About Jd2chm...

The project files of this HTML Help Were generated with Jd2chm version 0.3
Copyright © 2000-2003 Andre Burgaud
http://www.burgaud.com
log4j version 1.2.14
API Specification

Make sure to read the user manual in addition to this javadoc documentation.

See:
Description

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.chainsaw</td>
<td>Chainsaw is a GUI log viewer and filter for the log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.config</td>
<td>Package used in getting/setting component properties.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.jdbc</td>
<td>The JDBCAppender provides for sending log events to a database.</td>
</tr>
<tr>
<td>org.apache.log4j.jmx</td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
<tr>
<td>org.apache.log4j.lf5</td>
<td></td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td>org.apache.log4j.nt</td>
<td>Package for NT event logging.</td>
</tr>
<tr>
<td>org.apache.log4j.or</td>
<td>ObjectRenders are responsible for rendering messages depending on their class type.</td>
</tr>
<tr>
<td>org.apache.log4j.or.jms</td>
<td>This package contains the MessageRenderer which renders objects of type javax.jms.Message.</td>
</tr>
<tr>
<td>org.apache.log4j.or.sax</td>
<td>This package contains the AttributesRenderer which renders object of class org.xml.sax.Attributes.</td>
</tr>
<tr>
<td>Package</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>org.apache.log4j.performance</td>
<td>Package to measure the performance of the different log4j components.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
<tr>
<td>org.apache.log4j.xml.examples</td>
<td>Example usage of log4j with XML (including source code).</td>
</tr>
</tbody>
</table>

Make sure to read the [user manual](#) in addition to this javadoc documentation.

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For All Packages

Package Hierarchies:

- org.apache.log4j
- org.apache.log4j.chainsaw
- org.apache.log4j.config
- org.apache.log4j.helpers
- org.apache.log4j.jdbc
- org.apache.log4j.jmx
- org.apache.log4j.lf5
- org.apache.log4j.net
- org.apache.log4j.nt
- org.apache.log4j.or
- org.apache.log4j.or.jms
- org.apache.log4j.or.sax
- org.apache.log4j.performance
- org.apache.log4j.spi
- org.apache.log4j.varia
- org.apache.log4j.xml
- org.apache.log4j.xml.examples
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.jmx.**AbstractDynamicMBean** (implements javax.management.DynamicMBean, javax.management.MBeanRegistration)
    - class org.apache.log4j.jmx.**AppenderDynamicMBean**
    - class org.apache.log4j.jmx.**LayoutDynamicMBean**
    - class org.apache.log4j.jmx.**LoggerDynamicMBean** (implements javax.management.NotificationListener)
  - class org.apache.log4j.jmx.**Agent**
  - class org.apache.log4j.jmx.helpers.**AppenderAttachableImpl** (implements org.apache.log4j.spi.AppenderAttachable)
  - class org.apache.log4j.jmx.lf5.**AppenderFinalizer**
  - class org.apache.log4j.jmx.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.jmx.**AsyncAppender** (implements org.apache.log4j.spi.AppenderAttachable)
    - class org.apache.log4j.jmx.jdbc.**JDBCAppender** (implements org.apache.log4j.**Appender**)
    - class org.apache.log4j.jmx.net.**JMSAppender**
    - class org.apache.log4j.jmx.lf5.**LF5Appender**
    - class org.apache.log4j.jmx.nt.**NTEventLogAppender**
    - class org.apache.log4j.jmx.log4j.**NullAppender**
    - class org.apache.log4j.jmx.log4j.net.**SMTPAppender**
    - class org.apache.log4j.jmx.log4j.net.**SocketAppender**
    - class org.apache.log4j.jmx.log4j.net.**SocketHubAppender**
    - class org.apache.log4j.jmx.log4j.net.**SyslogAppender**
    - class org.apache.log4j.jmx.log4j.net.**TelnetAppender**
    - class org.apache.log4j.jmx.log4j.**WriterAppender**
      - class org.apache.log4j.jmx.**ConsoleAppender**
      - class org.apache.log4j.jmx.**FileAppender**
- class org.apache.log4j.DailyRollingFileAppender
- class org.apache.log4j.RollingFileAppender
  - class org.apache.log4j.varia.ExternallyRolledFileAppender
- class org.apache.log4j.or.sax.AttributesRenderer (implements org.apache.log4j.or.ObjectRenderer)
- class org.apache.log4j.or.BoundedFIFO
  - class org.apache.log4j.Logger
    - class org.apache.log4j.spi.RootCategory
    - class org.apache.log4j.spi.RootLogger
  - class java.awt.Container
    - class java.awt.Window (implements javax.accessibility.Accessible)
      - class java.awt.Frame (implements java.awt.MenuContainer)
        - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
      - class org.apache.log4j.chainsaw.Main
- class org.apache.log4j.helpers.CyclicBuffer
- class org.apache.log4j.xml.DOMConfigurator (implements org.apache.log4j.spi.Configurator)
- class org.apache.log4j.spi.Filter (implements org.apache.log4j.spi.OPTIONHandler)
  - class org.apache.log4j.varia.DenyAllFilter
  - class org.apache.log4j.varia.LevelMatchFilter
- class org.apache.log4j.varia.LevelRangeFilter
- class org.apache.log4j.varia.StringMatchFilter
- class java.text.Format (implements java.lang.Cloneable, java.io.Serializable)
- class java.text.DateFormat
  - class org.apache.log4j.helpers.AbsoluteTimeDateFormat
  - class org.apache.log4j.helpers.DateTimeDateFormat
  - class org.apache.log4j.helpers.ISO8601DateFormat
  - class org.apache.log4j.helpers.RelativeTimeDateFormat
- class org.apache.log4j.helpers.FormattingInfo
- class org.apache.log4j.net.JMSSink (implements javax.jms.MessageListener)
- class org.apache.log4j.Layout (implements org.apache.log4j.spi.OptionHandler)
  - class org.apache.log4j.helpers.DateLayout
  - class org.apache.log4j.TTCCLayout
  - class org.apache.log4j.HTMLLayout
  - class org.apache.log4j.PatternLayout
  - class org.apache.log4j.SimpleLayout
  - class org.apache.log4j.xml.XMLLayout
- class org.apache.log4j.performance.ListVsVector
- class org.apache.log4j.helpers.Loader
- class org.apache.log4j.spi.LocationInfo (implements java.io.Serializable)
- class org.apache.log4j.xml.Log4jEntityResolver (implements org.xml.sax.EntityResolver)
- class org.apache.log4j.spi.LoggingEvent (implements java.io.Serializable)
- class org.apache.log4j.lf5.LogLevel (implements java.io.Serializable)
- class org.apache.log4j.helpers.LogLog
- class org.apache.log4j.LogManager
- class org.apache.log4j.lf5.LogRecord (implements java.io.Serializable)
  - class org.apache.log4j.lf5.Log4JLogRecord
- class org.apache.log4j.MDC
- class org.apache.log4j.or.jms.MessageRenderer (implements org.apache.log4j.or.ObjectRenderer)
- class org.apache.log4j.NDC
- class org.apache.log4j.performance.NewVsSetLen
- class org.apache.log4j.helpers.NullEnumeration (implements java.utilEnumeration)
- class org.apache.log4j.helpers.OptionConverter
- class org.apache.log4j.helpers.PatternConverter
- class org.apache.log4j.helpers.PatternParser
- class org.apache.log4j.helpers.Priority
  - class org.apache.log4j.Priority.Level (implements java.io.Serializable)
- class org.apache.log4j.config.PropertyGetter
- class org.apache.log4j.config.PropertySetter
- class org.apache.log4j.or.RendererMap
- class org.apache.log4j.xml.varia.Roller
- class org.apache.log4j.xml.varia.SAXErrorHandler (implements org.xml.sax.ErrorHandler)
- class org.apache.log4j.net.SimpleSocketServer
- class org.apache.log4j.net.SocketNode (implements java.langRunnable)
- class org.apache.log4j.net.SocketServer
- class org.apache.log4j.lf5.StartLogFactor5
- class org.apache.log4j.performance.SystemTime
- class java.lang.Thread (implements java.lang.Runnable)
  - class org.apache.log4j.helpers.FileWatchdog
- class org.apache.log4j.net.TelnetAppender.SocketHandler
- class org.apache.log4j.or.ThreadGroupRenderer (implements org.apache.log4j.or.ObjectRenderer)
- class java.lang.ThreadLocal
  - class java.lang.InheritableThreadLocal
  - class org.apache.log4j.helpers.ThreadLocalMap
- class java.lang.Throwable (implements java.io.Serializable)
  - class java.lang.Exception
  - class org.apache.log4j.lf5.LogLevelFormatException
  - class org.apache.log4j.config.PropertySetterException
- class org.apache.log4j.spi ThrowableInformation (implements java.io.Serializable)
- class org.apache.log4j.helpers.Transform
- class java.io.Writer
  - class java.io.FilterWriter
  - class org.apache.log4j.helpers.QuietWriter
  - class org.apache.log4j.helpers.CountingQuietWriter
  - class org.apache.log4j.helpers.SyslogQuietWriter
  - class org.apache.log4j.performance.NOPWriter
  - class org.apache.log4j.helpers.SyslogWriter
- class org.apache.log4j.xml.examples.XMLSample
Interface Hierarchy

- interface org.apache.log4j.Appender
- interface org.apache.log4j.spi.AppenderAttachable
- interface org.apache.log4j.spi.Configurator
- interface org.apache.log4j.spi.ErrorCode
- interface org.apache.log4j.spi.HierarchyEventListener
- interface org.apache.log4j.spi.LoggerFactory
- interface org.apache.log4j.spi.LoggerRepository
- interface org.apache.log4j.lf5.LogRecordFilter
- interface org.apache.log4j.or.ObjectRenderer
- interface org.apache.log4j.spi.OptionHandler
  - interface org.apache.log4j.spi.ErrorHandler
- interface org.apache.log4j.config.PropertyGetter.PropertyCallback
- interface org.apache.log4j.spi.RendererSupport
- interface org.apache.log4j.spi.RepositorySelector
- interface org.apache.log4j.spi.TriggeringEventEvaluator
Package org.apache.log4j

The main log4j package.

See: Description

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appender</td>
<td>Implement this interface for your own strategies for outputting log statements.</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderSkeleton</td>
<td>Abstract superclass of the other appenders in the package.</td>
</tr>
<tr>
<td>AsyncAppender</td>
<td>The AsyncAppender lets users log events asynchronously.</td>
</tr>
<tr>
<td>BasicConfigurator</td>
<td>Use this class to quickly configure the package.</td>
</tr>
<tr>
<td>Category</td>
<td>This class has been deprecated and replaced by the Logger subclass.</td>
</tr>
<tr>
<td>ConsoleAppender</td>
<td>ConsoleAppender appends log events to System.out or System.err using a layout specified by the user.</td>
</tr>
<tr>
<td>DailyRollingFileAppender</td>
<td>DailyRollingFileAppender extends FileAppender so that the underlying file is rolled over at a user chosen frequency.</td>
</tr>
<tr>
<td>FileAppender</td>
<td>FileAppender appends log events to a file.</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>This class is specialized in retrieving loggers by name and also maintaining the logger hierarchy.</td>
</tr>
<tr>
<td>HTMLLayout</td>
<td>This layout outputs events in a HTML table.</td>
</tr>
<tr>
<td>Layout</td>
<td>Extend this abstract class to create your own log layout format.</td>
</tr>
<tr>
<td><strong>Level</strong></td>
<td>Defines the minimum set of levels recognized by the system, that is OFF, FATAL, ERROR, WARN, INFO, DEBUG and ALL.</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Logger</strong></td>
<td>This is the central class in the log4j package.</td>
</tr>
<tr>
<td><strong>LogManager</strong></td>
<td>Use the LogManager class to retrieve Logger instances or to operate on the current LoggerRepository.</td>
</tr>
<tr>
<td><strong>MDC</strong></td>
<td>The MDC class is similar to the NDC class except that it is based on a map instead of a stack.</td>
</tr>
<tr>
<td><strong>NDC</strong></td>
<td>The NDC class implements nested diagnostic contexts as defined by Neil Harrison in the article &quot;Patterns for Logging Diagnostic Messages&quot; part of the book &quot;Pattern Languages of Program Design 3&quot; edited by Martin et al.</td>
</tr>
<tr>
<td><strong>PatternLayout</strong></td>
<td>A flexible layout configurable with pattern string.</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
<td>Refrain from using this class directly, use the Level class instead.</td>
</tr>
<tr>
<td><strong>PropertyConfigurator</strong></td>
<td>Allows the configuration of log4j from an external file.</td>
</tr>
<tr>
<td><strong>RollingFileAppender</strong></td>
<td>RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.</td>
</tr>
<tr>
<td><strong>SimpleLayout</strong></td>
<td>SimpleLayout consists of the level of the log statement, followed by &quot; - &quot; and then the log message itself.</td>
</tr>
<tr>
<td><strong>TTCCLayout</strong></td>
<td>TTCC layout format consists of time, thread, category and nested diagnostic context information, hence the name.</td>
</tr>
<tr>
<td><strong>WriterAppender</strong></td>
<td>WriterAppender appends log events to a writer or an OutputStream depending on the user's choice.</td>
</tr>
</tbody>
</table>
Package org.apache.log4j Description

The main log4j package.

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j

Package Hierarchies:

All Packages

________________________________________________________________________
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.**AsyncAppender** (implements org.apache.log4j.spi.**AppenderAttachable**)
      - class org.apache.log4j.**WriterAppender**
        - class org.apache.log4j.**ConsoleAppender**
        - class org.apache.log4j.**FileAppender**
          - class org.apache.log4j.**DailyRollingFileAppender**
          - class org.apache.log4j.**RollingFileAppender**
    - class org.apache.log4j.**BasicConfigurator**
  - class org.apache.log4j.**Category** (implements org.apache.log4j.spi.**AppenderAttachable**)
    - class org.apache.log4j.**Logger**
    - class org.apache.log4j.**Layout** (implements org.apache.log4j.spi.**OptionHandler**)
      - class org.apache.log4j.helpers.**DateLayout**
      - class org.apache.log4j.**TTCCLayout**
      - class org.apache.log4j.**HTMLLayout**
      - class org.apache.log4j.**PatternLayout**
      - class org.apache.log4j.**SimpleLayout**
  - class org.apache.log4j.**LogManager**
  - class org.apache.log4j.**MDC**
  - class org.apache.log4j.**NDC**
  - class org.apache.log4j.**Priority**
    - class org.apache.log4j.**Level** (implements java.io.**Serializable**)
  - class org.apache.log4j.**PropertyConfigurator** (implements org.apache.log4j.spi.**Configurator**
Interface Hierarchy

- interface org.apache.log4j.Appender

Copyright 2000-2005 Apache Software Foundation.
# Uses of Package org.apache.log4j

<table>
<thead>
<tr>
<th>Packages that use org.apache.log4j</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.config</td>
<td>Package used in getting/setting component properties.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.jdbc</td>
<td>The JDBCAppender provides for sending log events to a database.</td>
</tr>
<tr>
<td>org.apache.log4j.jmx</td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
<tr>
<td>org.apache.log4j.lf5</td>
<td></td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td>org.apache.log4j.nt</td>
<td>Package for NT event logging.</td>
</tr>
<tr>
<td>org.apache.log4j.performance</td>
<td>Package to measure the performance of the different log4j components.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j used by org.apache.log4j</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Appenders</td>
<td>Implement this interface for your own strategies for outputting log statements.</td>
</tr>
</tbody>
</table>
**AppenderSkeleton**  
Abstract superclass of the other appenders in the package.

**Category**  
This class has been deprecated and replaced by the **Logger** subclass.

**FileAppender**  
FileAppender appends log events to a file.

**Layout**  
Extend this abstract class to create your own log layout format.

**Level**  
Defines the minimum set of levels recognized by the system, that is **OFF**, **FATAL**, **ERROR**, **WARN**, **INFO**, **DEBUG** and **ALL**.

**Logger**  
This is the central class in the log4j package.

**Priority**  
Refrain from using this class directly, use the **Level** class instead.

**WriterAppender**  
WriterAppender appends log events to a **Writer** or an **OutputStream** depending on the user's choice.

---

### Classes in org.apache.log4j used by org.apache.log4j.config

**Logger**  
This is the central class in the log4j package.

### Classes in org.apache.log4j used by org.apache.log4j.helpers

**Appender**  
Implement this interface for your own strategies for outputting log statements.

**Layout**
Extend this abstract class to create your own log layout format.

**Level**
Defines the minimum set of levels recognized by the system, that is OFF, FATAL, ERROR, WARN, INFO, DEBUG and ALL.

**Logger**
This is the central class in the log4j package.

---

## Classes in [org.apache.log4j](https://logging.apache.org/log4j/1.2/apidoc/) used by [org.apache.log4j.jdbc](https://logging.apache.org/log4j/1.2/apidoc/)

**Appender**
- Implement this interface for your own strategies for outputting log statements.

**AppenderSkeleton**
- Abstract superclass of the other appenders in the package.

---

## Classes in [org.apache.log4j](https://logging.apache.org/log4j/1.2/apidoc/) used by [org.apache.log4j.jmx](https://logging.apache.org/log4j/1.2/apidoc/)

**Appender**
- Implement this interface for your own strategies for outputting log statements.

**Category**
- This class has been deprecated and replaced by the [Logger](https://logging.apache.org/log4j/1.2/apidoc/) subclass.

**Layout**
- Extend this abstract class to create your own log layout format.

**Logger**
- This is the central class in the log4j package.

---

Classes in [org.apache.log4j](https://logging.apache.org/log4j/1.2/apidoc/) used by
### org.apache.log4j.lf5

**Appender**
Implement this interface for your own strategies for outputting log statements.

**AppenderSkeleton**
Abstract superclass of the other appenders in the package.

### Classes in org.apache.log4j used by org.apache.log4j.net

**Appender**
Implement this interface for your own strategies for outputting log statements.

**AppenderSkeleton**
Abstract superclass of the other appenders in the package.

**Layout**
Extend this abstract class to create your own log layout format.

### Classes in org.apache.log4j used by org.apache.log4j.nt

**Appender**
Implement this interface for your own strategies for outputting log statements.

**AppenderSkeleton**
Abstract superclass of the other appenders in the package.

**Layout**
Extend this abstract class to create your own log layout format.
**org.apache.log4j.performance**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appender</td>
<td>Implement this interface for your own strategies for outputting log statements.</td>
</tr>
<tr>
<td>AppenderSkeleton</td>
<td>Abstract superclass of the other appenders in the package.</td>
</tr>
<tr>
<td>Layout</td>
<td>Extend this abstract class to create your own log layout format.</td>
</tr>
</tbody>
</table>

**Classes in org.apache.log4j used by org.apache.log4j.spi**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appender</td>
<td>Implement this interface for your own strategies for outputting log statements.</td>
</tr>
<tr>
<td>Category</td>
<td>This class has been deprecated and replaced by the Logger subclass.</td>
</tr>
<tr>
<td>Level</td>
<td>Defines the minimum set of levels recognized by the system, that is OFF, FATAL, ERROR, WARN, INFO, DEBUG and ALL.</td>
</tr>
<tr>
<td>Logger</td>
<td>This is the central class in the log4j package.</td>
</tr>
<tr>
<td>Priority</td>
<td>Refrain from using this class directly, use the Level class instead.</td>
</tr>
</tbody>
</table>

**Classes in org.apache.log4j used by org.apache.log4j.varia**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appender</td>
<td>Implement this interface for your own strategies for outputting log statements.</td>
</tr>
</tbody>
</table>
### AppenderSkeleton
Abstract superclass of the other appenders in the package.

### FileAppender
FileAppender appends log events to a file.

### Level
Defines the minimum set of levels recognized by the system, that is `OFF`, `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG` and `ALL`.

### Logger
This is the central class in the log4j package.

### RollingFileAppender
RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.

### WriterAppender
WriterAppender appends log events to a `Writer` or an `OutputStream` depending on the user's choice.

---

### Classes in [org.apache.log4j](https://logging.apache.org/log4j/1.2/log4j.html) used by [org.apache.log4j.xml](https://logging.apache.org/log4j/1.2/log4j.xml)

#### Appender
Implement this interface for your own strategies for outputting log statements.

#### Layout
Extend this abstract class to create your own log layout format.

#### Logger
This is the central class in the log4j package.

---

Copyright 2000-2005 Apache Software Foundation.
public interface Appender

Implement this interface for your own strategies for outputting log statements.

Author:
Ceki Gülcü

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addFilter(Filter newFilter)</td>
<td>Add a filter to the end of the filter list.</td>
</tr>
<tr>
<td>void clearFilters()</td>
<td>Clear the list of filters by removing all the filters in it.</td>
</tr>
<tr>
<td>void close()</td>
<td>Release any resources allocated within the appender such as file handles, network connections, etc.</td>
</tr>
<tr>
<td>void doAppend(LoggingEvent event)</td>
<td>Log in Appender specific way.</td>
</tr>
<tr>
<td>ErrorHandler getErrorHandler()</td>
<td>Returns the ErrorHandler for this appender.</td>
</tr>
<tr>
<td>Filter getFilter()</td>
<td>Returns the head Filter.</td>
</tr>
<tr>
<td>Layout getLayout()</td>
<td>Returns this appenders layout.</td>
</tr>
<tr>
<td>String getName()</td>
<td>Get the name of this appender.</td>
</tr>
<tr>
<td>requiresLayout()</td>
<td></td>
</tr>
</tbody>
</table>
boolean Configurators call this method to determine if the appender requires a layout.

void setErrorHandler(ErrorHandler errorHandler)

Set the ErrorHandler for this appender.

void setLayout(Layout layout)

Set the Layout for this appender.

void setName(String name)

Set the name of this appender.

---

**Method Detail**

**addFilter**

public void addFilter(Filter newFilter)

Add a filter to the end of the filter list.

*Since:

0.9.0

**getFilter**

public Filter getFilter()

Returns the head Filter. The Filters are organized in a linked list and so all Filters on this Appender are available through the result.

*Returns:

the head Filter or null, if no Filters are present

*Since:

1.1

**clearFilters**

public void clearFilters()
Clear the list of filters by removing all the filters in it.

**Since:**

0.9.0

---

**close**

class method `close`

```java
public void close()
```

Release any resources allocated within the appender such as file handles, network connections, etc.

It is a programming error to append to a closed appender.

**Since:**

0.8.4

---

**doAppend**

class method `doAppend`

```java
public void doAppend(LoggingEvent event)
```

Log in Appender specific way. When appropriate, Loggers will call the `doAppend` method of appender implementations in order to log.

---

**getName**

get method `getName`

```java
public String getName()
```

Get the name of this appender. The name uniquely identifies the appender.

---

**setErrorHandler**

set method `setErrorHandler`

```java
public void setErrorHandler(ErrorHandler errorHandler)
```

Set the `ErrorHandler` for this appender.

**Since:**
getErrorHandler

public ErrorHandler getErrorHandler()

Returns the ErrorHandler for this appender.
Since: 1.1

setLayout

public void setLayout(Layout layout)

Set the Layout for this appender.
Since: 0.8.1

getLayout

public Layout getLayout()

Returns this appenders layout.
Since: 1.1

setName

public void setName(String name)

Set the name of this appender. The name is used by other components to identify this appender.
Since: 0.8.1
requiresLayout

public boolean requiresLayout()

Configurators call this method to determine if the appender requires a layout. If this method returns true, meaning that layout is required, then the configurator will configure an layout using the configuration information at its disposal. If this method returns false, meaning that a layout is not required, then layout configuration will be skipped even if there is available layout configuration information at the disposal of the configurator.

In the rather exceptional case, where the appender implementation admits a layout but can also work without it, then the appender should return true.

Since:

0.8.4
Class AppenderSkeleton

java.lang.Object
+-- org.apache.log4j.AppenderSkeleton

All Implemented Interfaces:
    Appender, OptionHandler

Direct Known Subclasses:
    AsyncAppender, JDBCApender, JMSAppender, LF5Appender, NTEventLogAppender, NullAppender, NullAppender, SMTPAppender, SocketAppender, SocketHubAppender, SyslogAppender, TelnetAppender, WriterAppender

Public abstract class AppenderSkeleton extends Object
implements Appender, OptionHandler

Abstract superclass of the other appenders in the package. This class provides the code for common functionality, such as support for threshold filtering and support for general filters.

Since: 0.8.1
Author: Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed</td>
<td>boolean</td>
<td>Is this appender closed?</td>
</tr>
<tr>
<td>errorHandler</td>
<td>ErrorHandler</td>
<td>It is assumed and enforced that errorHandler is never null.</td>
</tr>
<tr>
<td>headFilter</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Filter

The first filter in the filter chain.

protected

layout

The layout variable does not need to be set if the appender implementation has its own layout.

protected

String

name

Appenders are named.

protected

Filter

tailFilter

The last filter in the filter chain.

protected

Priority

threshold

There is no level threshold filtering by default.

Constructor Summary

AppenderSkeleton()

Method Summary

void activateOptions()

Derived appenders should override this method if option structure requires it.

void addFilter(Filter newFilter)

Add a filter to end of the filter list.

protected abstract void append(LoggingEvent event)

Subclasses of AppenderSkeleton should implement this method to perform actual logging.

void clearFilters()

Clear the filters chain.

void doAppend(LoggingEvent event)

This method performs threshold checks and invokes filters before delegating actual logging to the subclasses specific append(org.apache.log4j.spi.LoggingEvent) method.

void finalize()

Finalize this appender by calling the derived class'
**getErrorHandler()**
Return the currently set ErrorHandler for this Appender.

**getFilter()**
Returns the head Filter.

**getFirstFilter()**
Return the first filter in the filter chain for this Appender.

**getLayout()**
Returns the layout of this appender.

**getName()**
Returns the name of this FileAppender.

**getThreshold()**
Returns this appender's threshold level.

**isAsSevereAsThreshold(Priority priority)**
Check whether the message level is below the appender's threshold.

**setErrorHandler(ErrorHandler eh)**
Set the ErrorHandler for this Appender.

**setLayout(Layout layout)**
Set the layout for this appender.

**setName(String name)**
Set the name of this Appender.

**setThreshold(Priority threshold)**
Set the threshold level.

**Methods inherited from class java.lang.Object**
close, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Methods inherited from interface org.apache.log4j.Appender**
close, requiresLayout
Field Detail

layout

protected  Layout  layout

   The layout variable does not need to be set if the appender implementation has its own layout.

name

protected  String  name

   Appenders are named.

threshold

protected  Priority  threshold

   There is no level threshold filtering by default.

errorHandler

protected  ErrorHandler  errorHandler

   It is assumed and enforced that errorHandler is never null.

headFilter

protected  Filter  headFilter

   The first filter in the filter chain. Set to null initially.
tailFilter

protected Filter tailFilter

The last filter in the filter chain.

closed

protected boolean closed

Is this appender closed?

Constructor Detail

AppenderSkeleton

public AppenderSkeleton()

Method Detail

activateOptions

public void activateOptions()

Derived appenders should override this method if option structure requires it.

Specified by:
activateOptions in interface OptionHandler

addFilter

public void addFilter(Filter newFilter)

Add a filter to end of the filter list.

Specified by:
addFilter in interface Appender
Since: 0.9.0

append
protected abstract void append(LoggingEvent event)

Subclasses of AppenderSkeleton should implement this method to perform actual logging. See also AppenderSkeleton.doAppend method.
Since: 0.9.0

clearFilters
public void clearFilters()

Clear the filters chain.
Specified by:
clearFilters in interface Appender
Since: 0.9.0

finalize
public void finalize()

Finalize this appender by calling the derived class' close method.
Overrides:
finalize in class Object
Since: 0.8.4

getErrorHandler
public ErrorHandler getErrorHandler()

Return the currently set ErrorHandler for this Appender.

Specified by: getErrorHandler in interface Appender

Since: 0.9.0

getFilter

public Filter getFilter()

Returns the head Filter.

Specified by: getFilter in interface Appender

Since: 1.1

getFirstFilter

public final Filter getFirstFilter()

Return the first filter in the filter chain for this Appender. The return value may be null if no is filter is set.

getLayout

public Layout getLayout()

Returns the layout of this appender. The value may be null.

Specified by: getLayout in interface Appender

getName

public final String getName()
Returns the name of this FileAppender.

**Specified by:**

*getName* in interface *Appender*

---

**getThreshold**

```java
public Priority getThreshold()
```

Returns this appenders threshold level. See the *setThreshold* method for the meaning of this option.

**Since:**

1.1

---

**isAsSevereAsThreshold**

```java
public boolean isAsSevereAsThreshold(Priority priority)
```

Check whether the message level is below the appender's threshold. If there is no threshold set, then the return value is always true.

---

**doAppend**

```java
public void doAppend(LoggingEvent event)
```

This method performs threshold checks and invokes filters before delegating actual logging to the subclasses specific *append* method.

**Specified by:**

*doAppend* in interface *Appender*

---

**setErrorHandler**

```java
public void setErrorHandler(ErrorHandler eh)
```
Set the `ErrorHandler` for this Appender.

**Specified by:**
```
setErrorHandler in interface Appender
```

**Since:**
```
0.9.0
```

---

**setLayout**

``` java
public void setLayout(Layout layout)
```

Set the layout for this appender. Note that some appenders have their own (fixed) layouts or do not use one. For example, the `SocketAppender` ignores the layout set here.

**Specified by:**
```
setLayout in interface Appender
```

---

**setName**

``` java
public void setName(String name)
```

Set the name of this Appender.

**Specified by:**
```
setName in interface Appender
```

---

**setThreshold**

``` java
public void setThreshold(Priority threshold)
```

Set the threshold level. All log events with lower level than the threshold level are ignored by the appender.

In configuration files this option is specified by setting the value of the `Threshold` option to a level string, such as "DEBUG", "INFO" and so on.

**Since:**
<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>

Copyright 2000-2005 Apache Software Foundation.
Class AsyncAppender

public class AsyncAppender
extends AppenderSkeleton
implements AppenderAttachable

The AsyncAppender lets users log events asynchronously.

The AsyncAppender will collect the events sent to it and then dispatch them to all the appenders that are attached to it. You can attach multiple appenders to an AsyncAppender.

The AsyncAppender uses a separate thread to serve the events in its buffer.

Important note: The AsyncAppender can only be script configured using the DOMConfigurator.

Since:
0.9.1
Author:
Ceki Gülcü, Curt Arnold

Field Summary

| static int DEFAULT_BUFFER_SIZE | The default buffer size is set to 128 events. |
## Constructor Summary

**AsyncAppender()**
Create new instance.

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void [addAppender](Appender newAppender)</td>
<td>Add appender.</td>
</tr>
<tr>
<td>void [append](LoggingEvent event)</td>
<td>{@inheritDoc}</td>
</tr>
<tr>
<td>void <a href="">close</a></td>
<td>Close this AsyncAppender by interrupting the dispatcher thread which will process all pending events before exiting.</td>
</tr>
<tr>
<td>Enumeration <a href="">getAllAppenders</a></td>
<td>Get iterator over attached appenders.</td>
</tr>
<tr>
<td>Appender [getAppender](String name)</td>
<td>Get appender by name.</td>
</tr>
<tr>
<td>boolean <a href="">getBlocking</a></td>
<td>Gets whether appender should block calling thread when buffer is full.</td>
</tr>
<tr>
<td>int <a href="">getBufferSize</a></td>
<td>Gets the current buffer size.</td>
</tr>
<tr>
<td>boolean <a href="">getLocationInfo</a></td>
<td>Gets whether the location of the logging request call should be captured.</td>
</tr>
<tr>
<td>boolean [isAttached](Appender appender)</td>
<td>Determines if specified appender is attached.</td>
</tr>
<tr>
<td>void <a href="">removeAllAppenders</a></td>
<td>Removes and closes all attached appenders.</td>
</tr>
<tr>
<td>void [removeAppender](Appender appender)</td>
<td></td>
</tr>
</tbody>
</table>
Removes an appender.

```java
void removeAppender(String name)
    Remove appender by name.
```

```java
boolean requiresLayout()
    {@inheritDoc}
```

```java
void setBlocking(boolean value)
    Sets whether appender should wait if there is no space available in the event buffer or immediately return.
```

```java
void setBufferSize(int size)
    Sets the number of messages allowed in the event buffer before the calling thread is blocked (if blocking is true) or until messages are summarized and discarded.
```

```java
void setLocationInfo(boolean flag)
    The LocationInfo option takes a boolean value.
```

Methods inherited from class org.apache.log4j.AppenderSkeleton

```java
activateOptions, addFilter, clearFilters, doAppend, finalize, getErrorHandler, getFilter, getFirstFilter, getLayout, getName, getThreshold, isAsSevereAsThreshold, setErrorHandler, setLayout, setName, setThreshold
```

Methods inherited from class java.lang.Object

```java
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Field Detail

**DEFAULT_BUFFER_SIZE**

```java
public static final int DEFAULT_BUFFER_SIZE
    The default buffer size is set to 128 events.
```
AsyncAppender

public AsyncAppender()

Create new instance.

Method Detail

addAppender

public void addAppender(Appender newAppender)

Add appender.

Specified by:
addAppender in interface AppenderAttachable

Parameters:
newAppender - appender to add, may not be null.

append

public void append(LoggingEvent event)

{@inheritdoc}

Overrides:
append in class AppenderSkeleton

close

public void close()

Close this AsyncAppender by interrupting the dispatcher thread which will process all pending events before exiting.

getAllAppenders
public Enumeration getAllAppenders()

Get iterator over attached appenders.
Specified by:
getAllAppenders in interface AppenderAttachable

Returns:
iterator or null if no attached appenders.

getAppender

public Appender getAppender(String name)

Get appender by name.
Specified by:
getAppender in interface AppenderAttachable

Parameters:
name - name, may not be null.

Returns:
matching appender or null.

getLocationInfo

public boolean getLocationInfo()

Gets whether the location of the logging request call should be captured.

Returns:
the current value of the LocationInfo option.

isAttached

public boolean isAttached(Appender appender)

Determines if specified appender is attached.
Specified by:
isAttached in interface AppenderAttachable

Parameters:
appender - appender.

**Returns:**
true if attached.

---

**requiresLayout**

```java
public boolean requiresLayout()
{
    @inheritDoc
}
```

---

**removeAllAppenders**

```java
public void removeAllAppenders()

    Removes and closes all attached appenders.
    Specified by:
    removeAllAppenders in interface AppenderAttachable
```

---

**removeAppender**

```java
public void removeAppender(Appender appender)

    Removes an appender.
    Specified by:
    removeAppender in interface AppenderAttachable

    Parameters:
    appender - appender to remove.
```

---

**removeAppender**

```java
public void removeAppender(String name)

    Remove appender by name.
    Specified by:
    removeAppender in interface AppenderAttachable
```
Parameters:
  name - name.

---

**setLocationInfo**

```java
public void setLocationInfo(boolean flag)
```

The `LocationInfo` option takes a boolean value. By default, it is set to false which means there will be no effort to extract the location information related to the event. As a result, the event that will be ultimately logged will likely to contain the wrong location information (if present in the log format).

Location information extraction is comparatively very slow and should be avoided unless performance is not a concern.

Parameters:
  flag - true if location information should be extracted.

---

**setBufferSize**

```java
public void setBufferSize(int size)
```

Sets the number of messages allowed in the event buffer before the calling thread is blocked (if blocking is true) or until messages are summarized and discarded. Changing the size will not affect messages already in the buffer.

Parameters:
  size - buffer size, must be positive.

---

**getBufferSize**

```java
public int getBufferSize()
```

Gets the current buffer size.

Returns:
the current value of the BufferSize option.

### setBlocking

**public void setBlocking(boolean value)**

Sets whether appender should wait if there is no space available in the event buffer or immediately return.

**Parameters:**

- `value` - true if appender should wait until available space in buffer.

### getBlocking

**public boolean getBlocking()**

Gets whether appender should block calling thread when buffer is full. If false, messages will be counted by logger and a summary message appended after the contents of the buffer have been appended.

**Returns:**

- true if calling thread will be blocked when buffer is full.
public class BasicConfigurator extends Object

Use this class to quickly configure the package.

For file based configuration see PropertyConfigurator. For XML based configuration see DOMConfigurator.

Since: 0.8.1
Author: Ceki Gülcü

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected BasicConfigurator()</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void configure()</td>
</tr>
<tr>
<td>Add a ConsoleAppender that uses PatternLayout using the PatternLayout.TTCC_CONVERSION_PATTERN and prints to System.out to the root category.</td>
</tr>
<tr>
<td>static void configure(Appender appender)</td>
</tr>
<tr>
<td>Add appender to the root category.</td>
</tr>
<tr>
<td>static void resetConfiguration()</td>
</tr>
<tr>
<td>Reset the default hierarchy to its default.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

BasicConfigurator

protected BasicConfigurator()

Method Detail

configure

public static void configure()

Add a ConsoleAppender that uses PatternLayout using the PatternLayout.TTCC_CONVERSION_PATTERN and prints to System.out to the root category.

configure

public static void configure(Appender appender)

Add appender to the root category.
Parameters:
appender - The appender to add to the root category.

resetConfiguration

public static void resetConfiguration()

Reset the default hierarchy to its defaut. It is equivalent to calling Category.getDefaultHierarchy().resetConfiguration(). See
Hierarchy.resetConfiguration() for more details.
public class Category
extends Object
implements AppenderAttachable

This class has been deprecated and replaced by the Logger subclass. It will be kept around to preserve backward compatibility until mid 2003.

Logger is a subclass of Category, i.e. it extends Category. In other words, a logger is a category. Thus, all operations that can be performed on a category can be performed on a logger. Internally, whenever log4j is asked to produce a Category object, it will instead produce a Logger object. Log4j 1.2 will never produce Category objects but only Logger instances. In order to preserve backward compatibility, methods that previously accepted category objects still continue to accept category objects.

For example, the following are all legal and will work as expected.

    // Deprecated form:
    Category cat = Category.getInstance("foo.bar")

    // Preferred form for retrieving loggers:
    Logger logger = Logger.getLogger("foo.bar")

The first form is deprecated and should be avoided.
There is absolutely no need for new client code to use or refer to the Category class. Whenever possible, please avoid referring to it or using it.

See the short manual for an introduction on this class.

See the document entitled preparing for log4j 1.3 for a more detailed discussion.

Author:
  Ceki Gülcü, Anders Kristensen

---

**Field Summary**

<table>
<thead>
<tr>
<th>protected boolean</th>
<th>additive</th>
<th>Additivity is set to true by default, that is children inherit the appenders of their ancestors by default.</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Level</td>
<td>level</td>
<td>The assigned level of this category.</td>
</tr>
<tr>
<td>protected String</td>
<td>name</td>
<td>The name of this category.</td>
</tr>
<tr>
<td>protected Category</td>
<td>parent</td>
<td>The parent of this category.</td>
</tr>
<tr>
<td>protected LoggerRepository</td>
<td>repository</td>
<td></td>
</tr>
<tr>
<td>protected ResourceBundle</td>
<td>resourceBundle</td>
<td></td>
</tr>
</tbody>
</table>

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>protected Category(String name)</th>
<th>Category(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>This constructor created a new Category instance and sets its name.</td>
</tr>
</tbody>
</table>

---

**Method Summary**
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addAppender</code></td>
<td><code>Appender newAppender</code></td>
<td>Add <code>newAppender</code> to the list of appenders of this Category instance.</td>
</tr>
<tr>
<td><code>assertLog</code></td>
<td><code>boolean assertion, String msg</code></td>
<td>If <code>assertion</code> parameter is false, then logs <code>msg</code> as an <code>error</code> statement.</td>
</tr>
<tr>
<td><code>callAppenders</code></td>
<td><code>LoggingEvent event</code></td>
<td>Call the appenders in the hierarchy starting at this.</td>
</tr>
<tr>
<td><code>debug</code></td>
<td><code>Object message</code></td>
<td>Log a message object with the <code>DEBUG</code> level.</td>
</tr>
<tr>
<td><code>debug</code></td>
<td><code>Object message, Throwable t</code></td>
<td>Log a message object with the <code>DEBUG</code> level including the stack trace of the <code>Throwable</code> <code>t</code> passed as parameter.</td>
</tr>
<tr>
<td><code>error</code></td>
<td><code>Object message</code></td>
<td>Log a message object with the <code>ERROR</code> Level.</td>
</tr>
<tr>
<td><code>error</code></td>
<td><code>Object message, Throwable t</code></td>
<td>Log a message object with the <code>ERROR</code> level including the stack trace of the <code>Throwable</code> <code>t</code> passed as parameter.</td>
</tr>
<tr>
<td><code>exists</code></td>
<td><code>String name</code></td>
<td><code>Deprecated</code>. Please use <code>LogManager.exists(java.lang.String)</code> instead.</td>
</tr>
<tr>
<td><code>fatal</code></td>
<td><code>Object message</code></td>
<td>Log a message object with the <code>FATAL</code> Level.</td>
</tr>
<tr>
<td><code>fatal</code></td>
<td><code>Object message, Throwable t</code></td>
<td>Log a message object with the <code>FATAL</code> level including the stack trace of the <code>Throwable</code> <code>t</code> passed as parameter.</td>
</tr>
<tr>
<td><code>forcedLog</code></td>
<td><code>String fqcn, Priority level, Object message, Throwable t</code></td>
<td>This method creates a new logging event and logs the event without further checks.</td>
</tr>
<tr>
<td><code>getAdditivity</code></td>
<td></td>
<td>Get the additivity flag for this Category.</td>
</tr>
<tr>
<td>Class/Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td></td>
</tr>
</tbody>
</table>
| Enumeration | **getAllAppenders()**  
Get the appenders contained in this category as an **Enumeration**. |
| Appender | **getAppender(String name)**  
Look for the appender named as `name`. |
| Priority | **getChainedPriority()**  
**Deprecated.** Please use the `getEffectiveLevel()` method instead. |
| static Enumeration | **getCurrentCategories()**  
**Deprecated.** Please use `LogManager.getCurrentLoggers()` instead. |
| static LoggerRepository | **getDefaultHierarchy()**  
**Deprecated.** Please use `LogManager.getLoggerRepository()` instead. |
| Level | **getEffectiveLevel()**  
Starting from this category, search the category hierarchy for a non-null level and return it. |
| static LoggerRepository | **getHierarchy()**  
**Deprecated.** Please use `getLoggerRepository()` instead. |
| static Category | **getInstance(Class clazz)**  
**Deprecated.** Please make sure to use `Logger.getLogger(Class)` instead. |
| static Category | **getInstance(String name)**  
**Deprecated.** Make sure to use `Logger.getLogger(String)` instead. |
| Level | **getLevel()**  
Returns the assigned **Level**, if any, for this Category. |
| LoggerRepository | **getLoggerRepository()**  
Return the the **LoggerRepository** where this Category is attached. |
| String | **getName()**  
Return the category name. |
### Category

**getParent()**

Returns the parent of this category.

**getPriority()**

Deprecated. Please use `getLevel()` instead.

### ResourceBundle

**getResourceBundle()**

Returns the inherited `ResourceBundle` for this category.

**getResourceBundleString(String key)**

Returns the string resource corresponding to key in this category's inherited resource bundle.

**getRoot()**

Deprecated. Please use `Logger.getRootLogger()` instead.

### info(Object message)

Log a message object with the INFO Level.

### info(Object message, Throwable t)

Log a message object with the INFO level including the stack trace of the `Throwable t` passed as parameter.

### isAttached(Appender appender)

Is the appender passed as parameter attached to this category?

### isDebugEnabled()

Check whether this category is enabled for the DEBUG Level.

### isEnabledFor(Priority level)

Check whether this category is enabled for a given `Level` passed as parameter.

### isInfoEnabled()

Check whether this category is enabled for the info Level.

### l7dlog(Priority priority, String key, Object[] params, Throwable t)

Log a localized and parameterized message.

### l7dlog(Priority priority, String key, Throwable t)

Log a localized and parameterized message.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>log(Priority priority, Object message)</code></td>
<td>Log a localized message. This generic form is intended to be used by wrappers.</td>
</tr>
<tr>
<td><code>log(Priority priority, Object message, Throwable t)</code></td>
<td>This generic form is intended to be used by wrappers.</td>
</tr>
<tr>
<td><code>log(String callerFQCN, Priority level, Object message, Throwable t)</code></td>
<td>This is the most generic printing method.</td>
</tr>
<tr>
<td><code>removeAllAppenders()</code></td>
<td>Remove all previously added appenders from this Category instance.</td>
</tr>
<tr>
<td><code>removeAppender(Appender appender)</code></td>
<td>Remove the appender passed as parameter form the list of appenders.</td>
</tr>
<tr>
<td><code>removeAppender(String name)</code></td>
<td>Remove the appender with the name passed as parameter form the list of appenders.</td>
</tr>
<tr>
<td><code>setAdditivity(boolean additive)</code></td>
<td>Set the additivity flag for this Category instance.</td>
</tr>
<tr>
<td><code>setLevel(Level level)</code></td>
<td>Set the level of this Category.</td>
</tr>
<tr>
<td><code>setResourceBundle(ResourceBundle bundle)</code></td>
<td>Set the resource bundle to be used with localized logging methods <code>l7dlog(Priority, String, Throwable)</code> and <code>l7dlog(Priority, String, Object[], Throwable)</code>.</td>
</tr>
<tr>
<td><code>shutdown()</code></td>
<td>Deprecated. Please use</td>
</tr>
</tbody>
</table>
**Field Detail**

**name**

protected `String` name

The name of this category.

**level**

protected volatile `Level` level

The assigned level of this category. The `level` variable need not be assigned a value in which case it is inherited form the hierarchy.

**parent**

protected volatile `Category` parent

The parent of this category. All categories have at least one ancestor which is the root category.
resourceBundle

protected ResourceBundle resourceBundle

repository

protected LoggerRepository repository

additive

protected boolean additive

Additivity is set to true by default, that is children inherit the appenders of their ancestors by default. If this variable is set to false then the appenders found in the ancestors of this category are not used. However, the children of this category will inherit its appenders, unless the children have their additivity flag set to false too. See the user manual for more details.

Constructor Detail

Category

protected Category(String name)

This constructor created a new Category instance and sets its name.

It is intended to be used by sub-classes only. You should not create categories directly.

Parameters:
  name - The name of the category.

Method Detail
addAppender

