask.Status** STOPPED

**Method Detail**

**values**

public static **EvaluatorTimerTask.Status[]** values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```java
for (EvaluatorTimerTask.Status c : EvaluatorTimerTask.Status.values())
    System.out.println(c);
```
Returns:
an array containing the constants of this enum type, in the order they are declared

valueOf

public static EvaluatorTimerTask.Status valueOf(String name)

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
name - the name of the enum constant to be returned.

Returns:
the enum constant with the specified name

Throws:
IllegalArgumentException - if this enum type has no constant with the specified name
NullPointerException - if the argument is null

Copyright 2001-2011, Terracotta, Inc.
org.quartz.locality.constraint.evaluator  Class MemoryEvaluator

java.lang.Object  
   \java.util.TimerTask  
      \org.quartz.locality.constraint.evaluator.EvaluatorTimerTask  
         \org.quartz.locality.constraint.evaluator.MemoryEvaluator

All Implemented Interfaces:
   Runnable, Evaluator<MemoryConstraint>, PersistentEvaluator<MemoryConstraint,Long>

public class MemoryEvaluator
   extends EvaluatorTimerTask
   implements PersistentEvaluator<MemoryConstraint,Long>

Evaluates memory constraints

Author:
   Alex Snaps

Nested Class Summary

Nested classes/interfaces inherited from class
org.quartz.locality.constraint.evaluator.EvaluatorTimerTask
EvaluatorTimerTask.Status

Constructor Summary

MemoryEvaluator()

Method Summary

void initialize(String nodeName,
   ConcurrentMap<String,Long> map)
Will start the monitoring, to store available memory on the local node in the shared map.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean matches(MemoryConstraint constraint)</td>
<td>Evaluates the constraint against local node.</td>
</tr>
<tr>
<td>boolean matches(MemoryConstraint constraint, String node)</td>
<td>Verifies whether the constraint passed is validating on the node.</td>
</tr>
<tr>
<td>void monitor()</td>
<td>Performs the monitoring of the free memory on the local node.</td>
</tr>
<tr>
<td>List&lt;String&gt; suitedNodes(MemoryConstraint constraint)</td>
<td>Returns a list of best suited node (instanceId) based on the constraint.</td>
</tr>
<tr>
<td>void verify(MemoryConstraint constraint)</td>
<td>Always true</td>
</tr>
</tbody>
</table>

Methods inherited from class org.quartz.locality.constraint.evaluator.EvaluatorTimerTask:
cancelMonitoring, getInterval, getStatus, pauseMonitoring, resumeMonitoring, run, startMonitoring

Methods inherited from class java.util.TimerTask:
cancel, scheduledExecutionTime

Methods inherited from class java.lang.Object:
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Constructor Detail**

**MemoryEvaluator**

public MemoryEvaluator()
Method Detail

matches

public boolean matches(MemoryConstraint constraint)

Evaluates the constraint against local node.

Specified by:
matches in interface Evaluator<MemoryConstraint>

Parameters:
constraint - The constraint to evaluate

Returns:
true if local node is a match

See Also:
Constraint#matches(Object)

matches

public boolean matches(MemoryConstraint constraint, String node)

Verifies whether the constraint passed is validating on the node

Parameters:
constraint - The constraint to evaluate
node - The node to evaluate against

Returns:
true if node is a match

suitedNodes

public List<String> suitedNodes(MemoryConstraint constraint)

Returns a list of best suited node (instanceId) based on the constraint

Specified by:
**suitedNodes** in interface **Evaluator<MemoryConstraint>**

**Parameters:**
- constraint - To find best suited nodes for

**Outputs:**
- List of best suited Quartz instanceId

---

**verify**

```java
public void verify(MemoryConstraint constraint)
```

Always true

**Specified by:**
- **verify** in interface **Evaluator<MemoryConstraint>**

**Parameters:**
- constraint - The constraint to validate

---

**monitor**

```java
public void monitor()
```

Performs the monitoring of the free memory on the local node

**Specified by:**
- **monitor** in class **EvaluatorTimerTask**

---

**initialize**

```java
public void initialize(String nodeName, ConcurrentMap<String, Long> map)
```

Will start the monitoring, to store available memory on the local node in the shared map.

**Specified by:**
- **initialize** in interface **PersistentEvaluator<MemoryConstraint, Long>**
Parameters:
nodeName - Local instanceId
map - the shared persistent storage for the Evaluator
Class `NodeGroupEvaluator`

All Implemented Interfaces:

- `Evaluator`<`NodeGroupConstraint`>

```java
public class NodeGroupEvaluator
    extends Object
    implements Evaluator<NodeGroupConstraint>
```

Evaluator that matches job execution to particular nodes based on the `Trigger` and `JobDetail` groups they are in.

The rules for matching is loaded from `quartzLocality.properties` where:

```properties
# Example of node groups
org.quartz.locality.nodeGroup.group0 = node0
org.quartz.locality.nodeGroup.group1 = node1,node2

# Automagic dispatching based on trigger group
org.quartz.locality.nodeGroup.group0.triggerGroups = blueTriggers
org.quartz.locality.nodeGroup.group1.jobGroups = redJobs
```

Which, in this example, defines two groups: `group0` and `group1`. Automatically dispatching jobs being executed by `Triggers` of group `blueTriggers` to `group0` (and hence `node0`). While `JobDetails` of group `redJobs` will be dispatched to `group1` (either `node1` or `node2`)

**Author:**

Alex Snaps

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NodeGroupEvaluator()</code></td>
</tr>
<tr>
<td>Constructor initializing all the local data structures</td>
</tr>
</tbody>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String getNodeGroupForJobGroup(String jobGroup)</code></td>
<td>Retrieves the node group for a particular trigger group</td>
</tr>
<tr>
<td><code>String getNodeGroupForTriggerGroup(String triggerGroup)</code></td>
<td>Retrieves the node group for a particular trigger group</td>
</tr>
<tr>
<td><code>boolean isNodeInGroup(String nodeName, String group)</code></td>
<td>Checks whether a node is part of a group</td>
</tr>
<tr>
<td><code>boolean matches(NodeGroupConstraint constraint)</code></td>
<td>Evaluates the constraint against local node.</td>
</tr>
<tr>
<td><code>List&lt;String&gt; suitedNodes(NodeGroupConstraint constraint)</code></td>
<td>Returns a list of best suited node (instanceId) based on the constraint</td>
</tr>
<tr>
<td><code>void verify(NodeGroupConstraint constraint)</code></td>
<td>Verifies that the node group actually exists in the current configuration</td>
</tr>
</tbody>
</table>

## Constructor Detail

**NodeGroupEvaluator**

```java
public NodeGroupEvaluator()
```

Constructor initializing all the local data structures

## Method Detail

**matches**
public boolean matches(NodeGroupConstraint constraint)

Evaluates the constraint against local node.

**Specified by:**
matches in interface Evaluator<NodeGroupConstraint>

**Parameters:**
constraint - The constraint to evaluate

**Returns:**
true if local node is a match

**See Also:**
Constraint#matches(Object)

---

suitedNodes

public List<String> suitedNodes(NodeGroupConstraint constraint)

Returns a list of best suited node (instanceId) based on the constraint

**Specified by:**
suitedNodes in interface Evaluator<NodeGroupConstraint>

**Parameters:**
constraint - To find best suited nodes for

**Returns:**
List of best suited Quartz instanceId

---

verify

public void verify(NodeGroupConstraint constraint)
throws LocalityException

Verifies that the node group actually exists in the current configuration

**Specified by:**
verify in interface Evaluator<NodeGroupConstraint>

**Parameters:**
constraint - The constraint to validate

**Throws:**
LocalityException - if node group is unknown

**getNodeGroupForTriggerGroup**

```java
public String getNodeGroupForTriggerGroup(String triggerGroup)
```

Retrieves the node group for a particular trigger group

**Parameters:**

- triggerGroup - the TriggerKey's group

**Returns:**

- the matching node group, null if null

**getNodeGroupForJobGroup**

```java
public String getNodeGroupForJobGroup(String jobGroup)
```

Retrieves the node group for a particular trigger group

**Parameters:**

- jobGroup - the JobKey's group

**Returns:**

- the matching node group, null if null

**isNodeInGroup**

```java
public boolean isNodeInGroup(String nodeName, String group)
```

Checks whether a node is part of a group

**Parameters:**

- nodeName - the instanceId of the node
- group - the group of the node

**Returns:**

- true is node is part of group, false otherwise
Copyright 2001-2011, Terracotta, Inc.
Class OsEvaluator

java.lang.Object  
  org.quartz.locality.constraint.evaluator.OsEvaluator

All Implemented Interfaces:
  Evaluator<OsConstraint>, PersistentEvaluator<OsConstraint, Set<String>>

public class OsEvaluator
extends Object
implements PersistentEvaluator<OsConstraint, Set<String>>

Evaluator that persists all nodes' Operation Systems

Author:
  Alex Snaps

Constructor Summary

OsEvaluator()

Method Summary

void initialize(String nodeName, ConcurrentMap<String, Set<String>> storage)
  During its bootstrapping, the JobStore will call this method once on each node, passing a Clustered ConcurrentMap instance and the local instanceId of the Scheduler

boolean matches(OsConstraint constraint)
  Evaluates the constraint against local node.

List<String> suitedNodes(OsConstraint constraint)
  Returns a list of best suited node (instanceId) based on the constraint

void verify(OsConstraint constraint)
Verifies the validity of a constraint.

### Constructor Detail

**OsEvaluator**

```java
public OsEvaluator()
```

### Method Detail

**matches**

```java
public boolean matches(OsConstraint constraint)
```

Evaluates the constraint against local node.

**Specified by:**

matches in interface Evaluator<OsConstraint>

**Parameters:**

- constraint - The constraint to evaluate

**Returns:**

- true if local node is a match

**See Also:**

- Constraint#matches(Object)

---

**suitedNodes**

```java
public List<String> suitedNodes(OsConstraint constraint)
```

Returns a list of best suited node (instanceId) based on the constraint
Specified by:

suitedNodes in interface Evaluator<OsConstraint>

Parameters:

constraint - To find best suited nodes for

Returns:

List of best suited Quartz instanceId

---

verify

public void verify(OsConstraint constraint)

Verifies the validity of a constraint. Throws exception should the constraint fail to validate.

Specified by:

verify in interface Evaluator<OsConstraint>

Parameters:

constraint - The constraint to validate

---

initialize

public void initialize(String nodeName, ConcurrentMap<String, Set<String>> storage)

During its bootstrapping, the JobStore will call this method once on each node, passing a Clustered ConcurrentMap instance and the local instanceId of the Scheduler

Specified by:

initialize in interface PersistentEvaluator<OsConstraint, Set<String>>

Parameters:

nodeName - Local instanceId
storage - the shared persistent storage for the Evaluator
Copyright 2001-2011, Terracotta, Inc.
org.quartz.locality.constraint.evaluator

Interfaces  
Evaluator  
PersistentEvaluator

Classes  
CpuEvaluator  
EvaluatorTimerTask  
MemoryEvaluator  
NodeGroupEvaluator  
OsEvaluator

Enums  
EvaluatorTimerTask.Status
Package org.quartz.locality.constraint.evaluator

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Evaluator&lt;T extends Constraint&gt;</strong></td>
<td>An Evaluator will match and potentially return best suited nodes for a Job to execute on.</td>
</tr>
<tr>
<td><strong>PersistentEvaluator&lt;T extends Constraint,V&gt;</strong></td>
<td>An particular Evaluator that will be passed a clustered ConcurrentHashMap instance to store shared data.</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CpuEvaluator</strong></td>
<td>An evaluator that performs monitoring on a regular basis.</td>
</tr>
<tr>
<td><strong>EvaluatorTimerTask</strong></td>
<td>An evaluator that performs monitoring on a regular basis.</td>
</tr>
<tr>
<td><strong>MemoryEvaluator</strong></td>
<td>Evaluates memory constraints.</td>
</tr>
<tr>
<td><strong>NodeGroupEvaluator</strong></td>
<td>Evaluator that matches job execution to particular nodes based on the Trigger and JobDetail groups they are in.</td>
</tr>
<tr>
<td><strong>OsEvaluator</strong></td>
<td>Evaluator that persists all nodes' Operation Systems</td>
</tr>
</tbody>
</table>

### Enum Summary

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EvaluatorTimerTask.Status</strong></td>
<td></td>
</tr>
</tbody>
</table>
Hierarchy For Package
org.quartz.locality.constraint.evaluator

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.locality.constraint.evaluator.**NodeGroupEvaluator**
    (implements org.quartz.locality.constraint.evaluator.**Evaluator**<T>)
  - org.quartz.locality.constraint.evaluator.**OsEvaluator** (implements org.quartz.locality.constraint.evaluator.**PersistentEvaluator**<T,V>)
- java.util.**TimerTask** (implements java.lang.**Runnable**)
  - org.quartz.locality.constraint.evaluator.**EvaluatorTimerTask**
    - org.quartz.locality.constraint.evaluator.**CpuEvaluator**
      (implements org.quartz.locality.constraint.evaluator.**PersistentEvaluator**<T>)
    - org.quartz.locality.constraint.evaluator.**MemoryEvaluator**
      (implements org.quartz.locality.constraint.evaluator.**PersistentEvaluator**<T>
Interface Hierarchy

- org.quartz.locality.constraint.evaluator.Evaluator<T>
  - org.quartz.locality.constraint.evaluator.PersistentEvaluator<T,V>
Enum Hierarchy

- java.lang.**Object**
  - java.lang.**Enum<E>** (implements java.lang.**Comparable<T>, java.io.**Serializable**)
    - org.quartz.locality.constraint.evaluator.**EvaluatorTimerTask.Status**
# Uses of Package

`org.quartz.locality.constraint.evaluator`

## Packages that use `org.quartz.locality.constraint.evaluator`

<table>
<thead>
<tr>
<th>Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.quartz.locality.constraint.evaluator</code></td>
</tr>
<tr>
<td><code>org.quartz.locality.constraint.evaluator</code></td>
</tr>
<tr>
<td><code>org.terracotta.modules.ehcache.store</code></td>
</tr>
</tbody>
</table>

## Classes in `org.quartz.locality.constraint.evaluator` used by `org.quartz.locality.constraint.evaluator`

<table>
<thead>
<tr>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluator</td>
</tr>
<tr>
<td>Evaluator will match and potentially return best suited nodes for a Job to execute on.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvaluatorTimerTask</td>
</tr>
<tr>
<td>An evaluator that performs monitoring on a regular basis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvaluatorTimerTask.Status</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PersistentEvaluator</td>
</tr>
<tr>
<td>An particular Evaluator that will be passed a clustered <code>ConcurrentMap</code> instance to store shared data.</td>
</tr>
</tbody>
</table>

## Classes in `org.quartz.locality.constraint.evaluator` used by `org.terracotta.modules.ehcache.store`

<table>
<thead>
<tr>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluator</td>
</tr>
<tr>
<td>An Evaluator will match and potentially return best suited nodes for a Job to execute on.</td>
</tr>
</tbody>
</table>
Java Interface: PersistentEvaluator

**Type Parameters:**

- T - A clusterable value type for the Map

**All Superinterfaces:**

- `Evaluator<T>`

**All Known Implementing Classes:**

- `CpuEvaluator`, `MemoryEvaluator`, `OsEvaluator`

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>initialize</code></td>
<td>(String nodeName, ConcurrentHashMap&lt;String, V&gt; map)</td>
</tr>
<tr>
<td></td>
<td>During its bootstrapping, the JobStore will call this method once on each node, passing a Clustered ConcurrentHashMap instance and the local instanceId of the Scheduler</td>
</tr>
</tbody>
</table>

**Methods inherited from interface**

- `Evaluator` matches, suitedNodes, verify

---

**Author:**

Alex Snaps
Method Detail

initialize

void initialize(String nodeName, ConcurrentMap<String, V> map)

During its bootstrapping, the JobStore will call this method once on each node, passing a Clustered ConcurrentMap instance and the local instanceId of the Scheduler

Parameters:
nodeName - Local instanceId
map - the shared persistent storage for the Evaluator

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.locality.constraint.evaluator.CpuEvaluator

No usage of org.quartz.locality.constraint.evaluator.CpuEvaluator

Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

**org.quartz.locality.constraint.evaluator.Evaluator**

## Packages that use **Evaluator**
- org.quartz.locality.constraint.evaluator
- org.terracotta.modules.ehcache.store

## Uses of **Evaluator** in

**org.quartz.locality.constraint.evaluator**

## Subinterfaces of **Evaluator** in **org.quartz.locality.constraint.evaluator**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PersistentEvaluator&lt;T extends Constraint, V&gt;</td>
<td>An particular <strong>Evaluator</strong> that will be passed a clustered.ConcurrentMap instance to store shared data.</td>
</tr>
</tbody>
</table>

## Classes in **org.quartz.locality.constraint.evaluator** that implement **Evaluator**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CpuEvaluator</td>
<td></td>
</tr>
<tr>
<td>MemoryEvaluator</td>
<td>Evaluates memory constraints</td>
</tr>
<tr>
<td>NodeGroupEvaluator</td>
<td>Evaluator that matches job execution to particular nodes based on the <strong>Trigger</strong> and <strong>JobDetail</strong> groups they are in.</td>
</tr>
<tr>
<td>OsEvaluator</td>
<td>Evaluator that persists all nodes' Operation Systems</td>
</tr>
</tbody>
</table>

## Uses of **Evaluator** in
Classes in `org.terracotta.modules.ehcache.store` that implement `Evaluator`:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EhcacheEvaluator</td>
<td>Ehcache evaluator, which lets Jobs be executed on certain node, depending of the locality of the data in a Ehcache</td>
</tr>
</tbody>
</table>
## Uses of Class

**org.quartz.locality.constraint.evaluator.EvaluatorTimerTask**

### Packages that use **EvaluatorTimerTask**

org.quartz.locality.constraint.evaluator

### Uses of **EvaluatorTimerTask** in

**org.quartz.locality.constraint.evaluator**

### Subclasses of **EvaluatorTimerTask** in

**org.quartz.locality.constraint.evaluator**

<table>
<thead>
<tr>
<th>class</th>
<th>CpuEvaluator</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>MemoryEvaluator</td>
</tr>
<tr>
<td></td>
<td>Evaluates memory constraints</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
## Uses of Class

**org.quartz.locality.constraint.evaluator.EvaluatorTimerTask.Status**

### Packages that use **EvaluatorTimerTask.Status**

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
</tr>
</tbody>
</table>

### Uses of **EvaluatorTimerTask.Status** in

**org.quartz.locality.constraint.evaluator**

### Methods in **org.quartz.locality.constraint.evaluator** that return **EvaluatorTimerTask.Status**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the enum constant of this type with specified name.</td>
<td><code>EvaluatorTimerTask.Status.valueOf(String)</code></td>
</tr>
<tr>
<td>Returns an array containing the constants of this enum type, in the order they are declared.</td>
<td><code>EvaluatorTimerTask.Status.values()</code></td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.locality.constraint.evaluator.MemoryEvaluator

No usage of org.quartz.locality.constraint.evaluator.MemoryEvaluator

Copyright 2001-2011, Terracotta, Inc.
No usage of org.quartz.locality.constraint.evaluator.NodeGroupEvaluator
Uses of Class
org.quartz.locality.constraint.evaluator.OsEvaluator

No usage of org.quartz.locality.constraint.evaluator.OsEvaluator

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.locality.constraint.evaluator.PersistentEvaluator

## Packages that use PersistentEvaluator

<table>
<thead>
<tr>
<th>Package</th>
<th>PersistentEvaluator</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.locality.constraint.evaluator</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of PersistentEvaluator in
org.quartz.locality.constraint.evaluator

## Classes in org.quartz.locality.constraint.evaluator that implement PersistentEvaluator

<table>
<thead>
<tr>
<th>Class</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>CpuEvaluator</td>
<td></td>
</tr>
<tr>
<td>MemoryEvaluator</td>
<td>Evaluates memory constraints</td>
</tr>
<tr>
<td>OsEvaluator</td>
<td>Evaluator that persists all nodes’ Operation Systems</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
org.quartz.plugins Classes SchedulerPluginWithUserTransactionSupport
## Class Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedulerPluginWithUserTransactionSupport</td>
<td>Base class for plugins that wish to support having their start and shutdown methods run within a UserTransaction.</td>
</tr>
</tbody>
</table>
Hierarchy For Package org.quartz.plugins

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.plugins.**SchedulerPluginWithUserTransactionSupport**
    (implements org.quartz.spi.SchedulerPlugin)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.plugins

Packages that use org.quartz.plugins

org.quartz.plugins.xml

Classes in org.quartz.plugins used by org.quartz.plugins.xml

SchedulerPluginWithUserTransactionSupport

Base class for plugins that wish to support having their start and shutdown methods run within a UserTransaction.

Copyright 2001-2011, Terracotta, Inc.
**org.quartz.plugins Class**

**SchedulerPluginWithUserTransactionSupport**

```java
java.lang.Object
    org.quartz.plugins.SchedulerPluginWithUserTransactionSupport
```

**All Implemented Interfaces:**

- org.quartz.spi.SchedulerPlugin

**Direct Known Subclasses:**

- [XMLSchedulingDataProcessorPlugin](#)

---

```java
public abstract class SchedulerPluginWithUserTransactionSupport
    extends Object
    implements org.quartz.spi.SchedulerPlugin
```

Base class for plugins that wish to support having their start and shutdown methods run within a `UserTransaction`. This is often necessary if using the JobStoreCMT and the plugin interacts with jobs/triggers.

The subclass should implement `start(UserTransaction)` and `shutdown(UserTransaction)`. The `UserTransaction` will be non-null if property `wrapInUserTransaction` is set to true.

For convenience, this base class also provides an `initialize()` implementation which saves the scheduler and plugin name, as well as `getLog()` for logging.

---

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">SchedulerPluginWithUserTransactionSupport()</a></td>
</tr>
</tbody>
</table>

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">getLog()</a></td>
</tr>
</tbody>
</table>

---
Get the commons Logger for this class.

```java
protected String getName()
Get the name of this plugin.

protected Scheduler getScheduler()
Get this plugin's Scheduler.

boolean getWrapInUserTransaction()
Wrap the start() and shutdown() methods in a UserTransaction.

void initialize(String name, Scheduler scheduler)

void setWrapInUserTransaction(boolean wrapInUserTransaction)
Wrap the start() and shutdown() methods in a UserTransaction.

void shutdown()
Based on the value of wrapInUserTransaction, wraps the call to shutdown(UserTransaction) in a UserTransaction.

protected void shutdown(UserTransaction userTransaction)
Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.

void start()
Based on the value of wrapInUserTransaction, wraps the call to start(UserTransaction) in a UserTransaction.

protected void start(UserTransaction userTransaction)
Called when the associated Scheduler is started, in order to let the plug-in know it can now make calls into the scheduler if it needs to.
```

**Methods inherited from class java.lang.Object**

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Constructor Detail**
SchedulerPluginWithUserTransactionSupport

public SchedulerPluginWithUserTransactionSupport()

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
</table>

**start**

protected void **start**(UserTransaction userTransaction)

Called when the associated Scheduler is started, in order to let the plug-in know it can now make calls into the scheduler if it needs to.

If UserTransaction is not null, the plugin can call setRollbackOnly() on it to signal that the wrapped transaction should rollback.

**Parameters:**

userTransaction - The UserTransaction object used to provide a transaction around the start() operation. It will be null if wrapInUserTransaction is false or if the transaction failed to be started.

**shutdown**

protected void **shutdown**(UserTransaction userTransaction)

Called in order to inform the SchedulerPlugin that it should free up all of it's resources because the scheduler is shutting down.

If UserTransaction is not null, the plugin can call setRollbackOnly() on it to signal that the wrapped transaction should rollback.

**Parameters:**

userTransaction - The UserTransaction object used to provide a transaction around the shutdown() operation. It will be null if wrapInUserTransaction is false or if the transaction failed to be started.
**getLog**

protected org.slf4j.Logger getLog()

Get the commons Logger for this class.

---

**getName**

protected String getName()

Get the name of this plugin. Set as part of initialize().

---

**getScheduler**

protected Scheduler getScheduler()

Get this plugin's Scheduler. Set as part of initialize().

---

**initialize**

public void initialize(String name, Scheduler scheduler)
throws SchedulerException

Specified by:
initialize in interface org.quartz.spi.SchedulerPlugin

Throws:
SchedulerException

---

**getWrapInUserTransaction**

public boolean getWrapInUserTransaction()

Wrap the start() and shutdown() methods in a UserTransaction. This is necessary for some plugins if using the JobStoreCMT.
**setWrapInUserTransaction**

public void **setWrapInUserTransaction**(boolean wrapInUserTransaction)

Wrap the start() and shutdown() methods in a UserTransaction. This is necessary for some plugins if using the JobStoreCMT.

---

**start**

public void **start**()

Based on the value of **wrapInUserTransaction**, wraps the call to start(UserTransaction) in a UserTransaction.

**Specified by:**

start in interface org.quartz.spi.SchedulerPlugin

---

**shutdown**

public void **shutdown**()

Based on the value of **wrapInUserTransaction**, wraps the call to shutdown(UserTransaction) in a UserTransaction.

**Specified by:**

shutdown in interface org.quartz.spi.SchedulerPlugin

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.plugins.SchedulerPluginWithUserTransactionSupport

Packages that use
SchedulerPluginWithUserTransactionSupport
org.quartz.plugins.xml

Uses of
SchedulerPluginWithUserTransactionSupport in
org.quartz.plugins.xml

Subclasses of SchedulerPluginWithUserTransactionSupport in
org.quartz.plugins.xml

- XMLSchedulingDataProcessorPlugin
  This plugin loads XML file(s) to add jobs and schedule them with triggers as the scheduler is initialized, and can optionally periodically scan the file for changes.
**org.quartz.plugins.history**  
**Class** **LoggingJobHistoryPlugin**

**java.lang.Object**  
`org.quartz.plugins.history.LoggingJobHistoryPlugin`

**All Implemented Interfaces:**  
JobListener, org.quartz.spi.SchedulerPlugin

```
public class LoggingJobHistoryPlugin
extends Object
implements org.quartz.spi.SchedulerPlugin, JobListener
```

Logs a history of all job executions (and execution vetos) via the Jakarta Commons-Logging framework.

The logged message is customizable by setting one of the following message properties to a String that conforms to the syntax of java.util.MessageFormat.

**JobToBeFiredMessage** - available message data are:

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>String</td>
<td>The Job's Name.</td>
</tr>
<tr>
<td>1</td>
<td>String</td>
<td>The Job's Group.</td>
</tr>
<tr>
<td>2</td>
<td>Date</td>
<td>The current time.</td>
</tr>
<tr>
<td>3</td>
<td>String</td>
<td>The Trigger's name.</td>
</tr>
<tr>
<td>4</td>
<td>String</td>
<td>The Triggers's group.</td>
</tr>
<tr>
<td>5</td>
<td>Date</td>
<td>The scheduled fire time.</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
<td>The next scheduled fire time.</td>
</tr>
<tr>
<td>7</td>
<td>Integer</td>
<td>The re-fire count from the JobExecutionContext.</td>
</tr>
</tbody>
</table>

The default message text is: 

```
"Job {1}.{0} fired (by trigger {4}.{3}) at: {2, date, HH:mm:ss MM/dd/yyyy}" 
```

**JobSuccessMessage** - available message data are:

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>String</td>
<td>The Job's Name.</td>
</tr>
<tr>
<td>Element</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>String</td>
<td>The Job's Name.</td>
</tr>
<tr>
<td>1</td>
<td>String</td>
<td>The Job's Group.</td>
</tr>
<tr>
<td>2</td>
<td>Date</td>
<td>The current time.</td>
</tr>
<tr>
<td>3</td>
<td>String</td>
<td>The Trigger's name.</td>
</tr>
<tr>
<td>4</td>
<td>String</td>
<td>The Triggers's group.</td>
</tr>
<tr>
<td>5</td>
<td>Date</td>
<td>The scheduled fire time.</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
<td>The next scheduled fire time.</td>
</tr>
<tr>
<td>7</td>
<td>Integer</td>
<td>The re-fire count from the JobExecutionContext.</td>
</tr>
<tr>
<td>8</td>
<td>String</td>
<td>The message from the thrown JobExecution Exception.</td>
</tr>
</tbody>
</table>

The default message text is "Job {1}.{0} execution complete at {2, date, HH:mm:ss MM/dd/yyyy} and reports: {8}"

JobFailedMessage - available message data are:

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>String</td>
<td>The Job's Name.</td>
</tr>
<tr>
<td>1</td>
<td>String</td>
<td>The Job's Group.</td>
</tr>
<tr>
<td>2</td>
<td>Date</td>
<td>The current time.</td>
</tr>
<tr>
<td>3</td>
<td>String</td>
<td>The Trigger's name.</td>
</tr>
<tr>
<td>4</td>
<td>String</td>
<td>The Triggers's group.</td>
</tr>
<tr>
<td>5</td>
<td>Date</td>
<td>The scheduled fire time.</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
<td>The next scheduled fire time.</td>
</tr>
<tr>
<td>7</td>
<td>Integer</td>
<td>The re-fire count from the JobExecutionContext.</td>
</tr>
<tr>
<td>8</td>
<td>String</td>
<td>The message from the thrown JobExecution Exception.</td>
</tr>
</tbody>
</table>

The default message text is "Job {1}.{0} execution failed at {2, date, HH:mm:ss MM/dd/yyyy} and reports: {8}"
Date | The next scheduled fire time.
---|---
Integer | The re-fire count from the JobExecutionContext.
The default message text is "Job \{1\}.\{0\} was vetoed. It was to be fired (by trigger \{4\}.\{3\}) at: \{2, date, HH:mm:ss MM/dd/yyyy\}"

**Author:**
James House

---

### Constructor Summary

**LoggingJobHistoryPlugin()**

---

### Method Summary

<table>
<thead>
<tr>
<th>String</th>
<th>getJobFailedMessage()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the message that is logged when a Job fails its execution.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getJobSuccessMessage()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the message that is logged when a Job successfully completes its execution.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getJobToBeFiredMessage()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the message that is logged when a Job is about to execute.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getJobWasVetoedMessage()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the message that is logged when a Job execution is vetoed by a trigger listener.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected org.slf4j.Logger</th>
<th>getLog()</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getName()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the name of the JobListener.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>initialize(String name, Scheduler scheduler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>jobExecutionVetoed(JobExecutionContext context)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Called by the Scheduler when a JobDetail was about</td>
<td></td>
</tr>
</tbody>
</table>
to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

```java
void jobToBeExecuted(JobExecutionContext context)
    Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

void jobWasExecuted(JobExecutionContext context, JobExecutionException jobException)
    Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

void setJobFailedMessage(String jobFailedMessage)
    Set the message that is logged when a Job fails its execution.

void setJobSuccessMessage(String jobSuccessMessage)
    Set the message that is logged when a Job successfully completes its execution.

void setJobToBeFiredMessage(String jobToBeFiredMessage)
    Set the message that is logged when a Job is about to execute.

void setJobWasVetoedMessage(String jobWasVetoedMessage)
    Set the message that is logged when a Job execution is vetoed by a trigger listener.

void shutdown()
    Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.

void start()
```

Methods inherited from class java.lang.Object

```java
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```
LoggingJobHistoryPlugin

public LoggingJobHistoryPlugin()

Method Detail

getLog

protected org.slf4j.Logger getLog()

getJobSuccessMessage

public String getJobSuccessMessage()

Get the message that is logged when a Job successfully completes its execution.

getJobFailedMessage

public String getJobFailedMessage()

Get the message that is logged when a Job fails its execution.

getJobToBeFiredMessage

public String getJobToBeFiredMessage()

Get the message that is logged when a Job is about to execute.

setJobSuccessMessage

public void setJobSuccessMessage(String jobSuccessMessage)

Set the message that is logged when a Job successfully completes its
execution.