```java
public void addAppender(Appender newAppender)
```

Add `newAppender` to the list of appenders of this Category instance.

If `newAppender` is already in the list of appenders, then it won't be added again.

**Specified by:**

`addAppender` in interface `AppenderAttachable`

---

assertLog

```java
public void assertLog(boolean assertion,
                      String msg)
```

If `assertion` parameter is false, then logs `msg` as an `error` statement.

The `assert` method has been renamed to `assertLog` because `assert` is a language reserved word in JDK 1.4.

**Parameters:**

- `assertion` -
- `msg` - The message to print if `assertion` is false.

**Since:**

1.2

---

callAppenders

```java
public void callAppenders(LoggingEvent event)
```

Call the appenders in the hierarchy starting at `this`. If no appenders could be found, emit a warning.

This method calls all the appenders inherited from the hierarchy circumventing any evaluation of whether to log or not to log the particular log request.
Parameters:
    event - the event to log.

---

**debug**

```java
public void debug(Object message)
```

Log a message object with the `DEBUG` level.

This method first checks if this category is `DEBUG` enabled by comparing the level of this category with the `DEBUG` level. If this category is `DEBUG` enabled, then it converts the message object (passed as parameter) to a string by invoking the appropriate `ObjectRenderer`. It then proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

**WARNING** Note that passing a `Throwable` to this method will print the name of the `Throwable` but no stack trace. To print a stack trace use the `debug(Object, Throwable)` form instead.

Parameters:
    message - the message object to log.

---

**debug**

```java
public void debug(Object message, Throwable t)
```

Log a message object with the `DEBUG` level including the stack trace of the `Throwable` `t` passed as parameter.

See `debug(Object)` form for more detailed information.

Parameters:
    message - the message object to log.
    t - the exception to log, including its stack trace.
error

public void error(Object message)

Log a message object with the ERROR Level.

This method first checks if this category is ERROR enabled by comparing the level of this category with ERROR Level. If this category is ERROR enabled, then it converts the message object passed as parameter to a string by invoking the appropriate ObjectRenderer. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

WARNING Note that passing a Throwable to this method will print the name of the Throwable but no stack trace. To print a stack trace use the error(Object, Throwable) form instead.

Parameters:
message - the message object to log

error

public void error(Object message, Throwable t)

Log a message object with the ERROR level including the stack trace of the Throwable t passed as parameter.

See error(Object) form for more detailed information.

Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.

exists

public static Logger exists(String name)
**Deprecation.** Please use `LogManager.exists(java.lang.String)` instead.

If the named category exists (in the default hierarchy) then it returns a reference to the category, otherwise it returns `null`.

**Since:**
0.8.5

---

**fatal**

public void **fatal**(Object message)

Log a message object with the **FATAL** Level.

This method first checks if this category is **FATAL** enabled by comparing the level of this category with **FATAL** Level. If the category is **FATAL** enabled, then it converts the message object passed as parameter to a string by invoking the appropriate `ObjectRenderer`. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

**WARNING** Note that passing a `Throwable` to this method will print the name of the Throwable but no stack trace. To print a stack trace use the `fatal(Object, Throwable)` form instead.

**Parameters:**
message - the message object to log

---

**fatal**

public void **fatal**(Object message, Throwable t)

Log a message object with the **FATAL** level including the stack trace of the `Throwable` t passed as parameter.

See `fatal(Object)` for more detailed information.
Parameters:
message - the message object to log.
t - the exception to log, including its stack trace.

forcedLog

protected void forcedLog(String fqcn,
Priority level,
Object message,
Throwable t)

This method creates a new logging event and logs the event without further checks.

getAdditivity

public boolean getAdditivity()

Get the additivity flag for this Category instance.

getAllAppenders

public Enumeration getAllAppenders()

Get the appenders contained in this category as an Enumeration. If no appenders can be found, then a NullEnumeration is returned.

Specified by:
getAllAppenders in interface AppenderAttachable

Returns:
Enumeration An enumeration of the appenders in this category.

g.getAppender

public Appender getAppender(String name)

Look for the appender named as name.
Return the appender with that name if in the list. Return null otherwise.

Specified by:
getAppender in interface AppenderAttachable

getEffectiveLevel

public Level getEffectiveLevel()

Starting from this category, search the category hierarchy for a non-null level and return it. Otherwise, return the level of the root category.

The Category class is designed so that this method executes as quickly as possible.

getChainedPriority

public Priority getChainedPriority()

Deprecated. Please use the getEffectiveLevel() method instead.

getCurrentCategories

public static Enumeration getCurrentCategories()

Deprecated. Please use LogManager.getCurrentLoggers() instead.

Returns all the currently defined categories in the default hierarchy as an Enumeration.

The root category is not included in the returned Enumeration.
getDefaultHierarchy

public static LoggerRepository getDefaultHierarchy()

    Deprecated. Please use LogManager.getLoggerRepository() instead.

    Return the default Hierarchy instance.
    Since:
        1.0

getHierarchy

public LoggerRepository getHierarchy()

    Deprecated. Please use getLoggerRepository() instead.

    Return the the Hierarchy where this Category instance is attached.
    Since:
        1.1

getLoggerRepository

public LoggerRepository getLoggerRepository()

    Return the the LoggerRepository where this Category is attached.
    Since:
        1.2

getInstance

public static Category getInstance(String name)

    Deprecated. Make sure to use Logger.getLogger(String) instead.
public static Category getInstance(Class clazz)

    Deprecated. Please make sure to use Logger.getLogger(Class) instead.

getName

public final String getName()

    Return the category name.

getParent

public final Category getParent()

    Returns the parent of this category. Note that the parent of a given category may change during the lifetime of the category.

    The root category will return null.

    Since:
    1.2

getLevel

public final Level getLevel()

    Returns the assigned Level, if any, for this Category.

    Returns:
    Level - the assigned Level, can be null.

getPriority

public final Level getPriority()

    Deprecated. Please use getLevel() instead.
getRoot

public static final Category getRoot()

Deprecated. Please use Logger.getRootLogger() instead.

generateBundle

public ResourceBundle getResourceBundle()

Return the inherited ResourceBundle for this category.

This method walks the hierarchy to find the appropriate resource bundle. It will return the resource bundle attached to the closest ancestor of this category, much like the way priorities are searched. In case there is no bundle in the hierarchy then null is returned.

Since:
0.9.0

generateBundleString

protected String getResourceBundleString(String key)

Returns the string resource corresponding to key in this category’s inherited resource bundle. See also getResourceBundle().

If the resource cannot be found, then an error message will be logged complaining about the missing resource.

info

public void info(Object message)

Log a message object with the INFO Level.
This method first checks if this category is INFO enabled by comparing the level of this category with INFO Level. If the category is INFO enabled, then it converts the message object passed as parameter to a string by invoking the appropriate ObjectRenderer. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

**WARNING** Note that passing a Throwable to this method will print the name of the Throwable but no stack trace. To print a stack trace use the info(Object, Throwable) form instead.

**Parameters:**
- message - the message object to log

---

**info**

```java
public void info(Object message, Throwable t)
```

Log a message object with the INFO level including the stack trace of the Throwable t passed as parameter.

See info(Object) for more detailed information.

**Parameters:**
- message - the message object to log.
- t - the exception to log, including its stack trace.

---

**isAttached**

```java
public boolean isAttached(Appender appender)
```

Is the appender passed as parameter attached to this category?

**Specified by:**
- isAttached in interface AppenderAttachable
**isDebugEnabled**

```java
public boolean isDebugEnabled()
```

Check whether this category is enabled for the DEBUG Level.

This function is intended to lessen the computational cost of disabled log debug statements.

For some `cat` Category object, when you write,

```java
cat.debug("This is entry number: " + i);
```

You incur the cost constructing the message, concatenatioin in this case, regardless of whether the message is logged or not.

If you are worried about speed, then you should write

```java
if(cat.isDebugEnabled()) {
    cat.debug("This is entry number: " + i);
}
```

This way you will not incur the cost of parameter construction if debugging is disabled for `cat`. On the other hand, if the `cat` is debug enabled, you will incur the cost of evaluating whether the category is debug enabled twice. Once in `isDebugEnabled` and once in the `debug`.

This is an insignificant overhead since evaluating a category takes about 1% of the time it takes to actually log.

**Returns:**

- boolean - true if this category is debug enabled, false otherwise.

---

**isEnabledFor**

```java
public boolean isEnabledFor(Priority level)
```

Check whether this category is enabled for a given Level passed as
isDebugEnabled

public boolean isDebugEnabled()

Check whether this category is enabled for the debug Level. See also isDebugEnabled().

Returns:
boolean - true if this category is enabled for level debug, false otherwise.

isInfoEnabled

public boolean isInfoEnabled()

Check whether this category is enabled for the info Level. See also isDebugEnabled().

Returns:
boolean - true if this category is enabled for level info, false otherwise.

l7dlog

public void l7dlog(Priority priority, String key, Throwable t)

Log a localized message. The user supplied parameter key is replaced by its localized version from the resource bundle.

Since:
0.8.4

See Also:
setResourceBundle(java.util.ResourceBundle)

l7dlog

public void l7dlog(Priority priority, String key, Object[] params, Throwable t)

Log a localized and parameterized message. First, the user supplied key is searched in the resource bundle. Next, the resulting pattern is formatted using MessageFormat.format(String, Object[]) method.
with the user supplied object array params.

Since:

0.8.4

log

public void log(Priority priority, Object message, Throwable t)

This generic form is intended to be used by wrappers.

log

public void log(Priority priority, Object message)

This generic form is intended to be used by wrappers.

log

public void log(String callerFQCN, Priority level, Object message, Throwable t)

This is the most generic printing method. It is intended to be invoked by wrapper classes.

Parameters:

callerFQCN - The wrapper class' fully qualified class name.
level - The level of the logging request.
message - The message of the logging request.
t - The throwable of the logging request, may be null.

removeAllAppenders

public void removeAllAppenders()
Remove all previously added appenders from this Category instance.

This is useful when re-reading configuration information.

**Specified by:**

`removeAllAppenders` in interface `AppenderAttachable`

---

**removeAppender**

`public void removeAppender(Appender appender)`

Remove the appender passed as parameter form the list of appenders.

**Specified by:**

`removeAppender` in interface `AppenderAttachable`

**Since:**

0.8.2

---

**removeAppender**

`public void removeAppender(String name)`

Remove the appender with the name passed as parameter form the list of appenders.

**Specified by:**

`removeAppender` in interface `AppenderAttachable`

**Since:**

0.8.2

---

**setAdditivity**

`public void setAdditivity(boolean additive)`

Set the additivity flag for this Category instance.

**Since:**
0.8.1

**setLevel**

public void **setLevel**([Level](#) level)

Set the level of this Category. If you are passing any of `Level.DEBUG`, `Level.INFO`, `Level.WARN`, `Level.ERROR`, `Level.FATAL` as a parameter, you need to case them as `Level`.

As in

```
logger.setLevel((Level) Level.DEBUG);
```

Null values are admitted.

**setPriority**

public void **setPriority**([Priority](#) priority)

*Deprecated. Please use** `setLevel(org.apache.log4j.Level)` **instead.*

Set the level of this Category.

Null values are admitted.

**setResourceBundle**

public void **setResourceBundle**([ResourceBundle](#) bundle)

Set the resource bundle to be used with localized logging methods `l7dlog(Priority,String,Throwable)` and `l7dlog(Priority,String,Object[],Throwable)`. *Since:* 0.8.4
shutdown

public static void shutdown()

**Deprecated.** Please use `LogManager.shutdown()` instead.

Calling this method will *safely* close and remove all appenders in all the categories including root contained in the default hierarchy.

Some appenders such as `SocketAppender` and `AsyncAppender` need to be closed before the application exists. Otherwise, pending logging events might be lost.

The `shutdown` method is careful to close nested appenders before closing regular appenders. This is allows configurations where a regular appender is attached to a category and again to a nested appender.

**Since:**
1.0

---

warn

public void warn(Object message)

Log a message object with the **WARN** Level.

This method first checks if this category is **WARN** enabled by comparing the level of this category with **WARN** Level. If the category is **WARN** enabled, then it converts the message object passed as parameter to a string by invoking the appropriate `ObjectRenderer`. It proceeds to call all the registered appenders in this category and also higher in the hierarchy depending on the value of the additivity flag.

**WARNING** Note that passing a `Throwable` to this method will print the name of the Throwable but no stack trace. To print a stack trace use the `warn(Object, Throwable)` form instead.
**Parameters:**

message - the message object to log.

---

**warn**

```java
public void warn(Object message, Throwable t)
```

Log a message with the WARN level including the stack trace of the `Throwable` `t` passed as parameter.

See `warn(Object)` for more detailed information.

**Parameters:**

message - the message object to log.

t - the exception to log, including its stack trace.

---

Copyright 2000-2005 Apache Software Foundation.
public class **ConsoleAppender**
extends **WriterAppender**

ConsoleAppender appends log events to System.out or System.err using a layout specified by the user. The default target is System.out.

**Since:**
1.1

**Author:**
Ceki Gülcü, Curt Arnold

## Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>SYSTEM_ERR</td>
</tr>
<tr>
<td>static String</td>
<td>SYSTEM_OUT</td>
</tr>
</tbody>
</table>

protected String **target**

Fields inherited from class org.apache.log4j.**WriterAppender**

encoding, immediateFlush, qw
Fields inherited from class org.apache.log4j.AppenderSkeleton
closed, errorHandler, headFilter, layout, name, tailFilter, threshold

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsoleAppender()</td>
<td>Constructs an unconfigured appender.</td>
</tr>
<tr>
<td>ConsoleAppender(Layout layout)</td>
<td>Creates a configured appender.</td>
</tr>
<tr>
<td>ConsoleAppender(Layout layout, String target)</td>
<td>Creates a configured appender.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void activateOptions()</td>
<td>Prepares the appender for use.</td>
</tr>
<tr>
<td>protected void closeWriter()</td>
<td>{@inheritDoc}</td>
</tr>
<tr>
<td>boolean getFollow()</td>
<td>Gets whether the appender honors reassignments of System.out or System.err made after configuration.</td>
</tr>
<tr>
<td>String getTarget()</td>
<td>Returns the current value of the Target property.</td>
</tr>
<tr>
<td>void setFollow(boolean newValue)</td>
<td>Sets whether the appender honors reassignments of System.out or System.err made after configuration.</td>
</tr>
<tr>
<td>void setTarget(String value)</td>
<td>Sets the value of the Target option.</td>
</tr>
</tbody>
</table>

Methods inherited from class org.apache.log4j.WriterAppender
append, checkEntryConditions, close, createWriter, getEncoding, getImmediateFlush, requiresLayout, reset, setEncoding, setErrorHandler, setImmediateFlush, setWriter, subAppend, writeFooter, writeHeader
Methods inherited from class org.apache.log4j.AppenderSkeleton
- addFilter, clearFilters, doAppend, finalize, getErrorHandler, getFilter, getFirstFilter, getLayout, getName, getThreshold, isAsSevereAsThreshold, setLayout, setName, setThreshold

Methods inherited from class java.lang.Object
- clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SYSTEM_OUT

public static final String SYSTEM_OUT

SYSTEM_ERR

public static final String SYSTEM_ERR

target

protected String target

Constructor Detail

ConsoleAppender

public ConsoleAppender()

Constructs an unconfigured appender.
public *ConsoleAppender*(*Layout* layout)

> Creates a configured appender.
> **Parameters:**
> layout - layout, may not be null.

---

**ConsoleAppender**

public *ConsoleAppender*(*Layout* layout, *String* target)

> Creates a configured appender.
> **Parameters:**
> layout - layout, may not be null.
> target - target, either "System.err" or "System.out".

### Method Detail

**setTarget**

public void **setTarget**(*String* value)

Sets the value of the **Target** option. Recognized values are "System.out" and "System.err". Any other value will be ignored.

---

**getTarget**

public *String* **getTarget**()

Returns the current value of the **Target** property. The default value of the option is "System.out". See also **setTarget(java.lang.String)**.

---

**setFollow**

public final void **setFollow**(boolean newValue)
Sets whether the appender honors reassignments of System.out or System.err made after configuration.

**Parameters:**

newValue - if true, appender will use value of System.out or System.err in force at the time when logging events are appended.

**Since:**

1.2.13

---

**getFollow**

`public final boolean getFollow()`

Gets whether the appender honors reassignments of System.out or System.err made after configuration.

**Returns:**

true if appender will use value of System.out or System.err in force at the time when logging events are appended.

**Since:**

1.2.13

---

**activateOptions**

`public void activateOptions()`

Prepares the appender for use.

**Overrides:**

activateOptions in class WriterAppender

---

**closeWriter**

`protected final void closeWriter()`

{@inheritdoc}

**Overrides:**

closeWriter in class WriterAppender
public class DailyRollingFileAppender extends FileAppender

DailyRollingFileAppender extends FileAppender so that the underlying file is rolled over at a user chosen frequency.

The rolling schedule is specified by the DatePattern option. This pattern should follow the SimpleDateFormat conventions. In particular, you must escape literal text within a pair of single quotes. A formatted version of the date pattern is used as the suffix for the rolled file name.

For example, if the File option is set to /foo/bar.log and the DatePattern set to '. 'yyyy-MM-dd, on 2001-02-16 at midnight, the logging file /foo/bar.log will be copied to /foo/bar.log.2001-02-16 and logging for 2001-02-17 will continue in /foo/bar.log until it rolls over the next day.

Is is possible to specify monthly, weekly, half-daily, daily, hourly, or minutely rollover schedules.

<table>
<thead>
<tr>
<th>DatePattern</th>
<th>Rollover schedule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rollover at the</td>
<td>At midnight of May 31st, 2002 /foo/bar.log will be copied to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.yyyy-MM</td>
<td>beginning of each month</td>
<td>/foo/bar.log.2002-05. Logging for the month of June will be output to /foo/bar.log until it is also rolled over the next month.</td>
</tr>
<tr>
<td>.yyyy-ww</td>
<td>Rollover at the first day of each week. The first day of the week depends on the locale.</td>
<td>Assuming the first day of the week is Sunday, on Saturday midnight, June 9th 2002, the file /foo/bar.log will be copied to /foo/bar.log.2002-23. Logging for the 24th week of 2002 will be output to /foo/bar.log until it is rolled over the next week.</td>
</tr>
<tr>
<td>.yyyy-MM-dd</td>
<td>Rollover at midnight each day.</td>
<td>At midnight, on March 8th, 2002, /foo/bar.log will be copied to /foo/bar.log.2002-03-08. Logging for the 9th day of March will be output to /foo/bar.log until it is rolled over the next day.</td>
</tr>
<tr>
<td>.yyyy-MM-dd-a</td>
<td>Rollover at midnight and midday of each day.</td>
<td>At noon, on March 9th, 2002, /foo/bar.log will be copied to /foo/bar.log.2002-03-09-AM. Logging for the afternoon of the 9th will be output to /foo/bar.log until it is rolled over at midnight.</td>
</tr>
<tr>
<td>.yyyy-MM-dd-HH</td>
<td>Rollover at the top of every hour.</td>
<td>At approximately 11:00.000 o'clock on March 9th, 2002, /foo/bar.log will be copied to /foo/bar.log.2002-03-09-10. Logging for the 11th hour of the 9th of March will be output to /foo/bar.log until it is rolled over at the beginning of the next hour.</td>
</tr>
<tr>
<td>.yyyy-MM-dd-HH-mm</td>
<td>Rollover at the beginning of every minute.</td>
<td>At approximately 11:23,000, on March 9th, 2001, /foo/bar.log will be copied to /foo/bar.log.2001-03-09-10-22. Logging for the minute of 11:23 (9th of March) will be output to /foo/bar.log until it is rolled over the next minute.</td>
</tr>
</tbody>
</table>

Do not use the colon ":" character in anywhere in the DatePattern option.
The text before the colon is interpreted as the protocol specification of a URL which is probably not what you want.

**Author:**
Eirik Lygre, Ceki Gülcü

### Fields inherited from class `org.apache.log4j.FileAppender`
- bufferedIO
- bufferSize
- fileAppend
- fileName

### Fields inherited from class `org.apache.log4j.WriterAppender`
- encoding
- immediateFlush
- qw

### Fields inherited from class `org.apache.log4j.AppenderSkeleton`
- closed
- errorHandler
- headFilter
- layout
- name
- tailFilter
- threshold

### Constructor Summary

**DailyRollingFileAppender()**
The default constructor does nothing.

**DailyRollingFileAppender**(Layout layout, String filename, String datePattern)

- Instantiate a DailyRollingFileAppender and open the file designated by filename.

### Method Summary

**void activateOptions()**
- If the value of **File** is not null, then **FileAppender.setFile(java.lang.String)** is called with the values of **File** and **Append** properties.

**String getDatePattern()**
- Returns the value of the **DatePattern** option.

**void setDatePattern(String pattern)**
- The **DatePattern** takes a string in the same format as expected by **SimpleDateFormat**.
protected void subAppend(LoggingEvent event)
This method differentiates DailyRollingFileAppender from its super class.

Methods inherited from class org.apache.log4j.FileAppender
closeFile, getAppend, getBufferedIO, getBufferSize, getFile,
reset, setAppend, setBufferedIO, setBufferSize, setFile, setFile,
setQWForFiles

Methods inherited from class org.apache.log4j.WriterAppender
append, checkEntryConditions, close, closeWriter, createWriter,
getEncoding, getImmediateFlush, requiresLayout, setEncoding,
setErrorHandler, setImmediateFlush, setWriter, writeFooter,
writeHeader

Methods inherited from class org.apache.log4j.AppenderSkeleton
addFilter, clearFilters, doAppend, finalize, getErrorHandler,
getFilter, getFirstFilter, getLayout, getName, getThreshold,
isAsSevereAsThreshold, setLayout, setName, setThreshold

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString,
wait, wait, wait

Constructor Detail

DailyRollingFileAppender

public DailyRollingFileAppender()

The default constructor does nothing.

DailyRollingFileAppender

public DailyRollingFileAppender(Layout layout,
String filename,
String datePattern)
throws IOException

Instantiate a DailyRollingFileAppender and open the file designated by filename. The opened filename will become the output destination for this appender.

Method Detail

setDatePattern

public void setDatePattern(String pattern)

The DatePattern takes a string in the same format as expected by SimpleDateFormat. This option determines the rollover schedule.

datePattern

public String getDatePattern()

Returns the value of the DatePattern option.

activateOptions

public void activateOptions()

Description copied from class: FileAppender
If the value of File is not null, then FileAppender.setFile(java.lang.String) is called with the values of File and Append properties.

Overrides:
activateOptions in class FileAppender

subAppend
protected void subAppend(LoggingEvent event)

This method differentiates DailyRollingFileAppender from its superclass.

Before actually logging, this method will check whether it is time to do a rollover. If it is, it will schedule the next rollover time and then rollover.

Overrides:
subAppend in class WriterAppender
public class FileAppender
extends WriterAppender

FileAppender appends log events to a file.

Support for java.io.Writer and console appending has been deprecated and then removed. See the replacement solutions: WriterAppender and ConsoleAppender.

Author: Ceki Gülcü

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected boolean bufferedIO</td>
</tr>
<tr>
<td>protected int bufferSize</td>
</tr>
<tr>
<td>protected boolean fileAppend</td>
</tr>
<tr>
<td>protected fileName</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

**Fields inherited from class org.apache.log4j.**

**WriterAppender**

| encoding, immediateFlush, qw |

**Fields inherited from class org.apache.log4j.**

**AppenderSkeleton**

| closed, errorHandler, headFilter, layout, name, tailFilter, threshold |

**Constructor Summary**

**FileAppender()**

The default constructor does not do anything.

**FileAppender(Layout layout, String filename)**

Instantiate a FileAppender and open the file designated by filename.

**FileAppender(Layout layout, String filename, boolean append)**

Instantiate a FileAppender and open the file designated by filename.

**FileAppender(Layout layout, String filename, boolean append, boolean bufferedIO, int bufferSize)**

Instantiate a FileAppender and open the file designated by filename.

**Method Summary**

<table>
<thead>
<tr>
<th>void</th>
<th>activateOptions()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the value of File is not null, then <code>setFile(java.lang.String)</code> is called with the values of File and Append properties.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected void</th>
<th>closeFile()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closes the previously opened file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getAppend()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the value of the Append option.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getBufferedIO()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Get the value of the BufferedIO option.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>int getBufferSize()</code></td>
<td>Get the size of the IO buffer.</td>
</tr>
<tr>
<td><code>String getFile()</code></td>
<td>Returns the value of the <strong>File</strong> option.</td>
</tr>
<tr>
<td><code>protected void reset()</code></td>
<td>Close any previously opened file and call the parent's reset.</td>
</tr>
<tr>
<td><code>void setAppend(boolean flag)</code></td>
<td>The <strong>Append</strong> option takes a boolean value.</td>
</tr>
<tr>
<td><code>void setBufferedIO(boolean bufferedIO)</code></td>
<td>The <strong>BufferedIO</strong> option takes a boolean value.</td>
</tr>
<tr>
<td><code>void setBufferSize(int bufferSize)</code></td>
<td>Set the size of the IO buffer.</td>
</tr>
<tr>
<td><code>void setFile(String file)</code></td>
<td>The <strong>File</strong> property takes a string value which should be the name of the file to append to.</td>
</tr>
<tr>
<td><code>void setFile(String fileName, boolean append, boolean bufferedIO, int bufferSize)</code></td>
<td>Sets and <em>opens</em> the file where the log output will go.</td>
</tr>
<tr>
<td><code>protected void setQWForFiles(Writer writer)</code></td>
<td>Sets the quiet writer being used.</td>
</tr>
</tbody>
</table>

**Methods inherited from class org.apache.log4j.WriterAppender**

- `append`, `checkEntryConditions`, `close`, `closeWriter`, `createWriter`, `getEncoding`, `getImmediateFlush`, `requiresLayout`, `setEncoding`, `setErrorHandler`, `setImmediateFlush`, `setWriter`, `subAppend`, `writeFooter`, `writeHeader`

**Methods inherited from class org.apache.log4j.AppenderSkeleton**

- `addFilter`, `clearFilters`, `doAppend`, `finalize`, `getErrorHandler`, `getFilter`, `getFirstFilter`, `getLayout`, `getName`, `getThreshold`, `isAsSevereAsThreshold`, `setLayout`, `setName`, `setThreshold`

**Methods inherited from class java.lang.Object**

- `clone`, `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
Field Detail

fileAppend

protected boolean fileAppend

Controls file truncatation. The default value for this variable is true, meaning that by default a FileAppender will append to an existing file and not truncate it.

This option is meaningful only if the FileAppender opens the file.

fileName

protected String fileName

The name of the log file.

bufferedIO

protected boolean bufferedIO

Do we do bufferedIO?

bufferSize

protected int bufferSize

Determines the size of IO buffer be. Default is 8K.

Constructor Detail
FileAppender

public FileAppender()

    The default constructor does not do anything.

FileAppender

public FileAppender(Layout layout,
                     String filename,
                     boolean append,
                     boolean bufferedIO,
                     int bufferSize)
    throws IOException

    Instantiate a FileAppender and open the file designated by filename. The opened filename will become the output destination for this appender.

    If the append parameter is true, the file will be appended to. Otherwise, the file designated by filename will be truncated before being opened.

    If the bufferedIO parameter is true, then buffered IO will be used to write to the output file.

FileAppender

public FileAppender(Layout layout,
                     String filename,
                     boolean append)
    throws IOException

    Instantiate a FileAppender and open the file designated by filename. The opened filename will become the output destination for this appender.

    If the append parameter is true, the file will be appended to. Otherwise, the file designated by filename will be truncated before
FileAppender

public FileAppender(Layout layout, String filename)
throws IOException

Instantiate a FileAppender and open the file designated by filename. The opened filename will become the output destination for this appender.

The file will be appended to.

Method Detail

setFile

public void setFile(String file)

The File property takes a string value which should be the name of the file to append to.

Note that the special values "System.out" or "System.err" are no longer honored.

Note: Actual opening of the file is made when activateOptions() is called, not when the options are set.

getAppend

public boolean getAppend()

Returns the value of the Append option.

getFile
public String getFile()

    Returns the value of the File option.

activateOptions

public void activateOptions()

    If the value of File is not null, then setFile(java.lang.String) is called with the values of File and Append properties.

    Overrides: activateOptions in class WriterAppender

Since:
0.8.1

closeFile

protected void closeFile()

    Closes the previously opened file.

getBufferedIO

public boolean getBufferedIO()

    Get the value of the BufferedIO option.

    BufferedIO will significatnly increase performance on heavily loaded systems.

getBufferSize

public int getBufferSize()

    Get the size of the IO buffer.
setAppend

public void setAppend(boolean flag)

The Append option takes a boolean value. It is set to true by default. If true, then File will be opened in append mode by setFile (see above). Otherwise, setFile will open File in truncate mode.

Note: Actual opening of the file is made when activateOptions() is called, not when the options are set.

setBufferedIO

public void setBufferedIO(boolean bufferedIO)

The BufferedIO option takes a boolean value. It is set to false by default. If true, then File will be opened and the resulting Writer wrapped around a BufferedWriter. BufferedIO will significantly increase performance on heavily loaded systems.

setBufferSize

public void setBufferSize(int bufferSize)

Set the size of the IO buffer.

setFile

public void setFile(String fileName, boolean append, boolean bufferedIO, int bufferSize)

throws IOException

Sets and opens the file where the log output will go. The specified file must be writable.
If there was already an opened file, then the previous file is closed first.

Do not use this method directly. To configure a FileAppender or one of its subclasses, set its properties one by one and then call activateOptions.

Parameters:
- fileName - The path to the log file.
- append - If true will append to fileName. Otherwise will truncate fileName.

---

**setQWForFiles**

protected void setQWForFiles(Writer writer)

Sets the quiet writer being used. This method is overridden by RollingFileAppender.

---

**reset**

protected void reset()

Close any previously opened file and call the parent's reset.

Overrides:
- reset in class WriterAppender
Java Class Hierarchy

```
java.lang.Object
|  +--org.apache.log4j.Hierarchy
```

All Implemented Interfaces:
   LoggerRepository, RendererSupport

public class Hierarchy
extends Object
implements LoggerRepository, RendererSupport

This class is specialized in retrieving loggers by name and also maintaining the logger hierarchy.

*The casual user does not have to deal with this class directly.*

The structure of the logger hierarchy is maintained by the `getLogger(java.lang.String)` method. The hierarchy is such that children link to their parent but parents do not have any pointers to their children. Moreover, loggers can be instantiated in any order, in particular descendant before ancestor.

In case a descendant is created before a particular ancestor, then it creates a provision node for the ancestor and adds itself to the provision node. Other descendants of the same ancestor add themselves to the previously created provision node.

**Author:**
   Ceki Gülcü

---

**Constructor Summary**

```
Hierarchy(Logger root)
   Create a new logger hierarchy.
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addHierarchyEventListener(HierarchyEventListener listener)</td>
<td>Add a HierarchyEventListener event to the repository.</td>
</tr>
<tr>
<td>void addRenderer(Class classToRender, ObjectRenderer or)</td>
<td>Add an object renderer for a specific class.</td>
</tr>
<tr>
<td>void clear()</td>
<td>This call will clear all logger definitions from the internal hashtable.</td>
</tr>
<tr>
<td>void emitNoAppenderWarning(Category cat)</td>
<td></td>
</tr>
<tr>
<td>Logger exists(String name)</td>
<td>Check if the named logger exists in the hierarchy.</td>
</tr>
<tr>
<td>void fireAddAppenderEvent(Category logger, Appender appender)</td>
<td></td>
</tr>
<tr>
<td>Enumeration getCurrentCategories()</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>Enumeration getCurrentLoggers()</td>
<td>Returns all the currently defined categories in this hierarchy as an Enumeration.</td>
</tr>
<tr>
<td>Logger getILogger(String name)</td>
<td>Return a new logger instance named as the first parameter using the default factory.</td>
</tr>
<tr>
<td>Logger getILogger(String name, LoggerFactory factory)</td>
<td>Return a new logger instance named as the first parameter using factory.</td>
</tr>
<tr>
<td>RendererMap getRendererMap()</td>
<td>Get the renderer map for this hierarchy.</td>
</tr>
<tr>
<td>Logger getRootLogger()</td>
<td>Get the root of this hierarchy.</td>
</tr>
<tr>
<td>Level getThreshold()</td>
<td>Returns a Level representation of the enable state.</td>
</tr>
<tr>
<td>boolean isEnabled(int level)</td>
<td>This method will return true if this repository is disabled.</td>
</tr>
</tbody>
</table>
for level object passed as parameter and false otherwise.

```java
void overrideAsNeeded(String override)
    Deprecated. Deprecated with no replacement.
```

```java
void resetConfiguration()
    Reset all values contained in this hierarchy instance to their default.
```

```java
void setDisableOverride(String override)
    Deprecated. Deprecated with no replacement.
```

```java
void setRenderer(Class renderedClass, ObjectRenderer renderer)
    Used by subclasses to add a renderer to the hierarchy passed as parameter.
```

```java
void setThreshold(Level l)
    Enable logging for logging requests with level l or higher.
```

```java
void setThreshold(String levelStr)
    The string form of setThreshold(Level).
```

```java
void shutdown()
    Shutting down a hierarchy will safely close and remove all appenders in all categories including the root logger.
```

### Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Constructor Detail

#### Hierarchy

```java
public Hierarchy(Logger root)
    Create a new logger hierarchy.
```

**Parameters:**

- `root` - The root of the new hierarchy.
addRenderer

public void addRenderer(Class classToRender, ObjectRenderer or)

Add an object renderer for a specific class.

addHierarchyEventListener

public void addHierarchyEventListener(HierarchyEventListener listener)

Description copied from interface: LoggerRepository
Add a HierarchyEventListener event to the repository.
Specified by:
addHierarchyEventListener in interface LoggerRepository

clear

public void clear()

This call will clear all logger definitions from the internal hashtable. Invoking this method will irrevocably mess up the logger hierarchy.

You should really know what you are doing before invoking this method.

Since:
0.9.0

emitNoAppenderWarning

public void emitNoAppenderWarning(Category cat)

Specified by:
emitNoAppenderWarning in interface LoggerRepository

exists

public Logger exists(String name)

Check if the named logger exists in the hierarchy. If so return its reference, otherwise returns null.

Specified by:
exists in interface LoggerRepository

Parameters:
name - The name of the logger to search for.

setThreshold

public void setThreshold(String levelStr)

The string form of setThreshold(Level).

Specified by:
setThreshold in interface LoggerRepository

setThreshold

public void setThreshold(Level l)

Enable logging for logging requests with level 1 or higher. By default all levels are enabled.

Specified by:
setThreshold in interface LoggerRepository

Parameters:
l - The minimum level for which logging requests are sent to their appenders.

fireAddAppenderEvent
public void fireAddAppenderEvent(Category logger, Appender appender)

Specified by:
    fireAddAppenderEvent in interface LoggerRepository

getThreshold

public Level getThreshold()

Returns a Level representation of the enable state.
Specified by:
    getThreshold in interface LoggerRepository
Since:
    1.2

getLogger

public Logger getLogger(String name)

Return a new logger instance named as the first parameter using the default factory.

If a logger of that name already exists, then it will be returned. Otherwise, a new logger will be instantiated and then linked with its existing ancestors as well as children.

Specified by:
    getLogger in interface LoggerRepository
Parameters:
    name - The name of the logger to retrieve.

getLogger

public Logger getLogger(String name, LoggerFactory factory)
Return a new logger instance named as the first parameter using factory.

If a logger of that name already exists, then it will be returned. Otherwise, a new logger will be instantiated by the factory parameter and linked with its existing ancestors as well as children.

**Specified by:**

`getLogger` in interface `LoggerRepository`

**Parameters:**

name - The name of the logger to retrieve.
factory - The factory that will make the new logger instance.

---

**getCurrentLoggers**

```java
public Enumeration getCurrentLoggers()
```

Returns all the currently defined categories in this hierarchy as an `Enumeration`.

The root logger is *not* included in the returned `Enumeration`.

**Specified by:**

`getCurrentLoggers` in interface `LoggerRepository`

---

**getCurrentCategories**

```java
public Enumeration getCurrentCategories()
```

**Deprecated. Please use `getCurrentLoggers()` instead.**

**Description copied from interface: `LoggerRepository`**

Deprecated. Please use `LoggerRepository.getCurrentLoggers()` instead.

**Specified by:**

`getCurrentCategories` in interface `LoggerRepository`
getRendererMap

public RendererMap getRendererMap()

Get the renderer map for this hierarchy.

Specified by:
getRendererMap in interface RendererSupport

getRootLogger

public Logger getRootLogger()

Get the root of this hierarchy.

Specified by:
getRootLogger in interface LoggerRepository

Since:
0.9.0

isDisabled

public boolean isDisabled(int level)

This method will return true if this repository is disabled for level object passed as parameter and false otherwise. See also the threshold method.

Specified by:
isDisabled in interface LoggerRepository

overrideAsNeeded

public void overrideAsNeeded(String override)

Deprecated. Deprecated with no replacement.

resetConfiguration
public void resetConfiguration()

Reset all values contained in this hierarchy instance to their default. This removes all appenders from all categories, sets the level of all non-root categories to null, sets their additivity flag to true and sets the level of the root logger to DEBUG. Moreover, message disabling is set its default "off" value.

Existing categories are not removed. They are just reset.

This method should be used sparingly and with care as it will block all logging until it is completed.

Specified by:  
resetConfiguration in interface LoggerRepository  
Since:  
0.8.5

setDisableOverride

public void setDisableOverride(String override)

Deprecated. Deprecated with no replacement.

Does nothing.

setRenderer

public void setRenderer(Class renderedClass, ObjectRenderer renderer)

Used by subclasses to add a renderer to the hierarchy passed as parameter.
Specified by:  
setRenderer in interface RendererSupport

shutdown
public void shutdown()

Shutting down a hierarchy will safely close and remove all appenders in all categories including the root logger.

Some appenders such as SocketAppender and AsyncAppender need to be closed before the application exists. Otherwise, pending logging events might be lost.

The shutdown method is careful to close nested appenders before closing regular appenders. This is allows configurations where a regular appender is attached to a logger and again to a nested appender.

Specified by:
- shutdown in interface LoggerRepository

Since:
- 1.0
org.apache.log4j  Class HTMLLayout

java.lang.Object
|  +-- org.apache.log4j.Layout
|      |  +-- org.apache.log4j.HTMLLayout

All Implemented Interfaces:
    OptionHandler

public class HTMLLayout
extends Layout

This layout outputs events in a HTML table.

Author:
    Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>protected int BUF_SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String LOCATION_INFO_OPTION</td>
</tr>
<tr>
<td>Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
<tr>
<td>protected int MAX_CAPACITY</td>
</tr>
<tr>
<td>static String TITLE_OPTION</td>
</tr>
<tr>
<td>A string constant used in naming the option for setting the the HTML document title.</td>
</tr>
</tbody>
</table>

Fields inherited from class org.apache.log4j.Layout
LINE_SEP, LINE_SEP_LEN
## Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HTMLLayout()</code></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void activateOptions()</code></td>
<td>No options to activate.</td>
</tr>
<tr>
<td><code>String format(LoggingEvent event)</code></td>
<td>Implement this method to create your own layout format.</td>
</tr>
<tr>
<td><code>String getContentType()</code></td>
<td>Returns the content type output by this layout, i.e. “text/html”.</td>
</tr>
<tr>
<td><code>String getFooter()</code></td>
<td>Returns the appropriate HTML footers.</td>
</tr>
<tr>
<td><code>String getHeader()</code></td>
<td>Returns appropriate HTML headers.</td>
</tr>
<tr>
<td><code>boolean getLocationInfo()</code></td>
<td>Returns the current value of the <code>LocationInfo</code> option.</td>
</tr>
<tr>
<td><code>String getTitle()</code></td>
<td>Returns the current value of the <code>Title</code> option.</td>
</tr>
<tr>
<td><code>boolean ignoresThrowable()</code></td>
<td>The HTML layout handles the throwable contained in logging events.</td>
</tr>
<tr>
<td><code>void setLocationInfo(boolean flag)</code></td>
<td>The <code>LocationInfo</code> option takes a boolean value.</td>
</tr>
<tr>
<td><code>void setTitle(String title)</code></td>
<td>The <code>Title</code> option takes a String value.</td>
</tr>
</tbody>
</table>

## Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`
**Field Detail**

**BUF_SIZE**

protected final int BUF_SIZE

---

**MAX_CAPACITY**

protected final int MAX_CAPACITY

---

**LOCATION_INFO_OPTION**

public static final String LOCATION_INFO_OPTION

**Deprecated.** Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

A string constant used in naming the option for setting the the location information flag. Current value of this string constant is LocationInfo.

Note that all option keys are case sensitive.

---

**TITLE_OPTION**

public static final String TITLE_OPTION

A string constant used in naming the option for setting the the HTML document title. Current value of this string constant is Title.

---

**Constructor Detail**

**HTMLLayout**
public HTMLLayout()

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
</table>

**setLocationInfo**

public void setLocationInfo(boolean flag)

The **LocationInfo** option takes a boolean value. By default, it is set to false which means there will be no location information output by this layout. If the the option is set to true, then the file name and line number of the statement at the origin of the log statement will be output.

If you are embedding this layout within an [SMTPAppender](#) then make sure to set the **LocationInfo** option of that appender as well.

**getLocationInfo**

public boolean getLocationInfo()

Returns the current value of the **LocationInfo** option.

**setTitle**

public void setTitle(String title)

The **Title** option takes a String value. This option sets the document title of the generated HTML document.

Defaults to 'Log4J Log Messages'.

**getTitle**

public String getTitle()
Returns the current value of the **Title** option.

---

**getContentType**

```java
def getContentType()
  Returns the content type output by this layout, i.e "text/html".
  **Overrides:**
  getContentType in class Layout
```

---

**activateOptions**

```java
def activateOptions()
  No options to activate.
```

---

**format**

```java
public String format(LoggingEvent event)
  Description copied from class: Layout
  Implement this method to create your own layout format.
  **Overrides:**
  format in class Layout
```

---

**getHeader**

```java
public String getHeader()
  Returns appropriate HTML headers.
  **Overrides:**
  getHeader in class Layout
```

---

**getFooter**
public String getFooter()

    Returns the appropriate HTML footers.
    Overrides:
        getFooter in class Layout

ignoresThrowable

public boolean ignoresThrowable()

    The HTML layout handles the throwable contained in logging events.
    Hence, this method return false.
    Overrides:
        ignoresThrowable in class Layout
Class Layout

java.lang.Object
    +-- org.apache.log4j.Layout

All Implemented Interfaces:
    OptionHandler

Direct Known Subclasses:
    DateLayout, HTMLLayout, PatternLayout, SimpleLayout, XMLLayout

public abstract class Layout
extends Object
implements OptionHandler

Extend this abstract class to create your own log layout format.

Author:
    Ceki Gülcü

Field Summary

| static String | LINE_SEP |
| static int   | LINE_SEP_LEN |

Constructor Summary

Layout()
abstract **String** format(**LoggingEvent** event)
Implement this method to create your own layout format.

**getContentType()**
Returns the content type output by this layout.

**getFooter()**
Returns the footer for the layout format.

**getHeader()**
Returns the header for the layout format.

abstract **boolean** ignoresThrowable()
If the layout handles the throwable object contained within **LoggingEvent**, then the layout should return false.

---

**Methods inherited from class java.lang.****Object**
**clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait**

**Methods inherited from interface org.apache.log4j.spi.****OptionHandler**
**activateOptions**

---

**Field Detail**

**LINE_SEP**

public static final **String** **LINE_SEP**

**LINE_SEP_LEN**

public static final **int** **LINE_SEP_LEN**

**Constructor Detail**

**Layout**
public Layout()

Method Detail

format

public abstract String format(LoggingEvent event)

Implements this method to create your own layout format.

getContentType

public String getContentType()

Returns the content type output by this layout. The base class returns "text/plain".

getHeader

public String getHeader()

Returns the header for the layout format. The base class returns null.

getFooter

public String getFooter()

Returns the footer for the layout format. The base class returns null.

ignoresThrowable

public abstract boolean ignoresThrowable()

If the layout handles the throwable object contained within
LoggingEvent, then the layout should return false. Otherwise, if the layout ignores throwable object, then the layout should return true.

The SimpleLayout, TTCLayout, PatternLayout all return true. The XMLLayout returns false.

Since:
0.8.4
public class Level
extends Priority
implements Serializable

Defines the minimum set of levels recognized by the system, that is OFF, FATAL, ERROR, WARN, INFO/DEBUG and ALL.

The Level class may be subclassed to define a larger level set.

Author:
Ceki Gülcü

See Also:
Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>static Level</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The ALL has the lowest possible rank and is intended to turn on all logging.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static Level</th>
<th>DEBUG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The DEBUG Level designates fine-grained informational events that are most useful to debug an application.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static Level</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The ERROR level designates error events that might still allow the application to continue running.</td>
</tr>
<tr>
<td>static Level</td>
<td>FATAL</td>
</tr>
<tr>
<td>static Level</td>
<td>INFO</td>
</tr>
<tr>
<td>static Level</td>
<td>OFF</td>
</tr>
<tr>
<td>static Level</td>
<td>TRACE</td>
</tr>
<tr>
<td>static int</td>
<td>TRACE_INT</td>
</tr>
<tr>
<td>static Level</td>
<td>WARN</td>
</tr>
</tbody>
</table>

Fields inherited from class org.apache.log4j.Priority

ALL_INT, DEBUG_INT, ERROR_INT, FATAL_INT, INFO_INT, OFF_INT, WARN_INT

Constructor Summary

protected Level(int level, String levelStr, int syslogEquivalent)

Instantiate a Level object.

Method Summary

static Level toLevel(int val)

Convert an integer passed as argument to a level.

static Level toLevel(int val, Level defaultLevel)

Convert an integer passed as argument to a level.

static Level toLevel(String sArg)

Convert the string passed as argument to a level.
static Level toLevel(String sArg, Level defaultLevel)
Convert the string passed as argument to a level.

Methods inherited from class org.apache.log4j.Priority
equals, getAllPossiblePriorities, getSyslogEquivalent,
isGreaterOrEqual, toInt, toPriority, toPriority, toPriority,
toPriority, toString

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, wait,
wait, wait

Field Detail

TRACE_INT

public static final int TRACE_INT

TRACE level integer value.
Since:
1.2.12

OFF

public static final Level OFF

The OFF has the highest possible rank and is intended to turn off logging.

FATAL

public static final Level FATAL

The FATAL level designates very severe error events that will presumably lead the application to abort.
**ERROR**

public static final Level ERROR

The ERROR level designates error events that might still allow the application to continue running.

---

**WARN**

public static final Level WARN

The WARN level designates potentially harmful situations.

---

**INFO**

public static final Level INFO

The INFO level designates informational messages that highlight the progress of the application at coarse-grained level.

---

**DEBUG**

public static final Level DEBUG

The DEBUG Level designates fine-grained informational events that are most useful to debug an application.

---

**TRACE**

public static final Level TRACE

The TRACE Level designates finer-grained informational events than the DEBUG Since:
ALL

public static final Level ALL

The ALL has the lowest possible rank and is intended to turn on all logging.

Constructor Detail

Level

protected Level(int level,
               String levelStr,
               int syslogEquivalent)

Instantiate a Level object.

Method Detail

toLevel

public static Level toLevel(String sArg)

Convert the string passed as argument to a level. If the conversion fails, then this method returns DEBUG.

toLevel

public static Level toLevel(int val)

Convert an integer passed as argument to a level. If the conversion fails, then this method returns DEBUG.
public static Level toLevel(int val, Level defaultLevel)

Convert an integer passed as argument to a level. If the conversion fails, then this method returns the specified default.

toLevel

public static Level toLevel(String sArg, Level defaultLevel)

Convert the string passed as argument to a level. If the conversion fails, then this method returns the value of defaultLevel.
org.apache.log4j  Class Logger

java.lang.Object
   |
   +-- org.apache.log4j.Category
      |   +-- org.apache.log4j.Logger

All Implemented Interfaces:
   AppenderAttachable

Direct Known Subclasses:
   RootCategory, RootLogger

public class Logger
extends Category

This is the central class in the log4j package. Most logging operations, except configuration, are done through this class.

Since:
   log4j 1.2
Author:
   Ceki Gülcü

Fields inherited from class org.apache.log4j.Category
additive, level, name, parent, repository, resourceBundle

Constructor Summary

protected Logger(String name)

Method Summary

getLogger(Class clazz)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Logger</td>
<td>Shorthand for getLogger(clazz.getName()).</td>
</tr>
<tr>
<td>static Logger(String name)</td>
<td>Retrieve a logger named according to the value of the name parameter.</td>
</tr>
<tr>
<td>static Logger(String name, LoggerFactory factory)</td>
<td>Like getLogger(String) except that the type of logger instantiated depends on the type returned by the LoggerFactory.makeNewLoggerInstance(java.lang.String) method of the factory parameter.</td>
</tr>
<tr>
<td>static Logger()</td>
<td>Return the root logger for the current logger repository.</td>
</tr>
<tr>
<td>boolean isTraceEnabled()</td>
<td>Check whether this category is enabled for the TRACE Level.</td>
</tr>
<tr>
<td>void trace(Object message)</td>
<td>Log a message object with the TRACE level.</td>
</tr>
<tr>
<td>void trace(Object message, Throwable t)</td>
<td>Log a message object with the TRACE level including the stack trace of the Throwable passed as parameter.</td>
</tr>
</tbody>
</table>

Methods inherited from class org.apache.log4j.Category

- addAppender
- assertLog
- callAppenders
- debug
- debug
- error
- error
- exists
- fatal
- fatal
- forcedLog
- getAdditivity
- getAllAppenders
- getAppender
- getChainedPriority
- getCurrentCategories
- getDefaultHierarchy
- getEffectiveLevel
- getHierarchy
- getInstance
- getInstance
- getLevel
- getLoggerRepository
- getName
- getParent
- getPriority
- getResourceBundle
- getResourceBundleString
- getRoot
- info
- info
- isAttached
- isDebugEnabled
- isEnabledFor
- isEnabled
- log
- log
- log
- log
- log
- removeAllAppenders
- removeAppender
- removeAppender
- setAdditivity
- setLevel
- setPriority
- setResourceBundle
- shutdown
- warn
- warn

Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait
Constructor Detail

**Logger**

protected `Logger(String name)`

Method Detail

**getLogger**

public static `Logger getLogger(String name)`

Retrieve a logger named according to the value of the `name` parameter. If the named logger already exists, then the existing instance will be returned. Otherwise, a new instance is created.

By default, loggers do not have a set level but inherit it from their nearest ancestor with a set level. This is one of the central features of log4j.

**Parameters:**

- `name` - The name of the logger to retrieve.

**getLogger**

public static `Logger getLogger(Class clazz)`

Shorthand for `getLogger(clazz.getName())`.

**Parameters:**

- `clazz` - The name of `clazz` will be used as the name of the logger to retrieve. See `getLogger(String)` for more detailed information.

**getRootLogger**
public static Logger getRootLogger()

Return the root logger for the current logger repository.

The Logger.getName() method for the root logger always returns string value: "root". However, calling Logger.getLogger("root") does not retrieve the root logger but a logger just under root named "root".

In other words, calling this method is the only way to retrieve the root logger.

getLogger

public static Logger getLogger(String name, LoggerFactory factory)

Like getLogger(String) except that the type of logger instantiated depends on the type returned by the LoggerFactory.makeNewLoggerInstance(java.lang.String) method of the factory parameter.

This method is intended to be used by sub-classes.

Parameters:
name - The name of the logger to retrieve.
factory - A LoggerFactory implementation that will actually create a new Instance.

Since: 0.8.5

trace

public void trace(Object message)

Log a message object with the TRACE level.

Parameters:
message - the message object to log.

Since:
**trace**

public void **trace**(Object message,
  Throwable t)

Log a message object with the TRACE level including the stack trace of the Throwable t passed as parameter.

See **Category.debug(Object)** form for more detailed information.

**Parameters:**
- message - the message object to log.
- t - the exception to log, including its stack trace.

**Since:**
  1.2.12

**isTraceEnabled**

public boolean **isTraceEnabled**()

Check whether this category is enabled for the TRACE Level.

**Returns:**
- boolean - true if this category is enabled for level TRACE, false otherwise.

**Since:**
  1.2.12
public class LogManager extends Object

Use the LogManager class to retrieve Logger instances or to operate on the current LoggerRepository. When the LogManager class is loaded into memory the default initalization procedure is inititated. The default intialization procedure is described in the short log4j manual.

Author:  
Ceki Gülcü

| Field Summary |
|--------------|--------------------------|
| static String CONFIGURATOR_CLASS_KEY | Deprecated. This variable is for internal use only. It will become private in future versions. |
| static String DEFAULT_CONFIGURATION_FILE | Deprecated. This variable is for internal use only. It will become package protected in future versions. |
| static String DEFAULT_CONFIGURATION_KEY | Deprecated. This variable is for internal use only. It will become private in future versions. |
| static String DEFAULT_INIT_OVERRIDE_KEY | Deprecated. This variable is for internal use only. It will become private in future versions. |

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogManager()</td>
</tr>
</tbody>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static Logger exists(String name)</code></td>
<td></td>
<td>Retrieve the appropriate <code>Logger</code> instance.</td>
</tr>
<tr>
<td><code>static Enumeration getCurrentLoggers()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>static Logger getLogger(Class clazz)</code></td>
<td></td>
<td>Retrieve the appropriate <code>Logger</code> instance.</td>
</tr>
<tr>
<td><code>static Logger getLogger(String name)</code></td>
<td></td>
<td>Retrieve the appropriate <code>Logger</code> instance.</td>
</tr>
<tr>
<td><code>static Logger getLogger(String name, LoggerFactory factory)</code></td>
<td></td>
<td>Retrieve the appropriate <code>Logger</code> instance.</td>
</tr>
<tr>
<td><code>static LoggerRepository getLoggerRepository()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>static Logger getRootLogger()</code></td>
<td></td>
<td>Retrieve the appropriate root logger.</td>
</tr>
<tr>
<td><code>static void resetConfiguration()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>static void setRepositorySelector(RepositorySelector selector, Object guard)</code></td>
<td></td>
<td>Sets <code>LoggerFactory</code> but only if the correct <code>guard</code> is passed as parameter.</td>
</tr>
<tr>
<td><code>static void shutdown()</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Methods inherited from class java.lang.Object

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
DEFAULT_CONFIGURATION_FILE

public static final String DEFAULT_CONFIGURATION_FILE

*Deprecated.* This variable is for internal use only. It will become package protected in future versions.

DEFAULT_CONFIGURATION_KEY

public static final String DEFAULT_CONFIGURATION_KEY

*Deprecated.* This variable is for internal use only. It will become private in future versions.

CONFIGURATOR_CLASS_KEY

public static final String CONFIGURATOR_CLASS_KEY

*Deprecated.* This variable is for internal use only. It will become private in future versions.

DEFAULT_INIT_OVERRIDE_KEY

public static final String DEFAULT_INIT_OVERRIDE_KEY

*Deprecated.* This variable is for internal use only. It will become private in future versions.

Constructor Detail

LogManager

public LogManager()
### setRepositorySelector

**Method Signature**
```
public static void setRepositorySelector(RepositorySelector selector, Object guard) throws IllegalArgumentException
```

- **Sets** LoggerFactory but only if the correct `guard` is passed as parameter.

- Initially the guard is null. If the guard is `null`, then invoking this method sets the logger factory and the guard. Following invocations will throw a `IllegalArgumentException`, unless the previously set guard is passed as the second parameter.

- This allows a high-level component to set the `RepositorySelector` used by the LogManager.

- For example, when tomcat starts it will be able to install its own repository selector. However, if and when Tomcat is embedded within JBoss, then JBoss will install its own repository selector and Tomcat will use the repository selector set by its container, JBoss.

### getLoggerRepository

**Method Signature**
```
public static LoggerRepository getLoggerRepository()
```

###getRootLogger

**Method Signature**
```
public static Logger getRootLogger()
```

- Retrieve the appropriate root logger.

### getLoggers

**Method Signature**
```
public static Logger getLogger(String name)
```
Retrieve the appropriate Logger instance.

**getLogger**

public static Logger getLogger(Class clazz)

Retrieve the appropriate Logger instance.

**getLogger**

public static Logger getLogger(String name, LoggerFactory factory)

Retrieve the appropriate Logger instance.

**exists**

public static Logger exists(String name)

**getCurrentLoggers**

public static Enumeration getCurrentLoggers()

**shutdown**

public static void shutdown()

**resetConfiguration**

public static void resetConfiguration()
<table>
<thead>
<tr>
<th>SUMMARY</th>
<th>FIELD</th>
<th>CONSTR</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETAIL</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Overview | Package | Use | Tree | Deprecated | Index | Help | Log4j 1.2.14
PREV CLASS | NEXT CLASS | SUMMARY: INNER | FIELD | CONSTR | METHOD | FRAMES | NO FRAMES | DETAIL: FIELD | CONSTR | METHOD
public class MDC
extends Object

The MDC class is similar to the NDC class except that it is based on a map instead of a stack. It provides mapped diagnostic contexts. A Mapped Diagnostic Context, or MDC in short, is an instrument for distinguishing interleaved log output from different sources. Log output is typically interleaved when a server handles multiple clients near-simultaneously.

The MDC is managed on a per thread basis. A child thread automatically inherits a copy of the mapped diagnostic context of its parent.

The MDC class requires JDK 1.2 or above. Under JDK 1.1 the MDC will always return empty values but otherwise will not affect or harm your application.

Since:
  1.2
Author:
  Ceki Gülcü

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Object get(String key)</td>
<td>Get the context identified by the key parameter.</td>
</tr>
<tr>
<td>static Hashtable getContext()</td>
<td>Get the current thread's MDC as a hashtable.</td>
</tr>
<tr>
<td>static void put(String key, Object o)</td>
<td>Put a context value (the o parameter) as identified</td>
</tr>
</tbody>
</table>
static void put(String key, Object o)

Put a context value (the o parameter) as identified with the key parameter into the current thread's context map.

If the current thread does not have a context map it is created as a side effect.

public static Object get(String key)

Get the context identified by the key parameter.

This method has no side effects.

public static void remove(String key)
Remove the context identified by the key parameter.

**getContext**

```java
public static Hashtable getContext()
```

Get the current thread's MDC as a hashtable. This method is intended to be used internally.

---

Copyright 2000-2005 Apache Software Foundation.
public class NDC
extends Object

The NDC class implements nested diagnostic contexts as defined by Neil Harrison in the article "Patterns for Logging Diagnostic Messages" part of the book "Pattern Languages of Program Design 3" edited by Martin et al.

A Nested Diagnostic Context, or NDC in short, is an instrument to distinguish interleaved log output from different sources. Log output is typically interleaved when a server handles multiple clients near-simultaneously.

Interleaved log output can still be meaningful if each log entry from different contexts had a distinctive stamp. This is where NDCs come into play.

**Note that NDCs are managed on a per thread basis.** NDC operations such as push, pop(), clear(), getDepth() and setMaxDepth(int) affect the NDC of the current thread only. NDCs of other threads remain unaffected.

For example, a servlet can build a per client request NDC consisting the clients host name and other information contained in the the request. Cookies are another source of distinctive information. To build an NDC one uses the push operation. Simply put,

- Contexts can be nested.
- When entering a context, call NDC.push. As a side effect, if there is no nested diagnostic context for the current thread, this method will create it.
• When leaving a context, call \texttt{NDC.pop}.

• When exiting a thread make sure to call \texttt{NDC.remove()}.

There is no penalty for forgetting to match each \texttt{push} operation with a corresponding \texttt{pop}, except the obvious mismatch between the real application context and the context set in the NDC.

If configured to do so, \texttt{PatternLayout} and \texttt{TTCLayout} instances automatically retrieve the nested diagnostic context for the current thread without any user intervention. Hence, even if a servlet is serving multiple clients simultaneously, the logs emanating from the same code (belonging to the same category) can still be distinguished because each client request will have a different NDC tag.

Heavy duty systems should call the \texttt{remove()} method when leaving the run method of a thread. This ensures that the memory used by the thread can be freed by the Java garbage collector. There is a mechanism to lazily remove references to dead threads. In practice, this means that you can be a little sloppy and sometimes forget to call \texttt{remove()} before exiting a thread.

A thread may inherit the nested diagnostic context of another (possibly parent) thread using the \texttt{inherit} method. A thread may obtain a copy of its NDC with the \texttt{cloneStack} method and pass the reference to any other thread, in particular to a child.

\textbf{Since:} 0.7.0

\textbf{Author:} Ceki Gülcü

\begin{center}
\begin{tabular}{||p{2cm} | p{12cm}||}
\hline
\textbf{Method Summary} & \\
\hline
\texttt{static void clear()} & Clear any nested diagnostic information if any. \\
\hline
\texttt{static Stack cloneStack()} & Clone the diagnostic context for the current thread. \\
\hline
\end{tabular}
\end{center}
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String get()</td>
<td>Never use this method directly, use the LoggingEvent.getNDC() method instead.</td>
</tr>
<tr>
<td>static int getDepth()</td>
<td>Get the current nesting depth of this diagnostic context.</td>
</tr>
<tr>
<td>static void inherit(Stack stack)</td>
<td>Inherit the diagnostic context of another thread.</td>
</tr>
<tr>
<td>static String peek()</td>
<td>Looks at the last diagnostic context at the top of this NDC without removing it.</td>
</tr>
<tr>
<td>static String pop()</td>
<td>Clients should call this method before leaving a diagnostic context.</td>
</tr>
<tr>
<td>static void push(String message)</td>
<td>Push new diagnostic context information for the current thread.</td>
</tr>
<tr>
<td>static void remove()</td>
<td>Remove the diagnostic context for this thread.</td>
</tr>
<tr>
<td>static void setMaxDepth(int maxDepth)</td>
<td>Set maximum depth of this diagnostic context.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

---

Method Detail

**clear**

public static void clear()

Clear any nested diagnostic information if any. This method is useful in cases where the same thread can be potentially used over and over in different unrelated contexts.
This method is equivalent to calling the `setMaxDepth(int)` method with a zero `maxDepth` argument.

**Since:**
0.8.4c

---

### cloneStack

```java
public static Stack cloneStack()
```

Clone the diagnostic context for the current thread.

Internally a diagnostic context is represented as a stack. A given thread can supply the stack (i.e. diagnostic context) to a child thread so that the child can inherit the parent thread's diagnostic context.

The child thread uses the `inherit` method to inherit the parent's diagnostic context.

**Returns:**
Stack A clone of the current thread's diagnostic context.

---

### inherit

```java
public static void inherit(Stack stack)
```

Inherit the diagnostic context of another thread.

The parent thread can obtain a reference to its diagnostic context using the `cloneStack()` method. It should communicate this information to its child so that it may inherit the parent's diagnostic context.

The parent's diagnostic context is cloned before being inherited. In other words, once inherited, the two diagnostic contexts can be managed independently.

In java, a child thread cannot obtain a reference to its parent, unless
it is directly handed the reference. Consequently, there is no client-transparent way of inheriting diagnostic contexts. Do you know any solution to this problem?

**Parameters:**
- **stack** - The diagnostic context of the parent thread.

---

### get

**public static String get()**

*Never use this method directly, use the LoggingEvent.getNDC() method instead.*

---

### getDepth

**public static int getDepth()**

Get the current nesting depth of this diagnostic context.

**Since:**
- 0.7.5

**See Also:**
- setMaxDepth(int)

---

### pop

**public static String pop()**

Clients should call this method before leaving a diagnostic context.

The returned value is the value that was pushed last. If no context is available, then the empty string "" is returned.

**Returns:**
- String The innermost diagnostic context.
peek

public static String peek()

Looks at the last diagnostic context at the top of this NDC without removing it.

The returned value is the value that was pushed last. If no context is available, then the empty string """" is returned.

>Returns:
String The innermost diagnostic context.

push

public static void push(String message)

Push new diagnostic context information for the current thread.

The contents of the message parameter is determined solely by the client.

Parameters:
message - The new diagnostic context information.

remove

public static void remove()

Remove the diagnostic context for this thread.

Each thread that created a diagnostic context by calling push(java.lang.String) should call this method before exiting. Otherwise, the memory used by the thread cannot be reclaimed by the VM.

As this is such an important problem in heavy duty systems and because it is difficult to always guarantee that the remove method is
called before exiting a thread, this method has been augmented to lazily remove references to dead threads. In practice, this means that you can be a little sloppy and occasionally forget to call remove() before exiting a thread. However, you must call remove sometime. If you never call it, then your application is sure to run out of memory.

**setMaxDepth**

```java
public static void setMaxDepth(int maxDepth)
```

Set maximum depth of this diagnostic context. If the current depth is smaller or equal to `maxDepth`, then no action is taken.

This method is a convenient alternative to multiple `pop()` calls. Moreover, it is often the case that at the end of complex call sequences, the depth of the NDC is unpredictable. The `setMaxDepth` method circumvents this problem.

For example, the combination

```java
void foo() {
    int depth = NDC.getDepth();
    ... complex sequence of calls
    NDC.setMaxDepth(depth);
}
```

ensures that between the entry and exit of `foo` the depth of the diagnostic stack is conserved.

**Since:**

0.7.5

**See Also:**

`getDepth()`
Copyright 2000-2005 Apache Software Foundation.
public class PatternLayout extends Layout

A flexible layout configurable with pattern string.

The goal of this class is to format a LoggingEvent and return the results as a String. The results depend on the conversion pattern.

The conversion pattern is closely related to the conversion pattern of the printf function in C. A conversion pattern is composed of literal text and format control expressions called conversion specifiers.

You are free to insert any literal text within the conversion pattern.

Each conversion specifier starts with a percent sign (%) and is followed by optional format modifiers and a conversion character. The conversion character specifies the type of data, e.g. category, priority, date, thread name. The format modifiers control such things as field width, padding, left and right justification. The following is a simple example.

Let the conversion pattern be "%-5p [%t]: %m%n" and assume that the log4j environment was set to use a PatternLayout. Then the statements

```java
Category root = Category.getRoot();
root.debug("Message 1");
root.warn("Message 2");
```

would yield the output
Note that there is no explicit separator between text and conversion specifiers. The pattern parser knows when it has reached the end of a conversion specifier when it reads a conversion character. In the example above the conversion specifier \%-5p means the priority of the logging event should be left justified to a width of five characters. The recognized conversion characters are

<table>
<thead>
<tr>
<th>Conversion Character</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Used to output the category of the logging event. The category conversion specifier can be optionally followed by <em>precision specifier</em>, that is a decimal constant in brackets. If a precision specifier is given, then only the corresponding number of right most components of the category name will be printed. By default the category name is printed in full. For example, for the category name &quot;a.b.c&quot; the pattern <code>%c{2}</code> will output &quot;b.c&quot;.</td>
</tr>
<tr>
<td>C</td>
<td>Used to output the fully qualified class name of the caller issuing the logging request. This conversion specifier can be optionally followed by <em>precision specifier</em>, that is a decimal constant in brackets. If a precision specifier is given, then only the corresponding number of right most components of the class name will be printed. By default the class name is output in fully qualified form. For example, for the class name &quot;org.apache.xyz.SomeClass&quot;, the pattern <code>%C{1}</code> will output &quot;SomeClass&quot;.</td>
</tr>
</tbody>
</table>

**WARNING** Generating the caller class information is slow.
Thus, it’s use should be avoided unless execution speed is not an issue.

| d | Used to output the date of the logging event. The date conversion specifier may be followed by a *date format specifier* enclosed between braces. For example, %d{HH:mm:ss,SSS} or %d{dd MMM yyyy HH:mm:ss,SSS}. If no date format specifier is given then ISO8601 format is assumed. The date format specifier admits the same syntax as the time pattern string of the *SimpleDateFormat*. Although part of the standard JDK, the performance of *SimpleDateFormat* is quite poor. For better results it is recommended to use the log4j date formatatters. These can be specified using one of the strings "ABSOLUTE", "DATE" and "ISO8601" for specifying *AbsoluteTimeDateFormat*, *DateTimeDateFormat* and respectively *ISO8601DateFormat*. For example, %d{ISO8601} or %d{ABSOLUTE}. These dedicated date formatters perform significantly better than *SimpleDateFormat*. |
|---|

| F | Used to output the file name where the logging request was issued. **WARNING** Generating caller location information is extremely slow. It’s use should be avoided unless execution speed is not an issue. |
|---|

<p>| I | Used to output location information of the caller which generated the logging event. The location information depends on the JVM implementation but usually consists of the fully qualified name of the calling method followed by the callers source the file name and line number between parentheses. |</p>
<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Used to output the line number from where the logging request was issued.</td>
</tr>
<tr>
<td>M</td>
<td>Used to output the method name where the logging request was issued.</td>
</tr>
<tr>
<td>n</td>
<td>Outputs the platform dependent line separator character or characters. This conversion character offers practically the same performance as using non-portable line separator strings such as &quot;\n&quot;, or &quot;\n\n&quot;. Thus, it is the preferred way of specifying a line separator.</td>
</tr>
<tr>
<td>p</td>
<td>Used to output the priority of the logging event.</td>
</tr>
<tr>
<td>r</td>
<td>Used to output the number of milliseconds elapsed from the construction of the layout until the creation of the logging event.</td>
</tr>
<tr>
<td>t</td>
<td>Used to output the name of the thread that generated the logging event.</td>
</tr>
<tr>
<td>x</td>
<td>Used to output the NDC (nested diagnostic context) associated with the thread that generated the logging event.</td>
</tr>
</tbody>
</table>

The location information can be very useful. However, its generation is *extremely* slow. It's use should be avoided unless execution speed is not an issue.

**WARNING** Generating caller location information is extremely slow. It's use should be avoided unless execution speed is not an issue.
| **X** | Used to output the MDC (mapped diagnostic context) associated with the thread that generated the logging event. The X conversion character *must* be followed by the key for the map placed between braces, as in \%X{clientNumber} where clientNumber is the key. The value in the MDC corresponding to the key will be output. See [MDC](#) class for more details. |
| **%** | The sequence %%% outputs a single percent sign. |

By default the relevant information is output as is. However, with the aid of format modifiers it is possible to change the minimum field width, the maximum field width and justification.

The optional format modifier is placed between the percent sign and the conversion character.

The first optional format modifier is the *left justification flag* which is just the minus (-) character. Then comes the optional *minimum field width* modifier. This is a decimal constant that represents the minimum number of characters to output. If the data item requires fewer characters, it is padded on either the left or the right until the minimum width is reached. The default is to pad on the left (right justify) but you can specify right padding with the left justification flag. The padding character is space. If the data item is larger than the minimum field width, the field is expanded to accommodate the data. The value is never truncated.

This behavior can be changed using the *maximum field width* modifier which is designated by a period followed by a decimal constant. If the data item is longer than the maximum field, then the extra characters are removed from the *beginning* of the data item and not from the end. For example, if the maximum field width is eight and the data item is ten characters long, then the first two characters of the data item are dropped. This behavior deviates from the printf function in C where truncation is done from the end.

Below are various format modifier examples for the category conversion specifier.
<table>
<thead>
<tr>
<th>Format modifier</th>
<th>left justify</th>
<th>minimum width</th>
<th>maximum width</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>%20c</td>
<td>false</td>
<td>20</td>
<td>none</td>
<td>Left pad with spaces if the category name is less than 20 characters long.</td>
</tr>
<tr>
<td>%-20c</td>
<td>true</td>
<td>20</td>
<td>none</td>
<td>Right pad with spaces if the category name is less than 20 characters long.</td>
</tr>
<tr>
<td>%.30c</td>
<td>NA</td>
<td>none</td>
<td>30</td>
<td>Truncate from the beginning if the category name is longer than 30 characters.</td>
</tr>
<tr>
<td>%20.30c</td>
<td>false</td>
<td>20</td>
<td>30</td>
<td>Left pad with spaces if the category name is shorter than 20 characters. However, if category name is longer than 30 characters, then truncate from the beginning.</td>
</tr>
<tr>
<td>%-20.30c</td>
<td>true</td>
<td>20</td>
<td>30</td>
<td>Right pad with spaces if the category name is shorter than 20 characters. However, if category name is longer than 30 characters, then truncate from the beginning.</td>
</tr>
</tbody>
</table>

Below are some examples of conversion patterns.

%r [%t] %-5p %c %x - %m\n
This is essentially the TTCC layout.

%-6r [%15.15t] %-5p %30.30c %x - %m\n
Similar to the TTCC layout except that the relative time is right padded if less than 6 digits, thread name is right padded if less than 15 characters and truncated if longer and the category name is left padded if shorter than 30 characters and truncated if longer.

The above text is largely inspired from Peter A. Darnell and Philip E.

**Since:**
0.8.2

**Author:**
James P. Cakalic, Ceki Gülcü

### Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>int BUF_SIZE</td>
<td></td>
</tr>
<tr>
<td>static</td>
<td>String DEFAULT_CONVERSION_PATTERN</td>
<td>Default pattern string for log output.</td>
</tr>
<tr>
<td>protected</td>
<td>int MAX_CAPACITY</td>
<td></td>
</tr>
<tr>
<td>static</td>
<td>String TTCC_CONVERSION_PATTERN</td>
<td>A conversion pattern equivalent to the TTCCCCLayout.</td>
</tr>
</tbody>
</table>

Fields inherited from class org.apache.log4j.Layout
LINE_SEP, LINE_SEP_LEN

### Constructor Summary

- **PatternLayout()**
  Constructs a PatternLayout using the DEFAULT_LAYOUT_PATTERN.

- **PatternLayout(String pattern)**
  Constructs a PatternLayout using the supplied conversion pattern.

### Method Summary

- **void activateOptions()**
  Does not do anything as options become effective
protected `PatternParser` createPatternParser(String pattern)

Returns `PatternParser` used to parse the conversion string.

`String` format(LoggingEvent event)

 Produces a formatted string as specified by the conversion pattern.

`String` getConversionPattern()

 Returns the value of the `ConversionPattern` option.

`boolean` ignoresThrowable()

 The `PatternLayout` does not handle the throwable contained within `LoggingEvents`.

`void` setConversionPattern(String conversionPattern)

 Set the `ConversionPattern` option.

Methods inherited from class `org.apache.log4j.Layout`
`getContentType`, `getFooter`, `getHeader`

Methods inherited from class `java.lang.Object`
`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

**DEFAULT_CONVERSION_PATTERN**

`public static final String DEFAULT_CONVERSION_PATTERN`

Default pattern string for log output. Currently set to the string "%m%n" which just prints the application supplied message.

**TTCC_CONVERSION_PATTERN**

`public static final String TTCC_CONVERSION_PATTERN`
A conversion pattern equivalent to the TTCCCLayout. Current value is \%r [\%t] \%p \%c \%x - \%m\%n.

---

**BUF_SIZE**

protected final int BUF_SIZE

---

**MAX_CAPACITY**

protected final int MAX_CAPACITY

---

### Constructor Detail

#### PatternLayout

```java
public PatternLayout()
```

Constructs a PatternLayout using the DEFAULT_LAYOUT_PATTERN. The default pattern just produces the application supplied message.

---

#### PatternLayout

```java
public PatternLayout(String pattern)
```

Constructs a PatternLayout using the supplied conversion pattern.

---

### Method Detail

#### setConversionPattern

```java
public void setConversionPattern(String conversionPattern)
```

Set the ConversionPattern option. This is the string which controls
formatting and consists of a mix of literal content and conversion specifiers.

---

getConversionPattern

```java
class PatternLayout {
    public String getConversionPattern() {
        Returns the value of the ConversionPattern option.
    }
    
    activateOptions

```java
class PatternLayout {
    public void activateOptions() {
        Does not do anything as options become effective
    }
    
    ignoresThrowable

```java
class PatternLayout {
    public boolean ignoresThrowable() {
        The PatternLayout does not handle the throwable contained within LoggingEvents. Thus, it returns true.
        Overrides: ignoresThrowable in class Layout
        Since: 0.8.4
    }
    
    createPatternParser

```java
class PatternLayout {
    protected PatternParser createPatternParser(String pattern) {
        Returns PatternParser used to parse the conversion string. Subclasses may override this to return a subclass of PatternParser which recognize custom conversion characters.
        Since: 0.9.0
    }
}
public String format(LoggingEvent event)

    Produces a formatted string as specified by the conversion pattern.

    Overrides:
    format in class Layout

Copyright 2000-2005 Apache Software Foundation.
Class Priority

java.lang.Object
+-org.apache.log4j.Priority

Direct Known Subclasses:
  Level

public class Priority extends Object

Refrain from using this class directly, use the Level class instead.

Author:
  Ceki Gülcü

### Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>ALL_INT</td>
<td></td>
</tr>
<tr>
<td>static Priority</td>
<td>DEBUG</td>
<td>Deprecated. Use Level.DEBUG instead.</td>
</tr>
<tr>
<td>static int</td>
<td>DEBUG_INT</td>
<td></td>
</tr>
<tr>
<td>static Priority</td>
<td>ERROR</td>
<td>Deprecated. Use Level.ERROR instead.</td>
</tr>
<tr>
<td>static int</td>
<td>ERROR_INT</td>
<td></td>
</tr>
<tr>
<td>static Priority</td>
<td>FATAL</td>
<td>Deprecated. Use Level.FATAL instead.</td>
</tr>
<tr>
<td>static int</td>
<td>FATAL_INT</td>
<td></td>
</tr>
<tr>
<td>static Priority</td>
<td>INFO</td>
<td></td>
</tr>
</tbody>
</table>
### Constructor Summary

| protected `Priority()` | Default constructor for deserialization. |
| protected `Priority(int level, String levelStr, int syslogEquivalent)` | Instantiate a level object. |

### Method Summary

| boolean `equals(Object o)` | Two priorities are equal if their level fields are equal. |
| static `Priority[] getAllPossiblePriorities()` | Deprecated. This method will be removed with no replacement. |
| int `getSyslogEquivalent()` | Return the syslog equivalent of this priority as an integer. |
| boolean `isGreaterOrEqual(Priority r)` | Returns true if this level has a higher or equal level than the level passed as argument, false otherwise. |
| int `toInt()` | Returns the integer representation of this level. |
| static `Priority toPriority(int val)` | Deprecated. Please use the `Level.toLevel(int)` |

### Deprecated Messages

- Use `Level.INFO` instead.
- Use `Level.WARN` instead.
- This method will be removed with no replacement.
- Please use the `Level.toLevel(int)`
```java
static Priority toPriority(int val, Priority defaultPriority)
    Deprecated. Please use the Level.toLevel(int, Level) method instead.

static Priority toPriority(String sArg)
    Deprecated. Please use the Level.toLevel(String) method instead.

static Priority toPriority(String sArg, Priority defaultPriority)
    Deprecated. Please use the Level.toLevel(String, Level) method instead.

String toString()
    Returns the string representation of this priority.
```

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait

Field Detail

OFF_INT
public static final int OFF_INT

FATAL_INT
public static final int FATAL_INT

ERROR_INT
public static final int ERROR_INT

WARN_INT
public static final int WARN_INT

INFO_INT
public static final int INFO_INT

DEBUG_INT
public static final int DEBUG_INT

ALL_INT
public static final int ALL_INT

FATAL
public static final Priority FATAL

    Deprecated. Use Level.FATAL instead.

ERROR
public static final Priority ERROR

    Deprecated. Use Level.ERROR instead.

WARN
public static final Priority WARN

    Deprecated. Use Level.WARN instead.
INFO

public static final Priority INFO

Deprecated. Use Level.INFO instead.

DEBUG

public static final Priority DEBUG

Deprecated. Use Level.DEBUG instead.

Constructor Detail

Priority

protected Priority()

Default constructor for deserialization.

Priority

protected Priority(int level,
String levelStr,
int syslogEquivalent)

Instantiate a level object.

Method Detail

equals

public boolean equals(Object o)

Two priorities are equal if their level fields are equal.

Overrides:
getSyslogEquivalent

public final int getSyslogEquivalent()

Return the syslog equivalent of this priority as an integer.

isGreaterOrEqual

public boolean isGreaterOrEqual(Priority r)

Returns true if this level has a higher or equal level than the level passed as argument, false otherwise.

You should think twice before overriding the default implementation of isGreaterOrEqual method.

getAllPossiblePriorities

public static Priority[] getAllPossiblePriorities()

Deprecated. This method will be removed with no replacement.

Return all possible priorities as an array of Level objects in descending order.

toString

public final String toString()

Returns the string representation of this priority.

Overrides:

toString in class Object
**toInt**

public final int toInt()

    Returns the integer representation of this level.

**toPriority**

public static Priority toPriority(String sArg)

    Deprecated. Please use the Level.toLevel(String) method instead.

**toPriority**

public static Priority toPriority(int val)

    Deprecated. Please use the Level.toLevel(int) method instead.

**toPriority**

public static Priority toPriority(int val, Priority defaultPriority)

    Deprecated. Please use the Level.toLevel(int, Level) method instead.

**toPriority**

public static Priority toPriority(String sArg, Priority defaultPriority)

    Deprecated. Please use the Level.toLevel(String, Level) method instead.
Copyright 2000-2005 Apache Software Foundation.
public class PropertyConfigurator extends Object implements Configurator

Allows the configuration of log4j from an external file. See doConfigure(String, LoggerRepository) for the expected format.

It is sometimes useful to see how log4j is reading configuration files. You can enable log4j internal logging by defining the log4j.debug variable.

As of log4j version 0.8.5, at class initialization time class, the file log4j.properties will be searched from the search path used to load classes. If the file can be found, then it will be fed to the configure(java.net.URL) method.

The PropertyConfigurator does not handle the advanced configuration features supported by the DOMConfigurator such as support for Filters, custom ErrorHandlers, nested appenders such as the AsyncAppender, etc.

All option values admit variable substitution. The syntax of variable substitution is similar to that of Unix shells. The string between an opening "${" and closing "}" is interpreted as a key. The value of the substituted variable can be defined as a system property or in the configuration file itself. The value of the key is first searched in the system properties, and if not found there, it is then searched in the configuration file being parsed. The corresponding value replaces the ${variableName} sequence. For example, if java.home system property is set to /home/xyz, then every occurrence of the sequence ${java.home} will be interpreted as /home/xyz.
Since: 0.8.1
Author: Ceki Gülcü, Anders Kristensen

## Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td><code>LOGGER_FACTORY_KEY</code></td>
<td>Key for specifying the <code>LoggerFactory</code>.</td>
</tr>
<tr>
<td>protected</td>
<td><code>loggerFactory</code></td>
<td></td>
</tr>
<tr>
<td>protected</td>
<td><code>registry</code></td>
<td>Used internally to keep track of configured appenders.</td>
</tr>
</tbody>
</table>

### Fields inherited from interface `org.apache.log4j.spi.Configurator` INHERITED, NULL

## Constructor Summary

- `PropertyConfigurator()`

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void <code>configure()</code></td>
<td>Read configuration options from <code>properties</code>.</td>
</tr>
<tr>
<td>static void <code>configure(String configFilename)</code></td>
<td></td>
</tr>
<tr>
<td>static void <code>configure(URL configURL)</code></td>
<td>Read configuration options from url <code>configURL</code>.</td>
</tr>
<tr>
<td>static void <code>configureAndWatch(String configFilename)</code></td>
<td>Like <code>configureAndWatch(String, long)</code> except that the default delay as defined by <code>FileWatchdog.DEFAULT_DELAY</code> is used.</td>
</tr>
</tbody>
</table>
static void configureAndWatch(String configFilename, long delay)
   Read the configuration file configFilename if it exists.

protected void configureLoggerFactory(Properties props)
   Check the provided Properties object for a LoggerFactory entry specified by LOGGER_FACTORY_KEY.

void doConfigure(Properties properties, LoggerRepository hierarchy)
   Read configuration options from properties.

void doConfigure(String configFileName, LoggerRepository hierarchy)
   Read configuration from a file.

void doConfigure(URL configURL, LoggerRepository hierarchy)
   Read configuration options from url configURL.

protected void parseCatsAndRenderers(Properties props, LoggerRepository hierarchy)
   Parse non-root elements, such non-root categories and renderers.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

registry
protected Hashtable registry
   Used internally to keep track of configured appenders.

loggerFactory
protected LoggerFactory loggerFactory
**LOGGER_FACTORY_KEY**

```java
public static final String LOGGER_FACTORY_KEY
```

Key for specifying the `LoggerFactory`. Currently set to "log4j.loggerFactory".

## Constructor Detail

**PropertyConfigurator**

```java
public PropertyConfigurator()
```

## Method Detail

**doConfigure**

```java
public void doConfigure(String configFileName, LoggerRepository hierarchy)
```

Read configuration from a file. **The existing configuration is not cleared nor reset.** If you require a different behavior, then call `resetConfiguration` method before calling `doConfigure`.

The configuration file consists of statements in the format `key=value`. The syntax of different configuration elements are discussed below.

### Repository-wide threshold

The repository-wide threshold filters logging requests by level regardless of logger. The syntax is:

`log4j.threshold=[level]`

The level value can consist of the string values OFF, FATAL, ERROR, WARN, INFO, DEBUG, ALL or a custom level value. A custom level value can be specified in the form `level#classname`. By default the repository-wide threshold is set to the lowest possible
value, namely the level ALL.

**Appender configuration**

Appender configuration syntax is:

```java
# For appender named appenderName, set its class.
# Note: The appender name can contain dots.
log4j.appender.appenderName=fully.qualified.name.of.appender.class

# Set appender specific options.
log4j.appender.appenderName.option1=value1
...  
log4j.appender.appenderName.optionN=valueN
```

For each named appender you can configure its [Layout](#). The syntax for configuring an appender's layout is:

```java
log4j.appender.appenderName.layout=fully.qualified.name.of.layout.class
log4j.appender.appenderName.layout.option1=value1
...  
log4j.appender.appenderName.layout.optionN=valueN
```

**Configuring loggers**

The syntax for configuring the root logger is:

```java
log4j.rootLogger=[level], appenderName, appenderName, ...
```

This syntax means that an optional *level* can be supplied followed by appender names separated by commas.

The level value can consist of the string values OFF, FATAL, ERROR, WARN, INFO, DEBUG, ALL or a custom level value. A custom level value can be specified in the form `level#classname`.

If a level value is specified, then the root level is set to the corresponding level. If no level value is specified, then the root level remains untouched.

The root logger can be assigned multiple appenders.

Each *appenderName* (separated by commas) will be added to the
The named appender is defined using the appender syntax defined above.

For non-root categories the syntax is almost the same:

```
log4j.logger.logger_name=[level|INHERITED|NULL], appenderName, a
```

The meaning of the optional level value is discussed above in relation to the root logger. In addition however, the value INHERITED can be specified meaning that the named logger should inherit its level from the logger hierarchy.

If no level value is supplied, then the level of the named logger remains untouched.

By default categories inherit their level from the hierarchy. However, if you set the level of a logger and later decide that that logger should inherit its level, then you should specify INHERITED as the value for the level value. NULL is a synonym for INHERITED.

Similar to the root logger syntax, each `appenderName` (separated by commas) will be attached to the named logger.

See the [appender additivity rule](#) in the user manual for the meaning of the additivity flag.

### ObjectRenderers

You can customize the way message objects of a given type are converted to String before being logged. This is done by specifying an [ObjectRenderer](#) for the object type would like to customize.

The syntax is:

```
log4j.renderer.fully.qualified.name.of.rendered.class=fully.qualified.name.of.rendering.class
```

As in,

```
log4j.renderer.my.Fruit=my.FruitRenderer
```

### Logger Factories
The usage of custom logger factories is discouraged and no longer documented.

**Example**

An example configuration is given below. Other configuration file examples are given in the examples folder.

```xml
# Set options for appender named "A1".
# Appender "A1" will be a SyslogAppender
log4j.appender.A1=org.apache.log4j.net.SyslogAppender

# The syslog daemon resides on www.abc.net

# A1's layout is a PatternLayout, using the conversion pattern
# %r %5p %c{2} %M.%L %x %m
# Thus, the log output will include the relative time since the start of the application
# milliseconds, followed by the level of the log request,
# followed by the two rightmost components of the logger name,
# followed by the callers method name, followed by the line number
# the nested diagnostic context and finally the message itself.
# Refer to the documentation of PatternLayout for further information
# on the syntax of the ConversionPattern key.
log4j.appender.A1.layout.ConversionPattern=%-5r %-5p %c{2} %M.%L %x %m

# Set options for appender named "A2"
# A2 should be a RollingFileAppender, with maximum file size of
# using at most one backup file. A2's layout is TTCC, using the
# ISO8601 date format with context printing enabled.
log4j.appender.A2=org.apache.log4j.RollingFileAppender
log4j.appender.A2.MaxFileSize=10MB
log4j.appender.A2.MaxBackupIndex=1
log4j.appender.A2.layout=org.apache.log4j.TTCCLayout
log4j.appender.A2.layout.ContextPrinting=enabled
log4j.appender.A2.layout.DateFormat=ISO8601

# Root logger set to DEBUG using the A2 appender defined above.
log4j.rootLogger=DEBUG,A2

# Logger definitions:
# The SECURITY logger inherits is level from root. However, it's
# will go to A1 appender defined above. It's additivity is non-cumulative.
log4j.logger.SECURITY=INHERIT,A1
log4j.additivity.SECURITY=false
```
# Only warnings or above will be logged for the logger "SECURITY
# Output will go to A1.
log4j.logger.SECURITY.access=WARN

# The logger "class.of.the.day" inherits its level from the
# logger hierarchy. Output will go to the appender's of the root
# logger, A2 in this case.
log4j.logger.class.of.the.day=INHERIT

Refer to the setOption method in each Appender and Layout for
class specific options.

Use the # or ! characters at the beginning of a line for comments.

**Parameters:**

configFileName - The name of the configuration file where the
configuration information is stored.

---

**configure**

public static void configure(String configFilename)

---

**configure**

public static void configure(URL configURL)

Read configuration options from url configURL.

**Since:**

0.8.2

---

**configure**

public static void configure(Properties properties)

Read configuration options from properties. See
doConfigure(String, LoggerRepository) for the expected format.
configureAndWatch

public static void configureAndWatch(String configFilename)

Like configureAndWatch(String, long) except that the default delay as defined by FileWatchdog.DEFAULT_DELAY is used.

Parameters:
  configFilename - A file in key=value format.

configureAndWatch

public static void configureAndWatch(String configFilename, long delay)

Read the configuration file configFilename if it exists. Moreover, a thread will be created that will periodically check if configFilename has been created or modified. The period is determined by the delay argument. If a change or file creation is detected, then configFilename is read to configure log4j.

Parameters:
  configFilename - A file in key=value format.
  delay - The delay in milliseconds to wait between each check.

doConfigure

public void doConfigure(Properties properties, LoggerRepository hierarchy)

Read configuration options from properties. See doConfigure(String, LoggerRepository) for the expected format.

doConfigure

public void doConfigure(URL configURL, LoggerRepository hierarchy)

Read configuration options from url configURL.
Specified by:
   doConfigure in interface Configurator
Following copied from interface: org.apache.log4j.spi.Configurator

Parameters:
   url - The URL to parse
   repository - The hierarchy to operation upon.

configureLoggerFactory

protected void configureLoggerFactory(Properties props)

   Check the provided Properties object for a LoggerFactory entry
   specified by LOGGER_FACTORY_KEY. If such an entry exists, an attempt
   is made to create an instance using the default constructor. This
   instance is used for subsequent Category creations within this
   configurator.

   See Also:
   parseCatsAndRenderers(java.util.Properties,
       org.apache.log4j.spi.LoggerRepository)

parseCatsAndRenderers

protected void parseCatsAndRenderers(Properties props,
                                   LoggerRepository hierarchy)

   Parse non-root elements, such non-root categories and renderers.
Class RollingFileAppender

java.lang.Object
  |  +- org.apache.log4j.AppenderSkeleton
  |     |  +- org.apache.log4j.WriterAppender
  |     |     |  +- org.apache.log4j.FileAppender
  |     |     +-- org.apache.log4j.RollingFileAppender

All Implemented Interfaces:
  Appender, OptionHandler

Direct Known Subclasses:
  ExternallyRolledFileAppender

public class RollingFileAppender
extends FileAppender

RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.