**Parameters:**

jobSuccessMessage - String in java.text.MessageFormat syntax.

---

**setJobFailedMessage**

```java
public void setJobFailedMessage(String jobFailedMessage)
```

Set the message that is logged when a Job fails its execution.

**Parameters:**


---

**setJobToBeFiredMessage**

```java
public void setJobToBeFiredMessage(String jobToBeFiredMessage)
```

Set the message that is logged when a Job is about to execute.

**Parameters:**

jobToBeFiredMessage - String in java.text.MessageFormat syntax.

---

**getJobWasVetoedMessage**

```java
public String getJobWasVetoedMessage()
```

Get the message that is logged when a Job execution is vetoed by a trigger listener.

---

**setJobWasVetoedMessage**

```java
public void setJobWasVetoedMessage(String jobWasVetoedMessage)
```

Set the message that is logged when a Job execution is vetoed by a trigger listener.
Parameters:

initialize

```java
public void initialize(String name,
        Scheduler scheduler)
  throws SchedulerException
```

Called during creation of the Scheduler in order to give the
SchedulerPlugin a chance to initialize.

**Specified by:**
  initialize in interface org.quartz.spi.SchedulerPlugin

**Throws:**
  SchedulerConfigException - if there is an error initializing.
  SchedulerException

start

```java
public void start()
```

**Specified by:**
  start in interface org.quartz.spi.SchedulerPlugin

shutdown

```java
public void shutdown()
```

Called in order to inform the SchedulerPlugin that it should free up all of
it's resources because the scheduler is shutting down.

**Specified by:**
  shutdown in interface org.quartz.spi.SchedulerPlugin

getName
public String getName()

Description copied from interface: JobListener

Get the name of the JobListener.

Specified by:

generateName in interface JobListener

..........................................................................................................................

jobToBeExecuted

public void jobToBeExecuted(JobExecutionContext context)

Description copied from interface: JobListener

Called by the Scheduler when a JobDetail is about to be executed (an associated Trigger has occurred).

This method will not be invoked if the execution of the Job was vetoed by a TriggerListener.

Specified by:

generateJobToBeExecuted in interface JobListener

See Also:

JobListener.jobToBeExecuted(JobExecutionContext)

..........................................................................................................................

jobWasExecuted

public void jobWasExecuted(JobExecutionContext context,
                            JobExecutionException jobException)

Description copied from interface: JobListener

Called by the Scheduler after a JobDetail has been executed, and be for the associated Trigger's triggered(xx) method has been called.

Specified by:

generateJobWasExecuted in interface JobListener
See Also:

JobListener.jobWasExecuted(JobExecutionContext, JobExecutionException)

---

**jobExecutionVetoed**

public void **jobExecutionVetoed**(JobExecutionContext context)

**Description copied from interface:** JobListener

Called by the Scheduler when a JobDetail was about to be executed (an associated Trigger has occurred), but a TriggerListener vetoed it's execution.

**Specified by:**

jobExecutionVetoed in interface JobListener

See Also:

JobListener.jobExecutionVetoed(org.quartz.JobExecutionContext)
public class LoggingTriggerHistoryPlugin
extends Object
implements org.quartz.spi.SchedulerPlugin, TriggerListener

Logs a history of all trigger firings via the Jakarta Commons-Logging framework.

The logged message is customizable by setting one of the following message properties to a String that conforms to the syntax of java.util.MessageFormat.

TriggerFiredMessage - available message data are:

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>String</td>
<td>The Trigger's Name.</td>
</tr>
<tr>
<td>1</td>
<td>String</td>
<td>The Trigger's Group.</td>
</tr>
<tr>
<td>2</td>
<td>Date</td>
<td>The scheduled fire time.</td>
</tr>
<tr>
<td>3</td>
<td>Date</td>
<td>The next scheduled fire time.</td>
</tr>
<tr>
<td>4</td>
<td>Date</td>
<td>The actual fire time.</td>
</tr>
<tr>
<td>5</td>
<td>String</td>
<td>The Job's name.</td>
</tr>
<tr>
<td>6</td>
<td>String</td>
<td>The Job's group.</td>
</tr>
<tr>
<td>7</td>
<td>Integer</td>
<td>The re-fire count from the JobExecutionContext.</td>
</tr>
</tbody>
</table>

The default message text is "Trigger {1}.{0} fired job {6}.{5} at: {4, date, HH:mm:ss MM/dd/yyyy}".

TriggerMisfiredMessage - available message data are:

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>String</td>
<td>The Trigger's Name.</td>
</tr>
<tr>
<td>Element</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>String</td>
<td>The Trigger's Name.</td>
</tr>
<tr>
<td>1</td>
<td>String</td>
<td>The Trigger's Group.</td>
</tr>
<tr>
<td>2</td>
<td>Date</td>
<td>The scheduled fire time.</td>
</tr>
<tr>
<td>3</td>
<td>Date</td>
<td>The next scheduled fire time.</td>
</tr>
<tr>
<td>4</td>
<td>Date</td>
<td>The job completion time.</td>
</tr>
<tr>
<td>5</td>
<td>String</td>
<td>The Job's name.</td>
</tr>
<tr>
<td>6</td>
<td>String</td>
<td>The Job's group.</td>
</tr>
<tr>
<td>7</td>
<td>Integer</td>
<td>The re-fire count from the JobExecutionContext.</td>
</tr>
<tr>
<td>8</td>
<td>Integer</td>
<td>The trigger's resulting instruction code.</td>
</tr>
<tr>
<td>9</td>
<td>String</td>
<td>A human-readable translation of the trigger's resulting instruction code.</td>
</tr>
</tbody>
</table>

The default message text is "Trigger {1}.{0} completed firing job {6}.{5} at {4, date, HH:mm:ss MM/dd/yyyy} with resulting trigger instruction code: {9}"
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected org.slf4j.Logger getLog()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>String getName()</code></td>
<td></td>
<td>Get the name of the TriggerListener.</td>
</tr>
<tr>
<td><code>String getTriggerCompleteMessage()</code></td>
<td></td>
<td>Get the message that is printed upon the completion of a trigger's firing.</td>
</tr>
<tr>
<td><code>String getTriggerFiredMessage()</code></td>
<td></td>
<td>Get the message that is printed upon a trigger's firing.</td>
</tr>
<tr>
<td><code>String getTriggerMisfiredMessage()</code></td>
<td></td>
<td>Get the message that is printed upon a trigger's mis-firing.</td>
</tr>
<tr>
<td><code>void initialize(String name, Scheduler scheduler)</code></td>
<td></td>
<td>Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.</td>
</tr>
<tr>
<td><code>void setTriggerCompleteMessage(String triggerCompleteMessage)</code></td>
<td></td>
<td>Set the message that is printed upon the completion of a trigger's firing.</td>
</tr>
<tr>
<td><code>void setTriggerFiredMessage(String triggerFiredMessage)</code></td>
<td></td>
<td>Set the message that is printed upon a trigger's firing.</td>
</tr>
<tr>
<td><code>void setTriggerMisfiredMessage(String triggerMisfiredMessage)</code></td>
<td></td>
<td>Set the message that is printed upon a trigger's firing.</td>
</tr>
<tr>
<td><code>void shutdown()</code></td>
<td></td>
<td>Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.</td>
</tr>
<tr>
<td><code>void start()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>void triggerComplete(Trigger trigger, JobExecutionContext context, Trigger.CompletedExecutionInstruction triggerInstructionCode)</code></td>
<td></td>
<td>Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered(xx) method has been called.</td>
</tr>
</tbody>
</table>
| `void triggerFired(Trigger trigger, JobExecutionContext context)` | | Called by the Scheduler when a Trigger has fired, and it
### Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

**LoggingTriggerHistoryPlugin**

public LoggingTriggerHistoryPlugin()

### Method Detail

**getLog**

protected org.slf4j.Logger getLog()

**getTriggerCompleteMessage**

public String getTriggerCompleteMessage()

Get the message that is printed upon the completion of a trigger's firing.

**Returns:**

String
**getTriggerFiredMessage**

public String getTriggerFiredMessage()

Get the message that is printed upon a trigger's firing.

**Returns:**

String

---

**getTriggerMisfiredMessage**

public String getTriggerMisfiredMessage()

Get the message that is printed upon a trigger's mis-firing.

**Returns:**

String

---

**setTriggerCompleteMessage**

public void setTriggerCompleteMessage(String triggerCompleteMessage)

Set the message that is printed upon the completion of a trigger's firing.

**Parameters:**

triggerCompleteMessage - String in java.text.MessageFormat syntax.

---

**setTriggerFiredMessage**

public void setTriggerFiredMessage(String triggerFiredMessage)

Set the message that is printed upon a trigger's firing.

**Parameters:**

**setTriggerMisfiredMessage**

```java
public void setTriggerMisfiredMessage(String triggerMisfiredMessage)
```

Set the message that is printed upon a trigger's firing.

**Parameters:**

---

**initialize**

```java
public void initialize(String name, Scheduler scheduler)
    throws SchedulerException
```

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

**Specified by:**
- `initialize` in interface org.quartz.spi.SchedulerPlugin

**Throws:**
- SchedulerConfigException - if there is an error initializing.
- SchedulerException

---

**start**

```java
public void start()
```

**Specified by:**
- `start` in interface org.quartz.spi.SchedulerPlugin

---

**shutdown**

```java
public void shutdown()
```

Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.
Specified by:
    shutdown in interface org.quartz.spi.SchedulerPlugin

---

**getName**

```java
public String getName()
```

**Description copied from interface:** TriggerListener

Get the name of the TriggerListener.

**Specified by:**
    getName in interface TriggerListener

---

**triggerFired**

```java
public void triggerFired(Trigger trigger,
                        JobExecutionContext context)
```

**Description copied from interface:** TriggerListener

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed.

It is called before the vetoJobExecution(...) method of this interface.

**Specified by:**
    triggerFired in interface TriggerListener

**Parameters:**
    trigger - The Trigger that has fired.
    context - The JobExecutionContext that will be passed to the Job's execute(xx) method.

---

**triggerMisfired**

```java
public void triggerMisfired(Trigger trigger)
```
Description copied from interface: **TriggerListener**

Called by the Scheduler when a Trigger has misfired.

Consideration should be given to how much time is spent in this method, as it will affect all triggers that are misfiring. If you have lots of triggers misfiring at once, it could be an issue if this method does a lot.

**Specified by:**

- `triggerMisfired` in interface **TriggerListener**

**Parameters:**

- `trigger` - The Trigger that has misfired.

---

**triggerComplete**

```java
definition triggerComplete(Trigger trigger,
                        JobExecutionContext context,
                        Trigger.CompletedExecutionInstruction triggerInstructionCode)
```

Description copied from interface: **TriggerListener**

Called by the Scheduler when a Trigger has fired, it's associated JobDetail has been executed, and it's triggered(xx) method has been called.

**Specified by:**

- `triggerComplete` in interface **TriggerListener**

**Parameters:**

- `trigger` - The Trigger that was fired.
- `context` - The JobExecutionContext that was passed to the Job's execute(xx) method.
- `triggerInstructionCode` - the result of the call on the Trigger's triggered(xx) method.

---

**vetoJobExecution**

```java
definition vetoJobExecution(Trigger trigger,
                           JobExecutionContext context)
```

---
Description copied from interface: TriggerListener

Called by the Scheduler when a Trigger has fired, and it's associated JobDetail is about to be executed. If the implementation vetos the execution (via returning true), the job's execute method will not be called.

It is called after the triggerFired(..) method of this interface.

Specified by:
  vetoJobExecution in interface TriggerListener

Parameters:
  trigger - The Trigger that has fired.
  context - The JobExecutionContext that will be passed to the Job'sexecute(xx) method.
org.quartz.plugins.history  Classes  LoggingJobHistoryPlugin
LoggingTriggerHistoryPlugin
Package org.quartz.plugins.history

Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LoggingJobHistoryPlugin</strong></td>
<td>Logs a history of all job executions (and execution vetos) via the Jakarta Commons-Logging framework.</td>
</tr>
<tr>
<td><strong>LoggingTriggerHistoryPlugin</strong></td>
<td>Logs a history of all trigger firings via the Jakarta Commons-Logging framework.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.plugins.history

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.plugins.history.**LoggingJobHistoryPlugin** (implements org.quartz.**JobListener**, org.quartz.spi.SchedulerPlugin)
  - org.quartz.plugins.history.**LoggingTriggerHistoryPlugin** (implements org.quartz.spi.SchedulerPlugin, org.quartz.**TriggerListener**)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.plugins.history

No usage of org.quartz.plugins.history

<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Class</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Uses of Class
org.quartz.plugins.history.LoggingJobHistoryPlugin

No usage of org.quartz.plugins.history.LoggingJobHistoryPlugin

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.plugins.history.LoggingTriggerHistoryPlugin

No usage of org.quartz.plugins.history.LoggingTriggerHistoryPlugin

Copyright 2001-2011, Terracotta, Inc.
org.quartz.plugins.management Classes ShutdownHookPlugin
## Package org.quartz.plugins.management

### Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ShutdownHookPlugin</strong></td>
<td>This plugin catches the event of the JVM terminating (such as upon a CRTL-C) and tells the scheduler to shutdown.</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Hierarchy For Package
org.quartz.plugins.management

Package Hierarchies:
All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.plugins.management.**ShutdownHookPlugin** (implements org.quartz.spi.SchedulerPlugin)

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Uses of Package
org.quartz.plugins.management

No usage of org.quartz.plugins.management

Copyright 2001-2011, Terracotta, Inc.
public class ShutdownHookPlugin extends Object
implements org.quartz.spi.SchedulerPlugin

This plugin catches the event of the JVM terminating (such as upon a CRTL-C) and tells the scheduler to shutdown.

Author:
James House

See Also:
Scheduler.shutdown(boolean)
void \texttt{setCleanShutdown}(boolean b)

Set whether or not the plug-in is configured to cause a clean shutdown of the scheduler.

void \texttt{shutdown}()

Called in order to inform the \texttt{SchedulerPlugin} that it should free up all of it's resources because the scheduler is shutting down.

void \texttt{start}()

Methods inherited from class \texttt{java.lang.Object}

\texttt{clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait}

\section*{Constructor Detail}

\begin{description}
\item \texttt{ShutdownHookPlugin}
\end{description}

public \texttt{ShutdownHookPlugin}()

\section*{Method Detail}

\begin{description}
\item \texttt{isCleanShutdown}
\end{description}

public boolean \texttt{isCleanShutdown}()

Determine whether or not the plug-in is configured to cause a clean shutdown of the scheduler.

The default value is true.

\textbf{See Also:} \texttt{Scheduler.shutdown(boolean)}

\begin{description}
\item \texttt{setCleanShutdown}
\end{description}
public void setCleanShutdown(boolean b)

Set whether or not the plug-in is configured to cause a clean shutdown of the scheduler.

The default value is true.

See Also:
Scheduler.shutdown(boolean)

getLog

protected org.slf4j.Logger getLog()

initialize

public void initialize(String name, Scheduler scheduler)
throws SchedulerException

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

Specified by:
initialize in interface org.quartz.spi.SchedulerPlugin

Throws:
SchedulerConfigException - if there is an error initializing.
SchedulerException

start

public void start()

Specified by:
start in interface org.quartz.spi.SchedulerPlugin
shutdown

public void shutdown()

   Called in order to inform the SchedulerPlugin that it should free up all of its resources because the scheduler is shutting down.

   Specified by:
                 shutdown in interface org.quartz.spi.SchedulerPlugin
Uses of Class

org.quartz.plugins.management.ShutdownHookPlugin

No usage of org.quartz.plugins.management.ShutdownHookPlugin
org.quartz.plugins.xml Classes XMLSchedulingDataProcessorPlugin
This plugin loads XML file(s) to add jobs and schedule them with triggers as the scheduler is initialized, and can optionally periodically scan the file for changes.
Hierarchy For Package org.quartz.plugins.xml

Package Hierarchies:

All Packages
### Class Hierarchy

- java.lang.**Object**
  - org.quartz.plugins.**SchedulerPluginWithUserTransactionSupport**
    (implements org.quartz.spi.SchedulerPlugin)
  - org.quartz.plugins.xml.**XMLSchedulingDataProcessorPlugin**
    (implements org.quartz.jobs.**FileScanListener**)

---

**Overview**  **Package**  **Class**  **Use**  **Deprecated**  **Index**  **Help**

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.plugins.xml

No usage of org.quartz.plugins.xml

Copyright 2001-2011, Terracotta, Inc.
Class

XMLSchedulingDataProcessorPlugin

java.lang.Object
   ▼ org.quartz.plugins.SchedulerPluginWithUserTransactionSupport
      ▼ org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin

All Implemented Interfaces:
   FileScanListener, org.quartz.spi.SchedulerPlugin

public class XMLSchedulingDataProcessorPlugin

extends SchedulerPluginWithUserTransactionSupport
implements FileScanListener

This plugin loads XML file(s) to add jobs and schedule them with triggers as the scheduler is initialized, and can optionally periodically scan the file for changes.

The XML schema definition can be found here: http://www.quartz-scheduler.org/xml/job_scheduling_data_1_8.xsd

The periodically scanning of files for changes is not currently supported in a clustered environment.

If using this plugin with JobStoreCMT, be sure to set the plugin property wrapInUserTransaction to true. Also, if you have a positive scanInterval be sure to set org.quartz.scheduler.wrapJobExecutionInUserTransaction to true.

Author:
   James House, Pierre Awaragi, pl47ypus

See Also:
   XMLSchedulingDataProcessor

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected org.quartz.spi.ClassLoadHelper</td>
<td>classLoadHelper</td>
</tr>
</tbody>
</table>
## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;XMLSchedulingDataProcessorPlugin&gt;()</code></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>fileUpdated(String fileName)</code></td>
<td></td>
</tr>
<tr>
<td>String <code>getFileNames()</code></td>
<td>Comma separated list of file names (with paths) to the XML files that should be read.</td>
</tr>
<tr>
<td>long <code>getScanInterval()</code></td>
<td>The interval (in seconds) at which to scan for changes to the file.</td>
</tr>
<tr>
<td>void <code>initialize(String name, Scheduler scheduler)</code></td>
<td>Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.</td>
</tr>
<tr>
<td>boolean <code>isFailOnFileNotFound()</code></td>
<td>Whether or not initialization of the plugin should fail (throw an exception) if the file cannot be found.</td>
</tr>
<tr>
<td>void <code>processFile(String filePath)</code></td>
<td></td>
</tr>
<tr>
<td>void <code>setFailOnFileNotFound(boolean failOnFileNotFound)</code></td>
<td>Whether or not initialization of the plugin should fail (throw an exception) if the file cannot be found.</td>
</tr>
<tr>
<td>void <code>setFileNames(String fileNames)</code></td>
<td>The file name (and path) to the XML file that should be read.</td>
</tr>
<tr>
<td>void <code>setScanInterval(long scanInterval)</code></td>
<td>The interval (in seconds) at which to scan for changes to the file.</td>
</tr>
<tr>
<td>void <code>shutdown()</code></td>
<td>Overriden to ignore <code>wrapInUserTransaction</code> because shutdown() does not interact with the Scheduler.</td>
</tr>
<tr>
<td>void <code>start(UserTransaction userTransaction)</code></td>
<td>Called when the associated Scheduler is started, in order to let the plug-in know it can now make calls into the scheduler if it needs to.</td>
</tr>
</tbody>
</table>
Methods inherited from class org.quartz.plugins.SchedulerPluginWithUserTransactionSupport
getLog, getName, getScheduler, getWrapInUserTransaction, setWrapInUserTransaction, shutdown, start

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

classLoadHelper

protected org.quartz.spi.ClassLoadHelper classLoadHelper

Constructor Detail

XMLSchedulingDataProcessorPlugin

public XMLSchedulingDataProcessorPlugin()

Method Detail

getFileNames

public String getFileNames()

Comma separated list of file names (with paths) to the XML files that should be read.

setFileNames

public void setFileNames(String fileNames)
The file name (and path) to the XML file that should be read.

---

**getScanInterval**

```java
g public long getScanInterval()
```

The interval (in seconds) at which to scan for changes to the file. If the file has been changed, it is re-loaded and parsed. The default value for the interval is 0, which disables scanning.

**Returns:**
Returns the scanInterval.

---

**setScanInterval**

```java
g public void setScanInterval(long scanInterval)
```

The interval (in seconds) at which to scan for changes to the file. If the file has been changed, it is re-loaded and parsed. The default value for the interval is 0, which disables scanning.

**Parameters:**
- scanInterval - The scanInterval to set.

---

**isFailOnFileNotFound**

```java
g public boolean isFailOnFileNotFound()
```

Whether or not initialization of the plugin should fail (throw an exception) if the file cannot be found. Default is true.

---

**setFailOnFileNotFound**

```java
g public void setFailOnFileNotFound(boolean failOnFileNotFound)
```
Whether or not initialization of the plugin should fail (throw an exception) if the file cannot be found. Default is true.

**initialize**

```java
public void initialize(String name,
                        Scheduler scheduler)
    throws SchedulerException
```

Called during creation of the Scheduler in order to give the SchedulerPlugin a chance to initialize.

**Specified by:**
- `initialize` in interface `org.quartz.spi.SchedulerPlugin`

**Overrides:**
- `initialize` in class `SchedulerPluginWithUserTransactionSupport`

**Throws:**
- `SchedulerConfigException` - if there is an error initializing.
- `SchedulerException`

**start**

```java
public void start(UserTransaction userTransaction)
```

Description copied from class:
- `SchedulerPluginWithUserTransactionSupport`

Called when the associated Scheduler is started, in order to let the plug-in know it can now make calls into the scheduler if it needs to.

If UserTransaction is not null, the plugin can call setRollbackOnly() on it to signal that the wrapped transaction should rollback.

**Overrides:**
- `start` in class `SchedulerPluginWithUserTransactionSupport`

**Parameters:**
- `userTransaction` - The UserTransaction object used to provide a transaction around the start() operation. It will be null if
wrapInUserTransaction is false or if the transaction failed to be started.

shutdown

public void shutdown()

Overriden to ignore wrapInUserTransaction because shutdown() does not interact with the Scheduler.

Specified by:
shutdown in interface org.quartz.spi.SchedulerPlugin

Overrides:
shutdown in class SchedulerPluginWithUserTransactionSupport

processFile

public void processFile(String filePath)

fileUpdated

public void fileUpdated(String fileName)

Specified by:
fileUpdated in interface FileScanListener

See Also:
FileScanListener.fileUpdated(java.lang.String)
Uses of Class
org.quartz.plugins.xml.XMLSchedulingDataProcessor

No usage of org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin
Class CascadingClassLoadHelper

extends Object

implements org.quartz.spi.ClassLoadHelper

A ClassLoadHelper uses all of the ClassLoadHelper types that are found in this package in its attempts to load a class, when one scheme is found to work, it is promoted to the scheme that will be used first the next time a class is loaded (in order to improve performance).

This approach is used because of the wide variance in class loader behavior between the various environments in which Quartz runs (e.g. disparate application servers, stand-alone, mobile devices, etc.). Because of this disparity, Quartz ran into difficulty with a one class-load style fits-all design. Thus, this class loader finds the approach that works, then 'remembers' it.

Author: 
  jhouse, pl47ypus

See Also:
  ClassLoadHelper, LoadingLoaderClassLoadHelper, SimpleClassLoadHelper, ThreadContextClassLoadHelper, InitThreadContextClassLoadHelper

Constructor Summary

CascadingClassLoadHelper()
### Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassLoader</td>
<td>getClassLoader()</td>
<td>Enable sharing of the &quot;best&quot; class-loader with 3rd party.</td>
</tr>
<tr>
<td>URL</td>
<td>getResource(String name)</td>
<td>Finds a resource with a given name.</td>
</tr>
<tr>
<td>InputStream</td>
<td>getResourceAsStream(String name)</td>
<td>Finds a resource with a given name.</td>
</tr>
<tr>
<td>void</td>
<td>initialize()</td>
<td>Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to &quot;steal&quot; the class loader off of the calling thread, which is the thread that is initializing Quartz.</td>
</tr>
<tr>
<td>Class</td>
<td>loadClass(String name)</td>
<td>Return the class with the given name.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

### Constructor Detail

**CascadingClassLoadHelper**

```java
public CascadingClassLoadHelper()
```

### Method Detail

**initialize**

```java
public void initialize()
```

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.
### loadClass

```java
class loadClass(String name)
```

**Throws:**

- `ClassNotFoundException`

Returns the class with the given name.

**Specified by:**

- `loadClass in interface org.quartz.spi.ClassLoadHelper`

**Throws:**

- `ClassNotFoundException`

### getResource

```java
public URL getResource(String name)
```

Finds a resource with a given name. This method returns null if no resource with this name is found.

**Specified by:**

- `getResource in interface org.quartz.spi.ClassLoadHelper`

**Parameters:**

- `name` - name of the desired resource

**Returns:**

- a java.net.URL object

### getResourceAsStream

```java
public InputStream getResourceAsStream(String name)
```

Finds a resource with a given name. This method returns null if no resource with this name is found.

**Specified by:**
getResourceAsStream in interface org.quartz.spi.ClassLoadHelper

**Parameters:**
- name - name of the desired resource

**Returns:**
a java.io.InputStream object

---

**getClassLoader**

public ClassLoader getClassLoader()

Enable sharing of the "best" class-loader with 3rd party.

**Specified by:**
- getClassLoader in interface org.quartz.spi.ClassLoadHelper

**Returns:**
- the class-loader user be the helper.

---

Copyright 2001-2011, Terracotta, Inc.
public class HostnameInstanceIdGenerator

extends Object

implements org.quartz.spi.InstanceIdGenerator

InstanceIdGenerator that names the scheduler instance using just the machine hostname.

This class is useful when you know that your scheduler instance will be the only one running on a particular machine. Each time the scheduler is restarted, it will get the same instance id as long as the machine is not renamed.

See Also:
InstanceIdGenerator, SimpleInstanceIdGenerator

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostnameInstanceIdGenerator()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>String generateInstanceId()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object: clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Constructor Detail

HostnameInstanceIdGenerator

public HostnameInstanceIdGenerator()

Method Detail

generateInstanceId

public String generateInstanceId()
  throws SchedulerException

Specified by:
generateInstanceId in interface org.quartz.spi.InstanceIdGenerator

Throws:
  SchedulerException

Copyright 2001-2011, Terracotta, Inc.
Class InitThreadContextClassLoadHelper

public class InitThreadContextClassLoadHelper
extends Object
implements org.quartz.spi.ClassLoadHelper

A ClassLoadHelper that uses either the context class loader of the thread that initialized Quartz.

Author:
jhouse, pl47ypus

See Also:
ClassLoadHelper, ThreadContextClassLoadHelper,
SimpleClassLoadHelper, CascadingClassLoadHelper,
LoadingLoaderClassLoadHelper

Constructor Summary

InitThreadContextClassLoadHelper()

Method Summary

<table>
<thead>
<tr>
<th>ClassLoader</th>
<th>getClassLoader()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enable sharing of the class-loader with 3rd party.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>URL</th>
<th>getResource(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Finds a resource with a given name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InputStream</th>
<th>getResourceAsStream(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Finds a resource with a given name.</td>
</tr>
</tbody>
</table>
### Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
Return the class with the given name.

**Specified by:**
loadClass in interface org.quartz.spi.ClassLoadHelper

**Throws:**
ClassNotFoundException

---

**getResource**

`public URL getResource(String name)`

Finds a resource with a given name. This method returns null if no resource with this name is found.

**Specified by:**
getResource in interface org.quartz.spi.ClassLoadHelper

**Parameters:**
- name - name of the desired resource

**Returns:**
a java.net.URL object

---

**getResourceAsStream**

`public InputStream getResourceAsStream(String name)`

Finds a resource with a given name. This method returns null if no resource with this name is found.

**Specified by:**
getResourceAsStream in interface org.quartz.spi.ClassLoadHelper

**Parameters:**
- name - name of the desired resource

**Returns:**
a java.io.InputStream object
public ClassLoader getClassLoader()

Enable sharing of the class-loader with 3rd party.

**Specified by:**

getClassLoader in interface org.quartz.spi.ClassLoadHelper

**Returns:**

the class-loader user be the helper.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FRAMES</th>
<th>NO FRAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>
public class LoadingLoaderClassLoadHelper

extends Object

implements org.quartz.spi.ClassLoadHelper

A ClassLoadHelper that uses either the loader of it's own class
(this.getClass().getClassLoader().loadClass( .. )).

Author:
   jhouse, pl47ypus

See Also:
   ClassLoadHelper, InitThreadContextClassLoadHelper,
   SimpleClassLoadHelper, CascadingClassLoadHelper

---

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadingLoaderClassLoadHelper()</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassLoader</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>URL</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>InputStream</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
</tbody>
</table>
Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

<table>
<thead>
<tr>
<th>Class</th>
<th>loadClass(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Return the class with the given name.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LoadingLoaderClassLoadHelper

public LoadingLoaderClassLoadHelper()

Method Detail

initialize

public void initialize()

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

Specified by:
initialize in interface org.quartz.spi.ClassLoadHelper

loadClass

public Class loadClass(String name)
throws ClassNotFoundException

Return the class with the given name.
Specified by:
    loadClass in interface org.quartz.spi.ClassLoadHelper

Throws:
    ClassNotFoundException

getResource

public URL getResource(String name)

Finds a resource with a given name. This method returns null if no resource with this name is found.

Specified by:
    getResource in interface org.quartz.spi.ClassLoadHelper

Parameters:
    name - name of the desired resource

Returns:
    a java.net.URL object

getResourceAsStream

public InputStream getResourceAsStream(String name)

Finds a resource with a given name. This method returns null if no resource with this name is found.

Specified by:
    getResourceAsStream in interface org.quartz.spi.ClassLoadHelper

Parameters:
    name - name of the desired resource

Returns:
    a java.io.InputStream object

classLoader

public ClassLoader getClassLoader()
Enable sharing of the class-loader with 3rd party.

**Specified by:**
getClassLoader in interface `org.quartz.spi.ClassLoadHelper`

**Returns:**
the class-loader user be the helper.
org.quartz.simpl Classes

- CascadingClassLoadHelper
- HostnameInstanceIdGenerator
- InitThreadContextClassLoadHelper
- LoadingLoaderClassLoadHelper
- PropertySettingJobFactory
- RAMJobStore
- SimpleClassLoadHelper
- SimpleInstanceIdGenerator
- SimpleJobFactory
- SimpleThreadPool
- SimpleTimeBroker
- SystemPropertyInstanceIdGenerator
- ThreadContextClassLoadHelper
- ZeroSizeThreadPool
**Package org.quartz.simpl**

Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.

See: [Description](#)

### Class Summary

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CascadingClassLoadHelper</strong></td>
<td>A ClassLoadHelper uses all of the classes found in this package in its attempts to load a class; when a scheme is found to work, it is promoted and used first the next time a class is loaded (to improve performance).</td>
</tr>
<tr>
<td><strong>HostnameInstanceIdGenerator</strong></td>
<td>InstanceIdGenerator that names the scheduler instance using the machine hostname.</td>
</tr>
<tr>
<td><strong>InitThreadContextClassLoadHelper</strong></td>
<td>A ClassLoadHelper that uses either the context class loader thread that initialized Quartz.</td>
</tr>
<tr>
<td><strong>LoadingLoaderClassLoadHelper</strong></td>
<td>A ClassLoadHelper that uses either the thread or the loader of its own (this.getClass().getClassLoader()).</td>
</tr>
<tr>
<td><strong>PropertySettingJobFactory</strong></td>
<td>A JobFactory that instantiates the Job instance, typically a no-arg constructor, or more specifically: class.newInstance(), then attempts to set all values in the JobDataMap onto bean properties of the Job.</td>
</tr>
<tr>
<td><strong>RAMJobStore</strong></td>
<td>This class implements a JobStore that uses RAM as its underlying device.</td>
</tr>
<tr>
<td><strong>SimpleClassLoadHelper</strong></td>
<td>A ClassLoadHelper that simply calls Class.forName(..).</td>
</tr>
<tr>
<td><strong>SimpleInstanceIdGenerator</strong></td>
<td>The default InstanceIdGenerator used by Quartz to automatically generate instance ID.</td>
</tr>
<tr>
<td><strong>SimpleJobFactory</strong></td>
<td>The default JobFactory used by Quartz - simply calls on the job class.</td>
</tr>
<tr>
<td><strong>SimpleThreadPool</strong></td>
<td>This is a class that implements a ThreadPool interface, providing a simple implementation of the ThreadPool interface.</td>
</tr>
<tr>
<td><strong>SimpleTimeBroker</strong></td>
<td>The interface to be implemented by classes that want to provide a mechanism by which the QuartzScheduler can synchronize classloading events.</td>
</tr>
<tr>
<td><strong>SystemPropertyInstanceIdGenerator</strong></td>
<td>InstanceIdGenerator that will use a system property the scheduler.</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ThreadContextClassLoadHelper</strong></td>
<td>A ClassLoadHelper that uses either the current thread's loader (Thread.currentThread().getContextClassLoader().loadClass(..)</td>
</tr>
<tr>
<td><strong>ZeroSizeThreadPool</strong></td>
<td>This is class is a simple implementation based on the ThreadPool interface.</td>
</tr>
</tbody>
</table>
Package org.quartz.simpl Description

Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.

See the Quartz project for more information.

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.simpl

Package Hierarchies:
  All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.simpl.**CascadingClassLoadHelper** (implements org.quartz.spi.ClassLoadHelper)
  - org.quartz.simpl.**HostnameInstanceIdGenerator** (implements org.quartz.spi.InstanceIdGenerator)
  - org.quartz.simpl.**InitThreadContextClassLoadHelper** (implements org.quartz.spi.ClassLoadHelper)
  - org.quartz.simpl.**LoadingLoaderClassLoadHelper** (implements org.quartz.spi.ClassLoadHelper)
  - org.quartz.simpl.**RAMJobStore** (implements org.quartz.spi.JobStore)
  - org.quartz.simpl.**SimpleClassLoadHelper** (implements org.quartz.spi.ClassLoadHelper)
  - org.quartz.simpl.**SimpleInstanceIdGenerator** (implements org.quartz.spi.InstanceIdGenerator)
  - org.quartz.simpl.**SimpleJobFactory** (implements org.quartz.spi.JobFactory)
    - org.quartz.simpl.**PropertySettingJobFactory**
  - org.quartz.simpl.**SimpleThreadPool** (implements org.quartz.spi.ThreadPool)
  - org.quartz.simpl.**SimpleTimeBroker** (implements org.quartz.spi.TimeBroker)
  - org.quartz.simpl.**SystemPropertyInstanceIdGenerator** (implements org.quartz.spi.InstanceIdGenerator)
  - org.quartz.simpl.**ThreadContextClassLoadHelper** (implements org.quartz.spi.ClassLoadHelper)
  - org.quartz.simpl.**ZeroSizeThreadPool** (implements org.quartz.spi.ThreadPool)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.simpl

<table>
<thead>
<tr>
<th>Packages that use org.quartz.simpl</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.simpl</td>
</tr>
<tr>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.quartz.simpl used by org.quartz.simpl</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleJobFactory</td>
</tr>
<tr>
<td>The default JobFactory used by Quartz - simply calls newInstance() on the job class.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
public class PropertySettingJobFactory
extends SimpleJobFactory

A JobFactory that instantiates the Job instance (using the default no-arg constructor, or more specifically: class.newInstance()), and then attempts to set all values in the JobExecutionContext's JobDataMap onto bean properties of the Job.

Author: jhouse

See Also: JobFactory, SimpleJobFactory, JobExecutionContext.getMergedJobDataMap(), setWarnIfPropertyNotFound(boolean), setThrowIfPropertyNotFound(boolean)

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertySettingJobFactory()</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean isThrowIfPropertyNotFound()</td>
<td>Whether the JobInstantiation should fail and throw and exception if a key (name) and value (type) found in the JobDataMap does not correspond to a proptery setter on the Job class.</td>
</tr>
</tbody>
</table>
boolean isWarnIfPropertyNotFound()

    Whether a warning should be logged if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

Job newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler scheduler)

protected void setBeanProps(Object obj, JobDataMap data)

void setThrowIfPropertyNotFound(boolean throwIfNotFound)

    Whether the JobInstantiation should fail and throw an exception if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

void setWarnIfPropertyNotFound(boolean warnIfNotFound)

    Whether a warning should be logged if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

Methods inherited from class org.quartz.simpl.SimpleJobFactory
getLog

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PropertySettingJobFactory

public PropertySettingJobFactory()

Method Detail

newJob
public Job newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler scheduler)
    throws SchedulerException

Specified by:
    newJob in interface org.quartz.spi.JobFactory

Overrides:
    newJob in class SimpleJobFactory

Throws:
    SchedulerException

setBeanProps

protected void setBeanProps(Object obj, JobDataMap data)
    throws SchedulerException

Throws:
    SchedulerException

isThrowIfPropertyNotFound

public boolean isThrowIfPropertyNotFound()

Whether the JobInstantiation should fail and throw and exception if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

Returns:
    Returns the throwIfNotFound.

setThrowIfPropertyNotFound

public void setThrowIfPropertyNotFound(boolean throwIfNotFound)

Whether the JobInstantiation should fail and throw and exception if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.
Parameters:
throwIfNotFound - defaults to false.

isWarnIfPropertyNotFound

public boolean isWarnIfPropertyNotFound()

Whether a warning should be logged if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

Returns:
Returns the warnIfNotFound.

setWarnIfPropertyNotFound

public void setWarnIfPropertyNotFound(boolean warnIfNotFound)

Whether a warning should be logged if a key (name) and value (type) found in the JobDataMap does not correspond to a property setter on the Job class.

Parameters:
warnIfNotFound - defaults to true.

Copyright 2001-2011, Terracotta, Inc.
org.quartz.simpl  Class RAMJobStore

java.lang.Object
  ↓ org.quartz.simpl.RAMJobStore

All Implemented Interfaces:
  org.quartz.spi.JobStore

public class RAMJobStore
extends Object
implements org.quartz.spi.JobStore

This class implements a JobStore that utilizes RAM as its storage device.

As you should know, the ramification of this is that access is extremely fast, but the data is completely volatile - therefore this JobStore should not be used if true persistence between program shutdowns is required.

Author:
  James House, Sharada Jambula, Eric Mueller

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected HashSet&lt;JobKey&gt;</td>
<td>blockedJobs</td>
</tr>
<tr>
<td>protected HashMap&lt;String,Calendar&gt;</td>
<td>calendarsByName</td>
</tr>
<tr>
<td>protected HashMap&lt;String,HashMap&lt;JobKey,org.quartz.simpl.JobWrapper&gt;&gt;</td>
<td>jobsByGroup</td>
</tr>
<tr>
<td>protected HashMap&lt;JobKey,org.quartz.simpl.JobWrapper&gt;</td>
<td>jobsByKey</td>
</tr>
<tr>
<td>protected Object</td>
<td>lock</td>
</tr>
<tr>
<td>protected long</td>
<td>misfireThreshold</td>
</tr>
</tbody>
</table>
protected `HashSet<String>` `pausedJobGroups`

protected `HashSet<String>` `pausedTriggerGroups`

protected `org.quartz.spi.SchedulerSignaler` `signaler`

protected `TreeSet<org.quartz.simpl.TriggerWrapper>` `timeTriggers`

protected `ArrayList<org.quartz.simpl.TriggerWrapper>` `triggers`

protected `HashMap<String, HashMap<TriggerKey, org.quartz.simpl.TriggerWrapper>>` `triggersByGroup`

protected `HashMap<TriggerKey, org.quartz.simpl.TriggerWrapper>` `triggersByKey`

---

**Constructor Summary**

`RAMJobStore()`

Create a new RAMJobStore.

---

**Method Summary**

`List<org.quartz.spi.OperableTrigger>` `acquireNextTriggers`(long `noLaterThan`, long `timeWindow`)

Get a handle to the next trigger to be fired, and mark it as 'reserved' within the calling scheduler.

`protected boolean` `applyMisfire`(org.quartz.simpl.TriggerWrapper `tw`)

`boolean` `checkExists`(JobKey `jobKey`)

Determine whether a Job with the given identifier already exists within the scheduler.

`boolean` `checkExists`(TriggerKey `triggerKey`)

Determine whether a Trigger with the given identifier already exists within the scheduler.

`void` `clearAllSchedulingData()`
Clear (delete!) all scheduling data -

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>List&lt;String&gt;</code> <code>getCalendarNames()</code></td>
<td>Get the names of all of the calendars.</td>
</tr>
<tr>
<td><code>long</code> <code>getEstimatedTimeToReleaseAndAcquireTrigger()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected String</code> <code>getFiredTriggerRecordId()</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code> <code>getJobGroupNames()</code></td>
<td>Get the names of all of the Job groups.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> <code>getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Get the names of all of the Jobs that match the specified matcher.</td>
</tr>
<tr>
<td><code>protected org.slf4j.Logger</code> <code>getLog()</code></td>
<td></td>
</tr>
<tr>
<td><code>long</code> <code>getMisfireThreshold()</code></td>
<td></td>
</tr>
<tr>
<td><code>int</code> <code>getNumberOfCalendars()</code></td>
<td>Get the number of calendars that are scheduled.</td>
</tr>
<tr>
<td><code>int</code> <code>getNumberOfJobs()</code></td>
<td>Get the number of JobDetails that are scheduled.</td>
</tr>
<tr>
<td><code>int</code> <code>getNumberOfTriggers()</code></td>
<td>Get the number of Triggers that are scheduled.</td>
</tr>
<tr>
<td><code>Set&lt;JobKey&gt;</code> <code>getPausedTriggerGroups()</code></td>
<td></td>
</tr>
<tr>
<td><code>List&lt;String&gt;</code> <code>getTriggerGroupNames()</code></td>
<td>Get the names of all of the Trigger groups.</td>
</tr>
<tr>
<td><code>Set&lt;TriggerKey&gt;</code> <code>getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; groupMatcher)</code></td>
<td>Get the names of all of the Triggers that are scheduled.</td>
</tr>
<tr>
<td><code>List&lt;org.quartz.spi.OperableTrigger&gt;</code> <code>getTriggersForJob(JobKey jobKey)</code></td>
<td>Get all of the Triggers that are associated with the given Job.</td>
</tr>
<tr>
<td><code>Trigger.TriggerState</code> <code>getTriggerState(TriggerKey triggerKey)</code></td>
<td>Get the current state of the identified Trigger.</td>
</tr>
<tr>
<td><code>protected ArrayList&lt;org.quartz.simpl.TriggerWrapper&gt;</code> <code>getTriggerWrappersForCalendar(String calendarName)</code></td>
<td>Get the Wrappers for the given Calendar.</td>
</tr>
</tbody>
</table>
ArrayList<org.quartz.simpl.TriggerWrapper> getTriggerWrappersForJob(JobKey jobKey)

void initialize(org.quartz.spi.ClassLoadHelper loadHelper, org.quartz.spi.SchedulerSignaler signaler)

Called by the QuartzScheduler before giving it a chance to initialize.

boolean isClustered()

void pauseAll()

Pause all triggers - equivalent of calling every group.

void pauseJob(JobKey jobKey)

Pause the JobDetail with the given Trigger.

List<String> pauseJobs(GroupMatcher<JobKey> matcher)

Pause all of the JobDetails in the Triggers.

void pauseTrigger(TriggerKey triggerKey)

Pause the Trigger with the given name.

List<String> pauseTriggers(GroupMatcher<TriggerKey>)

Pause all of the known Triggers matching.

protected String peekTriggers()

void releaseAcquiredTrigger(org.quartz.spi.OperableTrigger trigger)

Inform the JobStore that the scheduler released the acquired Trigger, that it had previously acquired.

boolean removeCalendar(String calName)

Remove (delete) the Calendar with the given name.

boolean removeJob(JobKey jobKey)

Remove (delete) the Job with the given reference.

boolean removeJobs(List<JobKey> jobKeys)

boolean removeTrigger(TriggerKey triggerKey)

Remove (delete) the Trigger with the given name.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>removeTriggers(List&lt;TriggerKey&gt; triggerTriggers)</code></td>
<td>Remove triggers specified in the trigger list.</td>
</tr>
<tr>
<td><code>replaceTrigger(TriggerKey triggerKey, org.quartz.spi.OperableTrigger newTrigger)</code></td>
<td>Replace a trigger with a new one.</td>
</tr>
<tr>
<td><code>resumeAll()</code></td>
<td>Resume (un-pause) all triggers - equivalent to <code>resumeTriggerGroup(group)</code></td>
</tr>
<tr>
<td><code>resumeJob(JobKey jobKey)</code></td>
<td>Resume (un-pause) the JobDetail for the given JobKey.</td>
</tr>
<tr>
<td><code>resumeJobs(GroupMatcher&lt;JobKey&gt; matcher)</code></td>
<td>Resume (un-pause) all of the JobDetails matching the given matcher.</td>
</tr>
<tr>
<td><code>resumeTrigger(TriggerKey triggerKey)</code></td>
<td>Resume (un-pause) the Trigger with the given TriggerKey.</td>
</tr>
<tr>
<td><code>resumeTriggers(GroupMatcher&lt;TriggerKey&gt; matcher)</code></td>
<td>Resume (un-pause) all of the Triggers matching the given matcher.</td>
</tr>
<tr>
<td><code>retrieveCalendar(String calName)</code></td>
<td>Retrieve the given Trigger with the given calendar name.</td>
</tr>
<tr>
<td><code>retrieveJob(JobKey jobKey)</code></td>
<td>Retrieve the JobDetail for the given JobKey.</td>
</tr>
<tr>
<td><code>retrieveTrigger(TriggerKey triggerKey)</code></td>
<td>Retrieve the given Trigger.</td>
</tr>
<tr>
<td><code>schedulerStarted()</code></td>
<td></td>
</tr>
<tr>
<td><code>setAllTriggersOfJobToState(JobKey jobKey)</code></td>
<td>Set all triggers of the given JobKey to a specific state.</td>
</tr>
<tr>
<td><code>setInstanceId(String schedInstId)</code></td>
<td>Set the instance ID of the scheduler.</td>
</tr>
<tr>
<td><code>setInstanceName(String schedName)</code></td>
<td>Set the instance name of the scheduler.</td>
</tr>
<tr>
<td><code>setMisfireThreshold(long misfireThreshold)</code></td>
<td>Set the misfire threshold. The number of milliseconds by which a trigger's next-fire-time, in order for it to be considered misfired, will be applied.</td>
</tr>
<tr>
<td><code>setThreadPoolSize(int poolSize)</code></td>
<td>Set the size of the thread pool.</td>
</tr>
</tbody>
</table>
void shutdown()
   Called by the QuartzScheduler to free up all of its resources because the scheduler is shutting down.

void storeCalendar(String name, Calendar calendar, boolean replaceExisting, boolean updateTriggers)
   Store the given Calendar.

void storeJob(JobDetail newJob, boolean replaceExisting)
   Store the given Job.

void storeJobAndTrigger(JobDetail newJob, org.quartz.spi.OperableTrigger newTrigger)
   Store the given JobDetail and Trigger.

void storeJobsAndTriggers(Map<JobDetail, boolean> replace)

void storeTrigger(org.quartz.spi.OperableTrigger newTrigger, boolean replaceExisting)
   Store the given Trigger.

boolean supportsPersistence()

void triggeredJobComplete(org.quartz.spi.JobDetail jobDetail,
                          Trigger.CompletedExecutionInstruction instruction)
   Inform the JobStore that the scheduler has completed the execution of the given Trigger (and the execution of its associated Job in the given JobDetail should be updated if the JobDetail has triggers associated with it).

List<org.quartz.spi.TriggerFiredResult> triggersFired(List<org.quartz.spi.OperableTrigger> triggers)
   Inform the JobStore that the scheduler is now firing the executing Job (executing its associated Job), that it had previously acquired (reserved).

**Methods inherited from class java.lang.Object**
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Field Detail**
jobsByKey
protected `HashMap<JobKey, org.quartz.simpl.JobWrapper>` jobsByKey

triggersByKey
protected `HashMap<TriggerKey, org.quartz.simpl.TriggerWrapper>` triggersByKey

jobsByGroup
protected `HashMap<String, HashMap<JobKey, org.quartz.simpl.JobWrapper>>` jobsByGroup

triggersByGroup
protected `HashMap<String, HashMap<TriggerKey, org.quartz.simpl.TriggerWrapper>>` triggersByGroup

timeTriggers
protected `TreeSet<org.quartz.simpl.TriggerWrapper>` timeTriggers

calendarsByName
protected `HashMap<String, Calendar>` calendarsByName

triggers
protected `ArrayList<org.quartz.simpl.TriggerWrapper>` triggers

lock
protected final `Object` lock
pausedTriggerGroups
protected `HashSet<String>` pausedTriggerGroups

pausedJobGroups
protected `HashSet<String>` pausedJobGroups

blockedJobs
protected `HashSet<JobKey>` blockedJobs

misfireThreshold
protected `long` misfireThreshold

signaler
protected `org.quartz.spi.SchedulerSignaler` signaler

### Constructor Detail

**RAMJobStore**

```java
public RAMJobStore()
```

Create a new RAMJobStore.

### Method Detail

**getLog**

```java
protected org.slf4j.Logger getLog()
```
**initialize**

```java
public void initialize(org.quartz.spi.ClassLoadHelper loadHelper,
                       org.quartz.spi.SchedulerSignaler signaler)
```

Called by the QuartzScheduler before the JobStore is used, in order to give the it a chance to initialize.

**Specified by:**

`initialize` in interface `org.quartz.spi.JobStore`

---

**schedulerStarted**

```java
public void schedulerStarted()
```

**Throws:**

`SchedulerException`

**Specified by:**

`schedulerStarted` in interface `org.quartz.spi.JobStore`

---

**getMisfireThreshold**

```java
public long getMisfireThreshold()
```

---

**setMisfireThreshold**

```java
public void setMisfireThreshold(long misfireThreshold)
```

The number of milliseconds by which a trigger must have missed its next-fire-time, in order for it to be considered "misfired" and thus have its misfire instruction applied.

**Parameters:**

`misfireThreshold` -
shutdown

public void shutdown()

Called by the QuartzScheduler to inform the JobStore that it should free up all of it's resources because the scheduler is shutting down.

Specified by:
    shutdown in interface org.quartz.spi.JobStore

supportsPersistence

public boolean supportsPersistence()

Specified by:
    supportsPersistence in interface org.quartz.spi.JobStore

clearAllSchedulingData

public void clearAllSchedulingData()  
throws JobPersistenceException

Clear (delete!) all scheduling data - all Jobs, Triggers Calendars.

Specified by:
    clearAllSchedulingData in interface org.quartz.spi.JobStore

Throws:
    JobPersistenceException

storeJobAndTrigger

public void storeJobAndTrigger(JobDetail newJob,  
org.quartz.spi.OperableTrigger newTri  
throws JobPersistenceException

Store the given JobDetail and Trigger.
Specified by:
    storeJobAndTrigger in interface org.quartz.spi.JobStore

Parameters:
    newJob - The JobDetail to be stored.
    newTrigger - The Trigger to be stored.

Throws:
    ObjectAlreadyExistsException - if a Job with the same name/group already exists.
    JobPersistenceException

---

storeJob

**public void storeJob(JobDetail newJob,**
    **boolean replaceExisting)**

throws ObjectAlreadyExistsException

Store the given Job.

Specified by:
    storeJob in interface org.quartz.spi.JobStore

Parameters:
    newJob - The Job to be stored.
    replaceExisting - If true, any Job existing in the JobStore with the same name & group should be over-written.

Throws:
    ObjectAlreadyExistsException - if a Job with the same name/group already exists, and replaceExisting is set to false.

---

removeJob

public boolean removeJob(JobKey jobKey)

Remove (delete) the Job with the given name, and any Trigger s that reference it.

Specified by:
    removeJob in interface org.quartz.spi.JobStore

Returns:
true if a Job with the given name & group was found and removed from the store.

---

**removeJobs**

public boolean **removeJobs**(List&lt;JobKey&gt; jobKeys)
    throws JobPersistenceException

  **Specified by:**
  removeJobs in interface org.quartz.spi.JobStore

  **Throws:**
  JobPersistenceException

---

**removeTriggers**

public boolean **removeTriggers**(List&lt;TriggerKey&gt; triggerKeys)
    throws JobPersistenceException

  **Specified by:**
  removeTriggers in interface org.quartz.spi.JobStore

  **Throws:**
  JobPersistenceException

---

**storeJobsAndTriggers**

public void **storeJobsAndTriggers**(Map&lt;JobDetail,List&lt;Trigger&gt;&gt; triggersAndJobs,
    boolean replace)
    throws ObjectAlreadyExistsException,
            JobPersistenceException

  **Specified by:**
  storeJobsAndTriggers in interface org.quartz.spi.JobStore

  **Throws:**
  ObjectAlreadyExistsException
  JobPersistenceException
storeTrigger

public void storeTrigger(org.quartz.spi.OperableTrigger newTrigger, boolean replaceExisting)
throws JobPersistenceException

Store the given Trigger.

Specified by:
storeTrigger in interface org.quartz.spi.JobStore

Parameters:
newTrigger - The Trigger to be stored.
replaceExisting - If true, any Trigger existing in the JobStore with the same name & group should be over-written.

Throws:
ObjectAlreadyExistsException - if a Trigger with the same name/group already exists, and replaceExisting is set to false.
JobPersistenceException

See Also:
#pauseTriggerGroup(SchedulingContext, String)

removeTrigger

public boolean removeTrigger(TriggerKey triggerKey)

Remove (delete) the Trigger with the given name.

Specified by:
removeTrigger in interface org.quartz.spi.JobStore

Returns:
true if a Trigger with the given name & group was found and removed from the store.

replaceTrigger

public boolean replaceTrigger(TriggerKey triggerKey, org.quartz.spi.OperableTrigger newTrig)
throws JobPersistenceException
Specified by:
replaceTrigger in interface org.quartz.spi.JobStore

Throws:
JobPersistenceException

See Also:

-------------------------------------------------------------------------------

retrieveJob

public JobDetail retrieveJob(JobKey jobKey)

Retrieve the JobDetail for the given Job.

Specified by:
retrieveJob in interface org.quartz.spi.JobStore

Returns:
The desired Job, or null if there is no match.

-------------------------------------------------------------------------------

retrieveTrigger

public org.quartz.spi.OperableTrigger retrieveTrigger(TriggerKey triggerKey)

Retrieve the given Trigger.