Author:
  Heinz Richter, Ceki Gülcü

Field Summary

| protected int  | maxBackupIndex | There is one backup file by default. |
| protected long| maxFileSize     | The default maximum file size is 10MB. |

Fields inherited from class org.apache.log4j.FileAppender
bufferedIO, bufferSize, fileAppend, fileName
Fields inherited from class org.apache.log4j.**WriterAppender**
encoding, immediateFlush, qw

Fields inherited from class org.apache.log4j.**AppenderSkeleton**
closed, errorHandler, headFilter, layout, name, tailFilter, threshold

**Constructor Summary**

**RollingFileAppender**()
The default constructor simply calls its **parents constructor**.

**RollingFileAppender**(Layout layout, String filename)
Instantiate a FileAppender and open the file designated by filename.

**RollingFileAppender**(Layout layout, String filename, boolean append)
Instantiate a RollingFileAppender and open the file designated by filename.

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getMaxBackupIndex()</td>
<td>Returns the value of the <strong>MaxBackupIndex</strong> option.</td>
</tr>
<tr>
<td>long getMaximumFileSize()</td>
<td>Get the maximum size that the output file is allowed to reach before being rolled over to backup files.</td>
</tr>
<tr>
<td>void rollover()</td>
<td>Implements the usual roll over behaviour.</td>
</tr>
<tr>
<td>void setFile(String fileName, boolean append, boolean bufferedIO, int bufferSize)</td>
<td>Sets and <strong>opens</strong> the file where the log output will go.</td>
</tr>
<tr>
<td>void setMaxBackupIndex(int maxBackups)</td>
<td>Set the maximum number of backup files to keep around.</td>
</tr>
<tr>
<td>void setMaxFileSize(String value)</td>
<td>Set the maximum size that the output file is allowed to reach before being rolled over to backup files.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void setMaximumFileSize(long maxFileSize)</code></td>
<td>Set the maximum size that the output file is allowed to reach before being rolled over to backup files.</td>
</tr>
<tr>
<td><code>protected void setQWForFiles(Writer writer)</code></td>
<td>Sets the quiet writer being used.</td>
</tr>
<tr>
<td><code>protected void subAppend(LoggingEvent event)</code></td>
<td>This method differentiates RollingFileAppender from its super class.</td>
</tr>
</tbody>
</table>

Methods inherited from class `org.apache.log4j.FileAppender`:
- `activateOptions`, `closeFile`, `getAppend`, `getBufferedIO`, `getBufferSize`, `getAppend`, `getBufferedIO`, `getBufferSize`, `setAppend`, `setBufferedIO`, `setBufferSize`, `setFile`

Methods inherited from class `org.apache.log4j.WriterAppender`:
- `append`, `checkEntryConditions`, `close`, `closeWriter`, `createWriter`, `getEncoding`, `getImmediateFlush`, `requiresLayout`, `setEncoding`, `setErrorHandler`, `setImmediateFlush`, `setWriter`, `writeFooter`, `writeHeader`

Methods inherited from class `org.apache.log4j.AppenderSkeleton`:
- `addFilter`, `clearFilters`, `doAppend`, `finalize`, `getErrorHandler`, `getFilter`, `getFirstFilter`, `getLayout`, `getName`, `getThreshold`, `isAsSevereAsThreshold`, `setLayout`, `setName`, `setThreshold`

Methods inherited from class `java.lang.Object`:
- `clone`, `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

**Field Detail**

**maxFileSize**

protected long **maxFileSize**

The default maximum file size is 10MB.
maxBackupIndex

protected int maxBackupIndex

There is one backup file by default.

Constructor Detail

RollingFileAppender

public RollingFileAppender()

The default constructor simply calls its parents constructor.

RollingFileAppender

public RollingFileAppender(Layout layout,
  String filename,
  boolean append)
  throws IOException

Instantiate a RollingFileAppender and open the file designated by filename. The opened filename will become the output destination for this appender.

If the append parameter is true, the file will be appended to. Otherwise, the file designated by filename will be truncated before being opened.

RollingFileAppender

public RollingFileAppender(Layout layout,
  String filename)
  throws IOException

Instantiate a FileAppender and open the file designated by filename.
The opened filename will become the output destination for this appender.

The file will be appended to.

**Method Detail**

**getMaxBackupIndex**

```java
public int getMaxBackupIndex()
```

Returns the value of the `MaxBackupIndex` option.

---

**getMaximumFileSize**

```java
public long getMaximumFileSize()
```

Get the maximum size that the output file is allowed to reach before being rolled over to backup files.

**Since:**

1.1

---

**rollOver**

```java
public void rollOver()
```

Implements the usual roll over behaviour.

If `MaxBackupIndex is positive, then files {File.1, ..., File.MaxBackupIndex -1} are renamed to {File.2, ..., File.MaxBackupIndex}. Moreover, File is renamed File.1 and closed. A new File is created to receive further log output.

If `MaxBackupIndex is equal to zero, then the File is truncated with no backup files created.
**setFile**

```java
public void setFile(String fileName,
        boolean append,
        boolean bufferedIO,
        int bufferSize)
    throws IOException
```

**Description copied from class: FileAppender**

Sets and opens the file where the log output will go. The specified file must be writable.

If there was already an opened file, then the previous file is closed first.

**Do not use this method directly. To configure a FileAppender or one of its subclasses, set its properties one by one and then call activateOptions.**

**Overrrides:**

`setFile` in class FileAppender

Following copied from class: org.apache.log4j.FileAppender

**Parameters:**

- **fileName** - The path to the log file.
- **append** - If true will append to `fileName`. Otherwise will truncate `fileName`.

---

**setMaxBackupIndex**

```java
public void setMaxBackupIndex(int maxBackups)
```

Set the maximum number of backup files to keep around.

The **MaxBackupIndex** option determines how many backup files are kept before the oldest is erased. This option takes a positive integer value. If set to zero, then there will be no backup files and the log file will be truncated when it reaches `MaxFileSize`. 
**setMaximumFileSize**

```java
class Name {
    public void setMaximumFileSize(long maxFileSize) {
        // Set the maximum size that the output file is allowed to reach before being rolled over to backup files.
        // This method is equivalent to setMaxFileSize(java.lang.String) except that it is required for differentiating the setter taking a long argument from the setter taking a String argument by the JavaBeans Introspector.
    }
}
```

**See Also:**
- `setMaxFileSize(String)`

**setMaxFileSize**

```java
class Name {
    public void setMaxFileSize(String value) {
        // Set the maximum size that the output file is allowed to reach before being rolled over to backup files.
        // In configuration files, the MaxFileSize option takes an long integer in the range 0 - 2^63. You can specify the value with the suffixes "KB", "MB" or "GB" so that the integer is interpreted being expressed respectively in kilobytes, megabytes or gigabytes. For example, the value "10KB" will be interpreted as 10240.
    }
}
```

**setQWForFiles**

```java
class Name {
    protected void setQWForFiles(Writer writer) {
        // Description copied from class: FileAppender
        // Sets the quiet writer being used. This method is overridden by RollingFileAppender.
        // Overrides:
        // setQWForFiles in class FileAppender
    }
}
```
subAppend

protected void subAppend(LoggingEvent event)

This method differentiates RollingFileAppender from its super class.

Overrides:
subAppend in class WriterAppender

Since:
0.9.0
org.apache.log4j  Class SimpleLayout

java.lang.Object  
   +-- org.apache.log4j.Layout  
   |    +-- org.apache.log4j.SimpleLayout

All Implemented Interfaces:  
   OptionHandler

public class SimpleLayout extends Layout

SimpleLayout consists of the level of the log statement, followed by " - " and then the log message itself. For example,

DEBUG - Hello world

Since:  
   version 0.7.0

PatternLayout offers a much more powerful alternative.

Author:  
   Ceki Gülcü

Fields inherited from class org.apache.log4j.Layout
LINE_SEP, LINE_SEP_LEN

Constructor Summary

SimpleLayout()
### Methods

**void activateOptions()**
Activate the options that were previously set with calls to option setters.

**String format(LoggingEvent event)**
Returns the log statement in a format consisting of the level, followed by " - " and then the message.

**boolean ignoresThrowable()**
The SimpleLayout does not handle the throwable contained within `LoggingEvents`.

### Methods inherited from class `org.apache.log4j.Layout`
- `getContentType`, `getFooter`, `getHeader`

### Methods inherited from class `java.lang.Object`
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Constructor Detail

#### SimpleLayout

```java
public SimpleLayout()
```

### Method Detail

#### activateOptions

```java
public void activateOptions()
```

Description copied from interface: `OptionHandler`
Activate the options that were previously set with calls to option setters.

This allows to defer activation of the options until all options have
been set. This is required for components which have related options that remain ambiguous until all are set.

For example, the FileAppender has the File and Append options both of which are ambiguous until the other is also set.

**format**

public String format(LoggingEvent event)

Returns the log statement in a format consisting of the level, followed by " - " and then the message. For example,

INFO - "A message"

The category parameter is ignored.

Overrides:
format in class Layout

Returns:
A byte array in SimpleLayout format.

**ignoresThrowable**

public boolean ignoresThrowable()

The SimpleLayout does not handle the throwable contained within LoggingEvents. Thus, it returns true.

Overrides:
ignoresThrowable in class Layout

Since:
version 0.8.4
org.apache.log4j  Class TTCCLayout

java.lang.Object
   +-- org.apache.log4j.Layout
      |   +-- org.apache.log4j.helpers.DateLayout
      |   +-- org.apache.log4j.TTCCLayout

All Implemented Interfaces:
   OptionHandler

public class TTCCLayout
extends DateLayout

TTCC layout format consists of time, thread, category and nested diagnostic context information, hence the name.

Each of the four fields can be individually enabled or disabled. The time format depends on the DateFormat used.

Here is an example TTCCLayout output with the RelativeTimeDateFormat.

176 [main] INFO org.apache.log4j.examples.Sort - Populating an arra
225 [main] INFO org.apache.log4j.examples.SortAlgo - Entered the so
262 [main] DEBUG org.apache.log4j.examples.SortAlgo.OUTER i=1 - Oute
276 [main] DEBUG org.apache.log4j.examples.SortAlgo.OUTER i=0 - Oute
304 [main] INFO org.apache.log4j.examples.SortAlgo.DUMP - Dump of i
317 [main] INFO org.apache.log4j.examples.SortAlgo.DUMP - Element [
331 [main] INFO org.apache.log4j.examples.SortAlgo.DUMP - Element [
343 [main] INFO org.apache.log4j.examples.Sort - The next log state
346 [main] ERROR org.apache.log4j.examples.SortAlgo.DUMP - Tried to
at org.apache.log4j.examples.SortAlgo.dump(SortAlgo.java:58)
at org.apache.log4j.examples.Sort.main(Sort.java:64)
467 [main] INFO org.apache.log4j.examples.Sort - Exiting main metho

The first field is the number of milliseconds elapsed since the start of the program. The second field is the thread outputting the log statement. The third field is the level, the fourth field is the category to which the
statement belongs.

The fifth field (just before the '-' ) is the nested diagnostic context. Note the nested diagnostic context may be empty as in the first two statements. The text after the '-' is the message of the statement.

**WARNING** Do not use the same TTCCLayout instance from within different appenders. The TTCCLayout is not thread safe when used in his way. However, it is perfectly safe to use a TTCCLayout instance from just one appender.

*PatternLayout* offers a much more flexible alternative.

**Author:**

Ceki Gülcü, Heinz Richter

---

### Field Summary

| protected StringBuffer buf |

### Fields inherited from class org.apache.log4j.helpers.*DateLayout*

| date, DATE_FORMAT_OPTION, dateFormat, NULL_DATE_FORMAT, pos, RELATIVE_TIME_DATE_FORMAT, TIMEZONE_OPTION |

### Fields inherited from class org.apache.log4j.*Layout*

| LINE_SEP, LINE_SEP_LEN |

### Constructor Summary

**TTCCLayout()**

Instantiate a TTCCLayout object with RelativeTimeDateFormat as the date formatter in the local time zone.

**TTCCLayout(String dateFormatType)**

Instantiate a TTCCLayout object using the local time zone.
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String format(LoggingEvent event)</code></td>
<td>In addition to the level of the statement and message, the returned byte array includes time, thread, category and NDC information.</td>
</tr>
<tr>
<td><code>boolean getCategoryPrefixing()</code></td>
<td>Returns value of the <code>CategoryPrefixing</code> option.</td>
</tr>
<tr>
<td><code>boolean getContextPrinting()</code></td>
<td>Returns value of the <code>ContextPrinting</code> option.</td>
</tr>
<tr>
<td><code>boolean getThreadPrinting()</code></td>
<td>Returns value of the <code>ThreadPrinting</code> option.</td>
</tr>
<tr>
<td><code>boolean ignoresThrowable()</code></td>
<td>The TTCCLayout does not handle the throwable contained within <code>LoggingEvents</code>.</td>
</tr>
<tr>
<td><code>void setCategoryPrefixing(boolean categoryPrefixing)</code></td>
<td>The <code>CategoryPrefixing</code> option specifies whether Category name is part of log output or not.</td>
</tr>
<tr>
<td><code>void setContextPrinting(boolean contextPrinting)</code></td>
<td>The <code>ContextPrinting</code> option specifies log output will include the nested context information belonging to the current thread.</td>
</tr>
<tr>
<td><code>void setThreadPrinting(boolean threadPrinting)</code></td>
<td>The <code>ThreadPrinting</code> option specifies whether the name of the current thread is part of log output or not.</td>
</tr>
</tbody>
</table>

**Methods inherited from class org.apache.log4j.helpers.DateLayout**

- `activateOptions`, `dateFormat`, `getDateFormat`, `getOptionStrings`, `getTimeZone`, `setDateFormat`, `setDateFormat`, `setDateFormat`, `setOption`, `setTimeZone`

**Methods inherited from class org.apache.log4j.Layout**

- `getContentType`, `getFooter`, `getHeader`

**Methods inherited from class java.lang.Object**

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`,
Field Detail

buf

protected final StringBuffer buf

Constructor Detail

TTCCLayout

public TTCCLayout()

   Instantiate a TTCCLayout object with RelativeTimeDateFormat as the date formatter in the local time zone.
   Since: 0.7.5

TTCCLayout

public TTCCLayout(String dateFormatType)

   Instantiate a TTCCLayout object using the local time zone. The DateFormat used will depend on the dateFormatType.

   This constructor just calls the DateLayout.setDateFormat(java.lang.String) method.

Method Detail

setThreadPrinting

public void setThreadPrinting(boolean threadPrinting)
The **ThreadPrinting** option specifies whether the name of the current thread is part of log output or not. This is true by default.

---

**getThreadPrinting**

```java
public boolean getThreadPrinting()
```

Returns value of the **ThreadPrinting** option.

---

**setCategoryPrefixing**

```java
public void setCategoryPrefixing(boolean categoryPrefixing)
```

The **CategoryPrefixing** option specifies whether **Category** name is part of log output or not. This is true by default.

---

**getCategoryPrefixing**

```java
public boolean getCategoryPrefixing()
```

Returns value of the **CategoryPrefixing** option.

---

**setContextPrinting**

```java
public void setContextPrinting(boolean contextPrinting)
```

The **ContextPrinting** option specifies log output will include the nested context information belonging to the current thread. This is true by default.

---

**getContextPrinting**

```java
public boolean getContextPrinting()
```

Returns value of the **ContextPrinting** option.
format

public String format(LoggingEvent event)

In addition to the level of the statement and message, the returned byte array includes time, thread, category and NDC information.

Time, thread, category and diagnostic context are printed depending on options.

Overrides:
    format in class Layout

Parameters:
    event - The event to format

ignoresThrowable

public boolean ignoresThrowable()

The TTCCLayout does not handle the throwable contained within LoggingEvents. Thus, it returns true.

Overrides:
    ignoresThrowable in class Layout

Since:
    version 0.8.4
public class **WriterAppender**
extends **AppenderSkeleton**

WriterAppender appends log events to a *Writer* or an *OutputStream* depending on the user's choice.

**Since:**
1.1

**Author:**
Ceki Gülcü

---

### Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected String</td>
<td><strong>encoding</strong></td>
<td>The encoding to use when writing.</td>
</tr>
<tr>
<td>protected boolean</td>
<td><strong>immediateFlush</strong></td>
<td>Immediate flush means that the underlying writer or output stream will be flushed at the end of each append operation.</td>
</tr>
<tr>
<td>protected <strong>QuietWriter</strong></td>
<td><strong>qw</strong></td>
<td>This is the <em>quietWriter</em> where we will write to.</td>
</tr>
</tbody>
</table>
Fields inherited from class org.apache.log4j.AppenderSkeleton

closed, errorHandler, headFilter, layout, name, tailFilter, threshold

Constructor Summary

WriterAppender()
This default constructor does nothing.

WriterAppender(Layout layout, OutputStream os)
Instantiate a WriterAppender and set the output destination to a new OutputStreamWriter initialized with os as its OutputStream.

WriterAppender(Layout layout, Writer writer)
Instantiate a WriterAppender and set the output destination to writer.

Method Summary

void activateOptions()
Does nothing.

void append(LoggingEvent event)
This method is called by the AppenderSkeleton.doAppend(org.apache.log4j.spi.LoggingEvent) method.

protected boolean checkEntryConditions()
This method determines if there is a sense in attempting to append.

void close()
Close this appender instance.

protected void closeWriter()
Close the underlying Writer.

protected OutputStreamWriter createWriter(OutputStream os)
Returns an OutputStreamWriter when passed an OutputStream.

String getEncoding()

boolean getImmediateFlush()
Returns value of the **ImmediateFlush** option.

```java
boolean requiresLayout()
```

The WriterAppender requires a layout.

```java
protected void reset()
```

Clear internal references to the writer and other variables.

```java
void setEncoding(String value)
```

```java
void setErrorHandler(ErrorHandler eh)
```

Set the **ErrorHandler** for this WriterAppender and the underlying **QuietWriter** if any.

```java
void setImmediateFlush(boolean value)
```

If the **ImmediateFlush** option is set to true, the appender will flush at the end of each write.

```java
void setWriter(Writer writer)
```

Sets the Writer where the log output will go.

```java
protected void subAppend(LoggingEvent event)
```

Actual writing occurs here.

```java
protected void writeFooter()
```

Write a footer as produced by the embedded layout `Layout.getFooter()` method.

```java
protected void writeHeader()
```

Write a header as produced by the embedded layout `Layout.getHeader()` method.

**Methods inherited from class org.apache.log4j.AppenderSkeleton**

- `addFilter`, `clearFilters`, `doAppend`, `finalize`, `getErrorHandler`, `getFilter`, `getFirstFilter`, `getLayout`, `getName`, `getThreshold`, `isAsSevereAsThreshold`, `setLayout`, `setName`, `setThreshold`

**Methods inherited from class java.lang.Object**

- `clone`, `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
**immediateFlush**

protected boolean immediateFlush

Immediate flush means that the underlying writer or output stream will be flushed at the end of each append operation. Immediate flush is slower but ensures that each append request is actually written. If immediateFlush is set to false, then there is a good chance that the last few logs events are not actually written to persistent media if and when the application crashes.

The immediateFlush variable is set to true by default.

**encoding**

protected String encoding

The encoding to use when writing.

The encoding variable is set to null by default which results in the utilization of the system's default encoding.

**qw**

protected QuietWriter qw

This is the quietWriter where we will write to.

### Constructor Detail

**WriterAppender**

public WriterAppender()

This default constructor does nothing.
**WriterAppender**

`public WriterAppender(Layout layout, OutputStream os)`

Instantiate a `WriterAppender` and set the output destination to a new `OutputStreamWriter` initialized with `os` as its `OutputStream`.

**WriterAppender**

`public WriterAppender(Layout layout, Writer writer)`

Instantiate a `WriterAppender` and set the output destination to `writer`.

The `writer` must have been previously opened by the user.

**Method Detail**

**setImmediateFlush**

`public void setImmediateFlush(boolean value)`

If the `ImmediateFlush` option is set to `true`, the appender will flush at the end of each write. This is the default behavior. If the option is set to `false`, then the underlying stream can defer writing to physical medium to a later time.

Avoiding the flush operation at the end of each append results in a performance gain of 10 to 20 percent. However, there is safety tradeoff involved in skipping flushing. Indeed, when flushing is skipped, then it is likely that the last few log events will not be recorded on disk when the application exits. This is a high price to pay even for a 20% performance gain.

**getImmediateFlush**
public boolean getImmediateFlush()

Returns value of the ImmediateFlush option.

activateOptions

public void activateOptions()

Does nothing.
Overrides:
activateOptions in class AppenderSkeleton

append

public void append(LoggingEvent event)

This method is called by the AppenderSkeleton.doAppend(org.apache.log4j.spi.LoggingEvent) method.

If the output stream exists and is writable then write a log statement to the output stream. Otherwise, write a single warning message to System.err.

The format of the output will depend on this appender's layout.

Overrides:
append in class AppenderSkeleton

checkEntryConditions

protected boolean checkEntryConditions()

This method determines if there is a sense in attempting to append.

It checks whether there is a set output target and also if there is a set layout. If these checks fail, then the boolean value false is returned.
close

public void close()

Close this appender instance. The underlying stream or writer is also closed.

Closed appenders cannot be reused.

Since:
0.8.4
See Also:
setWriter(java.io.Writer)

closeWriter

protected void closeWriter()

Close the underlying Writer.

createWriter

protected OutputStreamWriter createWriter(OutputStream os)

Returns an OutputStreamWriter when passed an OutputStream. The encoding used will depend on the value of the encoding property. If the encoding value is specified incorrectly the writer will be opened using the default system encoding (an error message will be printed to the loglog).

getEncoding

public String getEncoding()
setEncoding

public void setEncoding(String value)

setErrorHandler

public void setErrorHandler(ErrorHandler eh)

Set the ErrorHandler for this WriterAppender and also the underlying QuietWriter if any.

Overrides:
setErrorHandler in class AppenderSkeleton

setWriter

public void setWriter(Writer writer)

Sets the Writer where the log output will go. The specified Writer must be opened by the user and be writable.

The java.io.Writer will be closed when the appender instance is closed.

WARNING: Logging to an unopened Writer will fail.

Parameters:
writer - An already opened Writer.

subAppend

protected void subAppend(LoggingEvent event)

Actual writing occurs here.

Most subclasses of WriterAppender will need to override this method.

Since:
The `WriterAppender` requires a layout. Hence, this method returns `true`. Clear internal references to the writer and other variables. Subclasses can override this method for an alternate closing behavior.

Write a footer as produced by the embedded layout's `Layout.getFooter()` method.

Write a header as produced by the embedded layout's `Layout.getHeader()` method.
Copyright 2000-2005 Apache Software Foundation.
Package org.apache.log4j.chainsaw

Chainsaw is a GUI log viewer and filter for the log4j package.

See:  Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main</strong></td>
</tr>
</tbody>
</table>
Package org.apache.log4j.chainsaw Description

Chainsaw is a GUI log viewer and filter for the log4j package. By default it listens for `LoggingEvent` objects sent using the `SocketAppender` and displays them in a table. The events can be filtered based on:

- Level
- Thread name
- Logger
- Message
- NDC

All the details for each event can be displayed by selecting the event in the table.

Chainsaw also supports loading events logged to a file using the `XMLLayout` format. This is great for analysing log files, and means you do not need to keep Chainsaw running continuously. It is easy to add support for loading events from other sources like JDBC.

A picture is worth a thousand words:
Finally, why is it called chainsaw? Because it cuts your log (file) down to size. :-(
Requirements

Chainsaw is based on the Swing API which requires JDK 1.2 or later.
Running chainsaw

Setup

You need to include the log4j.jar in the classpath.

Usage

The command line usage is:

    java -D<property>=<value> org.apache.log4j.chainsaw.Main

The default behaviour of chainsaw can be changed by setting system properties using the -D<property>=<value> arguments to java. The following table describes what properties can be set:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chainsaw.port</td>
<td>Indicates which port to listen for connections on. Defaults to &quot;4445&quot;.</td>
</tr>
</tbody>
</table>
Configuring Log4J

You will need to configure log4j to send logging events to Chainsaw. Here is a sample log4j.properties file for sending logging events to Chainsaw.

```properties
log4j.rootLogger=DEBUG, CHAINSAW_CLIENT

log4j.appender.CHAINSAW_CLIENT=org.apache.log4j.net.SocketAppender
log4j.appender.CHAINSAW_CLIENT.RemoteHost=localhost
log4j.appender.CHAINSAW_CLIENT.Port=4445
log4j.appender.CHAINSAW_CLIENT.LocationInfo=true
```

Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Class</th>
<th>Use</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
<th>Log4j 1.2.14</th>
</tr>
</thead>
</table>

PREV  NEXT  FRAMES  NO FRAMES
Hierarchy For Package
org.apache.log4j.chainsaw

Package Hierarchies:
   All Packages
Class Hierarchy

- class java.lang.Object
  - class java.awt.Container
    - class java.awt.Window (implements javax.accessibility.Accessible)
    - class java.awt.Frame (implements java.awt.MenuContainer)
      - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
      - class org.apache.log4j.chainsaw.Main
No usage of org.apache.log4j.chainsaw
Class Main

```
public class Main extends JFrame
```

The main application.

**Author:**

Oliver Burn

**See Also:**

Serialized Form

---

### Inner classes inherited from class javax.swing.JFrame

<table>
<thead>
<tr>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFrame.AccessibleJFrame</td>
</tr>
</tbody>
</table>

---

### Inner classes inherited from class java.awt.Frame

<table>
<thead>
<tr>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame.AccessibleAWTFrame</td>
</tr>
</tbody>
</table>

---

### Inner classes inherited from class java.awt.Window

<table>
<thead>
<tr>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window</td>
</tr>
</tbody>
</table>

---
Window.AccessibleAWTWindow

Inner classes inherited from class java.awt.Container
Container.AccessibleAWTContainer

Inner classes inherited from class java.awt.Component
Component.AccessibleAWTComponent

Field Summary

static String PORT_PROP_NAME
name of property for port name

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO NOTHING ON CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Method Summary
```java
static void main(String[] aArgs)
    The main method.
```

<table>
<thead>
<tr>
<th>Methods inherited from class javax.swing.JFrame</th>
</tr>
</thead>
<tbody>
<tr>
<td>addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isRootPaneCheckingEnabled, paramString, processKeyEvent, processWindowEvent, remove, getContentPane, getRootPane, setContentPane, setDefaultCloseOperation, setGlassPane, setJMenuBar, setLayeredPane, setRootPane, setRootPaneCheckingEnabled, update</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from class java.awt.Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>addNotify, finalize, getCursorType, getFrames, getIconImage, getMenuBar, getState, getTitle, isResizable, remove, removeNotify, setCursor, setIconImage, setMenuBar, setResizable, setState, setTitle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from class java.awt.Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>addWindowListener, applyResourceBundle, applyResourceBundle, dispose, getFocusOwner, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getOwnedWindows, getOwner, getToolkit, getWarningString, hide, isShowing, pack, postEvent, processEvent, removeWindowListener, setCursor, show, toBack, toFront</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from class java.awt.Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processComponentEvent, remove, removeAll, removeContainerListener, setFont, validate, validateTree</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from class java.awt.Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener,</td>
</tr>
</tbody>
</table>
addPropertyChangeListener, bounds, checkImage, checkImage, 
coalesceEvents, contains, contains, createImage, createImage, 
disable, disableEvents, dispatchEvent, enable, enable, 
enableEvents, enableInputMethods, firePropertyChange, 
getBackground, getBounds, getBounds, getColorModel, 
getComponentOrientation, getCursor, getDropTarget, getFont, 
getFontMetrics, getForeground, getGraphics, getHeight, 
getInputMethodRequests, getLocation, getLocation, 
getLocationOnScreen, getName, getParent, getPeer, getSize, 
getSize, getTreeLock, getLimit, getLimit, getLimit, getLimit, getLimit, 
hasFocus, imageUpdate, inside, isDisplayable, isDoubleBuffered, 
isEnabled, isFocusTraversable, isLightweight, isOpaque, isValid, 
isVisible, keyDown, keyUp, list, list, list, list, location, lostFocus, 
move, nextFocus, paintAll, prepareImage, prepareImage, printAll, 
processComponentEvent, processFocusEvent, 
processHierarchyBoundsEvent, processHierarchyEvent, 
processInputMethodEvent, processMouseEvent, 
processMouseMotionEvent, removeComponentListener, 
removeFocusListener, removeHierarchyBoundsListener, 
removeHierarchyListener, removeInputMethodListener, 
removeKeyListener, removeMouseListener, removeMouseMotionListener, 
removePropertyChangeListener, removePropertyChangeListener, 
repaint, repaint, repaint, repaint, requestFocus, reshape, resize, 
resize, setBackground, setBounds, setSize, 
setComponentOrientation, setDropTarget, setEnabled, setForeground, 
setLocale, setLocation, setLocation, setName, setSize, setSize, 
setVisible, show, size, toString, transferFocus

Methods inherited from class java.lang. Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt. MenuContainer
getFont, postEvent

Field Detail

PORT_PROP_NAME

public static final String PORT_PROP_NAME
The main method.

Parameters:
- `aArgs` - ignored
## Package org.apache.log4j.config

Package used in getting/setting component properties.

See: [Description](#)

<table>
<thead>
<tr>
<th>Interface Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyGetter.PropertyCallback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PropertyGetter</strong></td>
</tr>
<tr>
<td><strong>PropertyPrinter</strong></td>
</tr>
<tr>
<td><strong>PropertySetter</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exception Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertySetterException</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.config Description

Package used in getting/setting component properties.

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.config

Package Hierarchies:

All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.config.**PropertyGetter**
  - class org.apache.log4j.config.**PropertyPrinter** (implements org.apache.log4j.config.**PropertyGetter.PropertyCallback**)
  - class org.apache.log4j.config.**PropertySetter**
  - class java.lang.**Throwable** (implements java.io.**Serializable**)
    - class java.lang.**Exception**
      - class org.apache.log4j.config.**PropertySetterException**
Interface Hierarchy

- interface org.apache.log4j.config.PropertyGetter.PropertyCallback
Uses of Package
org.apache.log4j.config

Packages that use org.apache.log4j.config

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.config</td>
<td>Package used in getting/setting component properties.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

Classes in org.apache.log4j.config used by org.apache.log4j.config

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyGetter.PropertyTypeCallback</td>
<td></td>
</tr>
<tr>
<td>PropertySetterException</td>
<td>Thrown when an error is encountered whilst attempting to set a property using the PropertySetter utility class.</td>
</tr>
</tbody>
</table>

Classes in org.apache.log4j.config used by org.apache.log4j.xml

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertySetter</td>
<td>General purpose Object property setter.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
### Class PropertyGetter

**java.lang.Object**

```
+-org.apache.log4j.config.PropertyGetter
```

Public class **PropertyGetter** extends **Object**

Used for inferring configuration information for a log4j's component.

**Author:**

Anders Kristensen

---

### Inner Class Summary

<table>
<thead>
<tr>
<th>static interface</th>
<th>PropertyGetter.PropertyCallback</th>
</tr>
</thead>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>protected static Object[]</th>
<th>NULL_ARG</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Object</td>
<td>obj</td>
</tr>
<tr>
<td>protected PropertyDescriptor[]</td>
<td>props</td>
</tr>
</tbody>
</table>

### Constructor Summary

**PropertyGetter**(Object obj)

Create a new PropertyGetter for the specified Object.

### Method Summary
static void getProperties(Object obj, PropertyGetter.PropertyCallback callback, String prefix)

void getProperties(PropertyGetter.PropertyCallback callback, String prefix)

protected boolean isHandledType(Class type)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

NULL_ARG

protected static final Object[] NULL_ARG

obj

protected Object obj

props

protected PropertyDescriptor[] props

Constructor Detail

PropertyGetter

public PropertyGetter(Object obj)
throws IntrospectionException
Create a new PropertyGetter for the specified Object. This is done in preparation for invoking `getProperties(PropertyGetter.PropertyCallback, String)` one or more times.

**Parameters:**
- `obj` - the object for which to set properties

### Method Detail

#### getProperties

```java
public static void getProperties(Object obj, PropertyGetter.PropertyCallback callback, String prefix)
```

#### getProperties

```java
public void getProperties(PropertyGetter.PropertyCallback callback, String prefix)
```

#### isHandledType

```java
protected boolean isHandledType(Class type)
```
public static interface PropertyGetter.PropertyCallback

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>void foundProperty</td>
<td>Object obj, String prefix, String name, Object value</td>
</tr>
</tbody>
</table>

Method Detail

foundProperty

public void foundProperty(Object obj, String prefix, String name, Object value)
public class PropertyPrinter
extends Object
implements PropertyGetter.PropertyCallback

Prints the configuration of the log4j default hierarchy (which needs to be auto-initialized) as a properties file on a PrintWriter.

Author:
Anders Kristensen

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>appenderNames</td>
<td>Hashtable</td>
</tr>
<tr>
<td>do Capitalize</td>
<td>boolean</td>
</tr>
<tr>
<td>layoutNames</td>
<td>Hashtable</td>
</tr>
<tr>
<td>numAppenders</td>
<td>int</td>
</tr>
<tr>
<td>out</td>
<td>PrintWriter</td>
</tr>
</tbody>
</table>

Constructor Summary

PropertyPrinter(PrintWriter out)
**PropertyPrinter** *(PrintWriter* `out`, boolean *doCapitalize)*

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static String capitalize(String name)</code></td>
<td></td>
</tr>
<tr>
<td><code>void foundProperty(Object obj, String prefix, String name, Object value)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected String genAppName()</code></td>
<td></td>
</tr>
<tr>
<td><code>protected boolean isGenAppName(String name)</code></td>
<td>Returns true if the specified appender name is considered to have been generated, that is, if it is of the form A[0-9]+.</td>
</tr>
<tr>
<td><code>static void main(String[] args)</code></td>
<td></td>
</tr>
<tr>
<td><code>void print(PrintWriter out)</code></td>
<td>Prints the configuration of the default log4j hierarchy as a Java properties file on the specified Writer.</td>
</tr>
<tr>
<td><code>protected void printOptions(PrintWriter out, Logger cat)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected void printOptions(PrintWriter out, Object obj, String fullname)</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`
numAppenders
protected int numAppenders

appenderNames
protected Hashtable appenderNames

layoutNames
protected Hashtable layoutNames

out
protected PrintWriter out

doCapitalize
protected boolean doCapitalize

Constructor Detail

PropertyPrinter
public PropertyPrinter(PrintWriter out)

PropertyPrinter
public PropertyPrinter(PrintWriter out, boolean doCapitalize)

Method Detail
genAppName

protected String genAppName()

isGenAppName

protected boolean isGenAppName(String name)

   Returns true if the specified appender name is considered to have been generated, that is, if it is of the form A[0-9]+.

print

public void print(PrintWriter out)

   Prints the configuration of the default log4j hierarchy as a Java properties file on the specified Writer.

   N.B. print() can be invoked only once!

printOptions

protected void printOptions(PrintWriter out, Logger cat)

printOptions

protected void printOptions(PrintWriter out, Object obj, String fullname)

foundProperty

public void foundProperty(Object obj, String prefix,
String name,
Object value)

Specified by:
foundProperty in interface PropertyGetter.PropertyCallback

capitalize

public static String capitalize(String name)

main

public static void main(String[] args)
public class PropertySetter extends Object

General purpose Object property setter. Clients repeatedly invokes
setProperty(name, value) in order to invoke setters on the Object
specified in the constructor. This class relies on the JavaBeans
Introspector to analyze the given Object Class using reflection.

Usage:

PropertySetter ps = new PropertySetter(anObject);
ps.set("name", "Joe");
ps.set("age", "32");
ps.set("isMale", "true");

will cause the invocations anObject.setName("Joe"),
anObject.setAge(32), and setMale(true) if such methods exist with those
signatures. Otherwise an IntrospectionException are thrown.

Since:
    1.1
Author:
    Anders Kristensen

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected obj</td>
<td>Object</td>
</tr>
<tr>
<td>protected props</td>
<td>PropertyDescriptor[]</td>
</tr>
</tbody>
</table>
### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PropertySetter</strong>(Object obj)</td>
<td>Create a new PropertySetter for the specified Object.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>activate</strong>()</td>
<td></td>
</tr>
<tr>
<td>protected Object</td>
<td><strong>convertArg</strong>(String val, Class type)</td>
<td>Convert val a String parameter to an object of a given type.</td>
</tr>
<tr>
<td>protected PropertyDescriptor</td>
<td><strong>getPropertyDescriptor</strong>(String name)</td>
<td></td>
</tr>
<tr>
<td>protected void</td>
<td><strong>introspect</strong>()</td>
<td>Uses JavaBeans Introspector to computer setters of object to be configured.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>setProperties</strong>(Object obj, Properties properties, String prefix)</td>
<td>Set the properties of an object passed as a parameter in one go.</td>
</tr>
<tr>
<td>void</td>
<td><strong>setProperties</strong>(Properties properties, String prefix)</td>
<td>Set the properites for the object that match the prefix passed as parameter.</td>
</tr>
<tr>
<td></td>
<td><strong>setProperty</strong>(PropertyDescriptor prop, String name, String value)</td>
<td>Set the named property given a PropertyDescriptor.</td>
</tr>
<tr>
<td></td>
<td><strong>setProperty</strong>(String name, String value)</td>
<td>Set a property on this PropertySetter's Object.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
### Field Detail

**obj**

protected `Object` obj

---

**props**

protected `PropertyDescriptor[]` props

### Constructor Detail

**PropertySetter**

public `PropertySetter(Object obj)`

Create a new PropertySetter for the specified Object. This is done in preparation for invoking `setProperty(java.lang.String, java.lang.String)` one or more times.

**Parameters:**

- `obj` - the object for which to set properties

### Method Detail

**introspect**

protected void `introspect()`

Uses JavaBeans `Introspector` to computer setters of object to be configured.

**setProperties**

public static void `setProperties(Object obj, Properties properties,`
Set the properties of an object passed as a parameter in one go. The properties are parsed relative to a prefix.

**Parameters:**
- `obj` - The object to configure.
- `properties` - A `java.util.Properties` containing keys and values.
- `prefix` - Only keys having the specified prefix will be set.

---

**setProperties**

```java
public void setProperties(Properties properties, String prefix)
```

Set the properties for the object that match the `prefix` passed as parameter.

---

**setProperty**

```java
public void setProperty(String name, String value)
```

Set a property on this PropertySetter's Object. If successful, this method will invoke a setter method on the underlying Object. The setter is the one for the specified property name and the value is determined partly from the setter argument type and partly from the value specified in the call to this method.

If the setter expects a String no conversion is necessary. If it expects an int, then an attempt is made to convert 'value' to an int using new `Integer(value)`. If the setter expects a boolean, the conversion is by new `Boolean(value)`.

**Parameters:**
- `name` - name of the property
- `value` - String value of the property
setProperty

public void setProperty(PropertyDescriptor prop, String name, String value) throws PropertySetterException

Set the named property given a PropertyDescriptor.

Parameters:

prop - A PropertyDescriptor describing the characteristics of the property to set.
name - The name of the property to set.
value - The value of the property.

convertArg

protected Object convertArg(String val, Class type)

Convert val a String parameter to an object of a given type.

getPropertyDescriptor

protected PropertyDescriptor getPropertyDescriptor(String name)

activate

public void activate()
public class PropertySetterException extends Exception

Thrown when an error is encountered whilst attempting to set a property using the PropertySetter utility class.

Since: 1.1

Author: Anders Kristensen

See Also: Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>protected</th>
<th>rootCause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throwable</td>
<td></td>
</tr>
</tbody>
</table>

Constructor Summary

PropertySetterException(String msg)

PropertySetterException(Throwable rootCause)
<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>String</strong></td>
</tr>
<tr>
<td>Returns descriptive text on the cause of this exception.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Throwable
- `fillInStackTrace`
- `getLocalizedMessage`
- `printStackTrace`, `printStackTrace`, `toString`  

Methods inherited from class java.lang.Object
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`  

<table>
<thead>
<tr>
<th>Field Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>rootCause</strong></td>
</tr>
</tbody>
</table>

protected `Throwable` `rootCause`  

<table>
<thead>
<tr>
<th>Constructor Detail</th>
</tr>
</thead>
</table>

**PropertySetterException**

public `PropertySetterException(String msg)`  

**PropertySetterException**

public `PropertySetterException(Throwable rootCause)`  

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
</table>
public String getMessage()

Returns descriptive text on the cause of this exception.

Overrides:

getMessage in class Throwable
Package org.apache.log4j.helpers

This package is used internally.

See:   Description

<table>
<thead>
<tr>
<th>Class Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsoluteTimeDateFormat</td>
<td>Formats a Date in the format &quot;HH:mm:ss,SSS&quot; for example, &quot;15:49:37,459&quot;.</td>
</tr>
<tr>
<td>AppenderAttachableImpl</td>
<td>A straightforward implementation of the AppenderAttachable interface.</td>
</tr>
<tr>
<td>BoundedFIFO</td>
<td>BoundedFIFO serves as the bounded first-in-first-out buffer heavily used by the AsyncAppender.</td>
</tr>
<tr>
<td>CountingQuietWriter</td>
<td>Counts the number of bytes written.</td>
</tr>
<tr>
<td>CyclicBuffer</td>
<td>CyclicBuffer is used by other appenders to hold LoggingEvents for immediate or differed display.</td>
</tr>
<tr>
<td>DateLayout</td>
<td>This abstract layout takes care of all the date related options and formatting work.</td>
</tr>
<tr>
<td>DateTimeDateFormat</td>
<td>Formats a Date in the format &quot;dd MMM yyyy HH:mm:ss,SSS&quot; for example, &quot;06 Nov 1994 15:49:37,459&quot;.</td>
</tr>
<tr>
<td>FileWatchdog</td>
<td>Check every now and then that a certain file has not changed.</td>
</tr>
<tr>
<td>FormattingInfo</td>
<td>FormattingInfo instances contain the information obtained when parsing formatting modifiers in conversion modifiers.</td>
</tr>
<tr>
<td>ISO8601DateFormat</td>
<td>Formats a Date in the format &quot;yyyy-MM-dd HH:mm:ss,SSS&quot; for example &quot;1999-11-27 15:49:37,459&quot;.</td>
</tr>
<tr>
<td>Loader</td>
<td>Load resources (or images) from various sources.</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LogLog</td>
<td>This class used to output log statements from within the log4j package.</td>
</tr>
<tr>
<td>NullEnumeration</td>
<td>An always-empty Enumerator.</td>
</tr>
<tr>
<td>OnlyOnceErrorHandler</td>
<td>The <code>OnlyOnceErrorHandler</code> implements log4j's default error handling policy which consists of emitting a message for the first error in an appender and ignoring all following errors.</td>
</tr>
<tr>
<td>OptionConverter</td>
<td>A convenience class to convert property values to specific types.</td>
</tr>
<tr>
<td>PatternConverter</td>
<td>PatternConverter is an abstract class that provides the formatting functionality that derived classes need.</td>
</tr>
<tr>
<td>PatternParser</td>
<td>Most of the work of the <code>PatternLayout</code> class is delegated to the PatternParser class.</td>
</tr>
<tr>
<td>QuietWriter</td>
<td>QuietWriter does not throw exceptions when things go wrong.</td>
</tr>
<tr>
<td>RelativeTimeDateFormat</td>
<td>Formats a <code>Date</code> by printing the number of milliseconds elapsed since construction of the format.</td>
</tr>
<tr>
<td>SyslogQuietWriter</td>
<td>SyslogQuietWriter extends QuietWriter by prepending the syslog level code before each printed String.</td>
</tr>
<tr>
<td>SyslogWriter</td>
<td>SyslogWriter is a wrapper around the java.net.DatagramSocket class so that it behaves like a java.io.Writer.</td>
</tr>
<tr>
<td>ThreadLocalMap</td>
<td><code>ThreadLocalMap</code> extends <code>InheritableThreadLocal</code> to bequeath a copy of the hashtable of the MDC of the parent thread.</td>
</tr>
<tr>
<td>Transform</td>
<td>Utility class for transforming strings.</td>
</tr>
</tbody>
</table>
This package is used internally.

Last modified: Sat Jul 3 15:12:58 MDT 1999

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.helpers

Package Hierarchies:

All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.helpers.**AppenderAttachableImpl** (implements org.apache.log4j.spi.**AppenderAttachable**)
  - class org.apache.log4j.helpers.**BoundedFIFO**
  - class org.apache.log4j.helpers.**CyclicBuffer**
- class java.text.**Format** (implements java.lang.**Cloneable**, java.io.**Serializable**)
  - class java.text.**DateFormat**
  - class org.apache.log4j.helpers.**AbsoluteTimeDateFormat**
  - class org.apache.log4j.helpers.**RelativeTimeDateFormat**
- class org.apache.log4j.helpers.**FormattingInfo**
- class org.apache.log4j.helpers.**DateLayout** (implements org.apache.log4j.spi.**OptionHandler**)
  - class org.apache.log4j.helpers.**Loader**
  - class org.apache.log4j.helpers.**LogLog**
  - class org.apache.log4j.helpers.**NullEnumeration** (implements java.util.**Enumeration**)
  - class org.apache.log4j.helpers.**OnlyOnceErrorHandler** (implements org.apache.log4j.spi.**ErrorHandler**)
  - class org.apache.log4j.helpers.**OptionConverter**
  - class org.apache.log4j.helpers.**PatternConverter**
  - class org.apache.log4j.helpers.**PatternParser**
- class java.lang.**Thread** (implements java.lang.**Runnable**)
  - class org.apache.log4j.helpers.**FileWatchdog**
  - class java.lang.**ThreadLocal**
  - class java.lang.**InheritableThreadLocal**
  - class org.apache.log4j.helpers.**ThreadLocalMap**
- class org.apache.log4j.helpers.**Transform**
- class java.io.**Writer**
  - class java.io.**FilterWriter**
  - class org.apache.log4j.helpers.**QuietWriter**
    - class org.apache.log4j.helpers.**CountingQuietWriter**
    - class org.apache.log4j.helpers.**SyslogQuietWriter**
  - class org.apache.log4j.helpers.**SyslogWriter**

Copyright 2000-2005 Apache Software Foundation.
## Uses of Package org.apache.log4j.helpers

### Packages that use org.apache.log4j.helpers

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for remote logging.</td>
</tr>
</tbody>
</table>

### Classes in org.apache.log4j.helpers used by org.apache.log4j

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateLayout</td>
<td>This abstract layout takes care of all the date related options and formatting work.</td>
</tr>
<tr>
<td>PatternParser</td>
<td>Most of the work of the PatternLayout class is delegated to the PatternParser class.</td>
</tr>
<tr>
<td>QuietWriter</td>
<td>QuietWriter does not throw exceptions when things go wrong.</td>
</tr>
</tbody>
</table>

### Classes in org.apache.log4j.helpers used by org.apache.log4j.helpers

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsoluteTimeDateFormat</td>
<td>Formats a Date in the format &quot;HH:mm:ss,SSS&quot; for example, &quot;15:49:37,459&quot;.</td>
</tr>
<tr>
<td>FormattingInfo</td>
<td>FormattingInfo instances contain the information obtained when parsing formatting modifiers in conversion modifiers.</td>
</tr>
<tr>
<td>NullEnumeration</td>
<td>An always-empty Enumerator.</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>PatternConverter</td>
<td>PatternConverter is an abstract class that provides the formatting functionality that derived classes need.</td>
</tr>
<tr>
<td>QuietWriter</td>
<td>QuietWriter does not throw exceptions when things go wrong.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j.helpers used by org.apache.log4j.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyclicBuffer</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
public class AbsoluteTimeDateFormat extends DateFormat

Formats a date in the format "HH:mm:ss,SSS" for example, "15:49:37,459".