Specified by:
retrieveTrigger in interface org.quartz.spi.JobStore

Returns:
The desired Trigger, or null if there is no match.

-------------------------------------------------------------------------------

checkExists

public boolean checkExists(JobKey jobKey)

Determine whether a Job with the given identifier already exists within the scheduler.
**checkExists**

public boolean checkExists(TriggerKey triggerKey)

Determine whether a Trigger with the given identifier already exists within the scheduler.

**Specified by:**
checkExists in interface org.quartz.spi.JobStore

**Parameters:**
triggerKey - the identifier to check for

**Returns:**
true if a Trigger exists with the given identifier

**Throws:**
SchedulerException

---

**getTriggerState**

public Trigger.TriggerState getTriggerState(TriggerKey triggerKey)

throws JobPersistenceException

Get the current state of the identified Trigger.

**Specified by:**
getTriggerState in interface org.quartz.spi.JobStore

**Throws:**
JobPersistenceException

**See Also:**
Trigger#NORMAL, Trigger#PAUSED, Trigger#COMPLETE,
storeCalendar

public void storeCalendar(String name, Calendar calendar, boolean replaceExisting, boolean updateTriggers)
    throws ObjectAlreadyExistsException

Store the given Calendar.

Specified by:
    storeCalendar in interface org.quartz.spi.JobStore

Parameters:
    calendar - The Calendar to be stored.
    replaceExisting - If true, any Calendar existing in the JobStore with the same name & group should be over-written.
    updateTriggers - If true, any Triggers existing in the JobStore that reference an existing Calendar with the same name with have their next fire time re-computed with the new Calendar.

Throws:
    ObjectAlreadyExistsException - if a Calendar with the same name already exists, and replaceExisting is set to false.

removeCalendar

public boolean removeCalendar(String calName)
    throws JobPersistenceException

Remove (delete) the Calendar with the given name.

If removal of the Calendar would result in Triggers pointing to non-existent calendars, then a JobPersistenceException will be thrown.

* 

Specified by:
removeCalendar in interface org.quartz.spi.JobStore

**Parameters:**
- calName - The name of the Calendar to be removed.

**Returns:**
- true if a Calendar with the given name was found and removed from the store.

** Throws:**
- JobPersistenceException

---

**retrieveCalendar**

public Calendar retrieveCalendar(String calName)

- Retrieve the given Trigger.

**Specified by:**
- retrieveCalendar in interface org.quartz.spi.JobStore

**Parameters:**
- calName - The name of the Calendar to be retrieved.

**Returns:**
- The desired Calendar, or null if there is no match.

---

**getNumberOfJobs**

public int getNumberOfJobs()

- Get the number of JobDetail s that are stored in the JobsStore.

**Specified by:**
- getNumberOfJobs in interface org.quartz.spi.JobStore

---

**getNumberOfTriggers**

public int getNumberOfTriggers()

- Get the number of Trigger s that are stored in the JobsStore.
Specified by:
getNumberOfTriggers in interface org.quartz.spi.JobStore

getNumberOfCalendars

public int getNumberOfCalendars()

Get the number of Calendar s that are stored in the JobsStore.

Specified by:
getNumberOfCalendars in interface org.quartz.spi.JobStore

getJobKeys

public Set&lt;JobKey&gt; getJobKeys(GroupMatcher&lt;JobKey&gt; matcher)

Get the names of all of the Job s that match the given groupMatcher.

Specified by:
getJobKeys in interface org.quartz.spi.JobStore

getCalendarNames

public List&lt;String&gt; getCalendarNames()

Get the names of all of the Calendar s in the JobStore.

If there are no Calendars in the given group name, the result should be a zero-length array (not null).

Specified by:
getCalendarNames in interface org.quartz.spi.JobStore

getTriggerKeys

public Set&lt;TriggerKey&gt; getTriggerKeys(GroupMatcher&lt;TriggerKey&gt; match
Get the names of all of the Trigger s that match the given groupMatcher.

**Specified by:**
getTriggerKeys in interface org.quartz.spi.JobStore

---

**getJobGroupNames**

public `List<String>` `getJobGroupNames()`

Get the names of all of the Job groups.

**Specified by:**
getJobGroupNames in interface org.quartz.spi.JobStore

---

**getTriggerGroupNames**

public `List<String>` `getTriggerGroupNames()`

Get the names of all of the Trigger groups.

**Specified by:**
getTriggerGroupNames in interface org.quartz.spi.JobStore

---

**getTriggersForJob**

public `List<org.quartz.spi.OperableTrigger>` `getTriggersForJob(JobKey)`

Get all of the Triggers that are associated to the given Job.

If there are no matches, a zero-length array should be returned.

**Specified by:**
getTriggersForJob in interface org.quartz.spi.JobStore

---

**getTriggerWrappersForJob**
protected `ArrayList<org.quartz.simpl.TriggerWrapper>` `getTriggerWrappersForCalendar`

protected `ArrayList<org.quartz.simpl.TriggerWrapper>` `getTriggerWrappersForCalendar`

---

**pauseTrigger**

public void `pauseTrigger(TriggerKey triggerKey)`

Pause the Trigger with the given name.

**Specified by:**
`pauseTrigger in interface org.quartz.spi.JobStore`

---

**pauseTriggers**

public `List<String> pauseTriggers(GroupMatcher<TriggerKey> matcher)`

Pause all of the known Triggers matching.

The JobStore should "remember" the groups paused, and impose the pause on any new triggers that are added to one of these groups while the group is paused.

**Specified by:**
`pauseTriggers in interface org.quartz.spi.JobStore`

---

**pauseJob**

public void `pauseJob(JobKey jobKey)`

Pause the JobDetail with the given name - by pausing all of its current Triggers.

**Specified by:**
pauseJob in interface org.quartz.spi.JobStore

pauseJobs

public List<String> pauseJobs(GroupMatcher<JobKey> matcher)

Pause all of the JobDetails in the given group - by pausing all of their Triggers.

The JobStore should "remember" that the group is paused, and impose the pause on any new jobs that are added to the group while the group is paused.

Specified by: pauseJobs in interface org.quartz.spi.JobStore

resumeTrigger

public void resumeTrigger(TriggerKey triggerKey)

Resume (un-pause) the Trigger with the given key.

If the Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by: resumeTrigger in interface org.quartz.spi.JobStore

resumeTriggers

public List<String> resumeTriggers(GroupMatcher<TriggerKey> matcher)

Resume (un-pause) all of the Triggers in the given group.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.
Specified by:
resumeTriggers in interface org.quartz.spi.JobStore

**resumeJob**

```java
public void resumeJob(JobKey jobKey)
```

Resume (un-pause) the JobDetail with the given name.

If any of the Job's Triggers missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
resumeJob in interface org.quartz.spi.JobStore

---

**resumeJobs**

```java
public Collection<String> resumeJobs(GroupMatcher<JobKey> matcher)
```

Resume (un-pause) all of the JobDetails in the given group.

If any of the Jobs had Triggers that missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
resumeJobs in interface org.quartz.spi.JobStore

---

**pauseAll**

```java
public void pauseAll()
```

Pause all triggers - equivalent of calling pauseTriggerGroup(group) on every group.

When resumeAll() is called (to un-pause), trigger misfire instructions WILL be applied.
Specified by:
  pauseAll in interface org.quartz.spi.JobStore
See Also:
  #resumeAll(SchedulingContext),
  #pauseTriggerGroup(SchedulingContext, String)

---

**resumeAll**

```java
public void resumeAll()
```

Resume (un-pause) all triggers - equivalent of calling `resumeTriggerGroup(group)` on every group.

If any Trigger missed one or more fire-times, then the Trigger's misfire instruction will be applied.

Specified by:
  resumeAll in interface org.quartz.spi.JobStore
See Also:
  #pauseAll(SchedulingContext)

---

**applyMisfire**

```java
protected boolean applyMisfire(org.quartz.simpl.TriggerWrapper tw)
```

---

**getFiredTriggerRecordId**

```java
protected String getFiredTriggerRecordId()
```

---

**acquireNextTriggers**

```java
public List<org.quartz.spi.OperableTrigger> acquireNextTriggers(
    long noLaterThan,
    int maxCount,
    long timeWindow)
```

Get a handle to the next trigger to be fired, and mark it as 'reserved' by the
calling scheduler.

**Specified by:**
acquireNextTriggers in interface org.quartz.spi.JobStore

**See Also:**
#releaseAcquiredTrigger(SchedulingContext, Trigger)

---

**releaseAcquiredTrigger**

```
public void releaseAcquiredTrigger(org.quartz.spi.OperableTrigger trigger)
```

Inform the JobStore that the scheduler no longer plans to fire the given Trigger, that it had previously acquired (reserved).

**Specified by:**
releaseAcquiredTrigger in interface org.quartz.spi.JobStore

---

**triggersFired**

```
public List<org.quartz.spi.TriggerFiredResult> triggersFired(List<org.quartz.spi.OperableTrigger> triggers)
```

Inform the JobStore that the scheduler is now firing the given Trigger (executing its associated Job), that it had previously acquired (reserved).

**Specified by:**
triggersFired in interface org.quartz.spi.JobStore

---

**triggeredJobComplete**

```
public void triggeredJobComplete(org.quartz.spi.OperableTrigger trigger, JobDetail jobDetail, Trigger.CompletedExecutionInstruction instruction)
```

Inform the JobStore that the scheduler has completed the firing of the given Trigger (and the execution its associated Job), and that the JobDataMap in the given JobDetail should be updated if the Job is stateful.
Specified by:
	triggeredJobComplete in interface org.quartz.spi.JobStore

---

**setAllTriggersOfJobToState**

protected void **setAllTriggersOfJobToState**(JobKey jobKey, int state)

---

**peekTriggers**

protected **String** peekTriggers()

---

**getPausedTriggerGroups**

public **Set** getPausedTriggerGroups()

throws JobPersistenceException

Specified by:
	pauseAllTriggers in interface org.quartz.spi.JobStore

Throws:
	JobPersistenceException

See Also:
	null

---

**setInstanceId**

public void **setInstanceId**(String schedInstId)

Specified by:
	setInstanceId in interface org.quartz.spi.JobStore

---

**setInstanceName**

public void **setInstanceName**(String schedName)
Specified by:
   setInstanceName in interface org.quartz.spi.JobStore

setThreadPoolSize

public void setThreadPoolSize(int poolSize)

Specified by:
   setThreadPoolSize in interface org.quartz.spi.JobStore

getEstimatedTimeToReleaseAndAcquireTrigger

public long getEstimatedTimeToReleaseAndAcquireTrigger()

Specified by:
   getEstimatedTimeToReleaseAndAcquireTrigger in interface org.quartz.spi.JobStore

isClustered

public boolean isClustered()

Specified by:
   isClustered in interface org.quartz.spi.JobStore

Copyright 2001-2011, Terracotta, Inc.
public class SimpleClassLoadHelper
extends Object
implements org.quartz.spi.ClassLoadHelper

A ClassLoadHelper that simply calls Class.forName(..).

Author:
jhouse, pl47ypus

See Also:
ClassLoadHelper, ThreadContextClassLoadHelper,
CascadingClassLoadHelper, LoadingLoaderClassLoadHelper

Constructor Summary

SimpleClassLoadHelper()
itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

### loadClass

```java
public Class loadClass(String name) throws ClassNotFoundException
```

Return the class with the given name.

### Constructor Detail

**SimpleClassLoadHelper**

```java
public SimpleClassLoadHelper()
```

### Method Detail

**initialize**

```java
public void initialize()
```

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

**Specified by:**

`initialize in interface org.quartz.spi.ClassLoadHelper`

**loadClass**

```java
public Class loadClass(String name)
```

Throws `ClassNotFoundException`

Return the class with the given name.

### Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
Specified by:
loadClass in interface org.quartz.spi.ClassLoadHelper

Throws:
ClassNotFoundException

---

**getResource**

public **URL** getResource**(String name)**

Finds a resource with a given name. This method returns null if no resource with this name is found.

Specified by:
getResource in interface org.quartz.spi.ClassLoadHelper

Parameters:
name - name of the desired resource

Returns:
a java.net.URL object

---

**getResourceAsStream**

public **InputStream** getResourceAsStream**(String name)**

Finds a resource with a given name. This method returns null if no resource with this name is found.

Specified by:
getResourceAsStream in interface org.quartz.spi.ClassLoadHelper

Parameters:
name - name of the desired resource

Returns:
a java.io.InputStream object

---

**getClassLoader**

public **ClassLoader** getClassLoader()
Enable sharing of the class-loader with 3rd party.

**Specified by:**
getClassLoader in interface org.quartz.spi.ClassLoadHelper

**Returns:**
the class-loader user be the helper.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Class SimpleInstanceIdGenerator

extends Object
implements org.quartz.spi.InstanceIdGenerator

The default InstanceIdGenerator used by Quartz when instance id is to be automatically generated. Instance id is of the form HOSTNAME + CURRENT_TIME.

See Also:
InstanceIdGenerator, HostnameInstanceIdGenerator

Constructor Summary

| SimpleInstanceIdGenerator() |

Method Summary

| String generateInstanceId() |

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
SimpleInstanceIdGenerator

public SimpleInstanceIdGenerator()

Method Detail

generateInstanceId

public String generateInstanceId() throws SchedulerException

Specified by:
generateInstanceId in interface org.quartz.spi.InstanceIdGenerator

Throws:
SchedulerException

Copyright 2001-2011, Terracotta, Inc.
Class SimpleJobFactory

All Implemented Interfaces:
    org.quartz.spi.JobFactory

Direct Known Subclasses:
    PropertySettingJobFactory

public class SimpleJobFactory
extends Object
implements org.quartz.spi.JobFactory

The default JobFactory used by Quartz - simply calls newInstance() on the job class.

Author:
    jhouse

See Also:
    JobFactory, PropertySettingJobFactory

Constructor Summary

SimpleJobFactory()  

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getLog()</td>
<td>protected org.slf4j.Logger</td>
</tr>
<tr>
<td>newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler Scheduler)</td>
<td></td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

SimpleJobFactory

public SimpleJobFactory()

Method Detail

getLog

protected org.slf4j.Logger getLog()

newJob

public Job newJob(org.quartz.spi.TriggerFiredBundle bundle, Scheduler Scheduler)
throws SchedulerException

Specified by:
newJob in interface org.quartz.spi.JobFactory

Throws:
SchedulerException

Copyright 2001-2011, Terracotta, Inc.
Class SimpleThreadPool

All Implemented Interfaces:
   org.quartz.spi.ThreadPool

public class SimpleThreadPool
extends Object
implements org.quartz.spi.ThreadPool

This is class is a simple implementation of a thread pool, based on the ThreadPool interface.

Runnable objects are sent to the pool with the runInThread(Runnable) method, which blocks until a Thread becomes available.

The pool has a fixed number of Threads, and does not grow or shrink based on demand.

Author:
   James House, Juergen Donnerstag

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SimpleThreadPool()</strong></td>
<td>Create a new (unconfigured) SimpleThreadPool.</td>
</tr>
<tr>
<td><strong>SimpleThreadPool(int threadCount, int threadPriority)</strong></td>
<td>Create a new SimpleThreadPool with the specified number of Threads that have the given priority.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>blockForAvailableThreads()</strong></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>protected void clearFromBusyWorkersList</td>
<td></td>
</tr>
<tr>
<td>protected List&lt;org.quartz.simpl.SimpleThreadPool.WorkerThread&gt;</td>
<td></td>
</tr>
<tr>
<td>createWorkerThreads</td>
<td></td>
</tr>
<tr>
<td>org.slf4j.Logger getLog</td>
<td></td>
</tr>
<tr>
<td>int getPoolSize</td>
<td></td>
</tr>
<tr>
<td>int getThreadPoolCount</td>
<td></td>
</tr>
<tr>
<td>String getThreadNamePrefix</td>
<td>Get the number of worker threads in the pool.</td>
</tr>
<tr>
<td>int getThreadPriority</td>
<td></td>
</tr>
<tr>
<td>void initialize</td>
<td></td>
</tr>
<tr>
<td>boolean isMakeThreadsDaemons</td>
<td></td>
</tr>
<tr>
<td>boolean isThreadsInheritContextClassLoaderOfInitializingThread</td>
<td></td>
</tr>
<tr>
<td>boolean isThreadsInheritGroupOfInitializingThread</td>
<td></td>
</tr>
<tr>
<td>protected void makeAvailable</td>
<td></td>
</tr>
<tr>
<td>boolean runInThread</td>
<td>Run the given Runnable object in the next available thread.</td>
</tr>
<tr>
<td>void setInstanceId</td>
<td></td>
</tr>
<tr>
<td>void setInstanceName</td>
<td></td>
</tr>
<tr>
<td>void setMakeThreadsDaemons</td>
<td></td>
</tr>
<tr>
<td>void setThreadCount</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>void setThreadNamePrefix(String prfx)</td>
<td>Set the thread name prefix of worker threads.</td>
</tr>
<tr>
<td>void setThreadPriority(int prio)</td>
<td>Set the thread priority of worker threads.</td>
</tr>
<tr>
<td>void setThreadsInheritContextClassLoaderOfInitializingThread()</td>
<td>Set whether worker threads inherit the context class loader of the initializing thread.</td>
</tr>
<tr>
<td>void setThreadsInheritGroupOfInitializingThread()</td>
<td>Set whether worker threads inherit the group of the initializing thread.</td>
</tr>
<tr>
<td>void shutdown()</td>
<td>Terminate any worker threads.</td>
</tr>
<tr>
<td>void shutdown(boolean waitForJobsToComplete)</td>
<td>Terminate any worker threads, optionally waiting for them to complete.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SimpleThreadPool

public SimpleThreadPool()

Create a new (unconfigured) SimpleThreadPool.

See Also:
setThreadCount(int), setThreadPriority(int)

SimpleThreadPool

public SimpleThreadPool(int threadCount,
Create a new SimpleThreadPool with the specified number of threads that have the given priority.

**Parameters:**
- threadCount - the number of worker threads in the pool, must be > 0.
- threadPriority - the thread priority for the worker threads.

**See Also:**
- Thread

---

**Method Detail**

**getLog**

```java
public org.slf4j.Logger getLog()
```

**getPoolSize**

```java
public int getPoolSize()
```

**Specified by:**
- getPoolSize in interface org.quartz.spi.ThreadPool

**setThreadCount**

```java
public void setThreadCount(int count)
```

Set the number of worker threads in the pool - has no effect after initialize() has been called.

**getThreadCount**

```java
public int getThreadCount()
```

Get the number of worker threads in the pool.
**setThreadPriority**

```java
public void setThreadPriority(int prio)
```

Set the thread priority of worker threads in the pool - has no effect after `initialize()` has been called.

**getThreadPriority**

```java
public int getThreadPriority()
```

Get the thread priority of worker threads in the pool.

**setThreadNamePrefix**

```java
public void setThreadNamePrefix(String prfx)
```

**getThreadNamePrefix**

```java
public String getThreadNamePrefix()
```

**isThreadsInheritContextClassLoaderOfInitializingThread**

```java
public boolean isThreadsInheritContextClassLoaderOfInitializingThread()
```

Returns:

Returns the `threadsInheritContextClassLoaderOfInitializingThread`.

**setThreadsInheritContextClassLoaderOfInitializingThread**

```java
public void setThreadsInheritContextClassLoaderOfInitializingThread(boolean inheritLoader)
```

Parameters:
inheritLoader - The threadsInheritContextClassLoaderOfInitializingThread to set.

---

**isThreadsInheritGroupOfInitializingThread**

```java
public boolean isThreadsInheritGroupOfInitializingThread()
```

---

**setThreadsInheritGroupOfInitializingThread**

```java
public void setThreadsInheritGroupOfInitializingThread(boolean inheritGroup)
```

---

**isMakeThreadsDaemons**

```java
public boolean isMakeThreadsDaemons()
```

**Returns:**
Returns the value of makeThreadsDaemons.

---

**setMakeThreadsDaemons**

```java
public void setMakeThreadsDaemons(boolean makeThreadsDaemons)
```

**Parameters:**
makeThreadsDaemons - The value of makeThreadsDaemons to set.

---

**setInstanceId**

```java
public void setInstanceId(String schedInstId)
```

**Specified by:**
setInstanceId in interface org.quartz.spi.ThreadPool

---

**setInstanceName**
public void setInstanceName(String schedName)

    Specified by:
    setInstanceName in interface org.quartz.spi.ThreadPool

initialize

public void initialize()
    throws SchedulerConfigException

    Specified by:
    initialize in interface org.quartz.spi.ThreadPool

        Throws:
        SchedulerConfigException

createWorkerThreads

protected List<org.quartz.simpl.SimpleThreadPool.WorkerThread> create

shutdown

public void shutdown()

    Terminate any worker threads in this thread group.
    Jobs currently in progress will complete.

shutdown

public void shutdown(boolean waitForJobsToComplete)

    Terminate any worker threads in this thread group.
    Jobs currently in progress will complete.

    Specified by:
shutdown in interface org.quartz.spi.ThreadPool

runInThread

public boolean runInThread(Runnable runnable)

Run the given Runnable object in the next available Thread. If while waiting the thread pool is asked to shut down, the Runnable is executed immediately within a new additional thread.

Specified by:
runInThread in interface org.quartz.spi.ThreadPool

Parameters:
runnable - the Runnable to be added.

blockForAvailableThreads

public int blockForAvailableThreads()

Specified by:
blockForAvailableThreads in interface org.quartz.spi.ThreadPool

makeAvailable

protected void makeAvailable(org.quartz.simpl.SimpleThreadPool.WorkerThread wt)

clearFromBusyWorkersList

protected void clearFromBusyWorkersList(org.quartz.simpl.SimpleThreadPool.WorkerThread wt)
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAME</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>
Class SimpleTimeBroker

extends Object
implements org.quartz.spi.TimeBroker

The interface to be implemented by classes that want to provide a mechanism by which the QuartzScheduler can reliably determine the current time.

In general, the default implementation of this interface (SimpleTimeBroker—which simply uses System.currentTimeMillis()) is sufficient. However, situations may exist where this default scheme is lacking in its robustness—especially when Quartz is used in a clustered configuration. For example, if one or more of the machines in the cluster has a system time that varies by more than a few seconds from the clocks on the other systems in the cluster, scheduling confusion will result.

Author:
James House

See Also:
QuartzScheduler

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleTimeBroker()</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date getCurrentTime()</td>
</tr>
</tbody>
</table>
Get the current time, simply using `new Date()`.

```java
void initialize()
void shutdown()
```

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

**Constructor Detail**

**SimpleTimeBroker**

```java
public SimpleTimeBroker()
```

**Method Detail**

**getCurrentTime**

```java
public Date getCurrentTime()
```

Get the current time, simply using `new Date()`.

**Specified by:**

`getCurrentTime` in interface `org.quartz.spi.TimeBroker`

**initialize**

```java
public void initialize()
```

**Specified by:**

`initialize` in interface `org.quartz.spi.TimeBroker`

**Throws:**
public void shutdown()

    Specified by:
    shutdown in interface org.quartz.spi.TimeBroker
Class `SystemPropertyInstanceIdGenerator`

- extends `Object`
- implements `org.quartz.spi.InstanceIdGenerator`

All Implemented Interfaces:
- `org.quartz.spi.InstanceIdGenerator`

---

### Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String</code></td>
<td><code>SYSTEM_PROPERTY</code></td>
<td>System property to read the instanceID from</td>
</tr>
</tbody>
</table>

### Constructor Summary

- `SystemPropertyInstanceIdGenerator()`

### Method Summary

- `String generateInstanceId()`
  - Returns the cluster wide value for this scheduler instance's id, based on a system property

Methods inherited from class `java.lang.Object`
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait |

### Field Detail

**SYSTEM_PROPERTY**

```java
public static final String SYSTEM_PROPERTY
```

System property to read the instanceId from

**See Also:**
- [Constant Field Values](#)

### Constructor Detail

**SystemPropertyInstanceIdGenerator**

```java
public SystemPropertyInstanceIdGenerator()
```

### Method Detail

**generateInstanceId**

```java
public String generateInstanceId() throws SchedulerException
```

Returns the cluster wide value for this scheduler instance's id, based on a system property

**Specified by:**
- `generateInstanceId` in interface `org.quartz.spi.InstanceIdGenerator`

**Returns:**
- the value of the `system_property`

**Throws:**
- `SchedulerException` - Shouldn't a value be found
Copyright 2001-2011, Terracotta, Inc.
public class **ThreadContextClassLoadHelper**

extends **Object**
implements org.quartz.spi.ClassLoadHelper

A ClassLoadHelper that uses either the current thread's context class loader (Thread.currentThread().getContextClassLoader().loadClass( .. )).

**Author:**

jhous, pl47ypus

**See Also:**

ClassLoadHelper, InitThreadContextClassLoadHelper, SimpleClassLoadHelper, CascadingClassLoadHelper, LoadingLoaderClassLoadHelper

---

### Constructor Summary

<table>
<thead>
<tr>
<th>ThreadContextClassLoadHelper()</th>
</tr>
</thead>
</table>

---

### Method Summary

<table>
<thead>
<tr>
<th>ClassLoader</th>
<th>getClassLoader()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enable sharing of the class-loader with 3rd party.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>URL</th>
<th>getResource(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Finds a resource with a given name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InputStream</th>
<th>getResourceAsStream(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Finds a resource with a given name.</td>
</tr>
</tbody>
</table>
void initialize()

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

Class loadClass(String name)

Return the class with the given name.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ThreadContextClassLoadHelper

public ThreadContextClassLoadHelper()

Method Detail

initialize

public void initialize()

Called to give the ClassLoadHelper a chance to initialize itself, including the opportunity to "steal" the class loader off of the calling thread, which is the thread that is initializing Quartz.

Specified by:
initialize in interface org.quartz.spi.ClassLoadHelper

loadClass

public Class loadClass(String name)
throws ClassNotFoundException
Return the class with the given name.

**Specified by:**
loadClass in interface org.quartz.spi.ClassLoadHelper

**Throws:**
ClassNotFoundException

---

**getResource**

```java
public URL getResource(String name)
```

Finds a resource with a given name. This method returns null if no resource with this name is found.

**Specified by:**
getResource in interface org.quartz.spi.ClassLoadHelper

**Parameters:**
- name - name of the desired resource

**Returns:**
a java.net.URL object

---

**getResourceAsStream**

```java
public InputStream getResourceAsStream(String name)
```

Finds a resource with a given name. This method returns null if no resource with this name is found.

**Specified by:**
generateResourceAsStream in interface org.quartz.spi.ClassLoadHelper

**Parameters:**
- name - name of the desired resource

**Returns:**
a java.io.InputStream object
public ClassLoader getClassLoader()

Enable sharing of the class-loader with 3rd party.

**Specified by:**
getClassLoader in interface org.quartz.spi.ClassLoadHelper

**Returns:**
the class-loader user be the helper.
org.quartz.simpl Class ZeroSizeThreadPool

java.lang.Object
   └ org.quartz.simpl.ZeroSizeThreadPool

All Implemented Interfaces:
   org.quartz.spi.ThreadPool

public class ZeroSizeThreadPool

extends Object
implements org.quartz.spi.ThreadPool

This is class is a simple implementation of a zero size thread pool, based on the ThreadPool interface.

The pool has zero Threads and does not grow or shrink based on demand. Which means it is obviously not useful for most scenarios. When it may be useful is to prevent creating any worker threads at all - which may be desirable for the sole purpose of preserving system resources in the case where the scheduler instance only exists in order to schedule jobs, but which will never execute jobs (e.g. will never have start() called on it).

Author:
   Wayne Fay

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroSizeThreadPool()</td>
<td>Create a new ZeroSizeThreadPool.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int blockForAvailableThreads()</td>
<td></td>
</tr>
<tr>
<td>org.slf4j.Logger getLog()</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>int getPoolSize()</code></td>
<td></td>
</tr>
<tr>
<td><code>void initialize()</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean runInThread(Runnable runnable)</code></td>
<td></td>
</tr>
<tr>
<td><code>void setInstanceId(String schedInstId)</code></td>
<td></td>
</tr>
<tr>
<td><code>void setInstanceName(String schedName)</code></td>
<td></td>
</tr>
<tr>
<td><code>void shutdown()</code></td>
<td></td>
</tr>
<tr>
<td><code>void shutdown(boolean waitForJobsToComplete)</code></td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.** **Object**

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

**Constructor Detail**

**ZeroSizeThreadPool**

```java
public ZeroSizeThreadPool()
```

Create a new ZeroSizeThreadPool.