Since:
0.7.5

Author:
Ceki Gülcü, Andrew Vajoczki

See Also:
Serialized Form

Field Summary

| static String | ABS_TIME_DATE_FORMAT | String constant used to specify AbsoluteTimeDateFormat in layouts. |
| static String | DATE_AND_TIME_DATE_FORMAT | String constant used to specify DateTimeDateFormat in layouts. |
### Fields inherited from class java.text.DateFormat

<table>
<thead>
<tr>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM_PM_FIELD, calendar, DATE_FIELD, DAY_OF_WEEK_FIELD, DAY_OF_MONTH_FIELD, DAY_OF_YEAR_FIELD, DEFAULT, ERA_FIELD, FULL, HOUR_OF_DAY0_FIELD, HOUR_OF_DAY1_FIELD, HOUR0_FIELD, HOUR1_FIELD, LONG, MEDIUM, MILLISECOND_FIELD, MINUTE_FIELD, MONTH_FIELD, numberFormat, SECOND_FIELD, SHORT, TIMEZONE_FIELD, WEEK_OF_MONTH_FIELD, WEEK_OF_YEAR_FIELD, YEAR_FIELD</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsoluteTimeZoneDateFormat()</td>
</tr>
<tr>
<td>AbsoluteTimeZoneDateFormat(TimeZone timeZone)</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringBuffer format(Date date, StringBuffer sbuf, FieldPosition fieldPosition)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Date parse(String s, ParsePosition pos)</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.text.DateFormat:

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, format, format, getAvailableLocales, getCalendar, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getTimeInstance, getTimeInstance, getTimeInstance, getTimeInstance, getTimeInstance, hashCode, isLenient, parse, parseObject, getAvailableLocales, getCalendar</td>
</tr>
</tbody>
</table>

Methods inherited from class java.text.Format:

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>format, parseObject</td>
</tr>
</tbody>
</table>
Field Detail

**ABS_TIME_DATE_FORMAT**

```java
public static final String ABS_TIME_DATE_FORMAT
```

String constant used to specify `AbsoluteTimeDateFormat` in layouts. Current value is **ABSOLUTE**.

---

**DATE_AND_TIME_DATE_FORMAT**

```java
public static final String DATE_AND_TIME_DATE_FORMAT
```

String constant used to specify `DateTimeDateFormat` in layouts. Current value is **DATE**.

---

**ISO8601_DATE_FORMAT**

```java
public static final String ISO8601_DATE_FORMAT
```

String constant used to specify `ISO8601DateFormat` in layouts. Current value is **ISO8601**.

Constructor Detail

**AbsoluteTimeDateFormat**

```java
public AbsoluteTimeDateFormat()
```
AbsoluteTimeDateFormat

public AbsoluteTimeDateFormat(TimeZone timeZone)

Method Detail

format

public StringBuffer format(Date date, StringBuffer sbuf, FieldPosition fieldPosition)

Appends to sbuf the time in the format "HH:mm:ss,SSS" for example, "15:49:37,459"

Overrides: format in class DateFormat

Parameters:
  date - the date to format
  sbuf - the string buffer to write to
  fieldPosition - remains untouched

parse

public Date parse(String s, ParsePosition pos)

This method does not do anything but return null.

Overrides: parse in class DateFormat
public class AppenderAttachableImpl extends Object implements AppenderAttachable

A straightforward implementation of the AppenderAttachable interface.

Since:
version 0.9.1

Author:
Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>protected Vector appenderList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array of appenders.</td>
</tr>
</tbody>
</table>

Constructor Summary

AppenderAttachableImpl()

Method Summary

<table>
<thead>
<tr>
<th>void addAppender(Appender newAppender)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach an appender.</td>
</tr>
</tbody>
</table>

| int appendLoopOnAppenders(LoggingEvent event) |
Call the `doAppend` method on all attached appenders.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Enumeration getAllAppenders()</code></td>
<td>Get all attached appenders as an Enumeration.</td>
</tr>
<tr>
<td><code>Appender getAppender(String name)</code></td>
<td>Look for an attached appender named as <code>name</code>.</td>
</tr>
<tr>
<td><code>boolean isAttached(Appender appender)</code></td>
<td>Returns <code>true</code> if the specified appender is in the list of attached appenders, <code>false</code> otherwise.</td>
</tr>
<tr>
<td><code>void removeAllAppenders()</code></td>
<td>Remove and close all previously attached appenders.</td>
</tr>
<tr>
<td><code>void removeAppender(Appender appender)</code></td>
<td>Remove the appender passed as parameter form the list of attached appenders.</td>
</tr>
<tr>
<td><code>void removeAppender(String name)</code></td>
<td>Remove the appender with the name passed as parameter form the list of appenders.</td>
</tr>
</tbody>
</table>

### Methods inherited from class `java.lang.Object`

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Field Detail

**appenderList**

**protected Vector appenderList**

Array of appenders.

### Constructor Detail

**AppenderAttachableImpl**

**public AppenderAttachableImpl()**
Method Detail

addAppender

public void addAppender(Appender newAppender)

Attach an appender. If the appender is already in the list in won't be added again.

Specified by:
addAppender in interface AppenderAttachable

appendLoopOnAppenders

public int appendLoopOnAppenders(LoggingEvent event)

Call the doAppend method on all attached appenders.

getAllAppenders

public Enumeration getAllAppenders()

Get all attached appenders as an Enumeration. If there are no attached appenders null is returned.

Specified by:
getAllAppenders in interface AppenderAttachable

Returns:
Enumeration An enumeration of attached appenders.

getAppender

public Appender getAppender(String name)

Look for an attached appender named as name.

Return the appender with that name if in the list. Return null
otherwise.

**Specified by:**

`getAppender` in interface `AppenderAttachable`

---

**isAttached**

```java
public boolean isAttached(Appender appender)
```

Returns `true` if the specified appender is in the list of attached appenders, `false` otherwise.

**Specified by:**

`isAttached` in interface `AppenderAttachable`

**Since:**

1.2

---

**removeAllAppenders**

```java
public void removeAllAppenders()
```

Remove and close all previously attached appenders.

**Specified by:**

`removeAllAppenders` in interface `AppenderAttachable`

---

**removeAppender**

```java
public void removeAppender(Appender appender)
```

Remove the appender passed as parameter form the list of attached appenders.

**Specified by:**

`removeAppender` in interface `AppenderAttachable`
public void removeAppender(String name)

Remove the appender with the name passed as parameter form the list of appenders.

**Specified by:**
removeAppender in interface AppenderAttachable
public class **BoundedFIFO**
extends **Object**

BoundedFIFO serves as the bounded first-in-first-out buffer heavily used by the AsyncAppender.

**Since:**
version 0.9.1

**Author:**
Ceki Gülcü

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BoundedFIFO</strong>(int maxSize)</td>
<td>Instantiate a new BoundedFIFO with a maximum size passed as argument.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get()</code></td>
<td>Get the first element in the buffer.</td>
</tr>
<tr>
<td><code>getMaxSize()</code></td>
<td>Get the maximum size of the buffer.</td>
</tr>
<tr>
<td><code>isFull()</code></td>
<td>Return true if the buffer is full, that is, whether the number of elements in the buffer equals the buffer size.</td>
</tr>
<tr>
<td><code>length()</code></td>
<td>Get the number of elements in the buffer.</td>
</tr>
<tr>
<td><code>put(LoggingEvent o)</code></td>
<td></td>
</tr>
</tbody>
</table>
Place a `LoggingEvent` in the buffer.

```java
type Resize

void resize(int newSize)
   Resize the buffer to a new size.

boolean wasEmpty()
   Returns true if there is just one element in the buffer.

boolean wasFull()
   Returns true if the number of elements in the buffer plus 1 equals the maximum buffer size, returns false otherwise.
```

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

## Constructor Detail

### BoundedFIFO

```java
type BoundedFIFO

public BoundedFIFO(int maxSize)
   Instantiate a new BoundedFIFO with a maximum size passed as argument.
```

## Method Detail

### get

```java
function get()

public LoggingEvent get()
   Get the first element in the buffer. Returns null if there are no elements in the buffer.
```

### put

```java
function put
```
public void put(LoggingEvent o)

    Place a LoggingEvent in the buffer. If the buffer is full then the event is silently dropped. It is the caller's responsibility to make sure that the buffer has free space.

getMaxSize

public int getMaxSize()

    Get the maximum size of the buffer.

isFull

public boolean isFull()

    Return true if the buffer is full, that is, whether the number of elements in the buffer equals the buffer size.

length

public int length()

    Get the number of elements in the buffer. This number is guaranteed to be in the range 0 to maxSize (inclusive).

resize

public void resize(int newSize)

    Resize the buffer to a new size. If the new size is smaller than the old size events might be lost.

Since:

    1.1
wasEmpty

public boolean wasEmpty()

    Returns true if there is just one element in the buffer. In other words, if there were no elements before the last put(org.apache.log4j.spi.LoggingEvent) operation completed.

wasFull

public boolean wasFull()

    Returns true if the number of elements in the buffer plus 1 equals the maximum buffer size, returns false otherwise.
public class CountingQuietWriter
extends QuietWriter

Counts the number of bytes written.

Since:
0.8.1

Author:
Heinz Richter, heinz.richter@frogdot.com

Field Summary

| protected long count |

Fields inherited from class org.apache.log4j.helpers.QuietWriter
errorHandler

Fields inherited from class java.io.FilterWriter
out

Fields inherited from class java.io.Writer
lock
### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CountingQuietWriter(Writer writer, ErrorHandler eh)</code></td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td><code>getCount()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setCount(long count)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>write(String string)</code></td>
</tr>
</tbody>
</table>

### Methods inherited from class `org.apache.log4j.helpers.QuietWriter`:
- `flush`, `setErrorHandler`

### Methods inherited from class `java.io.FilterWriter`:
- `close`, `write`, `write`, `write`

### Methods inherited from class `java.io.Writer`:
- `write`  

### Methods inherited from class `java.lang.Object`:
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Field Detail

<table>
<thead>
<tr>
<th>Field</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>count</code></td>
<td>protected long</td>
</tr>
</tbody>
</table>
constructor detail

countingquietwriter

public countingquietwriter(writer writer, errorhandler eh)

method detail

write

public void write(string string)

overrides:
write in class quietwriter

getcount

public long getcount()

setcount

public void setcount(long count)
Class CyclicBuffer

public class CyclicBuffer extends Object

CyclicBuffer is used by other appenders to hold LoggingEvents for immediate or differed display.

This buffer gives read access to any element in the buffer not just the first or last element.

Since: 0.9.0

Author: Ceki Gülcü

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyclicBuffer(int maxSize)</td>
<td>Instantiate a new CyclicBuffer of at most maxSize events.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add(LoggingEvent event)</td>
<td>Add an event as the last event in the buffer.</td>
</tr>
<tr>
<td>LoggingEvent get()</td>
<td>Get the oldest (first) element in the buffer.</td>
</tr>
<tr>
<td>LoggingEvent get(int i)</td>
<td>Get the i-th oldest event currently in the buffer.</td>
</tr>
<tr>
<td>int getMaxSize()</td>
<td></td>
</tr>
</tbody>
</table>
int length()
Get the number of elements in the buffer.

void resize(int newSize)
Resize the cyclic buffer to newSize.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CyclicBuffer

public CyclicBuffer(int maxSize)
throws IllegalArgumentException

Instantiate a new CyclicBuffer of at most maxSize events. The
maxSize argument must a positive integer.

Parameters:
maxSize - The maximum number of elements in the buffer.

Method Detail

add

public void add(LoggingEvent event)

Add an event as the last event in the buffer.

get

public LoggingEvent get(int i)

Get the ith oldest event currently in the buffer. If i is outside the range
0 to the number of elements currently in the buffer, then null is
getMaxSize

public int getMaxSize()

get

public LoggingEvent get()

Get the oldest (first) element in the buffer. The oldest element is removed from the buffer.

length

public int length()

Get the number of elements in the buffer. This number is guaranteed to be in the range 0 to maxSize (inclusive).

resize

public void resize(int newSize)

Resize the cyclic buffer to newSize.

Throws:

IllegalStateException - if newSize is negative.
<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>

PREVIOUS CLASS  NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

Log4j 1.2.14
PREV CLASS  NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

Log4j 1.2.14
public abstract class DateLayout extends Layout

This abstract layout takes care of all the date related options and formatting work.

Author:
Ceki Gülcü

---

Field Summary

<table>
<thead>
<tr>
<th></th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Date</td>
<td></td>
</tr>
</tbody>
</table>

|                     | DATE_FORMATOPTION
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>dateFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected DateFormat</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NULL_DATE_FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>String constant designating no time information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td></td>
</tr>
</tbody>
</table>
### Field Position

<table>
<thead>
<tr>
<th>static String RELATIVE_TIME_DATE_FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>String constant designating relative time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String TIMEZONE_OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
</tbody>
</table>

### Fields inherited from class org.apache.log4j.Layout

| LINE_SEP, LINE_SEP_LEN |

### Constructor Summary

**DateLayout()**

### Method Summary

**activateOptions()**

- Activate the options that were previously set with calls to option setters.

**dateFormat(StringBuffer buf, LoggingEvent event)**

**getDateFormat()**

- Returns value of the DateFormat option.

**getOptionStrings()**

- Deprecated. Use the setter method for the option directly instead of the generic setOption method.

**getTimeZone()**

- Returns value of the TimeZone option.

**setDateFormat(DateFormat dateFormat, TimeZone timeZone)**

- Sets the DateFormat used to format time and date in the zone determined by timeZone.

**setDateFormat(String dateFormat)**

- The value of the DateFormat option should be either an
void argument to the constructor of SimpleDateFormat or one of the strings "NULL", "RELATIVE", "ABSOLUTE", "DATE" or "ISO8601.

void setDateFormat(String dateFormatType, TimeZone timeZone)
Sets the DateFormat used to format date and time in the time zone determined by timeZone parameter.

void setOption(String option, String value)
Deprecated. Use the setter method for the option directly instead of the generic setOption method.

void setTimeZone(String timeZone)
The TimeZoneID option is a time zone ID string in the format expected by the TimeZone.getTimeZone(java.lang.String) method.

Methods inherited from class org.apache.log4j.Layout
format, getContentType, getFooter, getHeader, ignoresThrowable

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

NULL_DATE_FORMAT

public static final String NULL_DATE_FORMAT

String constant designating no time information. Current value of this constant is NULL.

RELATIVE_TIME_DATE_FORMAT

public static final String RELATIVE_TIME_DATE_FORMAT
String constant designating relative time. Current value of this constant is RELATIVE.

pos

protected FieldPosition pos

DATE_FORMAT_OPTION

public static final String DATE_FORMAT_OPTION

Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

TIMEZONE_OPTION

public static final String TIMEZONE_OPTION

Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

dateFormat

protected DateFormat dateFormat

date

protected Date date

Constructor Detail
DateLayout

public DateLayout()

### Method Detail

#### getOptionStrings

public String[] getOptionStrings()

**Deprecated.** Use the setter method for the option directly instead of the generic `setOption` method.

#### setOption

public void setOption(String option, String value)

**Deprecated.** Use the setter method for the option directly instead of the generic `setOption` method.

#### setDateFormat

public void setDateFormat(String dateFormat)

The value of the `DateFormat` option should be either an argument to the constructor of `SimpleDateFormat` or one of the strings "NULL", "RELATIVE", "ABSOLUTE", "DATE" or "ISO8601.

#### getDateFormat

public String getDateFormat()

Returns value of the `DateFormat` option.
setTimeZone
public void setTimeZone(String timeZone)

The TimeZoneID option is a time zone ID string in the format expected by the TimeZone.getTimeZone(java.lang.String) method.

getTimeZone
public String getTimeZone()

Returns value of the TimeZone option.

activateOptions
public void activateOptions()

Description copied from interface: OptionHandler
Activate the options that were previously set with calls to option setters.

This allows to defer activation of the options until all options have been set. This is required for components which have related options that remain ambiguous until all are set.

For example, the FileAppender has the File and Append options both of which are ambiguous until the other is also set.

dateFormat
public void dateFormat(StringBuffer buf, LoggingEvent event)

setDateFormat
public void setDateFormat(DateFormat dateFormat,
TimeZone timeZone)

Sets the DateFormat used to format time and date in the zone determined by timeZone.

**setDateFormat**

```java
public void setDateFormat(String dateFormatType, TimeZone timeZone)
```

Sets the DateFormat used to format date and time in the time zone determined by timeZone parameter. The DateFormat used will depend on the dateFormatType.

The recognized types are NULL_DATE_FORMAT,
RELATIVE_TIME_DATE_FORMAT
AbsoluteTimeDateFormat.ABS_TIME_DATE_FORMAT,
AbsoluteTimeDateFormat.DATE_AND_TIME_DATE_FORMAT and
AbsoluteTimeDateFormat.ISO8601_DATE_FORMAT. If the dateFormatType is not one of the above, then the argument is assumed to be a date pattern for SimpleDateFormat.
public class DateTimeDateFormat extends AbsoluteTimeDateFormat

Formats a date in the format "dd MMM yyyy HH:mm:ss,SSS" for example, "06 Nov 1994 15:49:37,459".

Since: 0.7.5
Author: Ceki Gülcü
See Also: Serialized Form

Fields inherited from class org.apache.log4j.helpers.AbsoluteTimeDateFormat
ABS_TIME_DATE_FORMAT, DATE_AND_TIME_DATE_FORMAT, ISO8601_DATE_FORMAT

Fields inherited from class java.text.DateFormat
AM_PM_FIELD, calendar, DATE_FIELD, DAY_OF_WEEK_FIELD, DAY_OF_MONTH_FIELD, DAY_OF_YEAR_FIELD, DEFAULT, ERA_FIELD, FULL, HOUR_OF_DAY0_FIELD, HOUR_OF_DAY1_FIELD, HOUR0_FIELD, HOUR1_FIELD, LONG, MEDIUM, MILLISECOND_FIELD, MINUTE_FIELD,
### Constructor Summary

**DateTimeDateFormat()**

**DateTimeDateFormat(TimeZone timeZone)**

### Method Summary

**StringBuffer format(Date date, StringBuffer sbuf, FieldPosition fieldPosition)**

Appends to `sbuf` the date in the format "dd MMM yyyy HH:mm:ss,SSS" for example, "06 Nov 1994 08:49:37,459".

**Date parse(String s, ParsePosition pos)**

This method does not do anything but return null.

### Methods inherited from class java.text.**DateFormat**

cloned, equals, format, format, getAvailableLocales, getCalendar, getDateInstance, getDateInstance, getDateInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getInstance, getNumberFormat, getTimeInstance, getTimeInstance, getTimeInstance, getTimeZone, hashCode, isLenient, parse, parseObject, setCalendar, setLenient, setNumberFormat, setTimeZone

### Methods inherited from class java.text.**Format**

format, parseObject

### Methods inherited from class java.lang.**Object**

finalize, getClass, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail
**dateTimeDateFormat**

public dateTimeDateFormat()

---

**dateTimeDateFormat**

public dateTimeDateFormat(TimeZone timeZone)

### Method Detail

**format**

public StringBuffer format(Date date,
   StringBuffer sbuf,
   FieldPosition fieldPosition)

Appends to `sbuf` the date in the format "dd MMM yyyy HH:mm:ss,SSS" for example, "06 Nov 1994 08:49:37,459".

**Overrides:**
format in class AbsoluteTimeDateFormat

**Parameters:**
- sbuf - the string buffer to write to

**parse**

public Date parse(String s,
   ParsePosition pos)

This method does not do anything but return null.

**Overrides:**
parse in class AbsoluteTimeDateFormat

---

**Overview** | **Package** | **Use** | **Tree** | **Deprecated** | **Index** | **Help** | Log4j 1.2.14
---|---|---|---|---|---|---|---
PREV CLASS | NEXT CLASS | SUMMARY: INNER | FIELD | CONSTR | METHOD | FRAMES | NO FRAMES | DETAIL | FIELD | CONSTR | METHOD

Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>Specified Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
</tr>
<tr>
<td>SUMMARY: INNER</td>
<td>FIELD</td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
</tr>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>

```
Overview  Package  Use  Tree  Deprecated  Index  Help  Log4j
```

```
1.2.14
PREV  CLASS  NEXT  CLASS  SUMMARY:  INNER  |  FIELD  |  CONSTR  |  METHOD  |
FRAMES  |  NO  FRAMES  |
DETAIL:  FIELD  |  CONSTR  |  METHOD  |
```
public abstract class FileWatchdog extends Thread

Check every now and then that a certain file has not changed. If it has, then call the doOnChange() method.

Since:
version 0.9.1

Author:
Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static long</td>
<td>DEFAULT_DELAY</td>
<td>The default delay between every file modification check, set to 60 seconds.</td>
</tr>
<tr>
<td>protected long</td>
<td>delay</td>
<td>The delay to observe between every check.</td>
</tr>
<tr>
<td>protected String</td>
<td>filename</td>
<td>The name of the file to observe for changes.</td>
</tr>
</tbody>
</table>

Fields inherited from class java.lang.Thread

MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY
Constructor Summary

```java
protected FileWatchdog(String filename)
```

Method Summary

```java
protected void checkAndConfigure()
protected abstract void doOnChange()
void run()
void setDelay(long delay)
```

Methods inherited from class java.lang.Thread

```java
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield
```

Methods inherited from class java.lang.Object

```java
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait
```

Field Detail

```
DEFAULT_DELAY
```

```java
public static final long DEFAULT_DELAY
```

The default delay between every file modification check, set to 60
seconds.

**filename**

```
protected String filename
```

The name of the file to observe for changes.

**delay**

```
protected long delay
```

The delay to observe between every check. By default set `DEFAULT_DELAY`.

### Constructor Detail

**FileWatchdog**

```
protected FileWatchdog(String filename)
```

### Method Detail

**setDelay**

```
public void setDelay(long delay)
```

Set the delay to observe between each check of the file changes.

**doOnChange**

```
protected abstract void doOnChange()
```
checkAndConfigure

protected void checkAndConfigure()

run

public void run()

Overrides:
    run in class Thread
<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: INNER</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 class FormattingInfo extends Object

FormattingInfo instances contain the information obtained when parsing formatting modifiers in conversion modifiers.

Since: 0.8.2
Author: Jim Cakalic, Ceki Gülcü

Constructor Summary

| FormattingInfo() |

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

FormattingInfo

public FormattingInfo()
Copyright 2000-2005 Apache Software Foundation.
public class **ISO8601DateFormat**
events **AbsoluteTimeDateFormat**

Formats a `Date` in the format "yyyy-MM-dd HH:mm:ss,SSS" for example "1999-11-27 15:49:37,459".

Refer to the [summary of the International Standard Date and Time Notation](https://www.iso.org/obp/ui/#query=ISO%208601) for more information on this format.

**Since:**
0.7.5

**Author:**
Ceki Gülcü, Andrew Vajoczki

**See Also:**
[Serialized Form](https://www.apache.org)
### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO8601DateFormat()</td>
<td></td>
</tr>
<tr>
<td>ISO8601DateFormat(TimeZone timeZone)</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringBuffer format(Date date, StringBuffer sbuf, FieldPosition fieldPosition)</td>
<td>Appends a date in the format &quot;YYYY-mm-dd HH:mm:ss,SSS&quot; to sbuf.</td>
</tr>
<tr>
<td>Date parse(String s, ParsePosition pos)</td>
<td>This method does not do anything but return null.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.text.DateFormat

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, format, format, getAvailableLocales, getCalendar, getDateInstance, getDateInstance, getDateInstance, getDateInstance, getDateInstance, getDateTimeInstance, getDateTimeInstance, getDateTimeInstance, getInstance, getNumberFormat, getTimeInstance, getTimeInstance, getTimeZone, hashCode, isLenient, parse, parseObject, setCalendar, setLenient, setNumberFormat, setTimeZone</td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class java.text.Format

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>format, parseObject</td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>finalize, getClass, notify, notifyAll, toString, wait, wait, wait</td>
<td></td>
</tr>
</tbody>
</table>
Constructor Detail

ISO8601DateFormat

public ISO8601DateFormat()

ISO8601DateFormat

public ISO8601DateFormat(TimeZone timeZone)

Method Detail

format

public StringBuffer format(Date date, StringBuffer sbuf, FieldPosition fieldPosition)

Appends a date in the format "YYYY-mm-dd HH:mm:ss,SSS" to sbuf. For example: "1999-11-27 15:49:37,459".

Overrides: format in class AbsoluteTimeDateFormat

Parameters:

sbuf - the StringBuffer to write to

parse

public Date parse(String s, ParsePosition pos)

This method does not do anything but return null.

Overrides: parse in class AbsoluteTimeDateFormat
<table>
<thead>
<tr>
<th>SUMMARY:</th>
<th>INNER</th>
<th>FIELD</th>
<th>CONSTR</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETAIL:</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
public class **Loader**
extends **Object**

Load resources (or images) from various sources.

**Author:**
Ceki Gülcü

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loader()</strong></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getResource(String resource)</strong></td>
<td>This method will search for resource in different places.</td>
</tr>
<tr>
<td><strong>getResource(String resource, Class clazz)</strong></td>
<td>Deprecated. <em>as of 1.2.</em></td>
</tr>
<tr>
<td><strong>isJava1()</strong></td>
<td>Are we running under JDK 1.x?</td>
</tr>
<tr>
<td><strong>loadClass(String clazz)</strong></td>
<td>If running under JDK 1.2 load the specified class using the Thread contextClassLoader if that fails try Class.forName.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.**Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll,
Constructor Detail

Loader

public Loader()

Method Detail

getResource

public static URL getResource(String resource, Class clazz)

    Deprecated. as of 1.2.

    Get a resource by delegating to getResource(String).

    Parameters:
    resource - resource name
    clazz - class, ignored.

    Returns:
    URL to resource or null.

getResource

public static URL getResource(String resource)

    This method will search for resource in different places. The search order is as follows:

    1. Search for resource using the thread context class loader under Java2. If that fails, search for resource using the class loader that loaded this class (Loader). Under JDK 1.1, only the the class loader that loaded this class (Loader) is used.
2. Try one last time with 
   `ClassLoader.getSystemResource(resource)`, that is is using the 
   system class loader in JDK 1.2 and virtual machine's built-in 
   class loader in JDK 1.1.

**isJava1**

```java
public static boolean isJava1()

    Are we running under JDK 1.x?
```

**loadClass**

```java
public static Class loadClass(String clazz)
    throws ClassNotFoundException

    If running under JDK 1.2 load the specified class using the Thread 
    contextClassLoader if that fails try Class.forName. Under JDK 1.1 
    only Class.forName is used.
```
public class LogLog
extends Object

This class used to output log statements from within the log4j package.

Log4j components cannot make log4j logging calls. However, it is sometimes useful for the user to learn about what log4j is doing. You can enable log4j internal logging by defining the log4j.configDebug variable.

All log4j internal debug calls go to System.out where as internal error messages are sent to System.err. All internal messages are prepended with the string "log4j: ".

Since: 0.8.2
Author: Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>CONFIG_DEBUG_KEY</td>
<td>Deprecated. Use DEBUG_KEY instead.</td>
</tr>
<tr>
<td>static String</td>
<td>DEBUG_KEY</td>
<td>Defining this value makes log4j print log4j-internal debug statements to System.out.</td>
</tr>
<tr>
<td>protected static boolean</td>
<td>debugEnabled</td>
<td></td>
</tr>
</tbody>
</table>

Constructor Summary
# Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>debug</strong> <em>(String msg)</em></td>
<td>This method is used to output log4j internal debug statements.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>debug</strong> <em>(String msg, Throwable t)</em></td>
<td>This method is used to output log4j internal debug statements.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>error</strong> <em>(String msg)</em></td>
<td>This method is used to output log4j internal error statements.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>error</strong> <em>(String msg, Throwable t)</em></td>
<td>This method is used to output log4j internal error statements.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>setInternalDebugging</strong> <em>(boolean enabled)</em></td>
<td>Allows to enable/disable log4j internal logging.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>setQuietMode</strong> <em>(boolean quietMode)</em></td>
<td>In quite mode no LogLog generates strictly no output, not even for errors.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>warn</strong> <em>(String msg)</em></td>
<td>This method is used to output log4j internal warning statements.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>warn</strong> <em>(String msg, Throwable t)</em></td>
<td>This method is used to output log4j internal warnings.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.*Object*

*clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait*

# Field Detail
**DEBUG_KEY**

public static final String DEBUG_KEY

Defining this value makes log4j print log4j-internal debug statements to System.out.

The value of this string is log4j.debug.

Note that the search for all option names is case sensitive.

---

**CONFIG_DEBUG_KEY**

public static final String CONFIG_DEBUG_KEY

*Deprecated. Use DEBUG_KEY instead.*

Defining this value makes log4j components print log4j-internal debug statements to System.out.

The value of this string is log4j.configDebug.

Note that the search for all option names is case sensitive.

---

**debugEnabled**

protected static boolean debugEnabled

---

**Constructor Detail**

**LogLog**

public LogLog()
setInternalDebugging

public static void setInternalDebugging(boolean enabled)

    Allows to enable/disable log4j internal logging.

---

debbug

public static void debug(String msg)

    This method is used to output log4j internal debug statements. Output goes to System.out.

---

debbug

public static void debug(String msg, Throwable t)

    This method is used to output log4j internal debug statements. Output goes to System.out.

---

erroor

public static void error(String msg)

    This method is used to output log4j internal error statements. There is no way to disable error statements. Output goes to System.err.

---

erroor

public static void error(String msg, Throwable t)

    This method is used to output log4j internal error statements. There is no way to disable error statements. Output goes to System.err.
setQuietMode

public static void setQuietMode(boolean quietMode)

In quite mode no LogLog generates strictly no output, not even for errors.

Parameters:
quietMode - A true for not

warn

public static void warn(String msg)

This method is used to output log4j internal warning statements. There is no way to disable warning statements. Output goes to System.err.

warn

public static void warn(String msg, Throwable t)

This method is used to output log4j internal warnings. There is no way to disable warning statements. Output goes to System.err.
public class NullEnumeration extends Object implements Enumeration

An always-empty Enumerator.

Since:
    version 1.0

Author:
    Anders Kristensen

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>static NullEnumeration</td>
<td>getInstance()</td>
</tr>
<tr>
<td>boolean</td>
<td>hasMoreElements()</td>
</tr>
<tr>
<td>Object</td>
<td>nextElement()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
### Method Detail

#### getInstance

```java
public static NullEnumeration getInstance()
```

#### hasMoreElements

```java
public boolean hasMoreElements()
```

**Specified by:**
- `hasMoreElements` in interface `Enumeration`

#### nextElement

```java
public Object nextElement()
```

**Specified by:**
- `nextElement` in interface `Enumeration`

---

Copyright 2000-2005 Apache Software Foundation.
public class OnlyOnceErrorHandler extends Object implements ErrorHandler

The OnlyOnceErrorHandler implements log4j's default error handling policy which consists of emitting a message for the first error in an appender and ignoring all following errors.

The error message is printed on System.err.

This policy aims at protecting an otherwise working application from being flooded with error messages when logging fails.

Since:
0.9.0

Author:
Ceki Gülcü

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnlyOnceErrorHandler()</td>
<td>void activateOptions()</td>
</tr>
</tbody>
</table>

No options to activate.
### void error(String message)
Print a the error message passed as parameter on System.err.

### void error(String message, Exception e, int errorCode)
Prints the message and the stack trace of the exception on System.err.

### void error(String message, Exception e, int errorCode, LoggingEvent event)
Prints the message and the stack trace of the exception on System.err.

### void setAppender(Appender appender)
Does not do anything.

### void setBackupAppender(Appender appender)
Does not do anything.

### void setLogger(Logger logger)
Does not do anything.

---

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

### Constructor Detail

**OnlyOnceErrorHandler**

```java
public OnlyOnceErrorHandler()
```

### Method Detail

**setLogger**

```java
public void setLogger(Logger logger)
```

Does not do anything.
Specified by:
   setLogger in interface ErrorHandler
Following copied from interface: org.apache.log4j.spi.ErrorHandler

Parameters:
   logger - One of the loggers that will be searched for the failing appender in view of replacement.

activateOptions

public void activateOptions()

   No options to activate.

Specified by:
   activateOptions in interface OptionHandler

error

public void error(String message, Exception e, int errorCode)

   Prints the message and the stack trace of the exception on System.err.

Specified by:
   error in interface ErrorHandler

error

public void error(String message, Exception e, int errorCode, LoggingEvent event)

   Prints the message and the stack trace of the exception on System.err.

Specified by:
   error in interface ErrorHandler
Following copied from interface: org.apache.log4j.spi.ErrorHandler

Parameters:
- message - The message associated with the error.
- e - The Exception that was thrown when the error occurred.
- errorCode - The error code associated with the error.
- event - The logging event that the failing appender is asked to log.

error

public void error(String message)

Print a the error message passed as parameter on System.err.

Specified by:
- error in interface ErrorHandler

setAppender

public void setAppender(Appender appender)

Does not do anything.

Specified by:
- setAppender in interface ErrorHandler

setBackupAppender

public void setBackupAppender(Appender appender)

Does not do anything.

Specified by:
- setBackupAppender in interface ErrorHandler
Copyright 2000-2005 Apache Software Foundation.
public class OptionConverter extends Object

A convenience class to convert property values to specific types.

Author:
Ceki Gülcü, Simon Kitching; Anders Kristensen

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String[] <strong>concatenateArrays</strong> (String[] l, String[] r)</td>
</tr>
<tr>
<td>static String <strong>convertSpecialChars</strong> (String s)</td>
</tr>
<tr>
<td>static String <strong>findAndSubst</strong> (String key, Properties props)</td>
</tr>
<tr>
<td>static String <strong>getSystemProperty</strong> (String key, String def)</td>
</tr>
<tr>
<td>static Object <strong>instantiateByClassName</strong> (String className, Class superClass, Object defaultValue)</td>
</tr>
<tr>
<td>static Object <strong>instantiateByKey</strong> (Properties props, String key, Class superClass, Object defaultValue)</td>
</tr>
<tr>
<td>static void <strong>selectAndConfigure</strong> (URL url, String clazz, LoggerRepository hierarchy)</td>
</tr>
<tr>
<td>static void <strong>substVars</strong> (String val, Properties props)</td>
</tr>
</tbody>
</table>
Perform variable substitution in string `val` from the values of keys found in the system properties.

```java
static String performVariableSubstitution(String val) {
    // Implementation
}
```

If value is "true", then `true` is returned.

```java
static boolean toBoolean(String value, boolean defaultValue) {
    // Implementation
}
```

Converts a standard or custom priority level to a `Level` object.

```java
static Level toLevel(String value, Level defaultValue) {
    // Implementation
}
```
Parameters:
  key - The key to search for.
  def - The default value to return.

Returns:
  the string value of the system property, or the default value if there is no property with that key.

Since:
  1.1

---

**instantiateByKey**

```java
public static Object instantiateByKey(Properties props, String key, Class superClass, Object defaultValue)
```

---

**toBoolean**

```java
public static boolean toBoolean(String value, boolean dEfault)
```

If value is "true", then true is returned. If value is "false", then true is returned. Otherwise, default is returned.

Case of value is unimportant.

---

**toInt**

```java
public static int toInt(String value, int dEfault)
```

---

**toLevel**

```java
public static Level toLevel(String value, Level defaultValue)
```

Converts a standard or custom priority level to a Level object.
If `value` is of form "level#classname", then the specified class' `toLevel` method is called to process the specified level string; if no '#' character is present, then the default `Level` class is used to process the level value.

As a special case, if the `value` parameter is equal to the string "NULL", then the value `null` will be returned.

If any error occurs while converting the value to a level, the `defaultValue` parameter, which may be `null`, is returned.

Case of `value` is insignificant for the level `level`, but is significant for the class name `part`, if present.

**Since:**

1.1

---

### toFileSize

```java
public static long toFileSize(String value, long defaultValue)
```

---

### findAndSubst

```java
public static String findAndSubst(String key, Properties props)
```

Find the value corresponding to `key` in `props`. Then perform variable substitution on the found value.

---

### instantiateByClassName

```java
public static Object instantiateByClassName(String className, Class superClass, Object defaultValue)
```

Instantiate an object given a class name. Check that the `className` is
a subclass of superClass. If that test fails or the object could not be instantiated, then defaultValue is returned.

Parameters:

- className - The fully qualified class name of the object to instantiate.
- superClass - The class to which the new object should belong.
- defaultValue - The object to return in case of non-fulfillment.

 substVars

public static String substVars(String val, Properties props)
throws IllegalArgumentException

Perform variable substitution in string val from the values of keys found in the system properties.

The variable substitution delimiters are ${ }.

For example, if the System properties contains "key=value", then the call

String s = OptionConverter.substituteVars("Value of key is \${key}.");

will set the variable s to "Value of key is value."

If no value could be found for the specified key, then the props parameter is searched, if the value could not be found there, then substitution defaults to the empty string.

For example, if system properties contains no value for the key "inexistentKey", then the call

String s = OptionConverter.substituteVars("Value of inexistentKey is []");

will set s to "Value of inexistentKey is []"

An IllegalArgumentException is thrown if val contains a start delimeter "${" which is not balanced by a stop delimeter "}".
Parameters:
  val - The string on which variable substitution is performed.

Throws:
  IllegalArgumentException - if val is malformed.
<table>
<thead>
<tr>
<th>Summary: Inner</th>
<th>Field</th>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail: Field</td>
<td>Constructor</td>
<td>Method</td>
<td></td>
</tr>
</tbody>
</table>
public abstract class PatternConverter extends Object

PatternConverter is an abstract class that provides the formatting functionality that derived classes need.

Conversion specifiers in a conversion patterns are parsed to individual PatternConverters. Each of which is responsible for converting a logging event in a converter specific manner.

Since: 0.8.2
Author: James P. Cakalic, Ceki Gülcü
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected abstract String</td>
<td>convert(LoggingEvent event)</td>
<td>Derived pattern converters must override this method in order to convert conversion specifiers in the correct way.</td>
</tr>
<tr>
<td>void</td>
<td>format(StringBuffer sbuf, LoggingEvent e)</td>
<td>A template method for formatting in a converter specific way.</td>
</tr>
<tr>
<td>void</td>
<td>spacePad(StringBuffer sbuf, int length)</td>
<td>Fast space padding method.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.**Object**
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Field Detail**

next

public **PatternConverter** next

**Constructor Detail**

**PatternConverter**

protected **PatternConverter()**

**PatternConverter**

protected **PatternConverter(FormattingInfo fi)**

**Method Detail**

convert
protected abstract String convert(LoggingEvent event)

Derived pattern converters must override this method in order to convert conversion specifiers in the correct way.

format

public void format(StringBuffer sbuf, LoggingEvent e)

A template method for formatting in a converter specific way.

spacePad

public void spacePad(StringBuffer sbuf, int length)

Fast space padding method.
**org.apache.log4j.helpers Class PatternParser**

```java
java.lang.Object
   +-- org.apache.log4j.helpers.PatternParser
```

public class PatternParser extends Object

Most of the work of the PatternLayout class is delegated to the PatternParser class.

It is this class that parses conversion patterns and creates a chained list of OptionConverters.

**Since:** 0.8.2

**Author:** [James P. Cakalic](mailto:), Ceki Gülcü, Anders Kristensen

### Field Summary

<table>
<thead>
<tr>
<th>protected</th>
<th>currentLiteral</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringBuffer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected</th>
<th>formattingInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormattingInfo</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected</th>
<th>patternLength</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td></td>
</tr>
</tbody>
</table>
PatternParser(String pattern)

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void</td>
<td>addConverter(PatternConverter pc)</td>
</tr>
<tr>
<td>protected String</td>
<td>extractOption()</td>
</tr>
<tr>
<td>protected int</td>
<td>extractPrecisionOption()</td>
</tr>
<tr>
<td>protected void</td>
<td>finalizeConverter(char c)</td>
</tr>
<tr>
<td>PatternConverter</td>
<td>parse()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
class, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

currentLiteral

protected StringBuffer currentLiteral

patternLength

protected int patternLength
protected int i

formattingInfo

protected FormattingInfo formattingInfo

pattern

protected String pattern

Constructor Detail

PatternParser

public PatternParser(String pattern)

Method Detail

extractOption

protected String extractOption()

extractPrecisionOption

protected int extractPrecisionOption()

    The option is expected to be in decimal and positive. In case of error, zero is returned.

parse

public PatternConverter parse()
finalizeConverter

protected void finalizeConverter(char c)

addConverter

protected void addConverter(PatternConverter pc)
<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: INNER  FIELD  CONSTR  METHOD

FRAMES  NO FRAMES
DETAIL: FIELD  CONSTR  METHOD
Class QuietWriter


Direct Known Subclasses: CountingQuietWriter, SyslogQuietWriter

public class QuietWriter extends FilterWriter

QuietWriter does not throw exceptions when things go wrong. Instead, it delegates error handling to its ErrorHandler.

Since: 0.7.3
Author: Ceki Gülcü

### Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>errorHandler</td>
</tr>
</tbody>
</table>

Fields inherited from class java.io.FilterWriter:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td></td>
</tr>
</tbody>
</table>

Fields inherited from class java.io.Writer:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>lock</td>
<td></td>
</tr>
</tbody>
</table>
## Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QuietWriter</strong></td>
<td><code>Writer writer, ErrorHandler errorHandler</code></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>flush()</code></td>
<td></td>
</tr>
<tr>
<td><code>setErrorHandler</code></td>
<td><code>ErrorHandler eh</code></td>
</tr>
<tr>
<td><code>write</code></td>
<td><code>String string</code></td>
</tr>
</tbody>
</table>

## Methods inherited from class `java.io.FilterWriter`

- `close`, `write`, `write`, `write`

## Methods inherited from class `java.io.Writer`

- `write`

## Methods inherited from class `java.lang.Object`

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

## Field Detail

- `ErrorHandler` `errorHandler`

## Constructor Detail

- `QuietWriter`
public "QuietWriter"(Writer writer,
               ErrorHandler errorHandler)

Method Detail

write

public void write(String string)

Overrides:
        write in class Writer

flush

public void flush()

Overrides:
        flush in class FilterWriter

setErrorHandler

public void setErrorHandler(ErrorHandler eh)
public class RelativeTimeDateFormat extends DateFormat

Formats a date by printing the number of milliseconds elapsed since construction of the format. This is the fastest printing DateFormat in the package.

Since:
    0.7.5

Author:
    Ceki Gülcü

See Also:
    Serialized Form

---

Field Summary

| protected long startTime |

Fields inherited from class java.text.DateField

AM_PM_FIELD, calendar, DATE_FIELD, DAY_OF_WEEK_FIELD,
DAY_OF_WEEK_IN_MONTH_FIELD, DAY_OF_YEAR_FIELD, DEFAULT, ERA_FIELD,
FULL, HOUR_OF_DAY0_FIELD, HOUR_OF_DAY1_FIELD, HOUR0_FIELD,
HOUR1_FIELD, LONG, MEDIUM, MILLISECOND_FIELD, MINUTE_FIELD,
### Constructor Summary

**RelativeTimeDateFormat**

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>format(Date date, StringBuffer sbuf, FieldPosition fieldPosition)</code></td>
<td>Appends to sbuf the number of milliseconds elapsed since the start of the application.</td>
</tr>
<tr>
<td><code>parse(String s, ParsePosition pos)</code></td>
<td>This method does not do anything but return <code>null</code>.</td>
</tr>
</tbody>
</table>

### Methods inherited from class `java.text.SimpleDateFormat`

- `clone`, `equals`, `format`, `format`, `getAvailableLocales`, `getCalendar`, `getDateTimeInstance`, `getInstance`, `getNumberFormat`, `getTimeInstance`, `getTimeZone`, `hashCode`, `isLenient`, `parse`, `parseObject`, `setCalendar`, `setLenient`, `setNumberFormat`, `setTimeZone`

### Methods inherited from class `java.text.Format`

- `format`, `parseObject`

### Methods inherited from class `java.lang.Object`

- `finalize`, `getClass`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Field Detail

**startTime**

protected final long **startTime**
RelativeTimeDateFormat

public RelativeTimeDateFormat()

Method Detail

format

public StringBuffer format(Date date,
                        StringBuffer sbuf,
                        FieldPosition fieldPosition)

Appends to sbuf the number of milliseconds elapsed since the start of the application.

Overrides:
    format in class DateFormat

Since:
    0.7.5

parse

public Date parse(String s,
                  ParsePosition pos)

This method does not do anything but return null.

Overrides:
    parse in class DateFormat
public class **SyslogQuietWriter** extends **QuietWriter**

SyslogQuietWriter extends QuietWriter by prepending the syslog level code before each printed String.

Since:

0.7.3

### Fields inherited from class org.apache.log4j.helpers.QuietWriter
- `errorHandler`

### Fields inherited from class java.io.FilterWriter
- `out`

### Fields inherited from class java.io.Writer
- `lock`

### Constructor Summary

```java
SyslogQuietWriter(Writer writer, int syslogFacility, ErrorHandler eh)
```
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void setLevel</td>
<td>int level</td>
</tr>
<tr>
<td>void setSyslogFacility</td>
<td>int syslogFacility</td>
</tr>
<tr>
<td>void write</td>
<td>String string</td>
</tr>
</tbody>
</table>

Methods inherited from class org.apache.log4j.helpers.*QuietWriter*
flush, setErrorHandler

Methods inherited from class java.io.*FilterWriter*
close, write, write, write

Methods inherited from class java.io.*Writer*
write

Methods inherited from class java.lang.*Object*
close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

**SyslogQuietWriter**

```java
public SyslogQuietWriter(Writer writer, int syslogFacility, ErrorHandler eh)
```

### Method Detail

**setLevel**
public void setLevel(int level)

setSyslogFacility

public void setSyslogFacility(int syslogFacility)

write

public void write(String string)

Overrides:
write in class QuietWriter
public class **SyslogWriter**
extends **Writer**

SyslogWriter is a wrapper around the java.net.DatagramSocket class so that it behaves like a java.io.Writer.

**Since:**
0.7.3

<table>
<thead>
<tr>
<th>Fields inherited from class java.io.Writer</th>
</tr>
</thead>
<tbody>
<tr>
<td>lock</td>
</tr>
</tbody>
</table>

### Constructor Summary

**SyslogWriter(String syslogHost)**

Constructs a new instance of SyslogWriter.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
</tr>
<tr>
<td>void flush()</td>
</tr>
<tr>
<td>void write(char[] buf, int off, int len)</td>
</tr>
<tr>
<td>void write(String string)</td>
</tr>
<tr>
<td>Methods inherited from class java.io.Writer</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>write, write, write</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, toString, wait, wait</td>
</tr>
</tbody>
</table>

## Constructor Detail

### SyslogWriter

```java
public SyslogWriter(String syslogHost)
```

Constructs a new instance of SyslogWriter.

**Parameters:**
- `syslogHost` - host name, may not be null. A port may be specified by following the name or IPv4 literal address with a colon and a decimal port number. To specify a port with an IPv6 address, enclose the IPv6 address in square brackets before appending the colon and decimal port number.

## Method Detail

### write

```java
public void write(char[] buf, int off, int len)
throws IOException
```

**Overrides:**
- `write` in class `Writer`
public void write(String string)
    throws IOException

    Overrides:
        write in class Writer

flush

close
public final class **ThreadLocalMap** extends **InheritableThreadLocal**

ThreadLocalMap extends **InheritableThreadLocal** to bequeath a copy of the hashtable of the MDC of the parent thread.

### Since:
1.2

### Author:
Ceki Gülcü

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ThreadLocalMap()</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object childValue</strong>(Object parentValue)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.**InheritableThreadLocal**
get, set

Methods inherited from class java.lang.**ThreadLocal**
initialValue
Constructor Detail

ThreadLocalMap

public ThreadLocalMap()

Method Detail

childValue

public final Object childValue(Object parentValue)

Overrides: childValue in class InheritableThreadLocal

Overview Package Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS SUMMARY: INNER | FIELD | CONSTR | METHOD FRAMES NO FRAMES DETAIL: FIELD | CONSTR | METHOD

Copyright 2000-2005 Apache Software Foundation.
## Overview

Log4j 1.2.14

PREV CLASS  NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

<table>
<thead>
<tr>
<th>FRAME</th>
<th>NO FRAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>
public class Transform
extends Object

Utility class for transforming strings.

Author:
Ceki Gülcü, Michael A. McAngus

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform()</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendEscapingCDATA(buf, str)</td>
<td>Ensures that embedded CDEnd strings (]]&gt;) are handled properly within message, NDC and throwable tag text.</td>
</tr>
<tr>
<td>escapeTags(input)</td>
<td>This method takes a string which may contain HTML tags (ie, &lt;b&gt;, &lt;table&gt;, etc) and replaces any '&lt;' and '&gt;' characters with respective predefined entity references.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Constructor Detail

Transform

public Transform()

Method Detail

escapeTags

public static String escapeTags(String input)

This method takes a string which may contain HTML tags (ie, <b>, <table>, etc) and replaces any '<' and '>' characters with respective predefined entity references.

Parameters:
input - The text to be converted.

Returns:
The input string with the characters '<' and '>' replaced with &lt; and &gt; respectively.

appendEscapingCDATA

public static void appendEscapingCDATA(StringBuffer buf, String str)

Ensures that embedded CDEnd strings ([]) are handled properly within message, NDC and throwable tag text.

Parameters:
buf - StringBuffer holding the XML data to this point. The initial CDStart () of the CDATA section are the responsibility of the calling method.
str - The String that is inserted into an existing CDATA Section within buf.
Copyright 2000-2005 Apache Software Foundation.
Package org.apache.log4j.jdbc

The JDBCAppender provides for sending log events to a database.

See: Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBCAppender</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.jdbc Description

The JDBCAppender provides for sending log events to a database.

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.jdbc

Package Hierarchies:

All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
  - class org.apache.log4j.jdbc.**JDBCAppender** (implements org.apache.log4j.**Appender**)

Copyright 2000-2005 Apache Software Foundation.
Uses of Package
org.apache.log4j.jdbc

No usage of org.apache.log4j.jdbc

Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
<th>Log4j 1.2.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>SUMMARY: INNER</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
</tr>
</tbody>
</table>
public class JDBCAppender extends AppenderSkeleton implements Appender

WARNING: This version of JDBCAppender is very likely to be completely replaced in the future. Moreover, it does not log exceptions. The JDBCAppender provides for sending log events to a database.

Each append call adds to an ArrayList buffer. When the buffer is filled each log event is placed in a sql statement (configurable) and executed. BufferSize, db URL, User, & Password are configurable options in the standard log4j ways.

The setSql(String sql) sets the SQL statement to be used for logging -- this statement is sent to a PatternLayout (either created automaticly by the appender or added by the user). Therefore by default all the conversion patterns in PatternLayout can be used inside of the statement. (see the test cases for examples)

Overriding the getLogStatement(org.apache.log4j.spi.LoggingEvent) method allows more explicit control of the statement used for logging.

For use as a base class:

- Override getConnection() to pass any connection you want. Typically this is used to enable application wide connection pooling.
Override closeConnection(Connection con) -- if you override getConnection make sure to implement closeConnection to handle the connection you generated. Typically this would return the connection to the pool it came from.

Override getLogStatement(LoggingEvent event) to produce specialized or dynamic statements. The default uses the sql option value.

Author:
Kevin Steppe (ksteppe@pacbell.net)

---

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>ArrayList</td>
<td>ArrayList holding the buffer of Logging Events.</td>
</tr>
<tr>
<td>bufferSize</td>
<td>int</td>
<td>size of LoggingEvent buffer before writing to the database.</td>
</tr>
<tr>
<td>connection</td>
<td>Connection</td>
<td>Connection used by default.</td>
</tr>
<tr>
<td>databasePassword</td>
<td>String</td>
<td>User to use for default connection handling</td>
</tr>
<tr>
<td>databaseURL</td>
<td>String</td>
<td>URL of the DB for default connection handling</td>
</tr>
<tr>
<td>databaseUser</td>
<td>String</td>
<td>User to connect as for default connection handling</td>
</tr>
<tr>
<td>removes</td>
<td>ArrayList</td>
<td>Helper object for clearing out the buffer</td>
</tr>
<tr>
<td>sqlStatement</td>
<td>String</td>
<td>Stores the string given to the pattern layout for conversion into a SQL statement, eg: insert into LogTable (Thread, Class, Message) values (&quot;%t&quot;, &quot;%c&quot;, &quot;%m&quot;).</td>
</tr>
</tbody>
</table>

Fields inherited from class org.apache.log4j.AppenderSkeleton

closed, errorHandler, headFilter, layout, name, tailFilter, threshold
## Constructor Summary

**JDBCAppender**

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>append</strong>&lt;br&gt;(LoggingEvent event)</td>
<td>Adds the event to the buffer.</td>
</tr>
<tr>
<td><strong>close</strong></td>
<td>Closes the appender, flushing the buffer first then closing the default connection if it is open.</td>
</tr>
<tr>
<td><strong>closeConnection</strong>&lt;br&gt;(Connection con)</td>
<td>Override this to return the connection to a pool, or to clean up the resource.</td>
</tr>
<tr>
<td><strong>execute</strong>&lt;br&gt;(String sql)</td>
<td>Override this to provide an alternate method of getting connections (such as caching).</td>
</tr>
<tr>
<td><strong>finalize</strong></td>
<td>closes the appender before disposal</td>
</tr>
<tr>
<td><strong>flushBuffer</strong></td>
<td>loops through the buffer of LoggingEvents, gets a sql string from getLogStatement() and sends it to execute().</td>
</tr>
<tr>
<td><strong>getBufferSize</strong></td>
<td></td>
</tr>
<tr>
<td><strong>getConnection</strong></td>
<td>Override this to link with your connection pooling system.</td>
</tr>
<tr>
<td><strong>getLogStatement</strong>&lt;br&gt;(LoggingEvent event)</td>
<td>By default getLogStatement sends the event to the required Layout object.</td>
</tr>
<tr>
<td><strong>getPassword</strong></td>
<td></td>
</tr>
<tr>
<td><strong>getSql</strong></td>
<td></td>
</tr>
</tbody>
</table>
String

Returns pre-formatted statement eg: insert into LogTable (msg) values ("%m")

String

getURL ()

String

getUser ()

boolean

requiresLayout ()

JDBCAppender requires a layout.

void

setBufferSize (int newBufferSize)

void

setDriver (String driverClass)

Ensures that the given driver class has been loaded for sql connection creation.

void

setPassword (String password)

void

setSql (String s)

void

setURL (String url)

void

setUser (String user)

Methods inherited from class org.apache.log4j.AppenderSkeleton
activateOptions, addFilter, clearFilters, doAppend, getErrorHandler, getFilter, getFirstFilter, getLayout, getName, getThreshold, isAsSevereAsThreshold, setErrorHandler, setLayout, setName, setThreshold

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface org.apache.log4j.Appender
addFilter, clearFilters, doAppend, getErrorHandler, getFilter, getLayout, getName, setErrorHandler, setLayout, setName
Field Detail

**databaseURL**

protected `String` **databaseURL**

URL of the DB for default connection handling

**databaseUser**

protected `String` **databaseUser**

User to connect as for default connection handling

**databasePassword**

protected `String` **databasePassword**

User to use for default connection handling

**connection**

protected `Connection` **connection**

Connection used by default. The connection is opened the first time it is needed and then held open until the appender is closed (usually at garbage collection). This behavior is best modified by creating a sub-class and overriding the `getConnection` and `closeConnection` methods.

**sqlStatement**

protected `String` **sqlStatement**
Stores the string given to the pattern layout for conversion into a SQL statement, eg: insert into LogTable (Thread, Class, Message) values ("%t", "%c", "%m"). Be careful of quotes in your messages! Also see PatternLayout.

---

**bufferSize**

protected int bufferSize

size of LoggingEvent buffer before writing to the database. Default is 1.

---

**buffer**

protected ArrayList buffer

ArrayList holding the buffer of Logging Events.

---

**removes**

protected ArrayList removes

Helper object for clearing out the buffer

---

**Constructor Detail**

**JDBCAppender**

public JDBCAppender()

---

**Method Detail**

**append**
public void append(LoggingEvent event)

    Adds the event to the buffer. When full the buffer is flushed.
    Overrides:
        append in class AppenderSkeleton

getLogStatement

protected String getLogStatement(LoggingEvent event)

    By default getLogStatement sends the event to the required Layout object. The layout will format the given pattern into a workable SQL string. Overriding this provides direct access to the LoggingEvent when constructing the logging statement.

execute

protected void execute(String sql)
    throws SQLException

    Override this to provide an alternate method of getting connections (such as caching). One method to fix this is to open connections at the start of flushBuffer() and close them at the end. I use a connection pool outside of JDBCAppender which is accessed in an override of this method.

closeConnection

protected void closeConnection(Connection con)

    Override this to return the connection to a pool, or to clean up the resource. The default behavior holds a single connection open until the appender is closed (typically when garbage collected).

getConnection
protected Connection getConnection() throws SQLException

Override this to link with your connection pooling system. By default this creates a single connection which is held open until the object is garbage collected.

---

close

public void close()

Closes the appender, flushing the buffer first then closing the default connection if it is open.

**Specified by:**

close in interface Appender

---

flushBuffer

public void flushBuffer()

loops through the buffer of LoggingEvents, gets a sql string from getLogStatement() and sends it to execute(). Errors are sent to the errorHandler. If a statement fails the LoggingEvent stays in the buffer!

---

finalize

public void finalize()

closes the appender before disposal

**Overrides:**

finalize in class AppenderSkeleton

---

requiresLayout

public boolean requiresLayout()
JDBCAppender requires a layout.

Specified by:

*requiresLayout* in interface *Appender*

---

**setSql**

```java
public void setSql(String s)
```

---

**getSql**

```java
public String getSql()
```

Returns pre-formatted statement eg: insert into LogTable (msg) values ("%m")

---

**setUser**

```java
public void setUser(String user)
```

---

**setURL**

```java
public void setURL(String url)
```

---

**setPassword**

```java
public void setPassword(String password)
```

---

**setBufferSize**

```java
public void setBufferSize(int newBufferSize)
```

---

**getUser**
public String getUser()

getURL

public String getURL()

getPassword

public String getPassword()

getBufferSize

public int getBufferSize()

setDriver

public void setDriver(String driverClass)

Ensures that the given driver class has been loaded for sql connection creation.
Package org.apache.log4j.jmx

This package lets you manage log4j settings using JMX.

See:  Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractDynamicMBean</td>
</tr>
<tr>
<td>Agent</td>
</tr>
<tr>
<td>AppenderDynamicMBean</td>
</tr>
<tr>
<td>HierarchyDynamicMBean</td>
</tr>
<tr>
<td>LayoutDynamicMBean</td>
</tr>
<tr>
<td>LoggerDynamicMBean</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.jmx Description

This package lets you manage log4j settings using JMX. It is unfortunately not of production quality.

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.jmx

Package Hierarchies:
   All Packages
Class Hierarchy

- class `java.lang.Object`
  - class `org.apache.log4j.jmx.AbstractDynamicMBean`
    (implements `javax.management.DynamicMBean`, `javax.management.MBeanRegistration`)
  - class `org.apache.log4j.jmx.AppenderDynamicMBean`
  - class `org.apache.log4j.jmx.HierarchyDynamicMBean`
  - class `org.apache.log4j.jmx.LayoutDynamicMBean`
  - class `org.apache.log4j.jmx.LoggerDynamicMBean`
    (implements `javax.management.NotificationListener`)
  - class `org.apache.log4j.jmx.Agent`
Uses of Package
org.apache.log4j.jmx

Packages that use org.apache.log4j.jmx

<table>
<thead>
<tr>
<th>Package</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.jmx</td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
</tbody>
</table>

Classes in org.apache.log4j.jmx used by org.apache.log4j.jmx

AbstractDynamicMBean

Copyright 2000-2005 Apache Software Foundation.
<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: INNER | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

Log4j 1.2.14
Class AbstractDynamicMBean

java.lang.Object
    +--- org.apache.log4j.jmx.AbstractDynamicMBean

All Implemented Interfaces:
    javax.management.DynamicMBean,
    javax.management.MBeanRegistration

Direct Known Subclasses:
    AppenderDynamicMBean, HierarchyDynamicMBean,
    LayoutDynamicMBean, LoggerDynamicMBean

public abstract class AbstractDynamicMBean
extends Object
implements javax.management.DynamicMBean,
javax.management.MBeanRegistration

Constructor Summary

AbstractDynamicMBean()  

Method Summary

javax.management.AttributeList
getAttributes(String[] attributeNames)
    Enables the to get the values of several attributes of the Dynamic MBean.

protected abstract Logger
getLogger()

void
postDeregister()

void
postRegister(Boolean registrationDone)
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>preDeregister()</td>
<td>void</td>
</tr>
<tr>
<td>preRegister()</td>
<td>void(javax.management.MBeanServer server,</td>
</tr>
<tr>
<td></td>
<td>javax.management.ObjectName name)</td>
</tr>
<tr>
<td>setAttributes()</td>
<td>void(javax.management.AttributeList attributes)</td>
</tr>
</tbody>
</table>

Sets the values of several attributes of the MBean, and returns the list of attributes that have

```java
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
```

```java
Methods inherited from interface javax.management.DynamicMBean
getAttribute, getMBeanInfo, invoke, setAttribute
```

### Constructor Detail

**AbstractDynamicMBean**

```java
public AbstractDynamicMBean()
```

### Method Detail

**getAttributes**

```java
public javax.management.AttributeList getAttributes(String[] attributes)
```

Enables the to get the values of several attributes of the Dynamic MBean.

**Specified by:**

getAttributes in interface javax.management.DynamicMBean
**setAttributes**

```java
public javax.management.AttributeList setAttributes(javax.management.AttributeList attributes)
```

Sets the values of several attributes of the Dynamic MBean, and returns the list of attributes that have been set.

**Specified by:**
- `setAttributes in interface javax.management.DynamicMBean`

---

**getLogger**

```java
protected abstract Logger getLogger()
```

---

**postDeregister**

```java
public void postDeregister()
```

**Specified by:**
- `postDeregister in interface javax.management.MBeanRegistration`

---

**postRegister**

```java
public void postRegister(Boolean registrationDone)
```

**Specified by:**
- `postRegister in interface javax.management.MBeanRegistration`

---

**preDeregister**

```java
public void preDeregister()
```

**Specified by:**
- `preDeregister in interface javax.management.MBeanRegistration`
preRegister

public javax.management.ObjectName preRegister(javax.management.MBeanServer server, javax.management.ObjectName name)

Specified by:
preRegister in interface javax.management.MBeanRegistration

Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
<th>Log4j 1.2.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: INNER</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
</tr>
</tbody>
</table>

SUMMARY:
- INNER
- FIELD
- CONSTR
- METHOD

DETAIL:
- FIELD
- CONSTR
- METHOD
public class Agent
extends Object

Constructor Summary

| Agent() |

Method Summary

| void start() |

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Agent

public Agent()

Method Detail

start
public void start()
<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: INNER</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>

Log4j 1.2.14
org.apache.log4j.jmx  Class AppenderDynamicMBean

java.lang.Object

|--org.apache.log4j.jmx.AbstractDynamicMBean
  |--org.apache.log4j.jmx.AppenderDynamicMBean

All Implemented Interfaces:
javax.management.DynamicMBean,
javax.management.MBeanRegistration

public class AppenderDynamicMBean
extends AbstractDynamicMBean

Constructor Summary

AppenderDynamicMBean(Appender appender)

Method Summary

Object getAttribute(String attributeName)

protected Logger getLogger()

javax.management.MBeanInfo getMBeanInfo()

Object invoke(String operationName, Object[] params,
String[] signature)

javax.management.ObjectName preRegister(javax.management.MBeanServer	server,
javax.management.ObjectName	name)

void setAttribute(javax.management.Attribute attrit
Methods inherited from class org.apache.log4j.jmx.AbstractDynamicMBean
getAttributes, postDeregister, postRegister, preDeregister, setAttributes

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AppenderDynamicMBean

public AppenderDynamicMBean(Appender appender) throws IntrospectionException

Method Detail

getMBeanInfo

public javax.management.MBeanInfo getMBeanInfo()

invoke

public Object invoke(String operationName,
Object[] params,
String[] signature)
throws javax.management.MBeanException,
javax.management.ReflectionException

getLogger
protected Logger getLogger()

Overrides:
getLogger in class AbstractDynamicMBean

getAttribute

public Object getAttribute(String attributeName)
throws javax.management.AttributeNotFoundException,
javax.management.MBeanException,
javax.management.ReflectionException

setAttribute

public void setAttribute(javax.management.Attribute attribute)
throws javax.management.AttributeNotFoundException,
javax.management.InvalidAttributeValueException,
javax.management.MBeanException,
javax.management.ReflectionException

preRegister

public javax.management.ObjectName preRegister(javax.management.MBeanServer server,
javax.management.ObjectName name)

Overrides:
preRegister in class AbstractDynamicMBean

Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>Class</th>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
</table>

This is a summary of the class with constructors and methods.
public class HierarchyDynamicMBean
extends AbstractDynamicMBean
implements HierarchyEventListener,
javax.management.MBeanRegistration,
javax.management.NotificationBroadcaster

Constructor Summary

| HierarchyDynamicMBean() |

Method Summary

<p>| void addAppenderEvent(Category logger, Appender event) |
| String addLoggerMBean(String name) |
| void addNotificationListener(javax.management.NotificationFilter filter) |
| Object getAttribute(String attributeName) |
| Logger getLogger() |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>javax.management.MBeanInfo</code></td>
<td><code>getMBeanInfo()</code></td>
</tr>
<tr>
<td><code>javax.management.MBeanNotificationInfo[]</code></td>
<td><code>getNotificationInfo()</code></td>
</tr>
<tr>
<td><code>Object</code></td>
<td><code>invoke(String operationName, Object[] params)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>postRegister(Boolean registrationDone)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>removeAppenderEvent(Category cat, Appender appender)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>removeNotificationListener(javax.management.NotificationListener listener)</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>setAttribute(javax.management.Attribute attribute)</code></td>
</tr>
</tbody>
</table>

Methods inherited from class `org.apache.log4j.jmx.AbstractDynamicMBean`:
- `getAttributes`, `postDeregister`, `preDeregister`, `preRegister`, `setAttributes`

Methods inherited from class `java.lang.Object`:
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

**Constructor Detail**

**HierarchyDynamicMBean**

public `HierarchyDynamicMBean()`
addLoggerMBean

public javax.management.ObjectName addLoggerMBean(String name)

addNotificationListener

public void addNotificationListener(javax.management.NotificationListener listener, javax.management.NotificationFilter filter, Object handback)

Specified by:
    addNotificationListener in interface javax.management.NotificationBroadcaster

getLogger

protected Logger getLogger()

Overrides:
    getLogger in class AbstractDynamicMBean

getMBeanInfo

public javax.management.MBeanInfo getMBeanInfo()

getNotificationInfo

public javax.management.MBeanNotificationInfo[] getNotificationInfo()

Specified by:
    getNotificationInfo in interface javax.management.NotificationBroadcaster

invoke
public Object invoke(String operationName,
        Object[] params,
        String[] signature)
throws javax.management.MBeanException,
        javax.management.ReflectionException

getAttribute

public Object getAttribute(String attributeName)
throws javax.management.AttributeNotFoundException,
        javax.management.MBeanException,
        javax.management.ReflectionException

addAppenderEvent

public void addAppenderEvent(Category logger,
        Appender appender)

        Specified by:
        addAppenderEvent in interface HierarchyEventListener

removeAppenderEvent

public void removeAppenderEvent(Category cat,
        Appender appender)

        Specified by:
        removeAppenderEvent in interface HierarchyEventListener

postRegister

public void postRegister(Boolean registrationDone)

        Overrides:
        postRegister in class AbstractDynamicMBean
removeNotificationListener

public void removeNotificationListener(javax.management.NotificationListener listener)
throws javax.management.ListenerNotFoundException

Specified by:
removeNotificationListener in interface javax.management.NotificationBroadcaster

setAttribute

public void setAttribute(javax.management.Attribute attribute)
throws javax.management.AttributeNotFoundException,
javax.management.InvalidAttributeValueException,
javax.management.MBeanException,
javax.management.ReflectionException
Class LayoutDynamicMBean

java.lang.Object

+- org.apache.log4j.jmx.AbstractDynamicMBean
  +- org.apache.log4j.jmx.LayoutDynamicMBean

All Implemented Interfaces:
javax.management.DynamicMBean,
javax.management.MBeanRegistration

public class LayoutDynamicMBean
extends AbstractDynamicMBean

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LayoutDynamicMBean(Layout layout)</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>getAttribute(String attributeName)</td>
</tr>
<tr>
<td>protected Logger</td>
<td>getLogger()</td>
</tr>
<tr>
<td>javax.management.MBeanInfo</td>
<td>getMBeanInfo()</td>
</tr>
<tr>
<td>Object</td>
<td>invoke(String operationName, Object[] params, String[] signature)</td>
</tr>
<tr>
<td>void</td>
<td>setAttribute(javax.management.Attribute attribute)</td>
</tr>
</tbody>
</table>

Methods inherited from class
AbstractDynamicMBean

getAttributes, postDeregister, postRegister, preDeregister, preRegister, setAttributes

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LayoutDynamicMBean

public LayoutDynamicMBean(Layout layout)
throws IntrospectionException

Method Detail

getMBeanInfo

public javax.management.MBeanInfo getMBeanInfo()

invoke

public Object invoke(String operationName,
Object[] params,
String[] signature)
throws javax.management.MBeanException,
javax.management.ReflectionException

getLogger

protected Logger getLogger()

Overrides:
getLogger in class AbstractDynamicMBean
getAttributes

public Object getAttribute(String attributeName)
    throws javax.management.AttributeNotFoundException,
             javax.management.MBeanException,
             javax.management.ReflectionException

setAttribute

public void setAttribute(javax.management.Attribute attribute)
    throws javax.management.AttributeNotFoundException,
           javax.management.InvalidAttributeValueException,
           javax.management.MBeanException,
           javax.management.ReflectionException
<table>
<thead>
<tr>
<th>SUMMARY</th>
<th>INNER</th>
<th>FIELD</th>
<th>CONSTR</th>
<th>METHOD</th>
<th>FIELD</th>
<th>CONSTR</th>
<th>METHOD</th>
<th>CONSTR</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAME</td>
<td>NO FRAME</td>
<td>FRAME</td>
<td>NO FRAME</td>
<td>FRAME</td>
<td>NO FRAME</td>
<td>FRAME</td>
<td>NO FRAME</td>
<td>FRAME</td>
<td>NO FRAME</td>
</tr>
</tbody>
</table>

Overview Package Use Tree Deprecated Index Help Log4j 1.2.14
PREV CLASS NEXT CLASS SUMMARY: INNER | FIELD | CONSTR | METHOD
FRAMES NO FRAMES DETAIL: FIELD | CONSTR | METHOD
org.apache.log4j.jmx Class LoggerDynamicMBean

java.lang.Object
  +-org.apache.log4j.jmx.AbstractDynamicMBean
    +-org.apache.log4j.jmx.LoggerDynamicMBean

All Implemented Interfaces:
  javax.management.DynamicMBean, EventListener,
  javax.management.MBeanRegistration,
  javax.management.NotificationListener

public class LoggerDynamicMBean extends AbstractDynamicMBean
  implements javax.management.NotificationListener

Constructor Summary

LoggerDynamicMBean(Logger logger)

Method Summary

Object getAttribute(String attributeName)

protected Logger getLogger()

javax.management.MBeanInfo getMBeanInfo()

void handleNotification(javax.management.Notification notification,
                        Object handback)

Object invoke(String operationName, Object[] params,
               String[] signature)
Methods inherited from class org.apache.log4j.jmx.AbstractDynamicMBean
getAttributes, postDeregister, preDeregister, preRegister, setAttributes

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LoggerDynamicMBean

public LoggerDynamicMBean(Logger logger)

Method Detail

handleNotification

public void handleNotification(javax.management.Notification notification notific Object handback)

Specified by:
handleNotification in interface javax.management.NotificationListener

getLogger
protected Logger getLogger()

Overrides:
  getLogger in class AbstractDynamicMBean

getMBeanInfo

public javax.management.MBeanInfo getMBeanInfo()

invoke

public Object invoke(String operationName,
  Object[] params,
  String[] signature)
  throws javax.management.MBeanException,
  javax.management.ReflectionException

getAttribute

public Object getAttribute(String attributeName)
  throws javax.management.AttributeNotFoundException,
  javax.management.MBeanException,
  javax.management.ReflectionException

setAttribute

public void setAttribute(javax.management.Attribute attribute)
  throws javax.management.AttributeNotFoundException,
  javax.management.MBeanException,
  javax.management.InvalidAttributeValueException,
  javax.management.ReflectionException

postRegister

public void postRegister(Boolean registrationDone)

Overrides:

postRegister in class AbstractDynamicMBean
Package org.apache.log4j.lf5

Interface Summary

| LogRecordFilter | An interface for classes which filters LogRecords. |

Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderFinalizer</td>
<td>AppenderFinalizer has a single method that will finalize resources associated with a LogBrokerMonitor in the event that the LF5Appender class is destroyed, and the class loader is garbage collected.</td>
</tr>
<tr>
<td>DefaultLF5Configurator</td>
<td>The DefaultLF5Configurator provides a default configuration for the LF5Appender.</td>
</tr>
<tr>
<td>LF5Appender</td>
<td>LF5Appender logs events to a swing based logging console.</td>
</tr>
<tr>
<td>Log4JLogRecord</td>
<td>A Log4JLogRecord encapsulates the details of your log4j LoggingEvent in a format usable by the LogBrokerMonitor.</td>
</tr>
<tr>
<td>LogLevel</td>
<td>The LogLevel class defines a set of standard logging levels.</td>
</tr>
<tr>
<td>LogRecord</td>
<td>LogRecord.</td>
</tr>
<tr>
<td>PassingLogRecordFilter</td>
<td>An implementation of LogRecordFilter which always returns true.</td>
</tr>
<tr>
<td>StartLogFactor5</td>
<td>Starts an instance of the LogFactor5 console for off-line viewing.</td>
</tr>
</tbody>
</table>

Exception Summary

<table>
<thead>
<tr>
<th>Exception Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogLevelFormatException</td>
<td>Thrown to indicate that the client has attempted to convert a string to one the LogLevel types, but the string does not</td>
</tr>
</tbody>
</table>
have the appropriate format.
Hierarchy For Package org.apache.log4j.lf5

Package Hierarchies:

All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.lf5.**AppenderFinalizer**
  - class org.apache.log4j.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.lf5.**LF5Appender**
  - class org.apache.log4j.lf5.**DefaultLF5Configurator** (implements org.apache.log4j.spi.**Configurator**)
  - class org.apache.log4j.lf5.**LogLevel** (implements java.io.**Serializable**)
    - class org.apache.log4j.lf5.**LogRecord** (implements java.io.**Serializable**)
      - class org.apache.log4j.lf5.**Log4JLogRecord**
    - class org.apache.log4j.lf5.**PassingLogRecordFilter** (implements org.apache.log4j.lf5.**LogRecordFilter**)
  - class org.apache.log4j.lf5.**StartLogFactor5**
  - class java.lang.**Throwable** (implements java.io.**Serializable**)
    - class java.lang.**Exception**
      - class org.apache.log4j.lf5.**LogLevelFormatException**
Interface Hierarchy

- interface org.apache.log4j.lf5.LogRecordFilter
## Uses of Package

org.apache.log4j.lf5

### Packages that use org.apache.log4j.lf5

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.lf5</td>
</tr>
</tbody>
</table>

### Classes in org.apache.log4j.lf5 used by org.apache.log4j.lf5

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderFinalizer</td>
<td>AppenderFinalizer has a single method that will finalize resources associated with a LogBrokerMonitor in the event that the LF5Appender class is destroyed, and the class loader is garbage collected.</td>
</tr>
<tr>
<td>LF5Appender</td>
<td>LF5Appender logs events to a swing based logging console.</td>
</tr>
<tr>
<td>LogLevel</td>
<td>The LogLevel class defines a set of standard logging levels.</td>
</tr>
<tr>
<td>LogLevelFormatException</td>
<td>Thrown to indicate that the client has attempted to convert a string to one the LogLevel types, but the string does not have the appropriate format.</td>
</tr>
<tr>
<td>LogRecord</td>
<td>LogRecord.</td>
</tr>
<tr>
<td>LogRecordFilter</td>
<td>An interface for classes which filters LogRecords.</td>
</tr>
</tbody>
</table>

---

Copyright 2000-2005 Apache Software Foundation.
public interface LogRecordFilter

An interface for classes which filters LogRecords. Implementations represent a rule or condition which LogRecords may pass or fail.

Author:
Richard Wan

See Also:
LogRecord

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean passes</td>
<td>public boolean passes(LogRecord record)</td>
</tr>
</tbody>
</table>

Method Detail

passes

public boolean passes(LogRecord record)

Returns:
true if the specified LogRecord satisfies whatever condition implementing class tests for.
Copyright 2000-2005 Apache Software Foundation.
public class AppenderFinalizer extends Object

AppenderFinalizer has a single method that will finalize resources associated with a LogBrokerMonitor in the event that the LF5Appender class is destroyed, and the class loader is garbage collected.

Author:
    Brent Sprecher

Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>defaultMonitor</td>
</tr>
<tr>
<td>org.apache.log4j.lf5.viewer.LogBrokerMonitor</td>
<td>protected</td>
</tr>
</tbody>
</table>

Constructor Summary

AppenderFinalizer(org.apache.log4j.lf5.viewer.LogBrokerMonitor defaultMonitor)

Method Summary

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>finalize()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait
Field Detail

_privateMonitor

protected org.apache.log4j.lf5.viewer.LogBrokerMonitor _defaultMonitor

Constructor Detail

AppenderFinalizer

public AppenderFinalizer (org.apache.log4j.lf5.viewer.LogBrokerMonitor defaultMonitor)

Method Detail

finalize

protected void finalize ()
throws Throwable

Overrides: finalize in class Object

Throws: Throwable -

Copyright 2000-2005 Apache Software Foundation.
Class DefaultLF5Configurator

All Implemented Interfaces:
  Configurator

public class DefaultLF5Configurator extends Object implements Configurator

The DefaultLF5Configurator provides a default configuration for the LF5Appender. Note: The preferred method for configuring a LF5Appender is to use the LF5Manager class. This class ensures that configuration does not occur multiple times, and improves system performance. Reconfiguring the monitor multiple times can result in unexpected behavior.

Author:
  Brent Sprecher

Fields inherited from interface org.apache.log4j.spi.Configurator
INHERITED, NULL

Method Summary

static void configure()
This method configures the LF5Appender using a default configuration file.

void doConfigure(URL configURL, LoggerRepository repository)
This is a dummy method that will throw an IllegalStateException if used.
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

configure

public static void configure()
        throws IOException

This method configures the LF5Appender using a default configuration file. The default configuration file is defaultconfig.properties.

Throws:
IOException

doConfigure

public void doConfigure(URL configURL, LoggerRepository repository)

This is a dummy method that will throw an IllegalStateException if used.

Specified by:
doConfigure in interface Configurator

Following copied from interface org.apache.log4j.spi.Configurator

Parameters:
url - The URL to parse
repository - The hierarchy to operation upon.
Class LF5Appender

public class LF5Appender extends AppenderSkeleton

LF5Appender logs events to a swing based logging console. The swing console supports turning categories on and off, multiple detail level views, as well as full text searching and many other capabilities.

Author:
Brent Sprecher

Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>static org.apache.log4j.lf5.viewer.LogBrokerMonitor _defaultLogMonitor</td>
</tr>
<tr>
<td>protected</td>
<td>static AppenderFinalizer finalizer</td>
</tr>
<tr>
<td>protected</td>
<td>org.apache.log4j.lf5.viewer.LogBrokerMonitor _logMonitor</td>
</tr>
</tbody>
</table>

Fields inherited from class org.apache.log4j.AppenderSkeleton

closed, errorHandler, headFilter, layout, name, tailFilter, threshold

Constructor Summary
**LF5Appender()**  
Constructs a LF5Appender using the default instance of the LogBrokerMonitor.

**LF5Appender**(org.apache.log4j.lf5.viewer.LogBrokerMonitor monitor)  
Constructs a LF5Appender using an instance of a LogBrokerMonitor supplied by the user.

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| void | **append**(LoggingEvent event)  
Appends a LoggingEvent |
| void | **close**()  
This method is an empty implementation of the close() method inherited from the org.apache.log4j.Appender interface. |
| boolean | **equals**(LF5Appender compareTo)  
The equals method compares two LF5Appenders and determines whether they are equal. |
| protected static org.apache.log4j.lf5.viewer.LogBrokerMonitor | **getLogBrokerMonitor**() |
| protected static int | **getScreenHeight**() |
| protected static int | **getScreenWidth**() |
| static void | **main**(String[] args) |
| boolean | **requiresLayout**()  
Returns a value that indicates if this appender requires a Layout. |
### Methods inherited from class org.apache.log4j.AppenderSkeleton

- `activateOptions`
- `addFilter`
- `clearFilters`
- `doAppend`
- `finalize`
- `getErrorHandler`
- `getFilter`
- `getFirstFilter`
- `getLayout`
- `getName`
- `getThreshold`
- `isAsSevereAsThreshold`
- `setErrorHandler`
- `setLayout`
- `setName`
- `setThreshold`

### Methods inherited from class java.lang.Object

- `clone`
- `equals`
- `getClass`
- `hashCode`
- `notify`
- `notifyAll`
- `toString`
- `wait`
- `wait`
- `wait`

### Field Detail

#### `_logMonitor`

protected `org.apache.log4j.lf5.viewer.LogBrokerMonitor` _logMonitor

#### `_defaultLogMonitor`

protected static `org.apache.log4j.lf5.viewer.LogBrokerMonitor` _defaultLogMonitor

#### `_finalizer`

protected static `AppenderFinalizer` _finalizer

### Constructor Detail
LF5Appender

public LF5Appender()

Constructs a LF5Appender using the default instance of the LogBrokerMonitor. This constructor should always be preferred over the LF5Appender(LogBrokerMonitor monitor) constructor, unless you need to spawn additional log monitoring windows.

LF5Appender

public LF5Appender(org.apache.log4j.lf5.viewer.LogBrokerMonitor monitor)

Constructs a LF5Appender using an instance of a LogBrokerMonitor supplied by the user. This constructor should only be used when you need to spawn additional log monitoring windows.

Parameters:
  monitor - An instance of a LogBrokerMonitor created by the user.

Method Detail

append

public void append(LoggingEvent event)

Appends a LoggingEvent record to the LF5Appender.

Overrides:
  append in class AppenderSkeleton

Parameters:
  event - The LoggingEvent to be appended.

close

public void close()

This method is an empty implementation of the close() method
inherited from the org.apache.log4j.Appender interface.

**requiresLayout**

```java
public boolean requiresLayout()
```

Returns a value that indicates whether this appender requires a Layout. This method always returns false. No layout is required for the LF5Appender.

**setCallSystemExitOnClose**

```java
public void setCallSystemExitOnClose(boolean callSystemExitOnClose)
```

This method is used to set the property that controls whether the LogBrokerMonitor is hidden or closed when a user exits the monitor. By default, the LogBrokerMonitor will hide itself when the log window is exited, and the swing thread will continue to run in the background. If this property is set to true, the LogBrokerMonitor will call System.exit(0) and will shut down swing thread and the virtual machine.

**Parameters:**
- `callSystemExitOnClose` - A boolean value indicating whether to call System.exit(0) when closing the log window.

**equals**

```java
public boolean equals(LF5Appender compareTo)
```

The equals method compares two LF5Appenders and determines whether they are equal. Two Appenders will be considered equal if, and only if, they both contain references to the same LogBrokerMonitor.

**Parameters:**
- `compareTo` - A boolean value indicating whether the two LF5Appenders are equal.
getLogBrokerMonitor

public org.apache.log4j.lf5.viewer.LogBrokerMonitor getLogBrokerMonitor

main

public static void main(String[] args)

setMaxNumberOfRecords

public void setMaxNumberOfRecords(int maxNumberOfRecords)

getDefaultInstance

protected static org.apache.log4j.lf5.viewer.LogBrokerMonitor getDefaultInstance

Returns:
The default instance of the LogBrokerMonitor.

getScreenWidth

protected static int getScreenWidth()

Returns:
the screen width from Toolkit.getScreenSize() if possible, otherwise returns 800
See Also:
Toolkit

gSCREENHEIGHT

protected static int getScreenWidth()
Returns:
the screen height from Toolkit.getScreenSize() if possible, otherwise returns 600

See Also:
Toolkit

genericDefaultMonitorWidth
protected static int getDefaultMonitorWidth()
genericDefaultMonitorHeight
protected static int getDefaultMonitorHeight()
public class Log4JLogRecord extends LogRecord

A Log4JLogRecord encapsulates the details of your log4j LoggingEvent in a format usable by the LogBrokerMonitor.

Author:
Brent Sprecher

See Also:
Serialized Form

Fields inherited from class org.apache.log4j.lf5.LogRecord
_category, _level, _location, _message, _millis, _ndc, _seqCount, _sequenceNumber, _thread, _thrown, _thrownStackTrace

Constructor Summary

Log4JLogRecord() Constructs an instance of a Log4JLogRecord.

Method Summary

boolean isSevereLevel() Determines which Priority levels will be displayed in colored font when the LogMonitorAppender renders this log
### Constructor Detail

**Log4JLogRecord**

```java
public Log4JLogRecord()
```

Constructs an instance of a Log4JLogRecord.

### Method Detail

**isSevereLevel**

```java
public boolean isSevereLevel()
```

Determines which Priority levels will be displayed in colored font when the LogMonitorAppender renders this log message. By default, messages will be colored red if they are of Priority ERROR or FATAL.

Overrides:
isSevereLevel in class LogRecord

Returns:
true if the log level is ERROR or FATAL.

setThrownStackTrace

public void setThrownStackTrace(ThrowableInformation throwableInfo)

Set stack trace information associated with this Log4JLogRecord. When this method is called, the stack trace in a String-based format is made available via the getThrownStackTrace() method.

Parameters:
throwableInfo - An org.apache.log4j.spi.ThrowableInformation to associate with this Log4JLogRecord.

See Also:
LogRecord.getThrownStackTrace()
Class LogLevel

java.lang.Object

---

org.apache.log4j.lf5(LogLevel)

All Implemented Interfaces:
  Serializable

public class LogLevel
extends Object
implements Serializable

The LogLevel class defines a set of standard logging levels. The logging Level objects are ordered and are specified by ordered integers. Enabling logging at a given level also enables logging at all higher levels.

Author:
  Michael J. Sikorsky, Robert Shaw, Brent Sprecher, Richard Hurst, Brad Marlborough

See Also:
  Serialized Form

Field Summary

<p>| protected String | _label |
| protected int    | _precedence |
| static LogLevel  | CONFIG |
| static LogLevel  | DEBUG |
| static LogLevel  | ERROR |</p>
<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LogLevel(String label, int precedence)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean <code>encompasses(LogLevel level)</code> Returns true if the level supplied is encompassed by this level.</td>
</tr>
<tr>
<td>boolean <code>equals(Object o)</code></td>
</tr>
<tr>
<td>static <code>List getAllDefaultLevels()</code></td>
</tr>
<tr>
<td>static <code>List getJdk14Levels()</code></td>
</tr>
<tr>
<td>String <code>getLabel()</code> Return the Label of the LogLevel.</td>
</tr>
<tr>
<td>static List</td>
</tr>
<tr>
<td>static Map</td>
</tr>
<tr>
<td>protected int</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>static void</td>
</tr>
<tr>
<td>static LogLevel</td>
</tr>
<tr>
<td>static void</td>
</tr>
<tr>
<td>static void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>static LogLevel</td>
</tr>
</tbody>
</table>

Convert a log level label into a LogLevel object.

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

FATAL

public static final LogLevel FATAL
ERROR

public static final LogLevel ERROR

WARN

public static final LogLevel WARN

INFO

public static final LogLevel INFO

DEBUG

public static final LogLevel DEBUG

SEVERE

public static final LogLevel SEVERE

WARNING

public static final LogLevel WARNING

CONFIG

public static final LogLevel CONFIG

FINE

public static final LogLevel FINE
FINER

public static final LogLevel FINER

FINEST

public static final LogLevel FINEST

_label

protected String _label

_precedence

protected int _precedence

(Constructor Detail)

LogLevel

public LogLevel(String label, int precedence)

(Method Detail)

getLabel

public String getLabel()

    Return the Label of the LogLevel.

encompasses
public boolean encompasses(LogLevel level)

Returns true if the level supplied is encompassed by this level. For example, LogLevel.SEVERE encompasses no other LogLevels and LogLevel.FINE encompasses all other LogLevels. By definition, a LogLevel encompasses itself.

valueOf

public static LogLevel.valueOf(String level)
throws LogLevelFormatException

Convert a log level label into a LogLevel object.

Parameters:
level - The label of a level to be converted into a LogLevel.

Returns:
LogLevel The LogLevel with a label equal to level.

Throws:
LogLevelFormatException - Is thrown when the level can not be converted into a LogLevel.

register

public static LogLevel register(LogLevel logLevel)

Registers a used defined LogLevel.

Parameters:
logLevel - The log level to be registered. Cannot be a default LogLevel

Returns:
LogLevel The replaced log level.

register

public static void register(LogLevel[] logLevels)
register

public static void register(List logLevels)

equals

public boolean equals(Object o)

Overrides:
equals in class Object

hashCode

public int hashCode()

Overrides:
hashCode in class Object

toString

public String toString()

Overrides:
toString in class Object

setLogLevelColorMap

public void setLogLevelColorMap(LogLevel level, Color color)

resetLogLevelColorMap

public static void resetLogLevelColorMap()
getLog4JLevels

public static List getLog4JLevels()

Returns:
A List of LogLevel objects that map to log4j Priority objects.

goingDk14Levels

public static List getJdk14Levels()

getAllDefaultLevels

public static List getAllDefaultLevels()

getLogLevelColorMap

public static Map getLogLevelColorMap()

goingPrecedence

protected int getPrecedence()
public abstract class LogRecord extends Object implements Serializable

LogRecord. A LogRecord encapsulates the details of your desired log request.

Author: Michael J. Sikorsky, Robert Shaw

See Also: Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>protected String</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected LogLevel</td>
<td>level</td>
</tr>
<tr>
<td>protected String</td>
<td>location</td>
</tr>
<tr>
<td>protected String</td>
<td>message</td>
</tr>
<tr>
<td>protected String</td>
<td>millis</td>
</tr>
<tr>
<td>protected String</td>
<td>ndc</td>
</tr>
<tr>
<td>------------------</td>
<td>-----</td>
</tr>
<tr>
<td>protected static long</td>
<td>seqCount</td>
</tr>
<tr>
<td>protected long</td>
<td>sequenceNumber</td>
</tr>
<tr>
<td>protected long</td>
<td>_sequenceNumber</td>
</tr>
<tr>
<td>protected String</td>
<td>thread</td>
</tr>
<tr>
<td>protected Throwable</td>
<td>_thrown</td>
</tr>
<tr>
<td>protected String</td>
<td>thrownStackTrace</td>
</tr>
</tbody>
</table>

### Constructor Summary

**LogRecord()**

### Method Summary

- **String** getCategory()  
  Get the category associated with this LogRecord.