**Method Detail**

**getLog**

```java
public org.slf4j.Logger getLog()
```
getPoolSize
public int getPoolSize()

Specified by:
getPoolSize in interface org.quartz.spi.ThreadPool

initialize
public void initialize()
    throws SchedulerConfigException

Specified by:
initialize in interface org.quartz.spi.ThreadPool
Throws:
SchedulerConfigException

shutdown
public void shutdown()

shutdown
public void shutdown(boolean waitForJobsToComplete)

Specified by:
shutdown in interface org.quartz.spi.ThreadPool

runInThread
public boolean runInThread(Runnable runnable)

Specified by:
rInThread in interface org.quartz.spi.ThreadPool
public int blockForAvailableThreads()

    Specified by:
    blockForAvailableThreads in interface org.quartz.spi.ThreadPool

setInstanceId

public void setInstanceId(String schedInstId)

    Specified by:
    setInstanceId in interface org.quartz.spi.ThreadPool

setInstanceName

public void setInstanceName(String schedName)

    Specified by:
    setInstanceName in interface org.quartz.spi.ThreadPool

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.simpl.CascadingClassLoadHelper**

No usage of org.quartz.simpl.CascadingClassLoadHelper

---

<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Class</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV  NEXT
FRAMES   NO FRAMES

Copyright 2001-2011, [Terracotta, Inc.](https://www.terracotta.com)
Uses of Class
org.quartz.simpl.HostnameInstanceIdGenerator

No usage of org.quartz.simpl.HostnameInstanceIdGenerator

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.simpl.InitThreadContextClassLoadHelper

No usage of org.quartz.simpl.InitThreadContextClassLoadHelper

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.simpl.LoadingLoaderClassLoadHelper

No usage of org.quartz.simpl.LoadingLoaderClassLoadHelper

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.simpl.PropertySettingJobFactory

No usage of org.quartz.simpl.PropertySettingJobFactory

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.simpl.RAMJobStore

No usage of org.quartz.simpl.RAMJobStore

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.simpl.SimpleClassLoadHelper

No usage of org.quartz.simpl.SimpleClassLoadHelper

Copyright 2001-2011, Terracotta, Inc.
Uses of Class org.quartz.simpl.SimpleInstanceIdGenerator

No usage of org.quartz.simpl.SimpleInstanceIdGenerator
# Uses of Class

**org.quartz.simpl.SimpleJobFactory**

## Packages that use SimpleJobFactory

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.simpl</td>
<td>Contains simple / light-weight implementations (with no dependencies on external libraries) of interfaces required by the org.quartz.core.QuartzScheduler.</td>
</tr>
</tbody>
</table>

## Uses of SimpleJobFactory in org.quartz.simpl

### Subclasses of SimpleJobFactory in org.quartz.simpl

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertySettingJobFactory</td>
<td>A JobFactory that instantiates the Job instance (using the default no-arg constructor, or more specifically: class.newInstance()), and then attempts to set all values in the JobExecutionContext's JobDataMap onto bean properties of the Job.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Uses of Class
org.quartz.simpl.SimpleThreadPool

No usage of org.quartz.simpl.SimpleThreadPool

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.simpl.SimpleTimeBroker

No usage of org.quartz.simpl.SimpleTimeBroker
Uses of Class
org.quartz.simpl.SystemPropertyInstanceIdGenerator

No usage of org.quartz.simpl.SystemPropertyInstanceIdGenerator

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.simpl.ThreadContextClassLoadHelper

No usage of org.quartz.simpl.ThreadContextClassLoadHelper
Uses of Class
org.quartz.simpl.ZeroSizeThreadPool

No usage of org.quartz.simpl.ZeroSizeThreadPool

Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils **Class CircularLossyQueue**<T>

```java
java.lang.Object
    | org.quartz.utils.CircularLossyQueue<T>
```

**Type Parameters:**

T - Type of the item's to add in this queue

---

public class **CircularLossyQueue**<T>

extends **Object**

An implementation of a CircularQueue data-structure. When the number of items added exceeds the maximum capacity, items that were added first are lost.

**Since:**

1.7

**Author:**

Abhishek Sanoujam

---

### Constructor Summary

**CircularLossyQueue**<T>(int size)

Constructs the circular queue with the specified capacity

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>depth()</td>
</tr>
<tr>
<td></td>
<td>Returns the number of items currently in the queue</td>
</tr>
<tr>
<td>boolean</td>
<td>isEmpty()</td>
</tr>
<tr>
<td></td>
<td>Returns true if the queue is empty, otherwise false</td>
</tr>
<tr>
<td>T</td>
<td>peek()</td>
</tr>
<tr>
<td></td>
<td>Returns value at the tail of the queue</td>
</tr>
<tr>
<td>void</td>
<td>push(T newVal)</td>
</tr>
<tr>
<td></td>
<td>Adds a new item</td>
</tr>
</tbody>
</table>
**toArray(T[] type)**

Returns an array of the current elements in the queue.

<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</td>
</tr>
</tbody>
</table>

## Constructor Detail

### CircularLossyQueue

```java
public CircularLossyQueue(int size)
```

Constructs the circular queue with the specified capacity

**Parameters:**

- **size** -

## Method Detail

### push

```java
public void push(T newVal)
```

Adds a new item

**Parameters:**

- **newVal** -

### toArray

```java
public T[] toArray(T[] type)
```

Returns an array of the current elements in the queue. The order of elements is in reverse order of the order items were added.
Parameters:
  type -

Returns:
An array containing the current elements in the queue. The first element of the array is the tail of the queue and the last element is the head of the queue.

---

peek

```java
public T peek()
```

Returns value at the tail of the queue

Returns:
Value at the tail of the queue

---

isEmpty

```java
public boolean isEmpty()
```

Returns true if the queue is empty, otherwise false

Returns:
true if the queue is empty, false otherwise

---

depth

```java
public int depth()
```

Returns the number of items currently in the queue

Returns:
the number of items in the queue
Copyright 2001-2011, Terracotta, Inc.
public class `ClassUtils` extends `Object`

### Constructor Summary

<table>
<thead>
<tr>
<th>ClassUtils()</th>
</tr>
</thead>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>isAnnotationPresent(Class&lt;?&gt; clazz, Class&lt;? extends Annotation&gt; a)</th>
</tr>
</thead>
</table>

### Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Constructor Detail

**ClassUtils**

**public ClassUtils()**

### Method Detail

**isAnnotationPresent**
public static boolean isAnnotationPresent(Class<?> clazz, Class<?> extends Annotation)
public interface ConnectionProvider

Implementations of this interface used by DBConnectionManager to provide connections from various sources.

Author:
Mohammad Rezaei

See Also:
DBConnectionManager, PoolingConnectionProvider, JNDIConnectionProvider, WeblogicConnectionProvider

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getConnection()</td>
<td>Returns: connection managed by this provider</td>
</tr>
<tr>
<td>shutdown()</td>
<td>Throws:</td>
</tr>
</tbody>
</table>

Method Detail

getConnection

getConnection() throws SQLException

Returns:
connection managed by this provider

Throws:
SQLException

shutdown

void shutdown() throws SQLException

Throws:
SQLException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
org.quartz.utils Class DBConnectionManager

java.lang.Object
└ org.quartz.utils.DBConnectionManager

public class DBConnectionManager
extends Object

Manages a collection of ConnectionProviders, and provides transparent access to their connections.

Author:
    James House, Sharada Jambula, Mohammad Rezaei
Se Also:
    ConnectionProvider, PoolingConnectionProvider, JNDIConnectionProvider, WeblogicConnectionProvider

Field Summary

<table>
<thead>
<tr>
<th>static String DB_PROPS_PREFIX</th>
</tr>
</thead>
</table>

Method Summary

void addConnectionProvider(String dataSourceName, ConnectionProvider provider)

Connection getConnection(String dsName)
    Get a database connection from the DataSource with the given name.

static DBConnectionManager getInstance()
    Get the class instance.

void shutdown(String dsName)
    Shuts down database connections from the DataSource with the given name, if applicable for the
underlying provider.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Field Detail

DB_PROPS_PREFIX

public static final String DB_PROPS_PREFIX

See Also:
Constant Field Values

Method Detail

addConnectionProvider

public void addConnectionProvider(String dataSourceName, ConnectionProvider provider)

getConnection

public Connection getConnection(String dsName)
throws SQLException

Get a database connection from the DataSource with the given name.

Returns:
a database connection

Throws:
SQLException - if an error occurs, or there is no DataSource with the given name.
**getInstance**

```java
public static DBCconnectionManager getInstance()

Get the class instance.

**Returns:**

an instance of this class
```

**shutdown**

```java
public void shutdown(String dsName)
throws SQLException

Shuts down database connections from the DataSource with the given name, if applicable for the underlying provider.

**Throws:**

SQLException - if an error occurs, or there is no DataSource with the given name.
org.quartz.utils **Class DirtyFlagMap**

```
java.lang.Object
   └ org.quartz.utils.DirtyFlagMap
```

**All Implemented Interfaces:**
- Serializable, Cloneable, Map

**Direct Known Subclasses:**
- StringKeyDirtyFlagMap

```java
class DirtyFlagMap
extends Object
implements Map, Cloneable, Serializable
```

An implementation of Map that wraps another Map and flags itself 'dirty' when it is modified.

**Author:**
James House

**See Also:**
Serialized Form

---

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested classes/interfaces inherited from interface java.util.Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map.Entry&lt;K, V&gt;</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>DirtyFlagMap()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a DirtyFlagMap that 'wraps' a HashMap.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DirtyFlagMap(int initialCapacity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a DirtyFlagMap that 'wraps' a HashMap that has the given initial</td>
</tr>
</tbody>
</table>
DirtyFlagMap(int initialCapacity, float loadFactor)

Create a DirtyFlagMap that 'wraps' a HashMap that has the given initial capacity and load factor.

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clear()</td>
<td>Clear the 'dirty' flag (set dirty flag to false).</td>
</tr>
<tr>
<td>void clearDirtyFlag()</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>Object clone()</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>boolean containsKey(Object key)</td>
<td>Determine whether the Map is flagged dirty.</td>
</tr>
<tr>
<td>boolean containsValue(Object val)</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>Set entrySet()</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>boolean equals(Object obj)</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>Object get(Object key)</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>Map getWrappedMap()</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>int hashCode()</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>boolean isEmpty()</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>Set keySet()</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
<tr>
<td>Object put(Object key, Object val)</td>
<td>Get a direct handle to the underlying Map.</td>
</tr>
</tbody>
</table>
void putAll(Map t)

Object remove(Object key)

int size()

Collection values()

Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DirtyFlagMap

public DirtyFlagMap()

Create a DirtyFlagMap that 'wraps' a HashMap.

See Also:
HashMap

DirtyFlagMap

public DirtyFlagMap(int initialCapacity)

Create a DirtyFlagMap that 'wraps' a HashMap that has the given initial capacity.

See Also:
HashMap

DirtyFlagMap
public DirtyFlagMap(int initialCapacity, float loadFactor)

Create a DirtyFlagMap that 'wraps' a HashMap that has the given initial capacity and load factor.

See Also:
HashMap

Method Detail

clearDirtyFlag

public void clearDirtyFlag()

Clear the 'dirty' flag (set dirty flag to false).

isDirty

public boolean isDirty()

Determine whether the Map is flagged dirty.

getWrappedMap

public Map getWrappedMap()

Get a direct handle to the underlying Map.

clear

public void clear()

Specified by:
clear in interface Map
containsKey

public boolean containsKey(Object key)

    Specified by:
        containsKey in interface Map

containsValue

public boolean containsValue(Object val)

    Specified by:
        containsValue in interface Map

diset

public Set entrySet()

    Specified by:
        entrySet in interface Map

equals

public boolean equals(Object obj)

    Specified by:
        equals in interface Map
        Overrides: equals in class Object

hashCode

public int hashCode()
**hashCode** in interface **Map**

**Overrides:**

**hashCode** in class **Object**

---

**get**

public **Object** get(**Object** key)

**Specified by:**

get in interface **Map**

---

**isEmpty**

public boolean isEmpty()

**Specified by:**

isEmpty in interface **Map**

---

**keySet**

public **Set** keySet()

**Specified by:**

keySet in interface **Map**

---

**put**

public **Object** put(**Object** key, **Object** val)

**Specified by:**

put in interface **Map**

---

**putAll**
public void putAll(Map t)

Specified by:

putAll in interface Map

remove

public Object remove(Object key)

Specified by:

remove in interface Map

size

public int size()

Specified by:

size in interface Map

values

public Collection values()

Specified by:

values in interface Map

clone

public Object clone()

Overrides:

clonet in class Object
Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils Class JNDIConnectionProvider

java.lang.Object  
  org.quartz.utils.JNDIConnectionProvider

All Implemented Interfaces:
   ConnectionProvider

public class JNDIConnectionProvider
  extends Object
  implements ConnectionProvider

A ConnectionProvider that provides connections from a DataSource that is managed by an application server, and made available via JNDI.

Author:
   James House, Sharada Jambula, Mohammad Rezaei, Patrick Lightbody, Srinivas Venkatatarangaiah

See Also:
   DBConnectionManager, ConnectionProvider, PoolingConnectionProvider

---

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNDIConnectionProvider(String jndiUrl, boolean alwaysLookup) Constructor</td>
<td></td>
</tr>
<tr>
<td>JNDIConnectionProvider(String jndiUrl, Properties jndiProps, boolean alwaysLookup) Constructor</td>
<td></td>
</tr>
</tbody>
</table>

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getConnection()</td>
<td></td>
</tr>
<tr>
<td>getLog()</td>
<td></td>
</tr>
</tbody>
</table>
boolean isAlwaysLookup()

void setAlwaysLookup(boolean b)

void shutdown()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JNDIConnectionProvider

public JNDIConnectionProvider(String jndiUrl, boolean alwaysLookup)

Constructor

Parameters:
  jndiUrl - The url for the datasource

JNDIConnectionProvider

public JNDIConnectionProvider(String jndiUrl, Properties jndiProps, boolean alwaysLookup)

Constructor

Parameters:
  jndiUrl - The URL for the DataSource
  jndiProps - The JNDI properties to use when establishing the InitialContext for the lookup of the given URL.
Method Detail

**getLog**

protected org.slf4j.Logger getLog()

**getConnection**

public Connection getConnection() throws SQLException

Specified by:
    getConnection in interface ConnectionProvider
Returns:
    connection managed by this provider
Throws:
    SQLException

**isAlwaysLookup**

public boolean isAlwaysLookup()

**setAlwaysLookup**

public void setAlwaysLookup(boolean b)

**shutdown**

public void shutdown() throws SQLException

Specified by:
    shutdown in interface ConnectionProvider
Throws:
    SQLException
**org.quartz.utils** **Class Key<T>**

```java
java.lang.Object
   └─org.quartz.utils.Key<T>
```

**All Implemented Interfaces:**
- `Serializable`, `Comparable<Key>`

**Direct Known Subclasses:**
- `JobKey`, `TriggerKey`

---

```java
public class Key<T>
    extends Object
    implements Serializable, Comparable<Key>
```

Object representing a job or trigger key.

**Author:**
- Jeffrey Wescott

**See Also:**
- `Serialized Form`

---

**Field Summary**

<table>
<thead>
<tr>
<th>static String DEFAULT_GROUP</th>
</tr>
</thead>
</table>
| The default group for scheduling entities, with the value "DEFAULT".

**Constructor Summary**

```java
Key(String name, String group)
```

Construct a new key with the given name and group.

**Method Summary**
### compareTo(Key o)

### static String createUniqueName(String group)

### boolean equals(Object obj)

### String getGroup()

Get the group portion of the key.

### String getName()

Get the name portion of the key.

### int hashCode()

### String toString()

Return the string representation of the key.

---

### Methods inherited from class java.lang.Object

clone, finalize, getClass, notify, notifyAll, wait, wait, wait

---

### Field Detail

#### DEFAULT_GROUP

public static final String DEFAULT_GROUP

The default group for scheduling entities, with the value "DEFAULT".

See Also:

[Constant Field Values](#)

---

### Constructor Detail

#### Key

public Key(String name,
Construct a new key with the given name and group.

**Parameters:**
- **name** - the name
- **group** - the group

### Method Detail

#### getName

```java
public String getName()
```

Get the name portion of the key.

**Returns:**
- the name

#### getGroup

```java
public String getGroup()
```

Get the group portion of the key.

**Returns:**
- the group

#### toString

```java
public String toString()
```

Return the string representation of the key. The format will be: `<group>.
<name>`.

** Overrides:**
- `toString` in class `Object`
Returns:
the string representation of the key

hashCode

public int hashCode()

Overrides:
hashCode in class Object

equals

public boolean equals(Object obj)

Overrides:
equals in class Object

compareTo

public int compareTo(Key o)

Specified by:
compareTo in interface Comparable<Key>

createUniqueName

public static String createUniqueName(String group)
org.quartz.utils

Interfaces  ConnectionProvider

Classes  CircularLossyQueue  ClassUtils  DBCConnectionManager  DirtyFlagMap  JNDIConnectionProvider  Key  PoolingConnectionProvider  PropertiesParser  StringKeyDirtyFlagMap  UpdateChecker
# Package org.quartz.utils

## Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionProvider</td>
<td>Implementations of this interface used by DBConnectionManager to provide connections from various sources.</td>
</tr>
</tbody>
</table>

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CircularLossyQueue&lt;T&gt;</td>
<td>An implementation of a CircularQueue data-structure.</td>
</tr>
<tr>
<td>ClassUtils</td>
<td></td>
</tr>
<tr>
<td>DBConnectionManager</td>
<td>Manages a collection of ConnectionProviders, and provides transparent access to their connections.</td>
</tr>
<tr>
<td>DirtyFlagMap</td>
<td>An implementation of Map that wraps another Map and flags itself 'dirty' when it is modified.</td>
</tr>
<tr>
<td>JNDIConnectionProvider</td>
<td>A ConnectionProvider that provides connections from a DataSource that is managed by an application server, and made available via JNDI.</td>
</tr>
<tr>
<td>Key&lt;T&gt;</td>
<td>Object representing a job or trigger key.</td>
</tr>
<tr>
<td>PoolingConnectionProvider</td>
<td>A ConnectionProvider implementation that creates its own pool of connections.</td>
</tr>
<tr>
<td>PropertiesParser</td>
<td>This is an utility class used to parse the properties.</td>
</tr>
<tr>
<td>StringKeyDirtyFlagMap</td>
<td>An implementation of Map that wraps another Map and flags itself 'dirty' when it is modified, enforces that all keys are Strings.</td>
</tr>
<tr>
<td>UpdateChecker</td>
<td>Check for updates and alert users if an update is available</td>
</tr>
</tbody>
</table>
Hierarchy For Package org.quartz.utils

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.utils.**CircularLossyQueue**<T>
  - org.quartz.utils.**ClassUtils**
  - org.quartz.utils.**DBConnectionManager**
  - org.quartz.utils.**DirtyFlagMap** (implements java.lang.**Cloneable**, java.util.**Map**<K,V>, java.io.**Serializable**)
    - org.quartz.utils.**StringKeyDirtyFlagMap**
  - org.quartz.utils.**JNDIConnectionProvider** (implements org.quartz.utils.**ConnectionProvider**)
  - org.quartz.utils.**Key**<T> (implements java.lang.**Comparable**<T>, java.io.**Serializable**)
  - org.quartz.utils.**PoolingConnectionProvider** (implements org.quartz.utils.**ConnectionProvider**)
  - org.quartz.utils.**PropertiesParser**
  - java.util.**TimerTask** (implements java.lang.**Runnable**)
    - org.quartz.utils.**UpdateChecker**
Interface Hierarchy

- org.quartz.utils.**ConnectionProvider**

Copyright 2001-2011, Terracotta, Inc.
# Uses of Package

**org.quartz.utils**

## Packages that use **org.quartz.utils**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.quartz</strong></td>
<td>The main package of <em>Quartz</em>, containing the client-side interfaces.</td>
</tr>
<tr>
<td><strong>org.quartz.impl.jdbcjobstore</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.impl.matchers</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.utils</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.utils.counter.sampled</strong></td>
<td></td>
</tr>
<tr>
<td><strong>org.quartz.utils.weblogic</strong></td>
<td></td>
</tr>
</tbody>
</table>

## Classes in **org.quartz.utils** used by **org.quartz**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DirtyFlagMap</strong></td>
<td>An implementation of <em>Map</em> that wraps another <em>Map</em> and flags itself 'dirty' when it is modified.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Object representing a job or trigger key.</td>
</tr>
<tr>
<td><strong>StringKeyDirtyFlagMap</strong></td>
<td>An implementation of <em>Map</em> that wraps another <em>Map</em> and flags itself 'dirty' when it is modified, enforces that all keys are Strings.</td>
</tr>
</tbody>
</table>

## Classes in **org.quartz.utils** used by **org.quartz.impl.jdbcjobstore**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
<td>Object representing a job or trigger key.</td>
</tr>
</tbody>
</table>
### org.quartz.impl.matchers

| Key | Object representing a job or trigger key. |

### Classes in org.quartz.utils used by org.quartz.utils

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConnectionProvider</strong></td>
<td>Implementations of this interface used by DBConnectionManager to provide connections from various sources.</td>
</tr>
<tr>
<td><strong>DBConnectionManager</strong></td>
<td>Manages a collection of ConnectionProviders, and provides transparent access to their connections.</td>
</tr>
<tr>
<td><strong>DirtyFlagMap</strong></td>
<td>An implementation of Map that wraps another Map and flags itself 'dirty' when it is modified.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Object representing a job or trigger key.</td>
</tr>
</tbody>
</table>

### Classes in org.quartz.utils used by org.quartz.utils.counter.sampled

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CircularLossyQueue</strong></td>
<td>An implementation of a CircularQueue data-structure.</td>
</tr>
</tbody>
</table>

### Classes in org.quartz.utils used by org.quartz.utils.weblogic

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConnectionProvider</strong></td>
<td>Implementations of this interface used by DBConnectionManager to provide connections from various sources.</td>
</tr>
</tbody>
</table>
org.quartz.utils Class PoolingConnectionProvider

java.lang.Object
  └ org.quartz.utils.PoolingConnectionProvider

All Implemented Interfaces:
  ConnectionProvider

public class PoolingConnectionProvider
  extends Object
  implements ConnectionProvider

A ConnectionProvider implementation that creates its own pool of connections.

This class uses DBCP, an Apache-Jakarta-Commons product.

Author:
  Sharada Jambula, James House, Mohammad Rezaei

See Also:
  DBConnectionManager, ConnectionProvider

---

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>DB_DRIVER</td>
<td>The JDBC database driver.</td>
</tr>
<tr>
<td>static</td>
<td>DB_IDLE_VALIDATION_SECONDS</td>
<td>The number of seconds between tests of idle connections - only enabled if the validation query property is set.</td>
</tr>
<tr>
<td>static</td>
<td>DB_MAX_CACHED_STATEMENTS_PER_CONNECTION</td>
<td>The maximum number of prepared statements that will be cached per connection in the pool.</td>
</tr>
<tr>
<td>static</td>
<td>DB_MAX_CONNECTIONS</td>
<td>The maximum number of database connections to have in the pool.</td>
</tr>
<tr>
<td>static</td>
<td>DB_PASSWORD</td>
<td></td>
</tr>
</tbody>
</table>
### Constructor Summary

**PoolingConnectionProvider**(*Properties config*)

Create a connection pool using the given properties.

**PoolingConnectionProvider**(*String dbDriver, String dbURL, String dbUser, String dbPassword, int maxConnections, String dbValidationQuery*)

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDataSource()</code></td>
<td>Get the C3PO ComboPooledDataSource created during initialization.</td>
</tr>
<tr>
<td><code>getConnection()</code></td>
<td></td>
</tr>
</tbody>
</table>
Field Detail

DB_DRIVER

public static final String DB_DRIVER

The JDBC database driver.

See Also:
Constant Field Values

DB_URL

public static final String DB_URL

The JDBC database URL.

See Also:
Constant Field Values

DB_USER

public static final String DB_USER

The database user name.

See Also:
Constant Field Values
DB_PASSWORD

public static final String DB_PASSWORD

    The database user password.

See Also:  
    Constant Field Values

DB_MAX_CONNECTIONS

public static final String DB_MAX_CONNECTIONS

    The maximum number of database connections to have in the pool. Default is 10.

See Also:  
    Constant Field Values

DB_MAX_CACHED_STATEMENTS_PER_CONNECTION

public static final String DB_MAX_CACHED_STATEMENTS_PER_CONNECTION

    The maximum number of prepared statements that will be cached per connection in the pool. Depending upon your JDBC Driver this may significantly help performance, or may slightly hinder performance. Default is 120, as Quartz uses over 100 unique statements. 0 disables the feature.

See Also:  
    Constant Field Values

DB_VALIDATION_QUERY

public static final String DB_VALIDATION_QUERY
The database sql query to execute every time a connection is returned to the pool to ensure that it is still valid.

See Also:
Constant Field Values

---

**DB_IDLE_VALIDATION_SECONDS**

```java
public static final String DB_IDLE_VALIDATION_SECONDS
```

The number of seconds between tests of idle connections - only enabled if the validation query property is set. Default is 50 seconds.

See Also:
Constant Field Values

---

**DB_VALIDATE_ON_CHECKOUT**

```java
public static final String DB_VALIDATE_ON_CHECKOUT
```

Whether the database sql query to validate connections should be executed every time a connection is retrieved from the pool to ensure that it is still valid. If false, then validation will occur on check-in. Default is false.

See Also:
Constant Field Values

---

**DEFAULT_DB_MAX_CONNECTIONS**

```java
public static final int DEFAULT_DB_MAX_CONNECTIONS
```

Default maximum number of database connections in the pool.

See Also:
Constant Field Values
DEFAULT_DB_MAX_CACHED_STATEMENTS_PER_CONNECTION

public static final int DEFAULT_DB_MAX_CACHED_STATEMENTS_PER_CONNECTION

Default maximum number of database connections in the pool.

See Also:
Constant Field Values

### Constructor Detail

**PoolingConnectionProvider**

```java
public PoolingConnectionProvider(String dbDriver,
                                 String dbURL,
                                 String dbUser,
                                 String dbPassword,
                                 int maxConnections,
                                 String dbValidationQuery)
throws SQLException,
       SchedulerException
```

**Throws:**

- SQLException
- SchedulerException

**PoolingConnectionProvider**

```java
public PoolingConnectionProvider(Properties config)
throws SchedulerException,
       SQLException
```

Create a connection pool using the given properties.

The properties passed should contain:

- **DB_DRIVER**- The database driver class name
- **DB_URL**- The database URL
- **DB_USER**- The database user
- **DB_PASSWORD**- The database password
- **DB_MAX_CONNECTIONS** - The maximum # connections in the pool, optional
- **DB_VALIDATION_QUERY** - The sql validation query, optional

**Parameters:**
- `config` - configuration properties

**Throws:**
- `SchedulerException`
- `SQLException`

## Method Detail

### getDataSource

```java
protected com.mchange.v2.c3p0.ComboPooledDataSource getDataSource()
```

Get the C3PO ComboPooledDataSource created during initialization.

This can be used to set additional data source properties in a subclass's constructor.

### getConnection

```java
public Connection getConnection()
```

Throws: `SQLException`

**Specified by:**
- `getConnection` in interface `ConnectionProvider`

**Returns:**
- connection managed by this provider

**Throws:**
- `SQLException`

### shutdown

```java
public void shutdown()
```

Throws: `SQLException`
Specified by:
shutdown in interface ConnectionProvider

Throws:
SQLException
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
public class **PropertiesParser**

extends **Object**

This is an utility class used to parse the properties.

**Author:**
James House

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PropertiesParser</strong>*(Properties props)**</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td><strong>getBooleanProperty</strong>*(String name)**</td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td><strong>getBooleanProperty</strong>*(String name, boolean def)**</td>
<td></td>
</tr>
<tr>
<td>byte</td>
<td><strong>getByteProperty</strong>*(String name)**</td>
<td></td>
</tr>
<tr>
<td>byte</td>
<td><strong>getByteProperty</strong>*(String name, byte def)**</td>
<td></td>
</tr>
<tr>
<td>char</td>
<td><strong>getCharProperty</strong>*(String name)**</td>
<td></td>
</tr>
<tr>
<td>char</td>
<td><strong>getCharProperty</strong>*(String name, char def)**</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td><strong>getDoubleProperty</strong>*(String name)**</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Method Name</td>
<td>Parameters</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>double</td>
<td><code>getDoubleProperty</code></td>
<td><code>String name</code>, <code>double def</code></td>
</tr>
<tr>
<td>float</td>
<td><code>getFloatProperty</code></td>
<td><code>String name</code></td>
</tr>
<tr>
<td>float</td>
<td><code>getFloatProperty</code></td>
<td><code>String name</code>, <code>float def</code></td>
</tr>
<tr>
<td>int[]</td>
<td><code>getIntArrayProperty</code></td>
<td><code>String name</code></td>
</tr>
<tr>
<td>int[]</td>
<td><code>getIntArrayProperty</code></td>
<td><code>String name</code>, <code>int[] def</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getIntProperty</code></td>
<td><code>String name</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getIntProperty</code></td>
<td><code>String name</code>, <code>int def</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getLongProperty</code></td>
<td><code>String name</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getLongProperty</code></td>
<td><code>String name</code>, <code>long def</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getPropertyGroup</code></td>
<td><code>String prefix</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getPropertyGroup</code></td>
<td><code>String prefix</code>, <code>boolean stripPrefix</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getPropertyGroup</code></td>
<td><code>String prefix</code>, <code>boolean stripPrefix</code>, <code>String[] excludedPrefixes</code></td>
</tr>
<tr>
<td>String[]</td>
<td><code>getPropertyGroups</code></td>
<td><code>String prefix</code></td>
</tr>
<tr>
<td>short</td>
<td><code>getShortProperty</code></td>
<td><code>String name</code></td>
</tr>
<tr>
<td>short</td>
<td><code>getShortProperty</code></td>
<td><code>String name</code>, <code>short def</code></td>
</tr>
<tr>
<td>String[]</td>
<td><code>getStringArrayProperty</code></td>
<td><code>String name</code></td>
</tr>
<tr>
<td>String[]</td>
<td><code>getStringArrayProperty</code></td>
<td><code>String name</code>, <code>String[] def</code></td>
</tr>
</tbody>
</table>
### String

**getStringProperty(String name)**

Get the trimmed String value of the property with the given name.

**getStringProperty(String name, String def)**

Get the trimmed String value of the property with the given name or the given default value if the value is null or empty after trimming.

### Properties

**getUnderlyingProperties()**

### Methods inherited from class java.lang.Object

*clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait*

### Constructor Detail

PropertiesParser

public PropertiesParser(Properties props)

### Method Detail

**getUnderlyingProperties**

public Properties getUnderlyingProperties()

**getStringProperty**

public String getStringProperty(String name)

Get the trimmed String value of the property with the given name. If the value is the empty String (after trimming), then it returns null.
**getStringProperty**

public `String` `getStringProperty(String name, String def)`

Get the trimmed String value of the property with the given name or the given default value if the value is null or empty after trimming.

**getStringArrayProperty**

public `String[]` `getStringArrayProperty(String name)`

**getStringArrayProperty**

public `String[]` `getStringArrayProperty(String name, String[] def)`

**getBooleanProperty**

public `boolean` `getBooleanProperty(String name)`

**getBooleanProperty**

public `boolean` `getBooleanProperty(String name, boolean def)`

**getByteProperty**

public `byte` `getByteProperty(String name)`

Throws: `NumberFormatException`
**getByteProperty**

```java
public byte getByteProperty(String name,
                             byte def)
    throws NumberFormatException
```

Throws:
- `NumberFormatException`

---

**getCharProperty**

```java
public char getCharProperty(String name)
```

---

**getCharProperty**

```java
public char getCharProperty(String name,
                             char def)
```

---

**getDoubleProperty**

```java
public double getDoubleProperty(String name)
    throws NumberFormatException
```

Throws:
- `NumberFormatException`

---

**getDoubleProperty**

```java
public double getDoubleProperty(String name,
                                 double def)
    throws NumberFormatException
```

Throws:
- `NumberFormatException`
getFloatProperty

public float getFloatProperty(String name)
    throws NumberFormatException

    Throws:
        NumberFormatException

getFloatProperty

public float getFloatProperty(String name, float def)
    throws NumberFormatException

    Throws:
        NumberFormatException

getIntProperty

public int getIntProperty(String name)
    throws NumberFormatException

    Throws:
        NumberFormatException

getIntProperty

public int getIntProperty(String name, int def)
    throws NumberFormatException

    Throws:
        NumberFormatException

getIntArrayProperty

public int[] getIntArrayProperty(String name)
Throws: 

`NumberFormatException`

---

**getIntArrayProperty**

public int[] `getIntArrayProperty`(String name, 
int[] def)

throws `NumberFormatException`

Throws: 

`NumberFormatException`

---

**getLongProperty**

public long `getLongProperty`(String name)

throws `NumberFormatException`

Throws: 

`NumberFormatException`

---

**getLongProperty**

public long `getLongProperty`(String name, 
long def)

throws `NumberFormatException`

Throws: 

`NumberFormatException`

---

**getShortProperty**

public short `getShortProperty`(String name)

throws `NumberFormatException`

Throws:
getShortProperty

```java
public short getShortProperty(String name, short def)
throws NumberFormatException
```

**Throws:**

NumberFormatException

getPropertyGroups

```java
public String[] getPropertyGroups(String prefix)
```

getPropertyGroup

```java
public Properties getPropertyGroup(String prefix)
```

getPropertyGroup

```java
public Properties getPropertyGroup(String prefix, boolean stripPrefix)
```

getPropertyGroup

```java
public Properties getPropertyGroup(String prefix, boolean stripPrefix, String[] excludedPrefixes)
```

Get all properties that start with the given prefix.

**Parameters:**

prefix - The prefix for which to search. If it does not end in a "." then one will be added to it for search purposes.
stripPrefix - Whether to strip off the given prefix in the result's keys.
excludedPrefixes - Optional array of fully qualified prefixes to exclude. For example if prefix is "a.b.c", then excludedPrefixes might be "a.b.c.ignore".

**Returns:**
Group of Properties that start with the given prefix, optionally have that prefix removed, and do not include properties that start with one of the given excluded prefixes.
Class StringKeyDirtyFlagMap

All Implemented Interfaces:
    Serializable, Cloneable, Map

Direct Known Subclasses:
    JobDataMap, SchedulerContext

public class StringKeyDirtyFlagMap
    extends DirtyFlagMap

    An implementation of Map that wraps another Map and flags itself 'dirty' when it is modified, enforces that all keys are Strings.

    All allowsTransientData flag related methods are deprecated as of version 1.6.

See Also:
    Serialized Form

<table>
<thead>
<tr>
<th>Nested Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested classes/interfaces inherited from interface java.util.Map</td>
</tr>
<tr>
<td>Map.Entry&lt;K, V&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringKeyDirtyFlagMap()</td>
</tr>
<tr>
<td>StringKeyDirtyFlagMap(int initialCapacity)</td>
</tr>
</tbody>
</table>
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean containsTransientData()</code></td>
<td><strong>Deprecated.</strong> JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.</td>
</tr>
<tr>
<td><code>boolean equals(Object obj)</code></td>
<td></td>
</tr>
<tr>
<td><code>boolean getAllowsTransientData()</code></td>
<td><strong>Deprecated.</strong> JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.</td>
</tr>
<tr>
<td><code>boolean getBoolean(String key)</code></td>
<td>Retrieve the identified boolean value from the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>char getChar(String key)</code></td>
<td>Retrieve the identified char value from the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>double getDouble(String key)</code></td>
<td>Retrieve the identified double value from the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>float getFloat(String key)</code></td>
<td>Retrieve the identified float value from the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>int getInt(String key)</code></td>
<td>Retrieve the identified int value from the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>String[] getKeys()</code></td>
<td>Get a copy of the Map's String keys in an array of Strings.</td>
</tr>
<tr>
<td><code>long getLong(String key)</code></td>
<td>Retrieve the identified long value from the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>String getString(String key)</code></td>
<td></td>
</tr>
</tbody>
</table>
Retrieve the identified String value from the StringKeyDirtyFlagMap.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hashCode()</code></td>
<td></td>
</tr>
<tr>
<td><code>put(Object key, Object value)</code></td>
<td>Adds the given Object value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>put(String key, boolean value)</code></td>
<td>Adds the given boolean value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>put(String key, char value)</code></td>
<td>Adds the given char value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>put(String key, double value)</code></td>
<td>Adds the given double value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>put(String key, float value)</code></td>
<td>Adds the given float value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>put(String key, int value)</code></td>
<td>Adds the given int value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>put(String key, long value)</code></td>
<td>Adds the given long value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>put(String key, String value)</code></td>
<td>Adds the given String value to the StringKeyDirtyFlagMap.</td>
</tr>
<tr>
<td><code>putAll(Map map)</code></td>
<td>Adds the name-value pairs in the given Map to the StringKeyDirtyFlagMap.</td>
</tr>
</tbody>
</table>

**Deprecated.** JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

**setAllowsTransientData(boolean allowsTransientData)**  
**Deprecated.** JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

**Methods inherited from class org.quartz.utils.DirtyFlagMap**

`clear, clearDirtyFlag, clone, containsKey, containsValue, entrySet, get, getWrappedMap, isDirty, isEmpty, keySet, remove,`
Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

StringKeyDirtyFlagMap

public StringKeyDirtyFlagMap()

StringKeyDirtyFlagMap

public StringKeyDirtyFlagMap(int initialCapacity)

StringKeyDirtyFlagMap

public StringKeyDirtyFlagMap(int initialCapacity, float loadFactor)

Method Detail

equals

public boolean equals(Object obj)

Specified by:
equals in interface Map

Overrides:
equals in class DirtyFlagMap

hashCode
public int hashCode()

Specified by:
hashCode in interface Map

Overrides:
hashCode in class DirtyFlagMap

getKeys

public String[] getKeys()

Get a copy of the Map's String keys in an array of Strings.

setAllowsTransientData

public void setAllowsTransientData(boolean allowsTransientData)

Deprecated. JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

Tell the StringKeyDirtyFlagMap that it should allow non-Serializable values. Enforces that the Map doesn't already include transient data.

getAllowsTransientData

public boolean getAllowsTransientData()

Deprecated. JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

Whether the StringKeyDirtyFlagMap allows non-Serializable values.

containsTransientData
public boolean containsTransientData()

**Deprecated.** JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

Determine whether any values in this Map do not implement Serializable. Always returns false if this Map is flagged to not allow transient data.

---

**removeTransientData**

public void removeTransientData()

**Deprecated.** JDBCJobStores no longer prune out transient data. If you include non-Serializable values in the Map, you will now get an exception when attempting to store it in a database.

Removes any data values in the map that are non-Serializable. Does nothing if this Map does not allow transient data.

---

**putAll**

public void putAll(Map map)

Adds the name-value pairs in the given Map to the StringKeyDirtyFlagMap.

All keys must be Strings.

**Specified by:**

putAll in interface Map

**Overrides:**

putAll in class DirtyFlagMap

---

**put**

public void put(String key,
Adds the given int value to the StringKeyDirtyFlagMap.

public void put(String key, int value)

Adds the given long value to the StringKeyDirtyFlagMap.

public void put(String key, long value)

Adds the given float value to the StringKeyDirtyFlagMap.

public void put(String key, float value)

Adds the given double value to the StringKeyDirtyFlagMap.

public void put(String key, double value)

Adds the given boolean value to the StringKeyDirtyFlagMap.

public void put(String key, boolean value)
char value)

Adds the given char value to the StringKeyDirtyFlagMap.

---

put

```java
public void put(String key, String value)
```

Adds the given String value to the StringKeyDirtyFlagMap.

---

put

```java
public Object put(Object key, Object value)
```

Adds the given Object value to the StringKeyDirtyFlagMap.

**Specified by:**
```
put in interface Map
```

**Overrides:**
```
put in class DirtyFlagMap
```

---

getInt

```java
public int getInt(String key)
```

Retrieve the identified int value from the StringKeyDirtyFlagMap.

**Throws:**
```
ClassCastException - if the identified object is not an Integer.
```

---

getLong

```java
public long getLong(String key)
```
Retrieve the identified long value from the StringKeyDirtyFlagMap.

**Throws:**

`ClassCastException` - if the identified object is not a Long.

---

**getFloat**

```java
public float getFloat(String key)
```

Retrieve the identified float value from the StringKeyDirtyFlagMap.

**Throws:**

`ClassCastException` - if the identified object is not a Float.

---

**getDouble**

```java
public double getDouble(String key)
```

Retrieve the identified double value from the StringKeyDirtyFlagMap.

**Throws:**

`ClassCastException` - if the identified object is not a Double.

---

**getBoolean**

```java
public boolean getBoolean(String key)
```

Retrieve the identified boolean value from the StringKeyDirtyFlagMap.

**Throws:**

`ClassCastException` - if the identified object is not a Boolean.

---

**getChar**

```java
public char getChar(String key)
```
Retrieve the identified char value from the StringKeyDirtyFlagMap.

Throws:
   ClassCastException - if the identified object is not a Character.

getString

public String getString(String key)

Retrieve the identified String value from the StringKeyDirtyFlagMap.

Throws:
   ClassCastException - if the identified object is not a String.

Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils Class UpdateChecker

java.lang.Object
<table>
<thead>
<tr>
<th>java.util.TimerTask</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils.UpdateChecker</td>
</tr>
</tbody>
</table>

All Implemented Interfaces:
Runnable

public class UpdateChecker extends TimerTask

Check for updates and alert users if an update is available

Author:
Hung Huynh

Constructor Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateChecker</td>
<td>()</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>checkForUpdate</td>
<td>()</td>
<td>This method ensures that there will be no exception thrown.</td>
</tr>
<tr>
<td>static void</td>
<td>main</td>
<td>(String[] args)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>run</td>
<td>()</td>
<td>Run the update check</td>
</tr>
</tbody>
</table>

Methods inherited from class java.util.TimerTask
cancel, scheduledExecutionTime
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

UpdateChecker

public UpdateChecker()  

Method Detail

run

class UpdateChecker

void run()

Run the update check

Specified by:
run in interface Runnable
Specified by:
run in class TimerTask

class UpdateChecker

checkForUpdate

public void checkForUpdate()

This method ensures that there will be no exception thrown.

main

public static void main(String[] args)
### Uses of Class

**org.quartz.utils.CircularLossyQueue**

<table>
<thead>
<tr>
<th>Packages that use CircularLossyQueue</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils.counter.sampled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of CircularLossyQueue in org.quartz.utils.counter.sampled</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Fields in org.quartz.utils.counter.sampled declared as CircularLossyQueue</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected CircularLossyQueue&lt;TimeStampedCounterValue&gt; history</td>
</tr>
<tr>
<td>SampledCounterImpl.history The history of this counter</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.ClassUtils

No usage of org.quartz.utils.ClassUtils

Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

org.quartz.utils.ConnectionProvider

## Packages that use ConnectionProvider

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils</td>
</tr>
<tr>
<td>org.quartz.utils.weblogic</td>
</tr>
</tbody>
</table>

## Uses of ConnectionProvider in org.quartz.utils

### Classes in org.quartz.utils that implement ConnectionProvider

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNDIConnectionProvider</td>
<td>A ConnectionProvider that provides connections from a DataSource that is managed by an application server, and made available via JNDI.</td>
</tr>
<tr>
<td>PoolingConnectionProvider</td>
<td>A ConnectionProvider implementation that creates its own pool of connections.</td>
</tr>
</tbody>
</table>

### Methods in org.quartz.utils with parameters of type ConnectionProvider

```java
void DBConnectionManager.addConnectionProvider(String dataSourceName, ConnectionProvider provider)
```

## Uses of ConnectionProvider in org.quartz.utils.weblogic

### Classes in org.quartz.utils.weblogic that implement ConnectionProvider
### WeblogicConnectionProvider

Provides connections via Weblogic's JTS driver.

---

**Overview**  **Package**  **Class**  **Tree**  **Deprecated**  **Index**  **Help**

PREV  NEXT  FRAMES  NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.DBConnectionManager

Packages that use DBConnectionManager
org.quartz.utils

Uses of DBConnectionManager in org.quartz.utils

Methods in org.quartz.utils that return DBConnectionManager

| static DBConnectionManager DBConnectionManager.getInstance() |
| Get the class instance. |

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.DirtyFlagMap

### Packages that use DirtyFlagMap

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.utils</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of DirtyFlagMap in org.quartz

<table>
<thead>
<tr>
<th>Subclasses of DirtyFlagMap in org.quartz</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class JobDataMap</td>
<td>Holds state information for Job instances.</td>
</tr>
<tr>
<td>class SchedulerContext</td>
<td>Holds context/environment data that can be made available to Jobs as they are executed.</td>
</tr>
</tbody>
</table>

### Uses of DirtyFlagMap in org.quartz.utils

<table>
<thead>
<tr>
<th>Subclasses of DirtyFlagMap in org.quartz.utils</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class StringKeyDirtyFlagMap</td>
<td>An implementation of Map that wraps another Map and flags itself 'dirty' when it is modified, enforces that all keys are Strings.</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.JNDIConnectionProvider

No usage of org.quartz.utils.JNDIConnectionProvider

Copyright 2001-2011, Terracotta, Inc.
# Uses of Class

**org.quartz.utils.Key**

## Packages that use Key

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz</td>
<td>The main package of Quartz, containing the client-side interfaces.</td>
</tr>
<tr>
<td>org.quartz.impl.jdbcjobstore</td>
<td></td>
</tr>
<tr>
<td>org.quartz.impl.matchers</td>
<td></td>
</tr>
<tr>
<td>org.quartz.utils</td>
<td></td>
</tr>
</tbody>
</table>

## Uses of Key in org.quartz

## Classes in org.quartz with type parameters of type Key

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matcher&lt;T extends Key&gt;</td>
<td>Matchers can be used in various Scheduler API methods to select the entities that should be operated upon.</td>
</tr>
</tbody>
</table>

## Subclasses of Key in org.quartz

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobKey</td>
<td>Uniquely identifies a JobDetail.</td>
</tr>
<tr>
<td>TriggerKey</td>
<td>Uniquely identifies a Trigger.</td>
</tr>
</tbody>
</table>

## Uses of Key in org.quartz.impl.jdbcjobstore

## Methods in org.quartz.impl.jdbcjobstore that return Key

```java
StdJDBCDelegate.selectTriggerForFireTime(Connection conn, ...
```
Select the trigger that will be fired at the given fire time.

```java
DriverDelegate.selectTriggerForFireTime(Connection conn, long fireTime)
```

Select the trigger that will be fired at the given fire time.

---

**Uses of Key in org.quartz.impl.matchers**

<table>
<thead>
<tr>
<th>Classes in org.quartz.impl.matchers with type parameters of type Key</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong> AndMatcher&lt;T extends Key&gt; Matches using an AND operator on two Matcher operands.</td>
</tr>
<tr>
<td><strong>class</strong> EverythingMatcher&lt;T extends Key&gt; Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td><strong>class</strong> GroupMatcher&lt;T extends Key&gt; Matches on group (ignores name) property of Keys.</td>
</tr>
<tr>
<td><strong>class</strong> KeyMatcher&lt;T extends Key&gt; Matches on the complete key being equal (both name and group).</td>
</tr>
<tr>
<td><strong>class</strong> NameMatcher&lt;T extends Key&gt; Matches on name (ignores group) property of Keys.</td>
</tr>
<tr>
<td><strong>class</strong> NotMatcher&lt;T extends Key&gt; Matches using an NOT operator on another Matcher.</td>
</tr>
<tr>
<td><strong>class</strong> OrMatcher&lt;T extends Key&gt; Matches using an OR operator on two Matcher operands.</td>
</tr>
<tr>
<td><strong>class</strong> StringMatcher&lt;T extends Key&gt; An abstract base class for some types of matchers.</td>
</tr>
</tbody>
</table>

---

**Fields in org.quartz.impl.matchers declared as Key**

```java
protected T KeyMatcher.compareTo
```

---

**Methods in org.quartz.impl.matchers with type parameters of type Key**
AndMatcher.\textbf{and}(Matcher<U> leftOperand, Matcher<U> rightOperand)

Create an AndMatcher that depends upon the result of both of the given matchers.

static
<U extends Key> AndMatcher<U>

KeyMatcher.\textbf{keyEquals}(U compareTo)

Create a KeyMatcher that matches Keys that equal the given key.

static
<U extends Key> KeyMatcher<U>

NotMatcher.\textbf{not}(Matcher<U> operand)

Create a NotMatcher that reverses the result of the given matcher.

static
<U extends Key> NotMatcher<U>

OrMatcher.\textbf{or}(Matcher<U> leftOperand, Matcher<U> rightOperand)

Create an OrMatcher that depends upon the result of at least one of the given matchers.

\textbf{Uses of Key in org.quartz.utils}

\textbf{Methods in org.quartz.utils with parameters of type Key}

\begin{itemize}
\item int \textbf{compareTo}(Key o)
\end{itemize}
Uses of Class

org.quartz.utils.PoolingConnectionProvider

No usage of org.quartz.utils.PoolingConnectionProvider

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.PropertiesParser

No usage of org.quartz.utils.PropertiesParser

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Class</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV</td>
<td>NEXT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.utils.StringKeyDirtyFlagMap

Packages that use StringKeyDirtyFlagMap

org.quartz | The main package of Quartz, containing the client-side interfaces.

Uses of StringKeyDirtyFlagMap in org.quartz

Subclasses of StringKeyDirtyFlagMap in org.quartz

<table>
<thead>
<tr>
<th>class</th>
<th>JobDataMap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Holds state information for Job instances.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
<th>SchedulerContext</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Holds context/environment data that can be made available to Jobs as they are executed.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.UpdateChecker

No usage of org.quartz.utils.UpdateChecker

Copyright 2001-2011, Terracotta, Inc.
Class CounterConfig

extends Object

Config for a simple Counter

Since: 1.8
Author: Abhishek Sanoujam

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CounterConfig(long initialValue)</td>
<td>Creates a config with the initial value</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter createCounter()</td>
<td>Creates and returns a Counter based on the initial value</td>
</tr>
<tr>
<td>long getInitialValue()</td>
<td>Gets the initial value</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Detail

CounterConfig

public CounterConfig(long initialValue)

Creates a config with the initial value

Parameters:

initialValue -

Method Detail

getInitialValue

public final long getInitialValue()

Gets the initial value

Returns:

the initial value of counters created by this config

createCounter

public Counter createCounter()

Creates and returns a Counter based on the initial value

Returns:

The counter created by this config
Class **CounterImpl**

**All Implemented Interfaces:**
- `Serializable`, `Counter`

**Direct Known Subclasses:**
- `SampledCounterImpl`

```java
public class CounterImpl extends Object implements Counter, Serializable
```

A simple counter implementation

**Since:**
1.8

**Author:**
Abhishek Sanoujam

**See Also:**
Serialized Form

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CounterImpl</strong>()</td>
<td>Default Constructor</td>
</tr>
<tr>
<td><strong>CounterImpl</strong>(long initialValue)</td>
<td>Constructor with initial value</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long <strong>decrement</strong>()</td>
<td>Decrement the counter by 1</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>long decrement(long amount)</td>
<td>Decrement the counter by given amount</td>
</tr>
<tr>
<td>long getAndSet(long newValue)</td>
<td>Returns the value of the counter and sets it to the new value</td>
</tr>
<tr>
<td>long getValue()</td>
<td>Gets current value of the counter</td>
</tr>
<tr>
<td>long increment()</td>
<td>Increment the counter by 1</td>
</tr>
<tr>
<td>long increment(long amount)</td>
<td>Increment the counter by given amount</td>
</tr>
<tr>
<td>void setValue(long newValue)</td>
<td>Sets the value of the counter to the supplied value</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

CounterImpl

public CounterImpl()

    Default Constructor


CounterImpl

public CounterImpl(long initialValue)

    Constructor with initial value

Parameters:
    initialValue -
### Method Detail

#### increment

```java
public long increment()
```

Increment the counter by 1

**Specified by:**
`increment` in interface `Counter`

**Returns:**
the value after incrementing

---

#### decrement

```java
public long decrement()
```

Decrement the counter by 1

**Specified by:**
`decrement` in interface `Counter`

**Returns:**
the value after decrementing

---

#### getAndSet

```java
public long getAndSet(long newValue)
```

Returns the value of the counter and sets it to the new value

**Specified by:**
`getAndSet` in interface `Counter`

**Returns:**
Returns the old value

---

#### getValue
public long getValue()

Gets current value of the counter

**Specified by:**
getValue in interface **Counter**

**Returns:**
current value of the counter

---

**increment**

public long increment(long amount)

Increment the counter by given amount

**Specified by:**
increment in interface **Counter**

**Returns:**
the value of the counter after incrementing

---

**decrement**

public long decrement(long amount)

Decrement the counter by given amount

**Specified by:**
decrement in interface **Counter**

**Returns:**
the value of the counter after decrementing

---

**setValue**

public void setValue(long newValue)

Sets the value of the counter to the supplied value
Specified by:

setParameter in interface Counter
public interface CounterManager

A Counter Manager that accepts a config to create counters. Creates counter's based on CounterConfig. This manages the lifecycle of a counter.

Since: 1.8
Author: Abhishek Sanoujam

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter createCounter</td>
<td>Creates a Counter based on the passed config</td>
</tr>
<tr>
<td>void shutdown</td>
<td>Shuts down this counter manager</td>
</tr>
<tr>
<td>void shutdownCounter</td>
<td>Shuts down the counter</td>
</tr>
</tbody>
</table>

### Method Detail

**createCounter**

**Counter createCounter(CounterConfig config)**

Creates a Counter based on the passed config

**Parameters:**

- config -
**Returns:**

The counter created and managed by this CounterManager

### shutdown

```java
void shutdown(boolean killTimer)
```

Shuts down this counter manager

### shutdownCounter