- **LogLevel** getLevel()  
  Get the level of this LogRecord.

- **String** getLocation()  
  Get the location in code where this LogRecord originated.

- **String** getMessage()  
  Get the message associated with this LogRecord.

- **long** getMillis()  
  Get the event time of this record in milliseconds from 1970.

- **String** getNDC()  
  Get the NDC (nested diagnostic context) for this record.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected long getNextId()</td>
<td>Get the sequence number associated with this LogRecord.</td>
</tr>
<tr>
<td>static long getSequenceNumber()</td>
<td>Get the sequence number associated with this LogRecord.</td>
</tr>
<tr>
<td>String getThreadDescription()</td>
<td>Get the thread description associated with this LogRecord.</td>
</tr>
<tr>
<td>Throwable getThrown()</td>
<td>Get the Throwable associated with this LogRecord.</td>
</tr>
<tr>
<td>String getThrownStackTrace()</td>
<td>Get the stack trace in a String-based format for the associated Throwable of this LogRecord.</td>
</tr>
<tr>
<td>boolean hasThrown()</td>
<td></td>
</tr>
<tr>
<td>boolean isFatal()</td>
<td></td>
</tr>
<tr>
<td>abstract boolean isSevereLevel()</td>
<td>Abstract method.</td>
</tr>
<tr>
<td>static void resetSequenceNumber()</td>
<td>Resets that sequence number to 0.</td>
</tr>
<tr>
<td>void setCategory(String category)</td>
<td>Set the category associated with this LogRecord.</td>
</tr>
<tr>
<td>void setLevel(LogLevel level)</td>
<td>Set the level of this LogRecord.</td>
</tr>
<tr>
<td>void setLocation(String location)</td>
<td>Set the location in code where this LogRecord originated.</td>
</tr>
<tr>
<td>void setMessage(String message)</td>
<td>Set the message associated with this LogRecord.</td>
</tr>
<tr>
<td>void setMillis(long millis)</td>
<td>Set the event time of this record.</td>
</tr>
<tr>
<td>void setNDC(String ndc)</td>
<td>Set the NDC (nested diagnostic context) for this record.</td>
</tr>
<tr>
<td>void setSequenceNumber(long number)</td>
<td></td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>Set the sequence number associated with this LogRecord.</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| **void** | **setThreadDescription** *(String threadDescription)*  
Set the thread description associated with this LogRecord. |
| **void** | **setThrown** *(Throwable thrown)*  
Set the Throwable associated with this LogRecord. |
| **void** | **setThrownStackTrace** *(String trace)*  
Set the ThrownStackTrace for the log record. |
| **String** | **toString()**  
Return a String representation of this LogRecord. |

**Methods inherited from class java.lang.**Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

---

**Field Detail**

**_seqCount**

protected static long _seqCount

**_level**

protected **LogLevel** _level

**_message**

protected **String** _message

**_sequenceNumber**
protected long _sequenceNumber

_millis

protected long _millis

_category

protected String _category

_thread

protected String _thread

_thrownStackTrace

protected String _thrownStackTrace

Thrown

protected Throwable _thrown

_ndc

protected String _ndc

_location

protected String _location

Constructor Detail
LogRecord

public LogRecord()

---

**Method Detail**

getLevel

public LogLevel getLevel()

Get the level of this LogRecord.

**Returns:**
The LogLevel of this record.

**See Also:**
setLevel(LogLevel), LogLevel

---

setLevel

public void setLevel(LogLevel level)

Set the level of this LogRecord.

**Parameters:**
level - The LogLevel for this record.

**See Also:**
getLevel(), LogLevel

---

isSevereLevel

public abstract boolean isSevereLevel()

Abstract method. Must be overridden to indicate what log level to show in red.

---

hasThrown
public boolean hasThrown()

Returns:
true if getThrown().toString() is a non-empty string.

isFatal

public boolean isFatal()

Returns:
true if isSevereLevel() or hasThrown() returns true.

category

public String getCategory()

Get the category associated with this LogRecord. For a more detailed description of what a category is see setCategory().

Returns:
The category of this record.

See Also:
setCategory(String)

setCategory

public void setCategory(String category)

Set the category associated with this LogRecord. A category represents a hierarchical dot (".") separated namespace for messages. The definition of a category is application specific, but a common convention is as follows:

When logging messages for a particular class you can use its class name: com.thoughtworks.framework.servlet.ServletServiceBroker.

Furthermore, to log a message for a particular method in a class add the method name:
com.thoughtworks.framework.servlet.ServletServiceBroker.init().

**Parameters:**
- category - The category for this record.

**See Also:**
- getCategory()

---

**getMessage**

```java
public String getMessage()
```

Get the message asssociated with this LogRecord.

**Returns:**
The message of this record.

**See Also:**
- setMessage(String)

---

**setMessage**

```java
public void setMessage(String message)
```

Set the message associated with this LogRecord.

**Parameters:**
- message - The message for this record.

**See Also:**
- getMessage()

---

**getSequenceNumber**

```java
public long getSequenceNumber()
```

Get the sequence number associated with this LogRecord.

Sequence numbers are generally assigned when a LogRecord is constructed. Sequence numbers start at 0 and increase with each newly constructed LogRecord.

**Returns:**
The sequence number of this record.
See Also:
```
setSequenceNumber(long)
```

### setSequenceNumber

```java
public void setSequenceNumber(long number)
```

Set the sequence number associated with this LogRecord. A sequence number will automatically be assigned to every newly constructed LogRecord, however, this method can override the value.

**Parameters:**
- `number` - The sequence number.

**See Also:**
```
getSequenceNumber()
```

### getMillis

```java
public long getMillis()
```

Get the event time of this record in milliseconds from 1970. When a LogRecord is constructed the event time is set but may be overridden by calling `setMillis()`;

**Returns:**
- The event time of this record in milliseconds from 1970.

**See Also:**
```
setMillis(long)
```

### setMillis

```java
public void setMillis(long millis)
```

Set the event time of this record. When a LogRecord is constructed the event time is set but may be overridden by calling this method.

**Parameters:**
- `millis` - The time in milliseconds from 1970.

**See Also:**
getThreadDescription

public String getThreadDescription()

Get the thread description associated with this LogRecord. When a LogRecord is constructed, the thread description is set by calling: Thread.currentThread().toString(). You may supply a thread description of your own by calling the setThreadDescription(String) method.

Returns:
The thread description of this record.

See Also:
setThreadDescription(String)

setThreadDescription

public void setThreadDescription(String threadDescription)

Set the thread description associated with this LogRecord. When a LogRecord is constructed, the thread description is set by calling: Thread.currentThread().toString(). You may supply a thread description of your own by calling this method.

Parameters:
threadDescription - The description of the thread for this record.

See Also:
getThreadDescription()

getThrownStackTrace

public String getThrownStackTrace()

Get the stack trace in a String-based format for the associated Throwable of this LogRecord. The stack trace in a String-based format is set when the setThrown(Throwable) method is called.
Why do we need this method considering that we have the getThrown() and setThrown() methods? A Throwable object may not be serializable, however, a String representation of it is. Users of LogRecords should generally call this method over getThrown() for the reasons of serialization.

**Returns:**
The Stack Trace for the associated Throwable of this LogRecord.

**See Also:**
setThrown(Throwables), getThrown()

---

**setThrownStackTrace**

```java
public void setThrownStackTrace(String trace)
```

Set the ThrowableStackTrace for the log record.

**Parameters:**
- trace - A String to associate with this LogRecord

**See Also:**
getThrownStackTrace()

---

**getThrown**

```java
public Throwable getThrown()
```

Get the Throwable associated with this LogRecord.

**Returns:**
The LogLevel of this record.

**See Also:**
setThrown(Throwables), getThrownStackTrace()

---

**setThrown**

```java
public void setThrown(Throwables thrown)
```

Set the Throwable associated with this LogRecord. When this
method is called, the stack trace in a String-based format is made available via the getThrownStackTrace() method.

Parameters:

thrown - A Throwable to associate with this LogRecord.

See Also:

getThrown(), getThrownStackTrace()

---

toString

public String toString()

Return a String representation of this LogRecord.

Overrides:

toString in class Object

---

getNDC

public String getNDC()

Get the NDC (nested diagnostic context) for this record.

Returns:

The string representing the NDC.

---

setNDC

public void setNDC(String ndc)

Set the NDC (nested diagnostic context) for this record.

Parameters:

ndc - A string representing the NDC.

---

getLocation

public String getLocation()
Get the location in code where this LogRecord originated.

**Returns:**
The string containing the location information.

---

**setLocation**

```java
public void setLocation(String location)
```

Set the location in code where this LogRecord originated.

**Parameters:**
- location - A string containing location information.

---

**resetSequenceNumber**

```java
public static void resetSequenceNumber()
```

Resets that sequence number to 0.

---

**getNextId**

```java
protected static long getNextId()
```

---

Copyright 2000-2005 Apache Software Foundation.
org.apache.log4j.lf5  Class PassingLogRecordFilter

java.lang.Object
   +--org.apache.log4j.lf5.PassingLogRecordFilter

All Implemented Interfaces:
   LogRecordFilter

public class PassingLogRecordFilter
extends Object
implements LogRecordFilter

An implementation of LogRecordFilter which always returns true.

Author:
    Richard Wan

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean passes</td>
<td>(LogRecord record)</td>
</tr>
<tr>
<td>void reset</td>
<td>Does nothing.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Constructor Detail

PassingLogRecordFilter

public PassingLogRecordFilter()

Method Detail

passes

public boolean passes(LogRecord record)

Specified by:
passes in interface LogRecordFilter

Returns:
true;

reset

public void reset()

Does nothing.
public class StartLogFactor5
extends Object

Starts an instance of the LogFactor5 console for off-line viewing.

Author:
Brad Marlborough, Richard Hurst

Constructor Summary

| StartLogFactor5() |

Method Summary

| static void main(String[] args) |
| Main - starts an instance of the LogFactor5 console and configures the console settings. |

Methods inherited from class java.lang.Object
clon e, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

StartLogFactor5
public StartLogFactor5()

**Method Detail**

**main**

```java
public static final void main(String[] args)
```

Main - starts an instance of the LogFactor5 console and configures the console settings.

---

Copyright 2000-2005 Apache Software Foundation.
public class LogLevelFormatException extends Exception

Thrown to indicate that the client has attempted to convert a string to one of the LogLevel types, but the string does not have the appropriate format.

Author:
   Michael J. Sikorsky, Robert Shaw

See Also:
   Serialized Form

Constructor Summary

| LogLevelFormatException(String message) |

Methods inherited from class java.lang.Throwable

| fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString |

Methods inherited from class java.lang.Object

| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |
public LogLevelFormatException(String message)
Package org.apache.log4j.net

Package for remote logging.

See: Description

<table>
<thead>
<tr>
<th>Class Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JMSAppender</strong></td>
<td>A simple appender that publishes events to a JMS Topic.</td>
</tr>
<tr>
<td><strong>JMSSink</strong></td>
<td>A simple application that consumes logging events sent by a JMSAppender.</td>
</tr>
<tr>
<td><strong>SimpleSocketServer</strong></td>
<td>A simple SocketNode based server.</td>
</tr>
<tr>
<td><strong>SMTPAppender</strong></td>
<td>Send an e-mail when a specific logging event occurs, typically on errors or fatal errors.</td>
</tr>
<tr>
<td><strong>SocketAppender</strong></td>
<td>Sends LoggingEvent objects to a remote a log server, usually a SocketNode.</td>
</tr>
<tr>
<td><strong>SocketHubAppender</strong></td>
<td>Sends LoggingEvent objects to a set of remote log servers, usually a SocketNodes.</td>
</tr>
<tr>
<td><strong>SocketNode</strong></td>
<td>Read LoggingEvent objects sent from a remote client using Sockets (TCP).</td>
</tr>
<tr>
<td><strong>SocketServer</strong></td>
<td>A SocketNode based server that uses a different hierarchy for each client.</td>
</tr>
<tr>
<td><strong>SyslogAppender</strong></td>
<td>Use SyslogAppender to send log messages to a remote syslog daemon.</td>
</tr>
<tr>
<td><strong>TelnetAppender</strong></td>
<td>The TelnetAppender is a log4j appender that specializes in writing to a read-only socket.</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.net Description

Package for remote logging.

Last modified: Tue Mar 21 20:28:14 MET 2000

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.net

Package Hierarchies:

All Packages

----------------------------------------
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.net.**JMSAppender**
    - class org.apache.log4j.net.**SMTPAppender**
    - class org.apache.log4j.net.**SocketAppender**
    - class org.apache.log4j.net.**SocketHubAppender**
    - class org.apache.log4j.net.**SyslogAppender**
    - class org.apache.log4j.net.**TelnetAppender**
  - class org.apache.log4j.net.**JMSSink** (implements javax.jms.**MessageListener**)
  - class org.apache.log4j.net.**SimpleSocketServer**
  - class org.apache.log4j.net.**SocketNode** (implements java.lang.**Runnable**)
  - class org.apache.log4j.net.**SocketServer**
  - class java.lang.**Thread** (implements java.lang.**Runnable**)
    - class org.apache.log4j.net.**TelnetAppender.SocketHandler**

Copyright 2000-2005 Apache Software Foundation.
Uses of Package org.apache.log4j.net

<table>
<thead>
<tr>
<th>Packages that use org.apache.log4j.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.net</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j.net used by org.apache.log4j.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>TelnetAppender</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
public class JMSAppender
extends AppenderSkeleton

A simple appender that publishes events to a JMS Topic. The events are serialized and transmitted as JMS message type ObjectMessage.

JMS topics and topic connection factories are administered objects that are retrieved using JNDI messaging which in turn requires the retrieval of a JNDI Context.

There are two common methods for retrieving a JNDI Context. If a file resource named jndi.properties is available to the JNDI API, it will use the information found therein to retrieve an initial JNDI context. To obtain an initial context, your code will simply call:

InitialContext jndiContext = new InitialContext();

Calling the no-argument InitialContext() method will also work from within Enterprise Java Beans (EJBs) because it is part of the EJB contract for application servers to provide each bean an environment naming context (ENC).

In the second approach, several predetermined properties are set and these properties are passed to the InitialContext constructor to connect to the naming service provider. For example, to connect to JBoss naming service one would write:

Properties env = new Properties();
env.put(Context.INITIAL_CONTEXT_FACTORY, "org.jnp.interfaces.NamingContextFactory");
env.put(Context.PROVIDER_URL, "jnp://hostname:1099");
env.put(Context.URL_PKG_PREFIXES, "org.jboss.naming:org.jnp.interfaces");
InitialContext jndiContext = new InitialContext(env);

where **hostname** is the host where the JBoss application server is running.

To connect to the the naming service of Weblogic application server one would write:

Properties env = new Properties();
env.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.WLInitialContextFactory");
env.put(Context.PROVIDER_URL, "t3://localhost:7001");
InitialContext jndiContext = new InitialContext(env);

Other JMS providers will obviously require different values. The initial JNDI context can be obtained by calling the no-argument InitialContext() method in EJBs. Only clients running in a separate JVM need to be concerned about the **jndi.properties** file and calling InitialContext.InitialContext() or alternatively correctly setting the different properties before calling InitialContext.InitialContext(java.util.Hashtable) method.

**Author:**

Ceki Gülcü

---

**Fields inherited from class org.apache.log4j.AppenderSkeleton**
closed, errorHandler, headFilter, layout, name, tailFilter, threshold

**Constructor Summary**

**JMSAppender()**

**Method Summary**

void **activateOptions()**
Options are activated and become effective only after calling this method.

void **append**(LoggingEvent event)

This method called by `AppenderSkeleton.doAppend` method to do most of the real appending work.

protected boolean **checkEntryConditions**()

void **close**()

Close this JMSAppender.

String **getInitialContextFactoryName**()

Returns the value of the `InitialContextFactoryName` option.

boolean **getLocationInfo**()

Returns value of the `LocationInfo` property which determines whether location (stack) info is sent to the remote subscriber.

String **getPassword**()

String **getProviderURL**()

String **getSecurityCredentials**()

String **getSecurityPrincipalName**()

String **getTopicBindingName**()

Returns the value of the `TopicBindingName` option.

protected TopicConnection **getTopicConnection**()

Returns the TopicConnection used for this appender.

String **getTopicConnectionFactoryBindingName**()

Returns the value of the `TopicConnectionFactoryBindingName` option.

protected TopicPublisher **getTopicPublisher**()

Returns the TopicPublisher used for this appender.

protected TopicSession **getTopicSession**()

Returns the TopicSession used for this appender.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getUsername()</td>
<td></td>
</tr>
<tr>
<td>protected Object lookup(Context ctx, String name)</td>
<td></td>
</tr>
<tr>
<td>boolean requiresLayout()</td>
<td>The JMSAppender sends serialized events and consequently does not require a layout.</td>
</tr>
<tr>
<td>void setInitialContextFactoryName(String initialContextFactoryName)</td>
<td>Setting the InitialContextFactoryName method will cause this JMSAppender instance to use the InitialContext.InitialContext(Hashtable) method instead of the no-argument constructor.</td>
</tr>
<tr>
<td>void setLocationInfo(boolean locationInfo)</td>
<td>If true, the information sent to the remote subscriber will include caller's location information.</td>
</tr>
<tr>
<td>void setPassword(String password)</td>
<td>The password to use when creating a topic session.</td>
</tr>
<tr>
<td>void setProviderURL(String providerURL)</td>
<td></td>
</tr>
<tr>
<td>void setSecurityCredentials(String securityCredentials)</td>
<td></td>
</tr>
<tr>
<td>void setSecurityPrincipalName(String securityPrincipalName)</td>
<td></td>
</tr>
<tr>
<td>void setTopicBindingName(String topicBindingName)</td>
<td>The TopicBindingName option takes a string value.</td>
</tr>
<tr>
<td>void setTopicConnectionFactoryBindingName(String tcfBindingName)</td>
<td>The TopicConnectionFactoryBindingName option takes a string value.</td>
</tr>
<tr>
<td>void setURLPkgPrefixes(String urlPkgPrefixes)</td>
<td></td>
</tr>
<tr>
<td>void setUserName(String userName)</td>
<td>The user name to use when creating a topic session.</td>
</tr>
</tbody>
</table>

Methods inherited from class org.apache.log4j.AppenderSkeleton:
addFilter, clearFilters, doAppend, finalize, getErrorHandler
Constructor Detail

JMSAppender

public JMSAppender()

Method Detail

setTopicConnectionFactoryBindingName

public void setTopicConnectionFactoryBindingName(String tcfBindingName)

The TopicConnectionFactoryBindingName option takes a string value. Its value will be used to lookup the appropriate TopicConnectionFactory from the JNDI context.

getTopicConnectionFactoryBindingName

public String getTopicConnectionFactoryBindingName()

Returns the value of the TopicConnectionFactoryBindingName option.

setTopicBindingName

public void setTopicBindingName(String topicBindingName)
The **TopicBindingName** option takes a string value. Its value will be used to lookup the appropriate **Topic** from the JNDI context.

---

### getTopicBindingName

```java
public String getTopicBindingName()
```

Returns the value of the **TopicBindingName** option.

---

### getLocationInfo

```java
public boolean getLocationInfo()
```

Returns value of the **LocationInfo** property which determines whether location (stack) info is sent to the remote subscriber.

---

### activateOptions

```java
public void activateOptions()
```

Options are activated and become effective only after calling this method.

**Overrides:**

`activateOptions` in class `AppenderSkeleton`

---

### lookup

```java
protected Object lookup(Context ctx,
                         String name)
          throws NamingException
```

---

### checkEntryConditions

```java
protected boolean checkEntryConditions()
```
close

public void close()

Close this JMSAppender. Closing releases all resources used by the appender. A closed appender cannot be re-opened.

append

public void append(LoggingEvent event)

This method called by AppenderSkeleton.doAppend(org.apache.log4j.spi.LoggingEvent) method to do most of the real appending work.

Overrides: append in class AppenderSkeleton

getInitialContextFactoryName

public String getInitialContextFactoryName()

Returns the value of the InitialContextFactoryName option. See setInitialContextFactoryName(java.lang.String) for more details on the meaning of this option.

setInitialContextFactoryName

public void setInitialContextFactoryName(String initialContextFactorName)

Setting the InitialContextFactoryName method will cause this JMSAppender instance to use the InitialContext.InitialContext(Hashtable) method instead of the no-argument constructor. If you set this option, you should also at least set the ProviderURL option.

See also setProviderURL(String).
getProviderURL
public String getProviderURL()

setProviderURL
public void setProviderURL(String providerURL)

setURLPkgPrefixes
public void setURLPkgPrefixes(String urlPkgPrefixes)

getSecurityCredentials
public String getSecurityCredentials()

setSecurityCredentials
public void setSecurityCredentials(String securityCredentials)

getSecurityPrincipalName
public String getSecurityPrincipalName()

setSecurityPrincipalName
public void setSecurityPrincipalName(String securityPrincipalName)

getUserName
public String getUsername()
setUserNam
public void setUserName(String userName)

The user name to use when creating a topic session. If you set this option, you should also set the Password option. See setPassword(String).

getPassword
public String getPassword()

setPassword
public void setPassword(String password)

The passsword to use when creating a topic session.

setLocationInfo
public void setLocationInfo(boolean locationInfo)

If true, the information sent to the remote subscriber will include caller's location information. By default no location information is sent to the subscriber.

getTopicConnection
protected TopicConnection getTopicConnection()

Returns the TopicConnection used for this appender. Only valid after activateOptions() method has been invoked.
protected TopicSession getTopicSession()

Returns the TopicSession used for this appender. Only valid after activateOptions() method has been invoked.

getTopicPublisher

protected TopicPublisher getTopicPublisher()

Returns the TopicPublisher used for this appender. Only valid after activateOptions() method has been invoked.

requiresLayout

public boolean requiresLayout()

The JMSAppender sends serialized events and consequently does not require a layout.
org.apache.log4j.net  Class JMSSink

java.lang.Object  
  +-- org.apache.log4j.net.JMSSink

All Implemented Interfaces:
  MessageListener

public class JMSSink
extends Object
implements MessageListener

A simple application that consumes logging events sent by a JMSAppender.

Author:
  Ceki Gülcü

Constructor Summary

| JMSSink( String tcfBindingName, String topicBindingName, String username, String password) |

Method Summary

| protected static object lookup(Context ctx, String name) |
| static void main(String[] args) |
| void onMessage(Message message) |

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
Constructor Detail

JMSSink

public JMSSink(String tcfBindingName,
                String topicBindingName,
                String username,
                String password)

Method Detail

main

public static void main(String[] args)
    throws Exception

onMessage

public void onMessage(Message message)

    Specified by:
        onMessage in interface MessageListener

lookup

protected static Object lookup(Context ctx,
                                String name)
    throws NamingException
Copyright 2000-2005 Apache Software Foundation.
public class SimpleSocketServer
extends Object

A simple SocketNode based server.

Usage: java org.apache.log4j.net.SimpleSocketServer port configFile

where port is a part number where the server listens and
cfgFile is a configuration file fed to the PropertyConfigurator

Since:
0.8.4
Author:
Ceki Gülcü

Constructor Summary

SimpleSocketServer()

Method Summary

static void main(String[] argv)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Constructor Detail

SimpleSocketServer

public SimpleSocketServer()

Method Detail

main

public static void main(String[] argv)
<table>
<thead>
<tr>
<th>SUMMARY:</th>
<th>INNER</th>
<th>FIELD</th>
<th>CONSTR</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETAIL:</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| FRAMELESS | NO FRAMES |

<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
<th>Log4j 1.2.14</th>
</tr>
</thead>
</table>

PREV CLASS | NEXT CLASS | SUMMAY: INNER | FIELD | CONSTR | METHOD |
FRAMES | NO FRAMES | DETAIL: FIELD | CONSTR | METHOD |
Class SMTPAppender

java.lang.Object
|-- org.apache.log4j.AppenderSkeleton
|   |-- org.apache.log4j.net.SMTPAppender

All Implemented Interfaces:
   Appender, OptionHandler

public class SMTPAppender extends AppenderSkeleton

Send an e-mail when a specific logging event occurs, typically on errors or fatal errors.

The number of logging events delivered in this e-mail depend on the value of BufferSize option. The SMTPAppender keeps only the last BufferSize logging events in its cyclic buffer. This keeps memory requirements at a reasonable level while still delivering useful application context.

Since: 1.0
Author: Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>protected</th>
<th>CyclicBuffer</th>
<th>cb</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td>TriggeringEventEvaluator</td>
<td>evaluator</td>
</tr>
<tr>
<td>protected</td>
<td>Message</td>
<td>msg</td>
</tr>
</tbody>
</table>
Fields inherited from class org.apache.log4j.**AppenderSkeleton**
closed, errorHandler, headFilter, layout, name, tailFilter, threshold

### Constructor Summary

**SMTPAppender()**
The default constructor will instantiate the appender with a **TriggeringEventEvaluator** that will trigger on events with level ERROR or higher.

**SMTPAppender(TriggeringEventEvaluator evaluator)**
Use evaluator passed as parameter as the **TriggeringEventEvaluator** for this SMTPAppender.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>activateOptions()</code></td>
<td>Activate the specified options, such as the smtp host, the recipient, from, etc.</td>
</tr>
<tr>
<td><code>addressMessage(Message msg)</code></td>
<td>Address message.</td>
</tr>
<tr>
<td><code>append(LoggingEvent event)</code></td>
<td>Perform SMTPAppender specific appending actions, mainly adding the event to a cyclic buffer and checking if the event triggers an e-mail to be sent.</td>
</tr>
<tr>
<td><code>checkEntryConditions()</code></td>
<td>This method determines if there is a sense in attempting to append.</td>
</tr>
<tr>
<td><code>close()</code></td>
<td>Release any resources allocated within the appender such as file handles, network connections, etc.</td>
</tr>
<tr>
<td><code>createSession()</code></td>
<td>Create mail session.</td>
</tr>
<tr>
<td><code>getBcc()</code></td>
<td>Get the bcc recipient addresses.</td>
</tr>
</tbody>
</table>
int `getBufferSize()`
Returns value of the `BufferSize` option.

String `getCC()`
Get the cc recipient addresses.

String `getEvaluatorClass()`
Returns value of the `EvaluatorClass` option.

String `getFrom()`
Returns value of the `From` option.

boolean `getLocationInfo()`
Returns value of the `LocationInfo` option.

boolean `getSMTPDebug()`
Get SMTP debug.

String `getSMTPHost()`
Returns value of the `SMTPHost` option.

String `getSMTPPassword()`
Get SMTP password.

String `getSMTPUsername()`
Get SMTP user name.

String `getSubject()`
Returns value of the `Subject` option.

String `getTo()`
Returns value of the `To` option.

boolean `requiresLayout()`
The `SMTPAppender` requires a layout.

protected void `sendBuffer()`
Send the contents of the cyclic buffer as an e-mail message.

void `setBcc`(String addresses)
Set the bcc recipient addresses.

void `setBufferSize`(int bufferSize)
The `BufferSize` option takes a positive integer representing the maximum number of logging events to collect in a cyclic buffer.

void `setCC`(String addresses)
Set the cc recipient addresses.
### setEvaluatorClass(String value)
- The **EvaluatorClass** option takes a string value representing the name of the class implementing the **TriggeringEventEvaluator** interface.

### setFrom(String from)
- The **From** option takes a string value which should be an e-mail address of the sender.

### setLocationInfo(boolean locationInfo)
- The **LocationInfo** option takes a boolean value.

### setSMTPDebug(boolean debug)
- Setting the **SmtpDebug** option to true will cause the mail session to log its server interaction to stdout.

### setSMTPHost(String smtpHost)
- The **SMTPHost** option takes a string value which should be a the host name of the SMTP server that will send the e-mail message.

### setSMTPPassword(String password)
- The **SmtpPassword** option takes a string value which should be the password required to authenticate against the mail server.

### setSMTPUsername(String username)
- The **SmtpUsername** option takes a string value which should be the username required to authenticate against the mail server.

### setSubject(String subject)
- The **Subject** option takes a string value which should be a the subject of the e-mail message.

### setTo(String to)
- The **To** option takes a string value which should be a comma separated list of e-mail address of the recipients.

---

**Methods inherited from class org.apache.log4j.AppenderSkeleton**
- addFilter, clearFilters, doAppend, finalize, getErrorHandler, getFilter, getFirstFilter, getLayout, getName, getThreshold, isAsSevereAsThreshold, setErrorHandler, setLayout, setName, setThreshold
Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

cb
protected CyclicBuffer cb

msg
protected Message msg

evaluator
protected TriggeringEventEvaluator evaluator

Constructor Detail

SMTPAppender

public SMTPAppender()

The default constructor will instantiate the appender with a TriggeringEventEvaluator that will trigger on events with level ERROR or higher.

SMTPAppender

public SMTPAppender(TriggeringEventEvaluator evaluator)
Use `evaluator` passed as parameter as the `TriggeringEventEvaluator` for this SMTPAppender.

## Method Detail

### activateOptions

```java
public void activateOptions()
```

Activate the specified options, such as the smtp host, the recipient, from, etc.

**Overrides:**

activateOptions in class `AppenderSkeleton`

### addressMessage

```java
protected void addressMessage(Message msg)
```

Address message.

**Parameters:**

`msg` - message, may not be null.

**Throws:**

`MessagingException` - thrown if error addressing message.

### createSession

```java
protected Session createSession()
```

Create mail session.

**Returns:**

mail session, may not be null.

### append
public void append(LoggingEvent event)

    Perform SMTPAppender specific appending actions, mainly adding the event to a cyclic buffer and checking if the event triggers an e-mail to be sent.
    **Overrides:**
    append in class AppenderSkeleton

checkEntryConditions

protected boolean checkEntryConditions()

    This method determines if there is a sense in attempting to append.

    It checks whether there is a set output target and also if there is a set layout. If these checks fail, then the boolean value false is returned.

close

public void close()

    **Description copied from interface: Appender**
    Release any resources allocated within the appender such as file handles, network connections, etc.

    It is a programming error to append to a closed appender.

geto

public String getTo()

    Returns value of the To option.

requiresLayout

public boolean requiresLayout()
The SMTPAppender requires a layout.

---

**sendBuffer**

```java
protected void sendBuffer()
```

Send the contents of the cyclic buffer as an e-mail message.

---

**getEvaluatorClass**

```java
public String getEvaluatorClass()
```

Returns value of the EvaluatorClass option.

---

**getFrom**

```java
public String getFrom()
```

Returns value of the From option.

---

**getSubject**

```java
public String getSubject()
```

Returns value of the Subject option.

---

**setFrom**

```java
public void setFrom(String from)
```

The From option takes a string value which should be a e-mail address of the sender.
setSubject

public void setSubject(String subject)

    The Subject option takes a string value which should be a the subject of the e-mail message.

setBufferSize

public void setBufferSize(int bufferSize)

    The BufferSize option takes a positive integer representing the maximum number of logging events to collect in a cyclic buffer. When the BufferSize is reached, oldest events are deleted as new events are added to the buffer. By default the size of the cyclic buffer is 512 events.

setSMTPHost

public void setSMTPHost(String smtpHost)

    The SMTPHost option takes a string value which should be a the host name of the SMTP server that will send the e-mail message.

getSMTPHost

public String getSMTPHost()

    Returns value of the SMTPHost option.

setTo

public void setTo(String to)

    The To option takes a string value which should be a comma separated list of e-mail address of the recipients.
getBufferSize

public int getBufferSize()

    Returns value of the BufferSize option.

setEvaluatorClass

public void setEvaluatorClass(String value)

    The EvaluatorClass option takes a string value representing the name of the class implementing the TriggeringEventEvaluator interface. A corresponding object will be instantiated and assigned as the triggering event evaluator for the SMTPAppender.

setLocationInfo

public void setLocationInfo(boolean locationInfo)

    The LocationInfo option takes a boolean value. By default, it is set to false which means there will be no effort to extract the location information related to the event. As a result, the layout that formats the events as they are sent out in an e-mail is likely to place the wrong location information (if present in the format).

    Location information extraction is comparatively very slow and should be avoided unless performance is not a concern.

getLocationInfo

public boolean getLocationInfo()

    Returns value of the LocationInfo option.
**setCc**

```java
public void setCc(String addresses)
```

Set the cc recipient addresses.

**Parameters:**
addresses - recipient addresses as comma separated string, may be null.

---

**getCc**

```java
public String getCc()
```

Get the cc recipient addresses.

**Returns:**
recipient addresses as comma separated string, may be null.

---

**setBcc**

```java
public void setBcc(String addresses)
```

Set the bcc recipient addresses.

**Parameters:**
addresses - recipient addresses as comma separated string, may be null.

---

**getBcc**

```java
public String getBcc()
```

Get the bcc recipient addresses.

**Returns:**
recipient addresses as comma separated string, may be null.

---

**setSMTPPassword**
public void setSMTPPassword(String password)

    The SmtpPassword option takes a string value which should be the password required to authenticate against the mail server.
    Parameters:
        password - password, may be null.

setSMTPUsername

public void setSMTPUsername(String username)

    The SmtpUsername option takes a string value which should be the username required to authenticate against the mail server.
    Parameters:
        username - user name, may be null.

setSMTPDebug

public void setSMTPDebug(boolean debug)

    Setting the SmtpDebug option to true will cause the mail session to log its server interaction to stdout. This can be useful when debugging the appender but should not be used during production because username and password information is included in the output.
    Parameters:
        debug - debug flag.

getSMTPPassword

public String getSMTPPassword()

    Get SMTP password.
    Returns:
        SMTP password, may be null.
getSMTPUsername

public String getSMTPUsername()

Get SMTP user name.

Returns:
SMTP user name, may be null.

getSMTPDebug

public boolean getSMTPDebug()

Get SMTP debug.

Returns:
SMTP debug flag.
org.apache.log4j.net  **Class SocketAppender**

```
java.lang.Object
   +-- org.apache.log4j.AppenderSkeleton
      +-- org.apache.log4j.net.SocketAppender
```

**All Implemented Interfaces:**

Appender, OptionHandler

---

public class **SocketAppender**
extends AppenderSkeleton

Sends LoggingEvent objects to a remote a log server, usually a SocketNode.

The SocketAppender has the following properties:

- If sent to a SocketNode, remote logging is non-intrusive as far as the log event is concerned. In other words, the event will be logged with the same time stamp, NDC, location info as if it were logged locally by the client.

- SocketAppenders do not use a layout. They ship a serialized LoggingEvent object to the server side.

- Remote logging uses the TCP protocol. Consequently, if the server is reachable, then log events will eventually arrive at the server.

- If the remote server is down, the logging requests are simply dropped. However, if and when the server comes back up, then event transmission is resumed transparently. This transparent reconnectiton is performed by a connector thread which periodically attempts to connect to the server.

- Logging events are automatically buffered by the native TCP implementation. This means that if the link to server is slow but still
faster than the rate of (log) event production by the client, the client will not be affected by the slow network connection. However, if the network connection is slower then the rate of event production, then the client can only progress at the network rate. In particular, if the network link to the the server is down, the client will be blocked.

On the other hand, if the network link is up, but the server is down, the client will not be blocked when making log requests but the log events will be lost due to server unavailability.

- Even if a SocketAppender is no longer attached to any category, it will not be garbage collected in the presence of a connector thread. A connector thread exists only if the connection to the server is down. To avoid this garbage collection problem, you should close() the the SocketAppender explicitly. See also next item.

Long lived applications which create/destroy many SocketAppender instances should be aware of this garbage collection problem. Most other applications can safely ignore it.

- If the JVM hosting the SocketAppender exits before the SocketAppender is closed either explicitly or subsequent to garbage collection, then there might be untransmitted data in the pipe which might be lost. This is a common problem on Windows based systems.

To avoid lost data, it is usually sufficient to close() the SocketAppender either explicitly or by calling the LogManager.shutdown() method before exiting the application.

Since: 0.8.4
Author: Ceki Gülcü

<table>
<thead>
<tr>
<th>Fields inherited from class org.apache.log4j.AppenderSkeleton</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed, errorHandler, headFilter, layout, name, tailFilter, threshold</td>
</tr>
</tbody>
</table>
### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SocketAppender</strong>()</td>
<td></td>
</tr>
<tr>
<td><strong>SocketAppender(InetAddress address, int port)</strong></td>
<td>Connects to remote server at address and port.</td>
</tr>
<tr>
<td><strong>SocketAppender(String host, int port)</strong></td>
<td>Connects to remote server at host and port.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>activateOptions()</td>
<td>void</td>
<td>Connect to the specified <strong>RemoteHost</strong> and <strong>Port</strong>.</td>
</tr>
<tr>
<td>append(LoggingEvent event)</td>
<td>void</td>
<td>Subclasses of AppenderSkeleton should implement this method to perform actual logging.</td>
</tr>
<tr>
<td>cleanUp()</td>
<td>void</td>
<td>Drop the connection to the remote host and release the underlying connector thread if it has been created</td>
</tr>
<tr>
<td>close()</td>
<td>void</td>
<td>Close this appender.</td>
</tr>
<tr>
<td>getLocationInfo()</td>
<td>boolean</td>
<td>Returns value of the <strong>LocationInfo</strong> option.</td>
</tr>
<tr>
<td>getPort()</td>
<td>int</td>
<td>Returns value of the <strong>Port</strong> option.</td>
</tr>
<tr>
<td>getReconnectionDelay()</td>
<td>int</td>
<td>Returns value of the <strong>ReconnectionDelay</strong> option.</td>
</tr>
<tr>
<td>getRemoteHost()</td>
<td>String</td>
<td>Returns value of the <strong>RemoteHost</strong> option.</td>
</tr>
<tr>
<td>requiresLayout()</td>
<td>boolean</td>
<td>The SocketAppender does not use a layout.</td>
</tr>
<tr>
<td>setLocationInfo(boolean locationInfo)</td>
<td>void</td>
<td>The <strong>LocationInfo</strong> option takes a boolean value.</td>
</tr>
<tr>
<td>setPort(int port)</td>
<td>void</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>The <strong>Port</strong> option takes a positive integer representing the port where the server is waiting for connections.</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><strong>setReconnectionDelay</strong>(int delay) The <strong>ReconnectionDelay</strong> option takes a positive integer representing the number of milliseconds to wait between each failed connection attempt to the server.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><strong>setRemoteHost</strong>(String host) The <strong>RemoteHost</strong> option takes a string value which should be the host name of the server where a <a href="#">SocketNode</a> is running.</td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class org.apache.log4j.**[AppenderSkeleton](#)

- addFilter
- clearFilters
- doAppend
- finalize
- getErrorHandler
- getFilter
- getFirstFilter
- getLayout
- getName
- getThreshold
- isAsSevereAsThreshold
- setErrorHandler
- setLayout
- setName
- setThreshold

**Methods inherited from class java.lang.**[Object](#)

- clone
- equals
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

**Constructor Detail**

**SocketAppender**

```java
class SocketAppender {
    public SocketAppender() {
    }
    public SocketAppender(InetAddress address, int port) {
        // Connects to remote server at address and port.
    }
}
```
public SocketAppender(String host, int port)

Connects to remote server at host and port.

Method Detail

activateOptions

public void activateOptions()

Connect to the specified RemoteHost and Port.

Overrides:
activateOptions in class AppenderSkeleton

close

public void close()

Close this appender.

This will mark the appender as closed and call then close method.

cleanUp

public void cleanUp()

Drop the connection to the remote host and release the underlying connector thread if it has been created

append

public void append(LoggingEvent event)

Description copied from class: AppenderSkeleton
Subclasses of `AppenderSkeleton` should implement this method to perform actual logging. See also `AppenderSkeleton.doAppend` method.

**Overrides:**

`append` in class `AppenderSkeleton`

---

**requiresLayout**

```java
public boolean requiresLayout()
```

The `SocketAppender` does not use a layout. Hence, this method returns `false`.

---

**setRemoteHost**

```java
public void setRemoteHost(String host)
```

The `RemoteHost` option takes a string value which should be the host name of the server where a `SocketNode` is running.

---

**getRemoteHost**

```java
public String getRemoteHost()
```

Returns value of the `RemoteHost` option.

---

**setPort**

```java
public void setPort(int port)
```

The `Port` option takes a positive integer representing the port where the server is waiting for connections.

---

**getPort**
public int getPort()

    Returns value of the Port option.

---

**setLocationInfo**

public void setLocationInfo(boolean locationInfo)

    The LocationInfo option takes a boolean value. If true, the information sent to the remote host will include location information. By default no location information is sent to the server.

---

**getLocationInfo**

public boolean getLocationInfo()

    Returns value of the LocationInfo option.

---

**setReconnectionDelay**

public void setReconnectionDelay(int delay)

    The ReconnectionDelay option takes a positive integer representing the number of milliseconds to wait between each failed connection attempt to the server. The default value of this option is 30000 which corresponds to 30 seconds.

    Setting this option to zero turns off reconnection capability.

---

**getReconnectionDelay**

public int getReconnectionDelay()

    Returns value of the ReconnectionDelay option.
Overview  Package  Use  Tree  Deprecated  Index  Help  Log4j
PREV CLASS  NEXT CLASS  FRAMES  NO FRAMES
SUMMARY:  INNER | FIELD | CONSTR | METHOD  DETAIL:  FIELD | CONSTR | METHOD

Copyright 2000-2005 Apache Software Foundation.
public class SocketHubAppender extends AppenderSkeleton

Sends LoggingEvent objects to a set of remote log servers, usually a SocketNodes.

Acts just like SocketAppender except that instead of connecting to a given remote log server, SocketHubAppender accepts connections from the remote log servers as clients. It can accept more than one connection. When a log event is received, the event is sent to the set of currently connected remote log servers. Implemented this way it does not require any update to the configuration file to send data to another remote log server. The remote log server simply connects to the host and port the SocketHubAppender is running on.

The SocketHubAppender does not store events such that the remote side will events that arrived after the establishment of its connection. Once connected, events arrive in order as guaranteed by the TCP protocol.

This implementation borrows heavily from the SocketAppender.

The SocketHubAppender has the following characteristics:

- If sent to a SocketNode, logging is non-intrusive as far as the log event is concerned. In other words, the event will be logged with the same time stamp, NDC, location info as if it were logged locally.
SocketHubAppender does not use a layout. It ships a serialized LoggingEvent object to the remote side.

SocketHubAppender relies on the TCP protocol. Consequently, if the remote side is reachable, then log events will eventually arrive at remote client.

If no remote clients are attached, the logging requests are simply dropped.

Logging events are automatically buffered by the native TCP implementation. This means that if the link to remote client is slow but still faster than the rate of (log) event production, the application will not be affected by the slow network connection. However, if the network connection is slower than the rate of event production, then the local application can only progress at the network rate. In particular, if the network link to the the remote client is down, the application will be blocked.

On the other hand, if the network link is up, but the remote client is down, the client will not be blocked when making log requests but the log events will be lost due to client unavailability.

The single remote client case extends to multiple clients connections. The rate of logging will be determined by the slowest link.

If the JVM hosting the SocketHubAppender exits before the SocketHubAppender is closed either explicitly or subsequent to garbage collection, then there might be untransmitted data in the pipe which might be lost. This is a common problem on Windows based systems.

To avoid lost data, it is usually sufficient to close() the SocketHubAppender either explicitly or by calling the LogManager.shutdown() method before exiting the application.

Author:
Mark Womack
### Constructor Summary

**SocketHubAppender()**

**SocketHubAppender(int _port)**

Connects to remote server at address and port.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>activateOptions()</strong></td>
<td>Set up the socket server on the specified port.</td>
</tr>
<tr>
<td>void <strong>append(LoggingEvent event)</strong></td>
<td>Append an event to all of current connections.</td>
</tr>
<tr>
<td>void <strong>cleanUp()</strong></td>
<td>Release the underlying ServerMonitor thread, and drop the connections to all connected remote servers.</td>
</tr>
<tr>
<td>void <strong>close()</strong></td>
<td>Close this appender.</td>
</tr>
<tr>
<td>boolean <strong>getLocationInfo()</strong></td>
<td>Returns value of the LocationInfo option.</td>
</tr>
<tr>
<td>int <strong>getPort()</strong></td>
<td>Returns value of the Port option.</td>
</tr>
<tr>
<td>boolean <strong>requiresLayout()</strong></td>
<td>The SocketHubAppender does not use a layout.</td>
</tr>
<tr>
<td>void <strong>setLocationInfo(boolean _locationInfo)</strong></td>
<td>The LocationInfo option takes a boolean value.</td>
</tr>
<tr>
<td>void <strong>setPort(int _port)</strong></td>
<td>The Port option takes a positive integer representing the port where the server is waiting for connections.</td>
</tr>
</tbody>
</table>

### Methods inherited from class org.apache.log4j.AppenderSkeleton
Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SocketHubAppender

public SocketHubAppender()

SocketHubAppender

public SocketHubAppender(int _port)

    Connects to remote server at address and port.

Method Detail

activateOptions

public void activateOptions()

    Set up the socket server on the specified port.
    Overrides:
    activateOptions in class AppenderSkeleton

close

public void close()
Close this appender.

This will mark the appender as closed and call then `cleanUp()` method.

---

**cleanUp**

```java
public void cleanUp()
```

Release the underlying ServerMonitor thread, and drop the connections to all connected remote servers.

---

**append**

```java
public void append(LoggingEvent event)
```

Append an event to all of current connections.

*Overrides:*

- `append` in class `AppenderSkeleton`

---

**requiresLayout**

```java
public boolean requiresLayout()
```

The SocketHubAppender does not use a layout. Hence, this method returns `false`.

---

**setPort**

```java
public void setPort(int _port)
```

The **Port** option takes a positive integer representing the port where the server is waiting for connections.
**getPort**

```java
class...

public int getPort()

    Returns value of the Port option.
```

**setLocationInfo**

```java
class...

public void setLocationInfo(boolean _locationInfo)

    The LocationInfo option takes a boolean value. If true, the information sent to the remote host will include location information. By default no location information is sent to the server.
```

**getLocationInfo**

```java
class...

public boolean getLocationInfo()

    Returns value of the LocationInfo option.
```
public class SocketNode extends Object implements Runnable

Read LoggingEvent objects sent from a remote client using Sockets (TCP). These logging events are logged according to local policy, as if they were generated locally.