```java
void shutdownCounter(Counter counter)
```

Shuts down the counter

**Parameters:**

- `counter` -
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAME</td>
</tr>
</tbody>
</table>
public class CounterManagerImpl
extends Object
implements CounterManager

An implementation of a CounterManager.

Since: 1.8
Author: Abhishek Sanoujam

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CounterManagerImpl(Timer timer)</td>
<td>Constructor that accepts a timer that will be used for scheduling sampled counter if any is created</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter createCounter(CounterConfig config)</td>
<td>Creates a Counter based on the passed config</td>
</tr>
<tr>
<td>void shutdown(boolean killTimer)</td>
<td>Shuts down this counter manager</td>
</tr>
<tr>
<td>void shutdownCounter(Counter counter)</td>
<td>Shuts down the counter</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
Constructor Detail

CounterManagerImpl

class CounterManagerImpl (Timer timer)

Constructor that accepts a timer that will be used for scheduling sampled counter if any is created

Method Detail

shutdown

public void shutdown (boolean killTimer)

Shuts down this counter manager

Specified by:

shutdown in interface CounterManager

createCounter

public Counter createCounter (CounterConfig config)

Creates a Counter based on the passed config

Specified by:

createCounter in interface CounterManager

Returns:

The counter created and managed by this CounterManager

shutdownCounter
public void shutdownCounter(Counter counter)

Shuts down the counter

**Specified by:**  
shutdownCounter in interface CounterManager
org.quartz.util.counter

Interfaces  
Counter  
CounterManager

Classes  
CounterConfig  
CounterImpl  
CounterManagerImpl
### Package org.quartz.utils.counter

#### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Counter</strong></td>
<td>A simple counter</td>
</tr>
<tr>
<td><strong>CounterManager</strong></td>
<td>A Counter Manager that accepts a config to create counters.</td>
</tr>
</tbody>
</table>

#### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CounterConfig</strong></td>
<td>Config for a simple Counter</td>
</tr>
<tr>
<td><strong>CounterImpl</strong></td>
<td>A simple counter implementation</td>
</tr>
<tr>
<td><strong>CounterManagerImpl</strong></td>
<td>An implementation of a CounterManager.</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Overview  Package  Class  Use  Deprecated  Index  Help
PREV  NEXT  FRAMES  NO FRAMES
Hierarchy For Package org.quartz.utils.counter

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.utils.counter.**CounterConfig**
  - org.quartz.utils.counter.**CounterImpl** (implements org.quartz.utils.counter.**Counter**, java.io.**Serializable**)
  - org.quartz.utils.counter.**CounterManagerImpl** (implements org.quartz.utils.counter.**CounterManager**)

Interface Hierarchy

- org.quartz.utils.counter.Counter
- org.quartz.utils.counter.CounterManager

Copyright 2001-2011, Terracotta, Inc.
# Uses of Package

**org.quartz.utils.counter**

## Packages that use `org.quartz.utils.counter`

<table>
<thead>
<tr>
<th>Package</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz/utils.counter</td>
<td></td>
</tr>
<tr>
<td>org.quartz/utils.counter/sample</td>
<td></td>
</tr>
</tbody>
</table>

## Classes in `org.quartz.utils.counter` used by `org.quartz.utils.counter`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter</td>
<td>A simple counter</td>
</tr>
<tr>
<td>CounterConfig</td>
<td>Config for a simple Counter</td>
</tr>
<tr>
<td>CounterManager</td>
<td>A Counter Manager that accepts a config to create counters.</td>
</tr>
</tbody>
</table>

## Classes in `org.quartz.utils.counter` used by `org.quartz.utils.counter/sample`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter</td>
<td>A simple counter</td>
</tr>
<tr>
<td>CounterConfig</td>
<td>Config for a simple Counter</td>
</tr>
<tr>
<td>CounterImpl</td>
<td>A simple counter implementation</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

### org.quartz.utils.counter.Counter

## Packages that use **Counter**

- org.quartz.utils.counter
- org.quartz.utils.counter.sampled

## Uses of **Counter** in **org.quartz.utils.counter**

## Classes in **org.quartz.utils.counter** that implement **Counter**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CounterImpl</td>
<td>A simple counter implementation</td>
</tr>
</tbody>
</table>

## Methods in **org.quartz.utils.counter** that return **Counter**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CounterConfig.createCounter()</td>
<td>Creates and returns a Counter based on the initial value</td>
</tr>
<tr>
<td>CounterManagerImpl.createCounter(CounterConfig config)</td>
<td>Creates a Counter based on the passed config</td>
</tr>
<tr>
<td>CounterManager.createCounter(CounterConfig config)</td>
<td>Creates a Counter based on the passed config</td>
</tr>
</tbody>
</table>

## Methods in **org.quartz.utils.counter** with parameters of type **Counter**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CounterManagerImpl.shutdownCounter(Counter counter)</td>
<td>Shuts down the counter</td>
</tr>
<tr>
<td>CounterManager.shutdownCounter(Counter counter)</td>
<td>Shuts down the counter</td>
</tr>
</tbody>
</table>
Uses of Counter in org.quartz.utils.counter.sampled

Subinterfaces of Counter in org.quartz.utils.counter.sampled

<table>
<thead>
<tr>
<th>interface</th>
<th>SampledCounter</th>
<th>Interface of a sampled counter -- a counter that keeps sampled values</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface</td>
<td>SampledRateCounter</td>
<td>Interface of a sampled rate counter -- a counter that keeps sampled values of rates</td>
</tr>
</tbody>
</table>

Classes in org.quartz.utils.counter.sampled that implement Counter

<table>
<thead>
<tr>
<th>class</th>
<th>SampledCounterImpl</th>
<th>An implementation of SampledCounter</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>SampledRateCounterImpl</td>
<td>An implementation of SampledRateCounter</td>
</tr>
</tbody>
</table>

Methods in org.quartz.utils.counter.sampled that return Counter

<table>
<thead>
<tr>
<th>Counter</th>
<th>SampledCounterConfig.createCounter()</th>
<th>Creates and returns a Counter based on the initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter</td>
<td>SampledRateCounterConfig.createCounter()</td>
<td>Creates and returns a Counter based on the initial value</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
### Uses of Class

**org.quartz.utils.counter.CounterConfig**

<table>
<thead>
<tr>
<th>Packages that use <strong>CounterConfig</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils.counter</td>
</tr>
<tr>
<td>org.quartz.utils.counter.sampled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of <strong>CounterConfig</strong> in <strong>org.quartz.utils.counter</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Methods in <strong>org.quartz.utils.counter</strong> with parameters of type <strong>CounterConfig</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Counter</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Counter</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of <strong>CounterConfig</strong> in <strong>org.quartz.utils.counter.sampled</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Subclasses of <strong>CounterConfig</strong> in <strong>org.quartz.utils.counter.sampled</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>class</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.counter.CounterImpl

Packages that use CounterImpl

org.quartz.utils.counter.sampled

Uses of CounterImpl in
org.quartz.utils.counter.sampled

<table>
<thead>
<tr>
<th>Subclasses of CounterImpl in org.quartz.utils.counter.sampled</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
</tr>
<tr>
<td>An implementation of SampledCounter</td>
</tr>
<tr>
<td>class</td>
</tr>
<tr>
<td>An implementation of SampledRateCounter</td>
</tr>
</tbody>
</table>

Overview Package Class Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
# Uses of Interface

org.quartz.utils.counter.CounterManager

## Packages that use CounterManager

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils.counter</td>
</tr>
</tbody>
</table>

## Uses of CounterManager in org.quartz.utils.counter

## Classes in org.quartz.utils.counter that implement CounterManager

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>CounterManagerImpl</td>
</tr>
</tbody>
</table>

An implementation of a CounterManager.

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.counter.CounterManagerImpl

No usage of org.quartz.utils.counter.CounterManagerImpl

Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils.counter.sampled

Interfaces
- SampledCounter
- SampledRateCounter

Classes
- SampledCounterConfig
- SampledCounterImpl
- SampledRateCounterConfig
- SampledRateCounterImpl
- TimeStampedCounterValue
## Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampledCounter</td>
<td>Interface of a sampled counter -- a counter that keeps sampled values</td>
</tr>
<tr>
<td>SampledRateCounter</td>
<td>Interface of a sampled rate counter -- a counter that keeps sampled values of rates</td>
</tr>
</tbody>
</table>

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampledCounterConfig</td>
<td>Config for a <a href="#">SampledCounter</a></td>
</tr>
<tr>
<td>SampledCounterImpl</td>
<td>An implementation of <a href="#">SampledCounter</a></td>
</tr>
<tr>
<td>SampledRateCounterConfig</td>
<td>An implementation of <a href="#">SampledCounterConfig</a></td>
</tr>
<tr>
<td>SampledRateCounterImpl</td>
<td>An implementation of <a href="#">SampledRateCounter</a></td>
</tr>
<tr>
<td>TimeStamipedCounterValue</td>
<td>A counter value at a particular time instance</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package
org.quartz.util.counter.sampled

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
- org.quartz.utils.counter.**CounterConfig**
  - org.quartz.utils.counter.sampled.**SampledCounterConfig**
  - org.quartz.utils.counter.sampled.**SampledRateCounterConfig**
- org.quartz.utils.counter.**CounterImpl** (implements org.quartz.utils.counter.**Counter**, java.io.**Serializable**)
  - org.quartz.utils.counter.sampled.**SampledCounterImpl** (implements org.quartz.utils.counter.sampled.**SampledCounter**)
  - org.quartz.utils.counter.sampled.**SampledRateCounterImpl** (implements org.quartz.utils.counter.sampled.**SampledRateCounter**)
- org.quartz.utils.counter.sampled.**TimeStampedCounterValue** (implements java.io.**Serializable**)
Interface Hierarchy

- org.quartz.utils.counter.Counter
  - org.quartz.utils.counter.sampled.SampledCounter
    - org.quartz.utils.counter.sampled.SampledRateCounter

Copyright 2001-2011, Terracotta, Inc.
# Uses of Package org.quartz.utils.counter.sampled

## Packages that use org.quartz.utils.counter.sampled

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils.counter.sampled</td>
<td></td>
</tr>
</tbody>
</table>

## Classes in org.quartz.utils.counter.sampled used by org.quartz.utils.counter.sampled

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampledCounter</td>
<td>Interface of a sampled counter -- a counter that keeps sampled values</td>
</tr>
<tr>
<td>SampledCounterConfig</td>
<td>Config for a SampledCounter</td>
</tr>
<tr>
<td>SampledCounterImpl</td>
<td>An implementation of SampledCounter</td>
</tr>
<tr>
<td>SampledRateCounter</td>
<td>Interface of a sampled rate counter -- a counter that keeps sampled values of rates</td>
</tr>
<tr>
<td>SampledRateCounterConfig</td>
<td>An implementation of SampledRateCounterConfig</td>
</tr>
<tr>
<td>TimeStampedCounterValue</td>
<td>A counter value at a particular time instance</td>
</tr>
</tbody>
</table>

---

Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils.counter.sampled  Interface SampledCounter

All Superinterfaces:
   Counter

All Known Subinterfaces:
   SampledRateCounter

All Known Implementing Classes:
   SampledCounterImpl, SampledRateCounterImpl

public interface SampledCounter
extends Counter

Interface of a sampled counter -- a counter that keeps sampled values

Since:
   1.8
Author:
   Abhishek Sanoujam

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeStampedCounterValue[] getAllSampleValues()</td>
<td>Returns all samples in history</td>
</tr>
<tr>
<td>long getAndReset()</td>
<td>Returns the current value of the counter and resets it to 0</td>
</tr>
<tr>
<td>TimeStampedCounterValue getMostRecentSample()</td>
<td>Returns the most recent sampled value</td>
</tr>
<tr>
<td>void shutdown()</td>
<td>Shutdown this counter</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.quartz.utils.counter.Counter
Method Detail

**shutdown**

```java
void shutdown()
```

Shutdown this counter

**getMostRecentSample**

```java
TimeStampedCounterValue getMostRecentSample()
```

Returns the most recent sampled value

**Returns:**
Value of the most recent sampled value

**getAllSampleValues**

```java
TimeStampedCounterValue[] getAllSampleValues()
```

Returns all samples in history

**Returns:**
An array containing the TimeStampedCounterValue's

**getAndReset**

```java
long getAndReset()
```

Returns the current value of the counter and resets it to 0
Returns:
current value of the counter
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Direct Known Subclasses:
SampledRateCounterConfig

public class SampledCounterConfig
extends CounterConfig

Config for a SampledCounter

Since:
1.7

Author:
Abhishek Sanoujam

Constructor Summary

SampledCounterConfig(int intervalSecs, int historySize,
boolean isResetOnSample, long initialValue)

Make a new timed counter config (duh)

Method Summary

createCounter()

Creates and returns a Counter based on the initial value

getHistorySize()

Returns the history size

getIntervalSecs()

Returns the interval time (seconds)

isResetOnSample()

Returns true if counters created from this config will reset on
Methods inherited from class org.quartz.utils.counter.CounterConfig
getInitialValue

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SampledCounterConfig

public SampledCounterConfig(int intervalSecs,
                          int historySize,
                          boolean isResetOnSample,
                          long initialValue)

Make a new timed counter config (duh)

Parameters:
   intervalSecs - the interval (in seconds) between sampling
   historySize - number of counter samples that will be retained in
                 memory
   isResetOnSample - true if the counter should be reset to 0 upon each
                     sample

Method Detail

getHistorySize

public int getHistorySize()  

Returns the history size

Returns:
The history size

---

getIntervalSecs

public int getIntervalSecs()

Returns the interval time (seconds)

**Returns:**
Interval of the sampling thread in seconds

---

isResetOnSample

public boolean isResetOnSample()

Returns true if counters created from this config will reset on each sample

**Returns:**
true if values are reset to the initial value after each sample

---

createCounter

public Counter createCounter()

Creates and returns a Counter based on the initial value

**Overrides:**
createCounter in class CounterConfig

**Returns:**
The counter created by this config

---

Overview | Package | Use | Tree | Deprecated | Index | Help
---|---|---|---|---|---|---
PREV CLASS | NEXT CLASS | SUMMARY: NESTED | FIELD | CONSTR | METHOD | FRAMES | NO FRAMES | DETAIL: FIELD | CONSTR | METHOD

Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils.counter.sampled Class SampledCounterImpl

java.lang.Object
  ↳ org.quartz.utils.counter.CounterImpl
      ↳ org.quartz.utils.counter.sampled.SampledCounterImpl

All Implemented Interfaces:
    Serializable, Counter, SampledCounter

Direct Known Subclasses:
    SampledRateCounterImpl

public class SampledCounterImpl
    extends CounterImpl
    implements SampledCounter

An implementation of SampledCounter

Since:
    1.7
Author:
    Abhishek Sanoujam
See Also:
    Serialized Form

Field Summary

| protected CircularLossyQueue<TimeStampedCounterValue> history | The history of this counter |
| protected boolean resetOnSample | Should the counter reset on each sample? |

Constructor Summary

SampledCounterImpl(SampledCounterConfig config)
todo GL how many threads is this creating? Constructor accepting a SampledCounterConfig

## Method Summary

<table>
<thead>
<tr>
<th>Class Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TimeStampedCounterValue[]</strong></td>
<td><strong>getAllSampleValues()</strong>&lt;br&gt;Returns all samples in history</td>
</tr>
<tr>
<td><strong>long</strong></td>
<td><strong>getAndReset()</strong>&lt;br&gt;Returns the current value of the counter and resets it to 0</td>
</tr>
<tr>
<td><strong>long</strong></td>
<td><strong>getIntervalMillis()</strong>&lt;br&gt;Returns the sampling thread interval in millis</td>
</tr>
<tr>
<td><strong>TimeStampedCounterValue</strong></td>
<td><strong>getMostRecentSample()</strong>&lt;br&gt;Returns the most recent sampled value</td>
</tr>
<tr>
<td><strong>TimerTask</strong></td>
<td><strong>getTimerTask()</strong>&lt;br&gt;Returns the timer task for this sampled counter</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>shutdown()</strong>&lt;br&gt;Shutdown this counter</td>
</tr>
</tbody>
</table>

## Methods inherited from class org.quartz.utils.counter.CounterImpl

decrement, decrement, getAndSet, getValue, increment, increment, setValue

## Methods inherited from class java.lang.Object

clon, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Methods inherited from interface org.quartz.utils.counter.Counter

decrement, decrement, getAndSet, getValue, increment, increment, setValue

## Field Detail

history
protected final CircularLossyQueue<TimeStampedCounterValue> history

    The history of this counter

resetOnSample

protected final boolean resetOnSample

    Should the counter reset on each sample?

Constructor Detail

SampledCounterImpl

public SampledCounterImpl(SampledCounterConfig config)

todo GL how many threads is this creating? Constructor accepting a SampledCounterConfig

Parameters:
    config -

Method Detail

getMostRecentSample

public TimeStampedCounterValue getMostRecentSample()

    Returns the most recent sampled value

    Specified by:
        getMostRecentSample in interface SampledCounter

    Returns:
        Value of the most recent sampled value

getAllSampleValues
public TimeStampedCounterValue[] getAllSampleValues()

Returns all samples in history

Specified by: getAllSampleValues in interface SampledCounter

Returns:
An array containing the TimeStampedCounterValue's

shutdown

public void shutdown()

Shutdown this counter

Specified by: shutdown in interface SampledCounter

getTimerTask

public TimerTask getTimerTask()

Returns the timer task for this sampled counter

Returns:
the timer task for this sampled counter

getIntervalMillis

public long getIntervalMillis()

Returns the sampling thread interval in millis

Returns:
the sampling thread interval in millis
getAndReset

public long getAndReset()

Returns the current value of the counter and resets it to 0

Specified by:
getAndReset in interface SampledCounter

Returns:
current value of the counter
Interface SampledRateCounter

public interface SampledRateCounter

extends SampledCounter

Interface of a sampled rate counter -- a counter that keeps sampled values of rates

Since:
1.8

Author:
Abhishek Sanoujam

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void decrement(long numerator, long denominator)</td>
<td>Decrements the numerator and denominator by the passed values</td>
</tr>
<tr>
<td>void increment(long numerator, long denominator)</td>
<td>Increments the numerator and denominator by the passed values</td>
</tr>
<tr>
<td>void setDenominatorValue(long newValue)</td>
<td>Sets the value of the denominator to the passed value</td>
</tr>
<tr>
<td>void setNumeratorValue(long newValue)</td>
<td>Sets the value of the numerator to the passed value</td>
</tr>
<tr>
<td>void setValue(long numerator, long denominator)</td>
<td>Sets the values of the numerator and denominator to the passed values</td>
</tr>
</tbody>
</table>

Methods inherited from interface
org.quartz.utils.counter.sampled.**SampledCounter**

**getAllSampleValues, getAndReset, getMostRecentSample, shutdown**

**Methods inherited from interface org.quartz.utils.counter.** **Counter**

decrement, decrement, getAndSet, getValue, increment, increment, setValue

**Method Detail**

**increment**

void **increment**(long numerator,

long denominator)

Increments the numerator and denominator by the passed values

**Parameters:**

- numerator -
- denominator -

**decrement**

void **decrement**(long numerator,

long denominator)

Decrements the numerator and denominator by the passed values

**Parameters:**

- numerator -
- denominator -

**setValue**

void **setValue**(long numerator,

long denominator)
Sets the values of the numerator and denominator to the passed values

**Parameters:**

- numerator
- denominator

---

**setNumeratorValue**

```java
void setNumeratorValue(long newValue)
```

Sets the value of the numerator to the passed value

**Parameters:**

- newValue

---

**setDenominatorValue**

```java
void setDenominatorValue(long newValue)
```

Sets the value of the denominator to the passed value

**Parameters:**

- newValue

---

Copyright 2001-2011, Terracotta, Inc.
public class SampledRateCounterConfig
extends SampledCounterConfig

An implementation of SampledCounterConfig

Since: 1.8
Author: Abhishek Sanoujam

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampledRateCounterConfig(int intervalSecs, int historySize, boolean isResetOnSample)</td>
<td>Constructor accepting the interval time in seconds, history-size and whether counters should reset on each sample or not.</td>
</tr>
<tr>
<td>SampledRateCounterConfig(int intervalSecs, int historySize, boolean isResetOnSample, long initialNumeratorValue, long initialDenominatorValue)</td>
<td>Constructor accepting the interval time in seconds, history-size and whether counters should reset on each sample or not.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter createCounter()</td>
<td>Creates and returns a Counter based on the initial value</td>
</tr>
</tbody>
</table>

Methods inherited from class org.quartz.utils.counter.sampled.SampledCounterConfig
getHistorySize, getIntervalSecs, isResetOnSample

Methods inherited from class org.quartz.utils.counter.CounterConfig
getInitialValue

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SampledRateCounterConfig

code

public SampledRateCounterConfig(int intervalSecs,
                                 int historySize,
                                 boolean isResetOnSample)

Constructor accepting the interval time in seconds, history-size and whether
counters should reset on each sample or not. Initial values of both
numerator and denominator are zeroes

Parameters:
    intervalSecs -
    historySize -
    isResetOnSample -

code

SampledRateCounterConfig

code

public SampledRateCounterConfig(int intervalSecs,
                                 int historySize,
                                 boolean isResetOnSample,
                                 long initialNumeratorValue,
                                 long initialDenominatorValue)

Constructor accepting the interval time in seconds, history-size and whether
counters should reset on each sample or not. Also the initial values for the
numerator and the denominator

**Parameters:**
- intervalSecs-
- historySize-
- isResetOnSample-
- initialNumeratorValue-
- initialDenominatorValue-

## Method Detail

### createCounter

```java
public Counter createCounter()
```

Creates and returns a Counter based on the initial value

**Overrides:**
- `createCounter` in class `SampledCounterConfig`

**Returns:**
The counter created by this config
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAME</td>
<td>NO FRAMES</td>
<td>SUMMARY: NESTED</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Class SampledRateCounterImpl

All Implemented Interfaces:
  Serializable, Counter, SampledCounter, SampledRateCounter

public class SampledRateCounterImpl
  extends SampledCounterImpl
  implements SampledRateCounter

An implementation of SampledRateCounter

Since:
  1.8

Author:
  Abhishek Sanoujam

See Also:
  Serialized Form

Field Summary

Fields inherited from class
org.quartz.utils.counter.sampled.SampledCounterImpl
history, resetOnSample

Constructor Summary

SampledRateCounterImpl(SampledRateCounterConfig config)
  Constructor accepting the config
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long <code>decrement()</code></td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>long <code>decrement</code>(long amount)</td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>void <code>decrement</code>(long numerator, long denominator)</td>
<td>Decr. the numerator and denominator by the passed values</td>
</tr>
<tr>
<td>long <code>getAndReset()</code></td>
<td>Returns the current value and resets it to 0</td>
</tr>
<tr>
<td>long <code>getAndSet</code>(long newValue)</td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>long <code>getMaxValue()</code></td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>long <code>getMinValue()</code></td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>long <code>getValue()</code></td>
<td>Gets current value of the counter</td>
</tr>
<tr>
<td>long <code>increment()</code></td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>long <code>increment</code>(long amount)</td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>void <code>increment</code>(long numerator, long denominator)</td>
<td>Increments the numerator and denominator by the passed values</td>
</tr>
<tr>
<td>void <code>setDenominatorValue</code>(long newValue)</td>
<td>Sets the value of the denominator to the passed value</td>
</tr>
<tr>
<td>void <code>setNumeratorValue</code>(long newValue)</td>
<td>Sets the value of the numerator to the passed value</td>
</tr>
<tr>
<td>void <code>setValue</code>(long newValue)</td>
<td>Thrown by <code>UnsupportedOperationException</code></td>
</tr>
<tr>
<td>void <code>setValue</code>(long numerator, long denominator)</td>
<td>Sets the values of the numerator and denominator to the passed values</td>
</tr>
</tbody>
</table>

## Methods inherited from class
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface org.quartz.utils.counter.sampled.SampledCounter
getAllSampleValues, getMostRecentSample, shutdown

Constructor Detail

SampledRateCounterImpl

public SampledRateCounterImpl(SampledRateCounterConfig config)

Constructor accepting the config

Parameters:
config -

Method Detail

setValue

public void setValue(long numerator,
               long denominator)

Sets the values of the numerator and denominator to the passed values

Specified by:
setValue in interface SampledRateCounter
increment

public void increment(long numerator,
                      long denominator)

Increments the numerator and denominator by the passed values

Specified by:
increment in interface SampledRateCounter

decrement

public void decrement(long numerator,
                      long denominator)

Decrements the numerator and denominator by the passed values

Specified by:
decrement in interface SampledRateCounter

setDenominatorValue

public void setDenominatorValue(long newValue)

Sets the value of the denominator to the passed value

Specified by:
setDenominatorValue in interface SampledRateCounter

setNumeratorValue

public void setNumeratorValue(long newValue)

Sets the value of the numerator to the passed value

Specified by:
setNumeratorValue in interface SampledRateCounter
**getValue**

```java
public long getValue()
```

Gets current value of the counter

**Specified by:**
`getValue` in interface `Counter`

**Overrides:**
`getValue` in class `CounterImpl`

**Returns:**
current value of the counter

---

**getAndReset**

```java
public long getAndReset()
```

Returns the current value of the counter and resets it to 0

**Specified by:**
`getAndReset` in interface `SampledCounter`

**Overrides:**
`getAndReset` in class `SampledCounterImpl`

**Returns:**
current value of the counter

---

**getAndSet**

```java
public long getAndSet(long newValue)
```

throws `UnsupportedOperationException`

**Specified by:**
`getAndSet` in interface `Counter`

**Overrides:**
`getAndSet` in class `CounterImpl`
Returns:
Returns the old value

---

**setValue**

public void **setValue**(long newValue)

throws **UnsupportedOperationException**

**Specified by:**
**setValue** in interface **Counter**

**Overrides:**
**setValue** in class **CounterImpl**

---

**decrement**

public long **decrement**()

throws **UnsupportedOperationException**

**Specified by:**
**decrement** in interface **Counter**

**Overrides:**
**decrement** in class **CounterImpl**

**Returns:**
the value after decrementing

---

**decrement**

public long **decrement**(long amount)

throws **UnsupportedOperationException**

**Specified by:**
**decrement** in interface **Counter**

**Overrides:**
decrement in class CounterImpl

Returns:
the value of the counter after decrementing

getMaxValue

public long getMaxValue()

throws UnsupportedOperationException

getMinValue

public long getMinValue()

throws UnsupportedOperationException

increment

public long increment()

throws UnsupportedOperationException

Specified by:
increment in interface Counter

Overrides:
increment in class CounterImpl

Returns:
the value after incrementing

increment

public long increment(long amount)

throws UnsupportedOperationException
Specified by:
  increment in interface Counter

Overrides:
  increment in class CounterImpl

Returns:
  the value of the counter after incrementing
Class TimeStampedCounterValue

public class TimeStampedCounterValue
    extends Object
    implements Serializable

A counter value at a particular time instance

Since: 1.8
Author: Abhishek Sanoujam
See Also: Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>TimeStampedCounterValue</th>
<th>(long timestamp, long value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor accepting the value of both timestamp and the counter value.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>getCounterValue()</td>
<td>Get the counter value</td>
</tr>
<tr>
<td>long</td>
<td>getTimestamp()</td>
<td>Get value of the timestamp</td>
</tr>
<tr>
<td>String</td>
<td>toString()</td>
<td></td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.**Object**

*clone*, *equals*, *finalize*, *getClass*, *hashCode*, *notify*, *notifyAll*, *wait*, *wait*, *wait*

### Constructor Detail

**TimeStampedCounterValue**

```java
public TimeStampedCounterValue(long timestamp, long value)
```

Constructor accepting the value of both timestamp and the counter value.

**Parameters:**
- timestamp
- value

### Method Detail

**getCounterValue**

```java
public long getCounterValue()
```

Get the counter value

**Returns:**
- The counter value

**getTimestamp**

```java
public long getTimestamp()
```

Get value of the timestamp

**Returns:**
- the timestamp associated with the current value
public String toString()

Overrides:

toString in class Object
Packages that use **SampledCounter**

org.quartz.utils.counter.sampled

---

Uses of **SampledCounter** in

org.quartz.utils.counter.sampled

---

Subinterfaces of **SampledCounter** in org.quartz.utils.counter.sampled

<table>
<thead>
<tr>
<th>interface</th>
<th>SampledRateCounter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interface of a sampled rate counter -- a counter that keeps sampled values of rates</td>
</tr>
</tbody>
</table>

---

Classes in org.quartz.utils.counter.sampled that implement **SampledCounter**

<table>
<thead>
<tr>
<th>class</th>
<th>SampledCounterImpl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An implementation of SampledCounter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
<th>SampledRateCounterImpl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An implementation of SampledRateCounter</td>
</tr>
</tbody>
</table>

---

Overview Package Class Tree Deprecated Index Help

Copyright 2001-2011, [Terracotta, Inc.](https://www.terracotta.com)
Packages that use **SampledCounterConfig**

*org.quartz.utils.counter.sampled*

---

Uses of **SampledCounterConfig** in *org.quartz.utils.counter.sampled*

---

Subclasses of **SampledCounterConfig** in *org.quartz.utils.counter.sampled*

<table>
<thead>
<tr>
<th>class</th>
<th><strong>SampledRateCounterConfig</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An implementation of <strong>SampledCounterConfig</strong></td>
</tr>
</tbody>
</table>

---

Constructors in *org.quartz.utils.counter.sampled* with parameters of type **SampledCounterConfig**

**SampledCounterImpl**

```java
SampledCounterImpl(SampledCounterConfig config)
```

dodo GL how many threads is this creating? Constructor accepting a **SampledCounterConfig**

---

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.org)
# Uses of Class

org.quartz.utils.counter.sampled.SampledCounterImpl

## Packages that use SampledCounterImpl

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils.counter.sampled</td>
</tr>
</tbody>
</table>

## Uses of SampledCounterImpl in org.quartz.utils.counter.sampled

## Subclasses of SampledCounterImpl in org.quartz.utils.counter.sampled

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampledRateCounterImpl</td>
</tr>
</tbody>
</table>

An implementation of SampledRateCounter

---

Overview | Package | Class | Tree | Deprecated | Index | Help

PREV NEXT

FRAMES NO FRAMES

Copyright 2001-2011, Terracotta, Inc.
Uses of Interface
org.quartz.utils.counter.sampled.SampledRateCounter

Packages that use SampledRateCounter
org.quartz.utils.counter.sampled

Uses of SampledRateCounter in
org.quartz.utils.counter.sampled

Classes in org.quartz.utils.counter.sampled that implement SampledRateCounter

<table>
<thead>
<tr>
<th>Class</th>
<th>SampledRateCounterImpl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An implementation of SampledRateCounter</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.counter.sampled.SampledRateCounter

Packages that use SampledRateCounterConfig
org.quartz.utils.counter.sampled

Uses of SampledRateCounterConfig in
org.quartz.utils.counter.sampled

 Constructors in org.quartz.utils.counter.sampled with parameters of type SampledRateCounterConfig
SampledRateCounterImpl(SampledRateCounterConfig config)
Constructor accepting the config

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.counter.sampled.SampledRateCounter

No usage of org.quartz.utils.counter.sampled.SampledRateCounterImpl

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.counter.sampled.TimeStampedCounterValue

## Packages that use TimeStampedCounterValue

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.utils.counter.sampled</td>
</tr>
</tbody>
</table>

## Uses of TimeStampedCounterValue in org.quartz.utils.counter.sampled

## Fields in org.quartz.utils.counter.sampled with type parameters of type TimeStampedCounterValue

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected CircularLossyQueue&lt;TimeStampedCounterValue&gt;</td>
<td>SampledCounterImpl.history</td>
<td>The history of this counter</td>
</tr>
</tbody>
</table>

## Methods in org.quartz.utils.counter.sampled that return TimeStampedCounterValue

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampledCounter.getAllSampleValues()</td>
<td>Returns all samples in history</td>
</tr>
<tr>
<td>SampledCounterImpl.getAllSampleValues()</td>
<td>Returns all samples in history</td>
</tr>
<tr>
<td>SampledCounter.getMostRecentSample()</td>
<td>Returns the most recent sampled value</td>
</tr>
<tr>
<td>SampledCounterImpl.getMostRecentSample()</td>
<td>Returns the most recent sampled value</td>
</tr>
</tbody>
</table>
Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils.weblogic Classes WeblogicConnectionProvider
## Package org.quartz.utils.weblogic

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WeblogicConnectionProvider</td>
<td>Provides connections via Weblogic's JTS driver.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.utils.weblogic

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.quartz.utils.weblogic.**WeblogicConnectionProvider** (implements org.quartz.utils.**ConnectionProvider**)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.utils.weblogic

No usage of org.quartz.utils.weblogic

Copyright 2001-2011, Terracotta, Inc.
org.quartz.utils.weblogic Class WeblogicConnectionProvider

java.lang.Object
   ↳ org.quartz.utils.weblogic.WeblogicConnectionProvider

All Implemented Interfaces:
   ConnectionProvider

public class WeblogicConnectionProvider
   extends Object
   implements ConnectionProvider

Provides connections via Weblogic's JTS driver.

Author:
   Mohammad Rezaei, James House

See Also:
   ConnectionProvider, DBConnectionManager

---

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>WeblogicConnectionProvider(String poolName)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection getConnection()</td>
</tr>
<tr>
<td>void shutdown()</td>
</tr>
</tbody>
</table>

| Methods inherited from class java.lang.Object |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait |
### Constructor Detail

**WeblogicConnectionProvider**

```java
public WeblogicConnectionProvider(String poolName)
```

### Method Detail

**getConnection**

```java
public Connection getConnection() throws SQLException
```

**Specified by:**
- `getConnection` in interface `ConnectionProvider`

**Returns:**
- connection managed by this provider

**Throws:**
- `SQLException`

**shutdown**

```java
public void shutdown() throws SQLException
```

**Specified by:**
- `shutdown` in interface `ConnectionProvider`

**Throws:**
- `SQLException`

---

Copyright 2001-2011, Terracotta, Inc.
Uses of Class
org.quartz.utils.weblogic.WeblogicConnectionProvider

No usage of org.quartz.utils.weblogic.WeblogicConnectionProvider

Copyright 2001-2011, Terracotta, Inc.
org.quartz.xml

Classes XMLSchedulingDataProcessor

Exceptions ValidationException
Package org.quartz.xml

Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLSchedulingDataProcessor</td>
<td>Parses an XML file that declares Jobs and their schedules (Triggers), and processes the related data.</td>
</tr>
</tbody>
</table>

Exception Summary

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValidationException</td>
<td>Reports JobSchedulingDataLoader validation exceptions.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Hierarchy For Package org.quartz.xml

Package Hierarchies:
  All Packages
Class Hierarchy

- java.lang.**Object**
  - java.lang.**Throwable** (implements java.io.**Serializable**)
    - java.lang.**Exception**
    - org.quartz.xml.**ValidationException**
  - org.quartz.xml.**XMLSchedulingDataProcessor** (implements org.xml.sax.**ErrorHandler**)

Copyright 2001-2011, Terracotta, Inc.
Uses of Package
org.quartz.xml

Packages that use org.quartz.xml

<table>
<thead>
<tr>
<th>Package</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.xml</td>
<td></td>
</tr>
</tbody>
</table>

Classes in org.quartz.xml used by org.quartz.xml

<table>
<thead>
<tr>
<th>Class</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValidationException</td>
<td>Reports JobSchedulingDataLoader validation exceptions.</td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
Class `ValidationException`

```java
java.lang.Object
  └ java.lang.Throwable
      └ java.lang.Exception
          └ org.quartz.xml.ValidationException
```

All Implemented Interfaces:
  Serializable

```java
public class ValidationException
  extends Exception
```

Reports JobSchedulingDataLoader validation exceptions.

**Author:**
  Chris Bonham

**See Also:**
  Serialized Form

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ValidationException()</code></td>
<td>Constructor for <code>ValidationException</code>.</td>
</tr>
<tr>
<td><code>ValidationException(Collection&lt;Exception&gt; errors)</code></td>
<td>Constructor for <code>ValidationException</code>.</td>
</tr>
<tr>
<td><code>ValidationException(String message)</code></td>
<td>Constructor for <code>ValidationException</code>.</td>
</tr>
<tr>
<td><code>ValidationException(String message, Collection&lt;Exception&gt; errors)</code></td>
<td>Constructor for <code>ValidationException</code>.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String getMessage()</code></td>
<td>Returns the detail message string.</td>
</tr>
</tbody>
</table>
**getValidationExceptions()**

Returns collection of errors.

<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Throwable</th>
</tr>
</thead>
<tbody>
<tr>
<td>fillInStackTrace, getCause, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait</td>
</tr>
</tbody>
</table>

**Constructor Detail**

**ValidationException**

```java
public ValidationException()
```

Constructor for ValidationException.

**ValidationException**

```java
public ValidationException(String message)
```

Constructor for ValidationException.

**Parameters:**
message - exception message.

**ValidationException**

```java
public ValidationException(Collection<Exception> errors)
```

Constructor for ValidationException.
**Parameters:**

   errors - collection of validation exceptions.

---

**ValidationException**

```java
public ValidationException(String message,
                           Collection<Exception> errors)
```

Constructor for ValidationException.

**Parameters:**

   message - exception message.
   errors - collection of validation exceptions.

---

**Method Detail**

**getValidationExceptions**

```java
public Collection getValidationExceptions()
```

Returns collection of errors.

**Returns:**

   collection of errors.

---

**getMessage**

```java
public String getMessage()
```

Returns the detail message string.

**Overrides:**

   getMessage in class Throwable

**Returns:**

   the detail message string.
Copyright 2001-2011, Terracotta, Inc.
public class XMLSchedulingDataProcessor

eextends Object
implements ErrorHandler

Parses an XML file that declares Jobs and their schedules (Triggers), and processes the related data. The xml document must conform to the format defined in "job_scheduling_data_1_8.xsd" The same instance can be used again and again, however a single instance is not thread-safe.

Since:
Quartz 1.8

Author:
James House, Past contributions from Chris Bonham, Past contributions from pl47ypus

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected org.quartz.spi.ClassLoadHelper</td>
<td>classLoadHelper</td>
</tr>
<tr>
<td>protected static java.text.SimpleDateFormat</td>
<td>dateFormat</td>
</tr>
<tr>
<td>protected List&lt;String&gt;</td>
<td>jobGroupsToDelete</td>
</tr>
<tr>
<td>protected List&lt;String&gt;</td>
<td>jobGroupsToNeverDelete</td>
</tr>
<tr>
<td>protected List&lt;JobKey&gt;</td>
<td>jobsToDelete</td>
</tr>
</tbody>
</table>
protected List<JobDetail> loadedJobs

protected List<Trigger> loadedTriggers

static String QUARTZ_NS

static String QUARTZ_SCHEMA_WEB_URL

static String QUARTZ_SYSTEM_ID_JAR_PREFIX

static String QUARTZ_XML_DEFAULT_FILE_NAME

static String QUARTZ_XSD_PATH_IN_JAR

protected List<String> triggerGroupsToDelete

protected List<String> triggerGroupsToNeverDelete

protected List<TriggerKey> triggersToDelete

protected Collection validationExceptions

protected static String XSD_DATE_FORMAT
    XML Schema dateTime datatype format.

Constructor Summary

XMLSchedulingDataProcessor(org.quartz.spi.ClassLoadHelper clh)
    Constructor for JobSchedulingDataLoader.

Method Summary

void addJobGroupToNeverDelete(String group)
    Add the given group to the list of job groups that will never be deleted by this processor, even if a pre-processing-command
protected void addJobToSchedule(JobDetail job)

void addTriggerGroupToNeverDelete(String group)
    Add the given group to the list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

protected void addTriggerToSchedule(Trigger trigger)

protected void addValidationException(SAXException e)
    Adds a detected validation exception.

protected void clearValidationExceptions()
    Resets the the number of detected validation exceptions.

void error(SAXParseException e)
    ErrorHandler interface.

protected void executePreProcessCommands(Scheduler scheduler)

void fatalError(SAXParseException e)
    ErrorHandler interface.

protected Boolean getBoolean(XPath xpath, String elementName, Document document)

protected InputStream getInputStream(String fileName)
    Returns an InputStream from the fileName as a resource.

List<String> getJobGroupsToNeverDelete()
    Get the (unmodifiable) list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

protected List<JobDetail> getLoadedJobs()
    Returns a List of jobs loaded from the xml file.

protected List<Trigger> getLoadedTriggers()
    Returns a List of triggers loaded from the xml file.

protected String getSystemIdForFileName(String fileName)
    For the given fileName, attempt to expand it to its full path for use as a system id.
getTriggerGroupsToNeverDelete()  
Get the (unmodifiable) list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

protected String getTrimmedToNullString(XPath xpath, String elementName, Node parentNode)

protected URL getURL(String fileName)
Returns an URL from the fileName as a resource.

protected void initDocumentParser()
Initializes the XML parser.

boolean isIgnoreDuplicates()
If true (and OverWriteExistingData is false) then any job/triggers encountered in this file that have names that already exist in the scheduler will be ignored, and no error will be produced.

boolean isOverWriteExistingData()
Whether the existing scheduling data (with same identifiers) will be overwritten.

protected void maybeThrowValidationException()
Throws a ValidationException if the number of validationExceptions detected is greater than zero.

protected void prepForProcessing()

protected void process(InputSource is)

protected void processFile()
Process the xml file in the default location (a file named "quartz_jobs.xml" in the current working directory).

protected void processFile(String fileName)
Process the xml file named fileName.

protected void processFile(String fileName, String systemId)
Process the xml file named fileName with the given system ID.

void processFileAndScheduleJobs(Scheduler sched, boolean overWriteExistingJobs)
Process the xml file in the default location, and schedule all of the jobs defined within it.

```java
void processFileAndScheduleJobs(String fileName, Scheduler sched)
```

Process the xml file in the given location, and schedule all of the jobs defined within it.

```java
void processFileAndScheduleJobs(String fileName, String systemId, Scheduler sched)
```

Process the xml file named `fileName` with the given system ID.

```java
boolean removeJobGroupToNeverDelete(String group)
```

Remove the given group to the list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

```java
boolean removeTriggerGroupToNeverDelete(String group)
```

Remove the given group to the list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

```java
protected Object resolveSchemaSource()
```

```java
protected void scheduleJobs(Scheduler sched)
```

Schedules the given sets of jobs and triggers.

```java
void setIgnoreDuplicates(boolean ignoreDuplicates)
```

If true (and `OverWriteExistingData` is false) then any job/triggers encountered in this file that have names that already exist in the scheduler will be ignored, and no error will be produced.

```java
protected void setOverWriteExistingData(boolean overWriteExistingData)
```

Whether the existing scheduling data (with same identifiers) will be overwritten.

```java
void warning(SAXParseException e)
```

ErrorHandler interface.
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

QUARTZ_NS

public static final String QUARTZ_NS

See Also:
   Constant Field Values

QUARTZ_SCHEMA_WEB_URL

public static final String QUARTZ_SCHEMA_WEB_URL

See Also:
   Constant Field Values

QUARTZ_XSD_PATH_IN_JAR

public static final String QUARTZ_XSD_PATH_IN_JAR

See Also:
   Constant Field Values

QUARTZ_XML_DEFAULT_FILE_NAME

public static final String QUARTZ_XML_DEFAULT_FILE_NAME

See Also:
   Constant Field Values
**QUARTZ_SYSTEM_ID_JAR_PREFIX**

public static final String QUARTZ_SYSTEM_ID_JAR_PREFIX

See Also:
Constant Field Values

---

**XSD_DATE_FORMAT**

protected static final String XSD_DATE_FORMAT

XML Schema dateTime datatype format.

See http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime

See Also:
Constant Field Values

---

**dateFormat**

protected static final SimpleDateFormat dateFormat

---

**jobGroupsToDelete**

protected List<String> jobGroupsToDelete

---

**triggerGroupsToDelete**

protected List<String> triggerGroupsToDelete

---

**jobsToDelete**

protected List<JobKey> jobsToDelete
triggersToDelete
protected List<TriggerKey> triggersToDelete

loadedJobs
protected List<JobDetail> loadedJobs

loadedTriggers
protected List<Trigger> loadedTriggers

validationExceptions
protected Collection validationExceptions

classLoadHelper
protected org.quartz.spi.ClassLoadHelper classLoadHelper

jobGroupsToNeverDelete
protected List<String> jobGroupsToNeverDelete

triggerGroupsToNeverDelete
protected List<String> triggerGroupsToNeverDelete

Constructor Detail

XMLSchedulingDataProcessor
public XMLSchedulingDataProcessor(org.quartz.spi.ClassLoadHelper clh throws ParserConfigurationException

Constructor for JobSchedulingDataLoader.

**Parameters:**
- clh - class-loader helper to share with digester.

**Throws:**
- ParserConfigurationException - if the XML parser cannot be configured as needed.

## Method Detail

### initDocumentParser

protected void initDocumentParser()

throws ParserConfigurationException

Initializes the XML parser.

**Throws:**
- ParserConfigurationException

### resolveSchemaSource

protected Object resolveSchemaSource()

### isOverWriteExistingData

public boolean isOverWriteExistingData()

Whether the existing scheduling data (with same identifiers) will be overwritten. If false, and IgnoreDuplicates is not false, and jobs or triggers with the same names already exist as those in the file, an error will occur.

**See Also:**
- isIgnoreDuplicates()
setOverWriteExistingData

protected void setOverWriteExistingData(boolean overWriteExistingData)

Whether the existing scheduling data (with same identifiers) will be overwritten. If false, and IgnoreDuplicates is not false, and jobs or triggers with the same names already exist as those in the file, an error will occur.

See Also:

setIgnoreDuplicates(boolean)

isIgnoreDuplicates

public boolean isIgnoreDuplicates()

If true (and OverWriteExistingData is false) then any job/triggers encountered in this file that have names that already exist in the scheduler will be ignored, and no error will be produced.

See Also:

isOverWriteExistingData()

setIgnoreDuplicates

public void setIgnoreDuplicates(boolean ignoreDuplicates)

If true (and OverWriteExistingData is false) then any job/triggers encountered in this file that have names that already exist in the scheduler will be ignored, and no error will be produced.

See Also:

setOverWriteExistingData(boolean)

addJobGroupToNeverDelete
public void addJobGroupToNeverDelete(String group)

Add the given group to the list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

Parameters:
  group -

removeJobGroupToNeverDelete

public boolean removeJobGroupToNeverDelete(String group)

Remove the given group to the list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

Parameters:
  group -

getJobGroupsToNeverDelete

public List<String> getJobGroupsToNeverDelete()

Get the (unmodifiable) list of job groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

Parameters:
  group -

addTriggerGroupToNeverDelete

public void addTriggerGroupToNeverDelete(String group)

Add the given group to the list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is
_parameters:
  group -

**removeTriggerGroupToNeverDelete**

```java
public boolean removeTriggerGroupToNeverDelete(String group)
```

Remove the given group to the list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

**Parameters:**
  group -

**getTriggerGroupsToNeverDelete**

```java
public List<String> getTriggerGroupsToNeverDelete()
```

Get the (unmodifiable) list of trigger groups that will never be deleted by this processor, even if a pre-processing-command to delete the group is encountered.

**Parameters:**
  group -

**processFile**

```java
protected void processFile() throws Exception
```

Process the xml file in the default location (a file named "quartz_jobs.xml" in the current working directory).

**Throws:**
  Exception
processFile

protected void processFile(String fileName)
    throws Exception

Process the xml file named fileName.

Parameters:
    fileName - meta data file name.

Throws:
    Exception

getSystemIdForFileName

protected String getSystemIdForFileName(String fileName)

For the given fileName, attempt to expand it to its full path for use as a system id.

See Also:
    getURL(String), processFile(), processFile(String),
    processFileAndScheduleJobs(Scheduler, boolean),
    #processFileAndScheduleJobs(String, Scheduler, boolean)

getURL

protected URL getURL(String fileName)

Returns an URL from the fileName as a resource.

Parameters:
    fileName - file name.

Returns:
    an URL from the fileName as a resource.

prepForProcessing
protected void prepForProcessing()

processFile

protected void processFile(String fileName, String systemId)
throws ValidationException, ParsingConfigurationException, SAXException, IOException, SchedulerException, ClassNotFoundException, ParseException, XPathException

Process the xml file named fileName with the given system ID.

Parameters:
- fileName - meta data file name.
- systemId - system ID.

Throws:
- ValidationException
- ParsingConfigurationException
- SAXException
- IOException
- SchedulerException
- ClassNotFoundException
- ParseException
- XPathException

processStreamAndScheduleJobs

public void processStreamAndScheduleJobs(InputStream stream, String systemId, Scheduler sched)
throws ValidationException, ParsingConfigurationException, SAXException, IOException, SchedulerException, ClassNotFoundException, NoSuchElementException
Process the xml file named `fileName` with the given system ID.

**Parameters:**
- `stream` - an input stream containing the xml content.
- `systemId` - system ID.

**Throws:**
- `ValidationException`
- `ParserConfigurationException`
- `SAXException`
- `XPathException`
- `IOException`
- `SchedulerException`
- `ClassNotFoundException`
- `ParseException`

```java
class ExampleClass {
    protected void process(InputSource is) throws SAXException, IOException, ParseException, XPathException, ClassNotFoundException {
    }
}
```

**Throws:**
- `SAXException`
- `IOException`
- `ParseException`
- `XPathException`
- `ClassNotFoundException`

```java
class ExampleClass {
    protected String getTrimmedToNullString(XPath xpath, String elementName, Node parentNode) throws XPathExpressionException {
    }
}
```

**Throws:**
**getBoolean**

```java
protected Boolean getBoolean(XPath xpath, String elementName, Document document) throws XPathExpressionException
```

**Throws:**
- `XPathExpressionException`

---

**processFileAndScheduleJobs**

```java
public void processFileAndScheduleJobs(Scheduler sched, boolean overwriteExistingJobs) throws SchedulerException, Exception
```

Process the xml file in the default location, and schedule all of the jobs defined within it.

**Throws:**
- `SchedulerException`
- `Exception`

---

**processFileAndScheduleJobs**

```java
public void processFileAndScheduleJobs(String fileName, Scheduler sched) throws Exception
```

Process the xml file in the given location, and schedule all of the jobs defined within it.

**Parameters:**
- `fileName` - meta data file name.

**Throws:**
- `Exception`
processFileAndScheduleJobs

```java
public void processFileAndScheduleJobs(String fileName, String systemId, Scheduler sched)
    throws Exception
```

Process the xml file in the given location, and schedule all of the jobs defined within it.

**Parameters:**
- `fileName` - meta data file name.

**Throws:**
- `Exception`

getLoadedJobs

```java
protected List<JobDetail> getLoadedJobs()
```

Returns a List of jobs loaded from the xml file.

**Returns:**
- a List of jobs.

getLoadedTriggers

```java
protected List<Trigger> getLoadedTriggers()
```

Returns a List of triggers loaded from the xml file.

**Returns:**
- a List of triggers.

gerInputStream

```java
protected InputStream getInputStream(String fileName)
```

getInputStream
Returns an InputStream from the fileName as a resource.

**Parameters:**
fileName - file name.

**Returns:**
an InputStream from the fileName as a resource.

---

**addJobToSchedule**

protected void addJobToSchedule(JobDetail job)

---

**addTriggerToSchedule**

protected void addTriggerToSchedule(Trigger trigger)

---

**executePreProcessCommands**

protected void executePreProcessCommands(Scheduler scheduler)
   throws SchedulerException

**Throws:**
SchedulerException

---

**scheduleJobs**

protected void scheduleJobs(Scheduler sched)
   throws SchedulerException

Schedules the given sets of jobs and triggers.

**Parameters:**
sched - job scheduler.

**Throws:**
SchedulerException - if the Job or Trigger cannot be added to the Scheduler, or there is an internal Scheduler error.
warning

public void warning(SAXParseException e)
    throws SAXException

ErrorHandler interface. Receive notification of a warning.

Specified by:
    warning in interface ErrorHandler
Parameters:
    e - The error information encapsulated in a SAX parse exception.
Throws:
    SAXException - Any SAX exception, possibly wrapping another exception.

error

public void error(SAXParseException e)
    throws SAXException

ErrorHandler interface. Receive notification of a recoverable error.

Specified by:
    error in interface ErrorHandler
Parameters:
    e - The error information encapsulated in a SAX parse exception.
Throws:
    SAXException - Any SAX exception, possibly wrapping another exception.

fatalError

public void fatalError(SAXParseException e)
    throws SAXException

ErrorHandler interface. Receive notification of a non-recoverable error.

Specified by:
fatalError in interface ErrorHandler

Parameters:
   e - The error information encapsulated in a SAX parse exception.

Throws:
   SAXException - Any SAX exception, possibly wrapping another exception.

addToValidations

protected void addToValidations(SAXException e)

Adds a detected validation exception.

Parameters:
   e - SAX exception.

clearValidationExceptions

protected void clearValidationExceptions()

Resets the number of detected validation exceptions.

maybeThrowValidationException

protected void maybeThrowValidationException()

   throws ValidationException

   Throws a ValidationException if the number of validationExceptions detected is greater than zero.

Throws:
   ValidationException - DTD validation exception.
# Uses of Class org.quartz.xml.ValidationException

## Packages that use ValidationException

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.quartz.xml</td>
</tr>
</tbody>
</table>

## Uses of ValidationException in org.quartz.xml

### Methods in org.quartz.xml that throw ValidationException

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protected void XMLSchedulingDataProcessor.maybeThrowValidationException()</code></td>
<td>Throws a ValidationException if the number of validationExceptions is greater than zero.</td>
</tr>
<tr>
<td><code>protected void XMLSchedulingDataProcessor.processFile(String fileName, String systemId)</code></td>
<td>Process the xmlfile named fileName with the given system ID.</td>
</tr>
<tr>
<td><code>void XMLSchedulingDataProcessor.processStreamAndScheduleJobs(InputStream input, String systemId, Scheduler sched)</code></td>
<td>Process the xmlfile named fileName with the given system ID.</td>
</tr>
</tbody>
</table>
Uses of Class
org.quartz.xml.XMLSchedulingDataProcessor

No usage of org.quartz.xml.XMLSchedulingDataProcessor

Copyright 2001-2011, Terracotta, Inc.
Class EhcacheEvaluator

public class EhcacheEvaluator

extends Object

implements Evaluator<EhcacheConstraint>

Ehcache evaluator, which lets Jobs be executed on certain node, depending of the locality of the data in a Ehcache

Author:
Alex Snaps

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EhcacheEvaluator(ConcurrentMap&lt;String, String&gt; nodeIdToInstanceId)</td>
<td>Constructor</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean matches(EhcacheConstraint constraint)</td>
<td>Evaluates the constraint against local node.</td>
</tr>
<tr>
<td>List&lt;String&gt; suitedNodes(EhcacheConstraint constraint)</td>
<td>Returns a list of best suited node (instanceId) based on the constraint</td>
</tr>
<tr>
<td>void verify(EhcacheConstraint constraint)</td>
<td>Doesn't do anything</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
Constructor Detail

EhcacheEvaluator

public EhcacheEvaluator(ConcurrentMap<String, String> nodeIdToInstanceId)

Constructor

Parameters:
nodeIdToInstanceId - the Map of nodeId (Terracotta) to instanceId (Quartz) mapping file (read only)

Method Detail

matches

public boolean matches(EhcacheConstraint constraint)

Evaluates the constraint against local node.

Specified by:
matches in interface Evaluator<EhcacheConstraint>

Parameters:
constraint - The constraint to evaluate

Returns:
true if local node is a match

See Also:
Constraint#matches(Object)

suitedNodes

public List<String> suitedNodes(EhcacheConstraint constraint)
Returns a list of best suited node (instanceId) based on the constraint

**Specified by:**

suitedNodes in interface Evaluator<EhcacheConstraint>

**Parameters:**

constraint - To find best suited nodes for

**Returns:**

List of best suited Quartz instanceId

---

**verify**

public void verify(EhcacheConstraint constraint)

Doesn't do anything

**Specified by:**

verify in interface Evaluator<EhcacheConstraint>

**Parameters:**

constraint - The constraint to validate

---

Copyright 2001-2011, Terracotta, Inc.
# EhcacheEvaluator

Ehcache evaluator, which lets Jobs be executed on certain node, depending on the locality of the data in a Ehcache.

---

Copyright 2001-2011, [Terracotta, Inc.](http://www.terracotta.com)
Hierarchy For Package
org.terracotta.modules.ehcache.store

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
- org.terracotta.modules.ehcache.store.**EhcacheEvaluator** (implements org.quartz.locality.constraint.evaluator.**Evaluator<T>**)

Copyright 2001-2011, **Terracotta, Inc.**
Uses of Package
org.terracotta.modules.ehcache.store

No usage of org.terracotta.modules.ehcache.store
Uses of Class
org.terracotta.modules.ehcache.store.EhcacheEvaluator

No usage of org.terracotta.modules.ehcache.store.EhcacheEvaluator

Copyright 2001-2011, Terracotta, Inc.
Interface Constraint<OP extends Enum,T>

Type Parameters:
  T - The value type the constraint will be evaluated against

All Superinterfaces:
  Serializable

All Known Implementing Classes:
  CpuConstraint, EhcacheConstraint, MemoryConstraint, NodeGroupConstraint, OsConstraint

public interface Constraint<OP extends Enum,T>

extends Serializable

A Constraint about the node on which a job will be executed A Constraint always needs a matching Evaluator type present on the classpath

Author:
  Alex Snaps

See Also:
  Evaluator
Interface Counter

All Known Subinterfaces:
- SampledCounter, SampledRateCounter

All Known Implementing Classes:
- CounterImpl, SampledCounterImpl, SampledRateCounterImpl

public interface Counter

A simple counter

**Since:**
1.8

**Author:**
Abhishek Sanoujam

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long decrement()</td>
<td>Decrement the counter by 1</td>
</tr>
<tr>
<td>long decrement(long amount)</td>
<td>Decrement the counter by given amount</td>
</tr>
<tr>
<td>long getAndSet(long newValue)</td>
<td>Returns the value of the counter and sets it to the new value</td>
</tr>
<tr>
<td>long getValue()</td>
<td>Gets current value of the counter</td>
</tr>
<tr>
<td>long increment()</td>
<td>Increment the counter by 1</td>
</tr>
<tr>
<td>long increment(long amount)</td>
<td>Increment the counter by given amount</td>
</tr>
<tr>
<td>void setValue(long newValue)</td>
<td>Sets the value of the counter to the supplied value</td>
</tr>
</tbody>
</table>
### Method Detail

#### increment

```java
long increment()
```

Increment the counter by 1

**Returns:**
the value after incrementing

#### decrement

```java
long decrement()
```

Decrement the counter by 1

**Returns:**
the value after decrementing

#### getAndSet

```java
long getAndSet(long newValue)
```

Returns the value of the counter and sets it to the new value

**Parameters:**
newValue -

**Returns:**
Returns the old value

#### getValue

```java
long getValue()
```

Gets current value of the counter
Returns:
current value of the counter

increment

long increment(long amount)

Increment the counter by given amount

Parameters:
  amount -

Returns:
the value of the counter after incrementing

decrement

long decrement(long amount)

Decrement the counter by given amount

Parameters:
  amount -

Returns:
the value of the counter after decrementing

setValue

void setValue(long newValue)

Sets the value of the counter to the supplied value

Parameters:
  newValue -
<table>
<thead>
<tr>
<th>SUMMARY: NESTED</th>
<th>FIELD</th>
<th>CONSTR</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2001-2011, Terracotta, Inc.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
</tr>
</thead>
</table>

PREV CLASS  NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
**Evaluator Interface**

`Evaluator<T extends Constraint>`

**Type Parameters:**

- `T`: The constraint this evaluator is uniquely responsible for.

**All Known Subinterfaces:**

- `PersistentEvaluator<T,V>`

**All Known Implementing Classes:**

- `CpuEvaluator`, `EhcacheEvaluator`, `MemoryEvaluator`, `NodeGroupEvaluator`, `OsEvaluator`

---

```java
public interface Evaluator<T extends Constraint>
```

An Evaluator will match and potentially return best suited nodes for a Job to execute on.

**Author:**

Alex Snaps

**See Also:**

`Constraint`

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean matches(T constraint)</code></td>
<td>Evaluates the constraint against local node.</td>
</tr>
<tr>
<td><code>List&lt;String&gt; suitedNodes(T constraint)</code></td>
<td>Returns a list of best suited node (instanceId) based on the constraint.</td>
</tr>
<tr>
<td><code>void verify(T constraint)</code></td>
<td>Verifies the validity of a constraint.</td>
</tr>
</tbody>
</table>

---

### Method Detail
matches

boolean matches(T constraint)

Evaluates the constraint against local node.

**Parameters:**
- constraint - The constraint to evaluate

**Returns:**
- true if local node is a match

**See Also:**
- Constraint#matches(Object)

suitedNodes

`List<String> suitedNodes(T constraint)`

Returns a list of best suited node (instanceId) based on the constraint

**Parameters:**
- constraint - To find best suited nodes for

**Returns:**
- List of best suited Quartz instanceId

verify

`void verify(T constraint)`

throws `LocalityException`

Verifies the validity of a constraint. Throws exception should the constraint fail to validate.

**Parameters:**
- constraint - The constraint to validate

**Throws:**
- `LocalityException` - if validation fails
Copyright 2001-2011, Terracotta, Inc.