For example, the socket node might decide to log events to a local file and also resent them to a second socket node.

Since:
  0.8.4

Author:
  Ceki Gülcü

---

## Constructor Summary

| SocketNode
| Socket socket, LoggerRepository hierarchy |

## Method Summary

| void run() |
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

SocketNode

public SocketNode(Socket socket,
LoggerRepository hierarchy)

Method Detail

run

public void run()

Specified by:
run in interface Runnable
public class **SocketServer**
extends **Object**

A **SocketNode** based server that uses a different hierarchy for each client.

**Usage:**
```java
java org.apache.log4j.net.SocketServer port configFileName configDir
```

where **port** is a part number where the server listens, 
**configFile** is a configuration file fed to the `PropertyConfigurator` a 
**configDir** is a path to a directory containing configuration files, p

The **configFile** is used to configure the log4j default hierarchy that the 
SocketServer will use to report on its actions.

When a new connection is opened from a previously unknown host, say 
foo.bar.net, then the SocketServer will search for a configuration file 
called foo.bar.net.lcf under the directory **configDir** that was passed as 
the third argument. If the file can be found, then a new hierarchy is 
instantiated and configured using the configuration file foo.bar.net.lcf. If 
and when the host foo.bar.net opens another connection to the server, 
then the previously configured hierarchy is used.

In case there is no file called foo.bar.net.lcf under the directory 
**configDir**, then the **generic** hierarchy is used. The generic hierarchy is 
configured using a configuration file called generic.lcf under the 
**configDir** directory. If no such file exists, then the generic hierarchy will 
be identical to the log4j default hierarchy.

Having different client hosts log using different hierarchies ensures the 
total independence of the clients with respect to their logging settings.

Currently, the hierarchy that will be used for a given request depends on 
the IP address of the client host. For example, two separate applicatons
running on the same host and logging to the same server will share the same hierarchy. This is perfectly safe except that it might not provide the right amount of independence between applications. The SocketServer is intended as an example to be enhanced in order to implement more elaborate policies.

Since:

1.0

Author:

Ceki Gülcü

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>SocketServer(File directory)</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void main(String[] argv)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

SocketServer

public SocketServer(File directory)

Method Detail

main
public static void main(String[] argv)
Class SyslogAppender

java.lang.Object
    +-- org.apache.log4j.AppenderSkeleton
        +-- org.apache.log4j.net.SyslogAppender

All Implemented Interfaces:
    Appender, OptionHandler

public class SyslogAppender
    extends AppenderSkeleton

Use SyslogAppender to send log messages to a remote syslog daemon.

Author:
    Ceki Gülcü, Anders Kristensen

Field Summary

<table>
<thead>
<tr>
<th>protected static int</th>
<th>FACILITY_OI</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>LOG_AUTH</td>
</tr>
<tr>
<td></td>
<td>security/authorization messages</td>
</tr>
<tr>
<td>static int</td>
<td>LOG_AUTHPRIV</td>
</tr>
<tr>
<td></td>
<td>security/authorization messages (private)</td>
</tr>
<tr>
<td>static int</td>
<td>LOG_CRON</td>
</tr>
<tr>
<td></td>
<td>clock daemon</td>
</tr>
<tr>
<td>static int</td>
<td>LOG_DAEMON</td>
</tr>
<tr>
<td></td>
<td>System daemons</td>
</tr>
<tr>
<td>static int</td>
<td>LOG_FTP</td>
</tr>
<tr>
<td></td>
<td>ftp daemon</td>
</tr>
<tr>
<td>static int</td>
<td>LOG_KERN</td>
</tr>
<tr>
<td></td>
<td>Kernel messages</td>
</tr>
<tr>
<td>static int LOG_LOCAL0</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>static int LOG_LOCAL1</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>static int LOG_LOCAL2</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>static int LOG_LOCAL3</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>static int LOG_LOCAL4</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>static int LOG_LOCAL5</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>static int LOG_LOCAL6</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>static int LOG_LOCAL7</td>
<td>reserved for local use</td>
</tr>
<tr>
<td>static int LOG_LPR</td>
<td>line printer subsystem</td>
</tr>
<tr>
<td>static int LOG_MAIL</td>
<td>Mail system</td>
</tr>
<tr>
<td>static int LOG_NEWS</td>
<td>network news subsystem</td>
</tr>
<tr>
<td>static int LOG_SYSLOG</td>
<td>messages generated internally by syslogd</td>
</tr>
<tr>
<td>static int LOG_USER</td>
<td>Random user-level messages</td>
</tr>
<tr>
<td>static int LOG_UUCP</td>
<td>UUCP subsystem</td>
</tr>
<tr>
<td>protected static int SYSLOG_HOST_OI</td>
<td></td>
</tr>
</tbody>
</table>

**Fields inherited from class org.apache.log4j.AppenderSkeleton**
closed, errorHandler, headFilter, layout, name, tailFilter, threshold
### Constructor Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyslogAppender</td>
<td></td>
</tr>
<tr>
<td>SyslogAppender(Layout layout, int syslogFacility)</td>
<td></td>
</tr>
<tr>
<td>SyslogAppender(Layout layout, String syslogHost, int syslogFacility)</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void activateOptions()</td>
<td>This method returns immediately as options are activated when they are set.</td>
</tr>
<tr>
<td>void append(LoggingEvent event)</td>
<td>Subclasses of AppenderSkeleton should implement this method to perform actual logging.</td>
</tr>
<tr>
<td>void close()</td>
<td>Release any resources held by this SyslogAppender.</td>
</tr>
<tr>
<td>String getFacility()</td>
<td>Returns the value of the Facility option.</td>
</tr>
<tr>
<td>static int getFacility(String facilityName)</td>
<td>Returns the integer value corresponding to the named syslog facility, or -1 if it couldn't be recognized.</td>
</tr>
<tr>
<td>boolean getFacilityPrinting()</td>
<td>Returns the value of the FacilityPrinting option.</td>
</tr>
<tr>
<td>static String getFacilityString(int syslogFacility)</td>
<td>Returns the specified syslog facility as a lower-case String, e.g.</td>
</tr>
<tr>
<td>String getSyslogHost()</td>
<td>Returns the value of the SyslogHost option.</td>
</tr>
<tr>
<td>boolean requiresLayout()</td>
<td>The SyslogAppender requires a layout.</td>
</tr>
</tbody>
</table>
void setFacility(String facilityName)
  Set the syslog facility.

void setFacilityPrinting(boolean on)
  If the FacilityPrinting option is set to true, the printed message will include the facility name of the application.

void setSyslogHost(String syslogHost)
  The SyslogHost option is the name of the the syslog host where log output should go.

Methods inherited from class org.apache.log4j.AppenderSkeleton
addFilter, clearFilters, doAppend, finalize, getErrorHandler, getFilter, getFirstFilter, getLayout, getName, getThreshold, isAsSevereAsThreshold, setErrorHandler, setLayout, setName, setThreshold

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

LOG_KERN

public static final int LOG_KERN
  Kernel messages

LOG_USER

public static final int LOG_USER
  Random user-level messages

LOG_MAIL
public static final int LOG_MAIL
     Mail system

LOG_DAEMON
public static final int LOG_DAEMON
     System daemons

LOG_AUTH
public static final int LOG_AUTH
     security/authorization messages

LOG_SYSLOG
public static final int LOG_SYSLOG
     messages generated internally by syslogd

LOG_LPR
public static final int LOG_LPR
     line printer subsystem

LOG_NEWS
public static final int LOG_NEWS
     network news subsystem
LOG_UUCP

public static final int LOG_UUCP

UUCP subsystem

LOG_CRON

public static final int LOG_CRON

clock daemon

LOG_AUTHPRIV

public static final int LOG_AUTHPRIV

security/authorization messages (private)

LOG_FTP

public static final int LOG_FTP

ftp daemon

LOG_LOCAL0

public static final int LOG_LOCAL0

reserved for local use

LOG_LOCAL1

public static final int LOG_LOCAL1

reserved for local use
LOG_LOCAL2
public static final int LOG_LOCAL2
    reserved for local use

LOG_LOCAL3
public static final int LOG_LOCAL3
    reserved for local use

LOG_LOCAL4
public static final int LOG_LOCAL4
    reserved for local use

LOG_LOCAL5
public static final int LOG_LOCAL5
    reserved for local use

LOG_LOCAL6
public static final int LOG_LOCAL6
    reserved for local use

LOG_LOCAL7
public static final int LOG_LOCAL7
SYSLOG_HOST_OI

protected static final int SYSLOG_HOST_OI

FACILITY_OI

protected static final int FACILITY_OI

Constructor Detail

SyslogAppender

class public SyslogAppender()

SyslogAppender

class public SyslogAppender(Layout layout, String syslogHost, int syslogFacility)

SyslogAppender

class public SyslogAppender(Layout layout, int syslogFacility)

Method Detail

close

class public void close()

Release any resources held by this SyslogAppender.
getFacilityString

public static String getFacilityString(int syslogFacility)

Returns the specified syslog facility as a lower-case String, e.g. "kern", "user", etc.

getFacility

public static int getFacility(String facilityName)

Returns the integer value corresponding to the named syslog facility, or -1 if it couldn't be recognized.

Parameters:
facilityName - one of the strings KERN, USER, MAIL, DAEMON, AUTH, SYSLOG, LPR, NEWS, UUCP, CRON, AUTHPRIV, FTP, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The matching is case-insensitive.

Since: 1.1

append

public void append(LoggingEvent event)

Description copied from class: AppenderSkeleton

Subclasses of AppenderSkeleton should implement this method to perform actual logging. See also AppenderSkeleton.doAppend method.

Overrides:
append in class AppenderSkeleton
**activateOptions**

```java
public void activateOptions()
```

This method returns immediately as options are activated when they are set.

**Overrides:**

activateOptions in class AppenderSkeleton

---

**requiresLayout**

```java
public boolean requiresLayout()
```

The SyslogAppender requires a layout. Hence, this method returns true.

**Since:**

0.8.4

---

**setSyslogHost**

```java
public void setSyslogHost(String syslogHost)
```

The **SyslogHost** option is the name of the the syslog host where log output should go. A non-default port can be specified by appending a colon and port number to a host name, an IPv4 address or an IPv6 address enclosed in square brackets. **WARNING** If the SyslogHost is not set, then this appender will fail.

---

**getSyslogHost**

```java
public String getSyslogHost()
```

Returns the value of the **SyslogHost** option.

---

**setFacility**
public void setFacility(String facilityName)

Set the syslog facility. This is the Facility option.

The facilityName parameter must be one of the strings KERN, USER, MAIL, DAEMON, AUTH, SYSLOG, LPR, NEWS, UUCP, CRON, AUTHPRIV, FTP, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. Case is unimportant.

Since:
0.8.1

getFacility

public String getFacility()

Returns the value of the Facility option.

setFacilityPrinting

public void setFacilityPrinting(boolean on)

If the FacilityPrinting option is set to true, the printed message will include the facility name of the application. It is false by default.

getFacilityPrinting

public boolean getFacilityPrinting()

Returns the value of the FacilityPrinting option.
org.apache.log4j.net

Class TelnetAppender

java.lang.Object
  +-- org.apache.log4j.AppenderSkeleton
     +-- org.apache.log4j.net.TelnetAppender

All Implemented Interfaces:
  Appender, OptionHandler

public class TelnetAppender
  extends AppenderSkeleton

The TelnetAppender is a log4j appender that specializes in writing to a read-only socket. The output is provided in a telnet-friendly way so that a log can be monitored over TCP/IP. Clients using telnet connect to the socket and receive log data. This is handy for remote monitoring, especially when monitoring a servlet.

Here is a list of the available configuration options:

<table>
<thead>
<tr>
<th>Name</th>
<th>Requirement</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>optional</td>
<td>This parameter determines the port to use for announcing log events. The default port is 23 (telnet).</td>
<td>5875</td>
</tr>
</tbody>
</table>

Author:
  Jay Funnell

Inner Class Summary

<table>
<thead>
<tr>
<th>protected class</th>
<th>TelnetAppender.SocketHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The SocketHandler class is used to accept connections from clients.</td>
</tr>
</tbody>
</table>
### Constructor Summary

**TelnetAppender()**

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>activateOptions()</strong></td>
<td>all of the options have been set, create the socket handler and wait for connections.</td>
</tr>
<tr>
<td>protected void <strong>append(LoggingEvent event)</strong></td>
<td>Handles a log event.</td>
</tr>
<tr>
<td>void <strong>close()</strong></td>
<td>shuts down the appender.</td>
</tr>
<tr>
<td>int <strong>getPort()</strong></td>
<td></td>
</tr>
<tr>
<td>boolean <strong>requiresLayout()</strong></td>
<td>This appender requires a layout to format the text to the attached client(s).</td>
</tr>
<tr>
<td>void <strong>setPort(int port)</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class org.apache.log4j.AppenderSkeleton

- addFilter, clearFilters, doAppend, finalize, getErrorHandler, getFilter, getFirstFilter, getLayout, getName, getThreshold, isAsSevereAsThreshold, setErrorHandler, setLayout, setName, setThreshold

### Methods inherited from class java.lang.Object

- clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Detail

TelnetAppender

public TelnetAppender()

Method Detail

requiresLayout

public boolean requiresLayout()

This appender requires a layout to format the text to the attached client(s).

activateOptions

public void activateOptions()

all of the options have been set, create the socket handler and wait for connections.

Overrides:

activateOptions in class AppenderSkeleton

getPort

public int getPort()

setPort

public void setPort(int port)

close
public void close()
    shuts down the appender.

append

protected void append(LoggingEvent event)
    Handles a log event. For this appender, that means writing the message to each connected client.
    Overrides:
        append in class AppenderSkeleton
protected class `TelnetAppender.SocketHandler` extends `Thread`

The `SocketHandler` class is used to accept connections from clients. It is threaded so that clients can connect/disconnect asynchronously.

### Fields inherited from class `java.lang.Thread`

- `MAX_PRIORITY`, `MIN_PRIORITY`, `NORM_PRIORITY`

### Constructor Summary

**TelnetAppender.SocketHandler**(int port)

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>finalize</strong>()</td>
<td>make sure we close all network connections when this handler is destroyed.</td>
</tr>
<tr>
<td><strong>run</strong>()</td>
<td>Continually accepts client connections.</td>
</tr>
</tbody>
</table>
void send(String message)
sends a message to each of the clients in telnet-friendly output.

Methods inherited from class java.lang.Thread
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TelnetAppender.SocketHandler

public TelnetAppender.SocketHandler(int port) throws IOException

Method Detail

finalize

public void finalize()

make sure we close all network connections when this handler is destroyed.

Overrides:

finalize in class Object

send
public void send(String message)

    sends a message to each of the clients in telnet-friendly output.

run

public void run()

    Continually accepts client connections. Client connections are refused when MAX_CONNECTIONS is reached.

Overrides:
    run in class Thread

Copyright 2000-2005 Apache Software Foundation.
Package org.apache.log4j.nt

Package for NT event logging.

See: Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTEventLogAppender</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.nt Description

Package for NT event logging.

Last modified: Sat Apr 29 14:30:12 MDT 2000

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.nt

Package Hierarchies:
- All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
  - class org.apache.log4j.nt.**NTEventLogAppender**

Copyright 2000-2005 Apache Software Foundation.
Uses of Package
org.apache.log4j.nt

No usage of org.apache.log4j.nt

Copyright 2000-2005 Apache Software Foundation.
org.apache.log4j.nt Class NTEventLogAppender

java.lang.Object
  +-- org.apache.log4j.AppenderSkeleton
      +-- org.apache.log4j.nt.NTEventLogAppender

All Implemented Interfaces:
  Appender, OptionHandler

public class NTEventLogAppender
extends AppenderSkeleton

Append to the NT event log system.

WARNING This appender can only be installed and used on a Windows system.

Do not forget to place the file NTEventLogAppender.dll in a directory that is on the PATH of the Windows system. Otherwise, you will get a java.lang.UnsatisfiedLinkError.

Author:
  Chris Taylor, Jim Cakalic

| Fields inherited from class org.apache.log4j.AppenderSkeleton |
| closed, errorHandler, headFilter, layout, name, tailFilter, threshold |

<p>| Constructor Summary |
| NTEventLogAppender() |
| NTEventLogAppender(Layout layout) |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NTEventLogAppender(String source)</code></td>
<td></td>
</tr>
<tr>
<td><code>NTEventLogAppender(String source, Layout layout)</code></td>
<td></td>
</tr>
<tr>
<td><code>NTEventLogAppender(String server, String source)</code></td>
<td></td>
</tr>
<tr>
<td><code>NTEventLogAppender(String server, String source, Layout layout)</code></td>
<td></td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>activateOptions()</code></td>
<td>Derived appenders should override this method if option structure requires it.</td>
</tr>
<tr>
<td><code>append(LoggingEvent event)</code></td>
<td>Subclasses of AppenderSkeleton should implement this method to perform actual logging.</td>
</tr>
<tr>
<td><code>close()</code></td>
<td>Release any resources allocated within the appender such as file handles, network connections, etc.</td>
</tr>
<tr>
<td><code>finalize()</code></td>
<td>Finalize this appender by calling the derived class' <code>close</code> method.</td>
</tr>
<tr>
<td><code>getSource()</code></td>
<td></td>
</tr>
<tr>
<td><code>requiresLayout()</code></td>
<td>The <code>NTEventLogAppender</code> requires a layout.</td>
</tr>
<tr>
<td><code>setSource(String source)</code></td>
<td>The <code>Source</code> option which names the source of the event.</td>
</tr>
</tbody>
</table>

Methods inherited from class org.apache.log4j.AppenderSkeleton

`addFilter`, `clearFilters`, `doAppend`, `getErrorHandler`, `getFilter`, `getFirstFilter`, `getLayout`, `getName`, `getThreshold`, `isAsSevereAsThreshold`, `setErrorHandler`, `setLayout`, `setName`, `setThreshold`
Methods inherited from class java.lang.**Object**
- clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Constructor Detail**

**NTEventLogAppender**

public static **NTEventLogAppender**(

)

**NTEventLogAppender**

public **NTEventLogAppender**(String source)

**NTEventLogAppender**

public **NTEventLogAppender**(String server, String source)

**NTEventLogAppender**

public **NTEventLogAppender**(Layout layout)

**NTEventLogAppender**

public **NTEventLogAppender**(String source, Layout layout)

**NTEventLogAppender**

public **NTEventLogAppender**(String server, String source, Layout layout)
close

```java
public void close()
```

**Description copied from interface: Appender**
Release any resources allocated within the appender such as file handles, network connections, etc.

It is a programming error to append to a closed appender.

activateOptions

```java
public void activateOptions()
```

**Description copied from class: AppenderSkeleton**
Derived appenders should override this method if option structure requires it.

**Overrides:**
activateOptions in class AppenderSkeleton

append

```java
public void append(LoggingEvent event)
```

**Description copied from class: AppenderSkeleton**
Subclasses of AppenderSkeleton should implement this method to perform actual logging. See also AppenderSkeleton.doAppend method.

**Overrides:**
append in class AppenderSkeleton

finalize

```java
public void finalize()
```
Description copied from class: AppenderSkeleton
Finalize this appender by calling the derived class' close method.

Overrides:
   finalize in class AppenderSkeleton

---

**setSource**

public void setSource(String source)

The Source option which names the source of the event. The current value of this constant is Source.

---

**getSource**

public String getSource()

---

**requiresLayout**

public boolean requiresLayout()

The NTEventLogAppender requires a layout. Hence, this method always returns true.

---

Copyright 2000-2005 Apache Software Foundation.
Package org.apache.log4j.or

ObjectRenders are responsible for rendering messages depending on their class type.

See: Description

**Interface Summary**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ObjectRenderer</strong></td>
<td>Implement this interface in order to render objects as strings.</td>
</tr>
</tbody>
</table>

**Class Summary**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RendererMap</strong></td>
<td>Map class objects to an ObjectRenderer.</td>
</tr>
<tr>
<td><strong>ThreadGroupRenderer</strong></td>
<td>Render ThreadGroup objects in a format similar to the information output by the ThreadGroup.list() method.</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.or Description

ObjectRenders are responsible for rendering messages depending on their class type.

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.or

Package Hierarchies:
All Packages
Class Hierarchy

- class java.lang.Object
  - class org.apache.log4j.or.RendererMap
  - class org.apache.log4j.or.ThreadGroupRenderer (implements org.apache.log4j.or.ObjectRenderer)
Interface Hierarchy

- interface org.apache.log4j.or.**ObjectRenderer**
Packages that use org.apache.log4j.or

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.or</td>
<td>ObjectRenderers are responsible for rendering messages depending on their class type.</td>
</tr>
<tr>
<td>org.apache.log4j.or.jms</td>
<td>This package contains the MessageRenderer which renders objects of type javax.jms.Message.</td>
</tr>
<tr>
<td>org.apache.log4j.or.sax</td>
<td>This package contains the AttributesRenderer which renders objects of class org.xml.sax.Attributes.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
</tbody>
</table>

Classes in org.apache.log4j.or used by org.apache.log4j

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectRenderer</td>
<td>Implement this interface in order to render objects as strings.</td>
</tr>
<tr>
<td>RendererMap</td>
<td>Map class objects to an ObjectRenderer.</td>
</tr>
</tbody>
</table>

Classes in org.apache.log4j.or used by org.apache.log4j.or

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectRenderer</td>
<td>Implement this interface in order to render objects as strings.</td>
</tr>
</tbody>
</table>
### Classes in `org.apache.log4j.or` used by `org.apache.log4j.or.jms`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectRenderer</td>
<td>Implement this interface in order to render objects as strings.</td>
</tr>
</tbody>
</table>

### Classes in `org.apache.log4j.or` used by `org.apache.log4j.or.sax`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectRenderer</td>
<td>Implement this interface in order to render objects as strings.</td>
</tr>
</tbody>
</table>

### Classes in `org.apache.log4j.or` used by `org.apache.log4j.spi`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectRenderer</td>
<td>Implement this interface in order to render objects as strings.</td>
</tr>
<tr>
<td>RendererMap</td>
<td>Map class objects to an <a href="#">ObjectRenderer</a>.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
public interface ObjectRenderer

Implement this interface in order to render objects as strings.

Since: 1.0

Author: Ceki Gülcü

Method Summary

| String doRender(Object o) | Render the object passed as parameter as a String. |

Method Detail

doRender

public String doRender(Object o)

Render the object passed as parameter as a String.
public class RendererMap
extends Object

Map class objects to an ObjectRenderer.

Since:
version 1.0
Author:
Ceki Gülcü

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>RendererMap()</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void addRenderer(RendererSupport repository, String renderedClassName, String renderingClassName)</td>
<td>Add a renderer to a hierarchy passed as parameter.</td>
</tr>
<tr>
<td>void clear()</td>
<td></td>
</tr>
<tr>
<td>String findAndRender(Object o)</td>
<td>Find the appropriate renderer for the class type of the o parameter.</td>
</tr>
<tr>
<td>ObjectRenderer get(Class clazz)</td>
<td>Search the parents of clazz for a renderer.</td>
</tr>
<tr>
<td>ObjectRenderer get(Object o)</td>
<td>Syntactic sugar method that calls get(Class) with the</td>
</tr>
</tbody>
</table>
class of the object parameter.

ObjectRenderer

```java
getDefaultRenderer()
```

```java
void put(Class clazz, ObjectRenderer or)
```

Register an `ObjectRenderer` for `clazz`.

Methods inherited from class java.lang.Object

```java
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

**Constructor Detail**

RendererMap

```java
public RendererMap()
```

**Method Detail**

addRenderer

```java
public static void addRenderer(RendererSupport repository, String renderedClassName, String renderingClassName)
```

Add a renderer to a hierarchy passed as parameter.

findAndRender

```java
public String findAndRender(Object o)
```

Find the appropriate renderer for the class type of the `o` parameter. This is accomplished by calling the `getClass` method. Once a renderer is found, it is applied on the object `o` and the result is returned as a `String`. 
get

public ObjectRenderer get(Object o)

Syntactic sugar method that calls get(Class) with the class of the object parameter.

get

public ObjectRenderer get(Class clazz)

Search the parents of clazz for a renderer. The renderer closest in the hierarchy will be returned. If no renderers could be found, then the default renderer is returned.

The search first looks for a renderer configured for clazz. If a renderer could not be found, then the search continues by looking at all the interfaces implemented by clazz including the super-interfaces of each interface. If a renderer cannot be found, then the search looks for a renderer defined for the parent (superclass) of clazz. If that fails, then all the interfaces implemented by the parent of clazz are searched and so on.

For example, if A0, A1, A2 are classes and X0, X1, X2, Y0, Y1 are interfaces where A2 extends A1 which in turn extends A0 and similarly X2 extends X1 which extends X0 and Y1 extends Y0. Let us also assume that A1 implements the Y0 interface and that A2 implements the X2 interface.

The table below shows the results returned by the get(A2.class) method depending on the renderers added to the map.

<table>
<thead>
<tr>
<th>Added renderers</th>
<th>Value returned by get(A2.class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0Renderer</td>
<td>A0Renderer</td>
</tr>
<tr>
<td>A0Renderer, A1Renderer</td>
<td>A1Renderer</td>
</tr>
<tr>
<td>X0Renderer</td>
<td>X0Renderer</td>
</tr>
<tr>
<td>A1Renderer, X0Renderer</td>
<td>X0Renderer</td>
</tr>
</tbody>
</table>
This search algorithm is not the most natural, although it is particularly easy to implement. Future log4j versions may implement a more intuitive search algorithm. However, the present algorithm should be acceptable in the vast majority of circumstances.

**getDefaultRenderer**

```java
public ObjectRenderer getDefaultRenderer()
```

**clear**

```java
public void clear()
```

**put**

```java
public void put(Class clazz, ObjectRenderer or)
```

Register an `ObjectRenderer` for `clazz`.

Copyright 2000-2005 Apache Software Foundation.
public class ThreadGroupRenderer
extends Object
implements ObjectRenderer

Render ThreadGroup objects in a format similar to the information output by the ThreadGroup.list() method.

Since: 1.0
Author: Ceki Gülcü
Constructor Detail

ThreadGroupRenderer

public ThreadGroupRenderer()

Method Detail

doRender

public String doRender(Object o)

Render a ThreadGroup object similar to the way that the ThreadGroup.list() method output information.

The output of a simple program consisting of one main thread is:

java.lang.ThreadGroup[name=main, maxpri=10]
Thread=[main,5,false]

The boolean value in thread information is the value returned by Thread.isDaemon().

Specified by:

doRender in interface ObjectRenderer
Package org.apache.log4j.or.jms

This package contains the MessageRenderer which renders objects of type javax.jms.Message.

See:   Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageRenderer</td>
</tr>
</tbody>
</table>
This package contains the MessageRenderer which renders objects of type `javax.jms.Message`.
Hierarchy For Package org.apache.log4j.or.jms

Package Hierarchies:

All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.or.jms.**MessageRenderer** (implements org.apache.log4j.or.**ObjectRenderer**)

Copyright 2000-2005 Apache Software Foundation.
Uses of Package
org.apache.log4j.or.jms

No usage of org.apache.log4j.or.jms

Copyright 2000-2005 Apache Software Foundation.
Class MessageRenderer

All Implemented Interfaces:
   ObjectRenderer

public class MessageRenderer
    extends Object
    implements ObjectRenderer

Render javax.jms.Message objects.

Since: 1.0
Author: Ceki Gülcü

Constructor Summary

MessageRenderer()

Method Summary

String doRender(Object o)
   Render a Message.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Detail

MessageRenderer

public MessageRenderer()

Method Detail

doRender

public String doRender(Object o)

Render a Message.

Specified by:
doRender in interface ObjectRenderer

Copyright 2000-2005 Apache Software Foundation.
Package org.apache.log4j.or.sax

This package contains the AttributesRenderer which renders object of class org.xml.sax.Attributes.

See: Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributesRenderer Render org.xml.sax.Attributes objects.</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.or.sax Description

This package contains the AttributesRenderer which renders object of class org.xml.sax.Attributes.
Hierarchy For Package org.apache.log4j.or.sax

Package Hierarchies:
   All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.or.sax.**AttributesRenderer** (implements org.apache.log4j.or.**ObjectRenderer**)

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.or.sax
<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: INNER</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAME</td>
</tr>
</tbody>
</table>
Class AttributesRenderer

org.apache.log4j.or.sax

java.lang.Object
+-org.apache.log4j.or.sax.AttributesRenderer

All Implemented Interfaces:
   ObjectRenderer

public class AttributesRenderer
extends Object
implements ObjectRenderer

Render org.xml.sax.Attributes objects.

Since:
   1.2
Author:
   Ceki Gülcü

Constructor Summary

AttributesRenderer()

Method Summary

String doRender(Object o)
   Render a Attributes.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
AttributesRenderer

public AttributesRenderer()

Method Detail

doRender

public String doRender(Object o)

Render a Attributes.
Specified by:
doRender in interface ObjectRenderer
# Package org.apache.log4j.performance

Package to measure the performance of the different log4j components.

See: Description

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ListVsVector</strong></td>
<td>Compares the performance of looping through a list versus a Vector.</td>
</tr>
<tr>
<td><strong>NewVsSetLen</strong></td>
<td>This program compares the cost of creating a new StringBuffer and converting it to a String versus keeping the same StringBuffer, setting its size to zero and then converting it to String.</td>
</tr>
<tr>
<td><strong>NOPWriter</strong></td>
<td>Extends Writer with methods that return immediately without doing anything.</td>
</tr>
<tr>
<td><strong>NullAppender</strong></td>
<td>A bogus appender which calls the format method of its layout object but does not write the result anywhere.</td>
</tr>
<tr>
<td><strong>SystemTime</strong></td>
<td>Measures the time required to make a System.currentTimeMillis() and Thread.currentThread().getName() calls.</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.performance

Description

Package to measure the performance of the different log4j components.

The org.apache.log4j.performance package is intended for internal use only. Consequently, the classes in this package are not included in the log4j.jar file.
Hierarchy For Package
org.apache.log4j.performance

Package Hierarchies:
All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.performance.**NullAppender**
  - class org.apache.log4j.performance.**ListVsVector**
  - class org.apache.log4j.performance.**NewVsSetLen**
  - class org.apache.log4j.performance.**SystemTime**
  - class java.io.**Writer**
    - class org.apache.log4j.performance.**NOPWriter**
Uses of Package
org.apache.log4j.performance

No usage of org.apache.log4j.performance

Copyright 2000-2005 Apache Software Foundation.
public class ListVsVector
extends Object

Compares the performance of looping through a list versus a Vector. Chain looping is *20* times faster than vector access on JDK 1.1.7B on NT

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListVsVector()</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void main(String[] args)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Constructor Detail**

ListVsVector

public ListVsVector()
public static void main(String[] args)
public class NewVsSetLen
extends Object

This program compares the cost of creating a new StringBuffer and converting it to a String versus keeping the same StringBuffer, setting its size to zero and then converting it to String.

The table below gives some figures.

<table>
<thead>
<tr>
<th>Total Message Length</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New Buffer</td>
<td>setLength</td>
<td>New Buffer</td>
<td>setLength</td>
</tr>
<tr>
<td>256</td>
<td>33</td>
<td>22</td>
<td>34</td>
<td>22</td>
</tr>
<tr>
<td>1024</td>
<td>58</td>
<td>41</td>
<td>59</td>
<td>45</td>
</tr>
<tr>
<td>4096</td>
<td>146</td>
<td>132</td>
<td>138</td>
<td>132</td>
</tr>
<tr>
<td>16384</td>
<td>617</td>
<td>593</td>
<td>593</td>
<td>609</td>
</tr>
<tr>
<td>65536</td>
<td>3218</td>
<td>3281</td>
<td>3093</td>
<td>3125</td>
</tr>
<tr>
<td>262144</td>
<td>14750</td>
<td>15125</td>
<td>14000</td>
<td>15500</td>
</tr>
<tr>
<td>1048576</td>
<td>87500</td>
<td>80000</td>
<td>60500</td>
<td>82000</td>
</tr>
</tbody>
</table>

The tests copy a message to a destination string buffer and then copy a 256 character buffer to another buffer the number of times as specified by the secondary loop length.

The setLength(0) method is usually faster. However, after copying a large string it becomes slow even when copying small strings.
This is due to a peculiarity in the `copy` method in StringBuffer class which creates a character array of the same length as the old buffer even if the vast majority of those characters are unused.

The tests were performed on Linux using IBM's JDK 1.1.8.

The test script is a crude model of what might happen in reality. If you remain unconvinced of its results, then please send your alternative measurement scenario.

---

### Constructor Summary

| NewVsSetLen() |

---

### Method Summary

| static void main(String[] args) |

### Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

---

### Constructor Detail

#### NewVsSetLen

public NewVsSetLen()

---

### Method Detail

#### main
public static void main(String[] args)
<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: INNER | FIELD | CONSTR | METHOD

FRAMES  NO FRAMES
DETAIL: FIELD | CONSTR | METHOD
public class NOPWriter extends Writer

Extends writer with methods that return immediately without doing anything. This class is used to measure the cost of constructing a log message but not actually writing to any device.

The org.apache.log4j.performance.NOPWriter class is intended for internal use only. Consequently, it is not included in the log4j.jar file.

Author:
Ceki Gülcü

Fields inherited from class java.io.Writer
lock

Constructor Summary
 NOPWriter()

Method Summary
 void close()
 void flush()
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td><code>void write(char[] cbuf)</code></td>
</tr>
<tr>
<td>write</td>
<td><code>void write(char[] cbuf, int off, int len)</code></td>
</tr>
<tr>
<td>write</td>
<td><code>void write(int b)</code></td>
</tr>
<tr>
<td>write</td>
<td><code>void write(String s)</code></td>
</tr>
<tr>
<td>write</td>
<td><code>void write(String s, int off, int len)</code></td>
</tr>
</tbody>
</table>

Methods inherited from class `java.lang.Object`:
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`  

## Constructor Detail

**NOPWriter**

```java
public NOPWriter()
```

## Method Detail

### write

```java
public void write(char[] cbuf)
```

Throws `IOException`  

**Overrides:**
- `write` in class `Writer`
int off,
    int len)
    throws IOException

Overrides:
    write in class Writer

write

class String
public void write(int b)
    throws IOException

Overrides:
    write in class Writer

write

class Writer
public void write(String s)
    throws IOException

Overrides:
    write in class Writer

write

class Writer
public void write(String s,
    int off,
    int len)
    throws IOException

Overrides:
    write in class Writer

flush

class Writer
public void flush()
    throws IOException
Overrides:

flush in class Writer

---

close

close() throws IOException

Overrides:

close in class Writer

---

Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
<th>Log4j 1.2.14</th>
</tr>
</thead>
</table>

PREV CLASS | NEXT CLASS | FRAMES | NO FRAMES | SUMMARY: INNER | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD |
package org.apache.log4j.performance;

import java.lang.Object;

public class NullAppender extends AppenderSkeleton {

  // A bogus appender which calls the format method of its layout object but
does not write the result anywhere.

  The org.apache.log4j.performance.NullAppender class is intended for
  internal use only. Consequently, it is not included in the log4j.jar file.

  Field Summary

  static String s
  String t

  Fields inherited from class org.apache.log4j.AppenderSkeleton
  closed, errorHandler, headFilter, layout, name, tailFilter, threshold

  Constructor Summary

  NullAppender()

NullAppender (Layout layout)

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>append(LoggingEvent event)</code></td>
<td>Subclasses of AppenderSkeleton should implement this method to perform actual logging.</td>
</tr>
<tr>
<td><code>close()</code></td>
<td>Release any resources allocated within the appender such as file handles, network connections, etc.</td>
</tr>
<tr>
<td><code>doAppend(LoggingEvent event)</code></td>
<td>This method performs threshold checks and invokes filters before delegating actual logging to the subclasses specific <code>AppenderSkeleton.append(org.apache.log4j.spi.LoggingEvent)</code> method.</td>
</tr>
<tr>
<td><code>requiresLayout()</code></td>
<td>This is a bogus appender but it still uses a layout.</td>
</tr>
</tbody>
</table>

Methods inherited from class org.apache.log4j.AppenderSkeleton:
- activateOptions
- addFilter
- clearFilters
- finalize
- getErrorHandler
- getFilter
- getFirstFilter
- getLayout
- getName
- getThreshold
- isAsSevereAsThreshold
- setErrorHandler
- setLayout
- setName
- setThreshold

Methods inherited from class java.lang.Object:
- clone
- equals
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

## Field Detail

**s**

public static String s
public String t

### Constructor Detail

**NullAppender**

public NullAppender()

public NullAppender(Layout layout)

### Method Detail

**close**

public void close()

*Description copied from interface: Appender*
Release any resources allocated within the appender such as file handles, network connections, etc.

It is a programming error to append to a closed appender.

**doAppend**

public void doAppend(LoggingEvent event)

*Description copied from class: AppenderSkeleton*
This method performs threshold checks and invokes filters before delegating actual logging to the subclasses specific

*Overrides:*
**doAppend** in class **AppenderSkeleton**

append

```java
public void append(LoggingEvent event)
```

**Description copied from class:** **AppenderSkeleton**
Subclasses of `AppenderSkeleton` should implement this method to perform actual logging. See also `AppenderSkeleton.doAppend` method.

**Overrides:**
`append` in class **AppenderSkeleton**

requiresLayout

```java
public boolean requiresLayout()
```

This is a bogus appender but it still uses a layout.

Copyright 2000-2005 Apache Software Foundation.
public class SystemTime extends Object

Measures the time required to make a System.currentTimeMillis() and Thread.currentThread().getName() calls.

On an 233Mhz NT machine (JDK 1.1.7B) the System.currentTimeMillis() call takes under half a microsecond to complete whereas the Thread.currentThread().getName() call takes about 4 micro-seconds.
public SystemTime()

### Method Detail

#### main

public static void main(String[] args)
Package org.apache.log4j.spi

Contains part of the System Programming Interface (SPI) needed to extend log4j.

See:   Description

<table>
<thead>
<tr>
<th>InterfaceSummary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderAttachable</td>
<td>Interface for attaching appenders to objects.</td>
</tr>
<tr>
<td>Configurator</td>
<td>Implemented by classes capable of configuring log4j using a URL.</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>This interface defines commonly encountered error codes.</td>
</tr>
<tr>
<td>ErrorHandler</td>
<td>Appenders may delegate their error handling to ErrorHandlers.</td>
</tr>
<tr>
<td>HierarchyEventListener</td>
<td>Listen to events occuring within a Hierarchy.</td>
</tr>
<tr>
<td>LoggerFactory</td>
<td>Implement this interface to create new instances of Logger or a sub-class of Logger.</td>
</tr>
<tr>
<td>LoggerRepository</td>
<td>A LoggerRepository is used to create and retrieve Loggers.</td>
</tr>
<tr>
<td>OptionHandler</td>
<td>A string based interface to configure package components.</td>
</tr>
<tr>
<td>RendererSupport</td>
<td></td>
</tr>
<tr>
<td>RepositorySelector</td>
<td>The LogManager uses one (and only one) RepositorySelector implementation to select the LoggerRepository for a particular application context.</td>
</tr>
<tr>
<td>TriggeringEventEvaluator</td>
<td>Implementations of this interface allow certain appenders to decide when to perform an appender specific action.</td>
</tr>
<tr>
<td>Class Summary</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td><strong>DefaultRepositorySelector</strong></td>
<td>Users should extend this class to implement customized logging event filtering.</td>
</tr>
<tr>
<td><strong>Filter</strong></td>
<td>The internal representation of caller location information.</td>
</tr>
<tr>
<td><strong>LocationInfo</strong></td>
<td>The internal representation of logging events.</td>
</tr>
<tr>
<td><strong>LoggingEvent</strong></td>
<td>RootCategory</td>
</tr>
<tr>
<td><strong>RootCategory</strong></td>
<td>RootLogger</td>
</tr>
<tr>
<td><strong>ThrowableInformation</strong></td>
<td>ThrowableInformation is log4j's internal representation of throwables.</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.spi Description

Contains part of the System Programming Interface (SPI) needed to extend log4j.

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.spi

Package Hierarchies:
    All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.**Category** (implements org.apache.log4j.spi.**AppenderAttachable**)
    - class org.apache.log4j.**Logger**
      - class org.apache.log4j.spi.**RootCategory**
      - class org.apache.log4j.spi.**RootLogger**
    - class org.apache.log4j.spi.**DefaultRepositorySelector** (implements org.apache.log4j.spi.**RepositorySelector**)
    - class org.apache.log4j.spi.**Filter** (implements org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.spi.**LocationInfo** (implements java.io.**Serializable**)
    - class org.apache.log4j.spi.**LoggingEvent** (implements java.io.**Serializable**)
    - class org.apache.log4j.spi.**ThrowableInformation** (implements java.io.**Serializable**)
Interface Hierarchy

- interface org.apache.log4j.spi.AppenderAttachable
- interface org.apache.log4j.spi.Configurator
- interface org.apache.log4j.spi.ErrorCode
- interface org.apache.log4j.spi.HierarchyEventListener
- interface org.apache.log4j.spi.LoggerFactory
- interface org.apache.log4j.spi.LoggerRepository
- interface org.apache.log4j.spi.OptionHandler
- interface org.apache.log4j.spi.ErrorHandler
- interface org.apache.log4j.spi.RendererSupport
- interface org.apache.log4j.spi.RepositorySelector
- interface org.apache.log4j.spi.TriggeringEventEvaluator

Copyright 2000-2005 Apache Software Foundation.
# Uses of Package

## org.apache.log4j.spi

<table>
<thead>
<tr>
<th>Packages that use <code>org.apache.log4j.spi</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.apache.log4j</code></td>
<td>The main log4j package.</td>
</tr>
<tr>
<td><code>org.apache.log4j.helpers</code></td>
<td>This package is used internally.</td>
</tr>
<tr>
<td><code>org.apache.log4j.jdbc</code></td>
<td>The JDBCAppender provides for sending log events to a database.</td>
</tr>
<tr>
<td><code>org.apache.log4j.jmx</code></td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
<tr>
<td><code>org.apache.log4j.lf5</code></td>
<td></td>
</tr>
<tr>
<td><code>org.apache.log4j.net</code></td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td><code>org.apache.log4j.nt</code></td>
<td>Package for NT event logging.</td>
</tr>
<tr>
<td><code>org.apache.log4j.or</code></td>
<td>ObjectRenders are responsible for rendering messages depending on their class type.</td>
</tr>
<tr>
<td><code>org.apache.log4j.performance</code></td>
<td>Package to measure the performance of the different log4j components.</td>
</tr>
<tr>
<td><code>org.apache.log4j.spi</code></td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td><code>org.apache.log4j.varia</code></td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td><code>org.apache.log4j.xml</code></td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in <code>org.apache.log4j.spi</code> used by <code>org.apache.log4j</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppenderAttachable</code></td>
<td>Interface for attaching appenders to objects.</td>
</tr>
</tbody>
</table>
### Classes in `org.apache.log4j.spi` used by `org.apache.log4j.helpers`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderAttachable</td>
<td>Interface for attaching appenders to objects.</td>
</tr>
<tr>
<td>ErrorHandler</td>
<td>Appenders may delegate their error handling to ErrorHandlers.</td>
</tr>
<tr>
<td>LoggerFactory</td>
<td>Implement this interface to create new instances of Logger or a sub-class of Logger.</td>
</tr>
<tr>
<td>LoggerRepository</td>
<td>A <code>LoggerRepository</code> is used to create and retrieve Loggers.</td>
</tr>
<tr>
<td>LoggingEvent</td>
<td>The internal representation of logging events.</td>
</tr>
<tr>
<td>OptionHandler</td>
<td>A string based interface to configure package components.</td>
</tr>
<tr>
<td>RendererSupport</td>
<td></td>
</tr>
<tr>
<td>RepositorySelector</td>
<td>The LogManager uses one (and only one) RepositorySelector implementation to select the <code>LoggerRepository</code> for a particular application context.</td>
</tr>
</tbody>
</table>
A *LoggerRepository* is used to create and retrieve Loggers.

<table>
<thead>
<tr>
<th><strong>LoggingEvent</strong></th>
<th>The internal representation of logging events.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OptionHandler</strong></td>
<td>A string based interface to configure package components.</td>
</tr>
</tbody>
</table>

**Classes in** [org.apache.log4j.spi](https://logging.apache.org/log4j/2.x/api/index.html) **used by** [org.apache.log4j.jdbc](https://logging.apache.org/log4j/2.x/api/index.html)

<table>
<thead>
<tr>
<th><strong>LoggingEvent</strong></th>
<th>The internal representation of logging events.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OptionHandler</strong></td>
<td>A string based interface to configure package components.</td>
</tr>
</tbody>
</table>

**Classes in** [org.apache.log4j.spi](https://logging.apache.org/log4j/2.x/api/index.html) **used by** [org.apache.log4j.jmx](https://logging.apache.org/log4j/2.x/api/index.html)

| **HierarchyEventListener** | Listen to events occurring within a [Hierarchy](https://logging.apache.org/log4j/2.x/api/index.html). |

**Classes in** [org.apache.log4j.spi](https://logging.apache.org/log4j/2.x/api/index.html) **used by** [org.apache.log4j.lf5](https://logging.apache.org/log4j/2.x/api/index.html)

<table>
<thead>
<tr>
<th><strong>Configurator</strong></th>
<th>Implemented by classes capable of configuring log4j using a URL.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LoggerRepository</strong></td>
<td>A <em>LoggerRepository</em> is used to create and retrieve Loggers.</td>
</tr>
<tr>
<td><strong>LoggingEvent</strong></td>
<td>The internal representation of logging events.</td>
</tr>
<tr>
<td><strong>OptionHandler</strong></td>
<td></td>
</tr>
<tr>
<td>Classes in org.apache.log4j.spi used by org.apache.log4j.net</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>LoggerRepository</strong></td>
<td></td>
</tr>
<tr>
<td>A LoggerRepository is used to create and retrieve Loggers.</td>
<td></td>
</tr>
<tr>
<td><strong>LoggingEvent</strong></td>
<td></td>
</tr>
<tr>
<td>The internal representation of logging events.</td>
<td></td>
</tr>
<tr>
<td><strong>OptionHandler</strong></td>
<td></td>
</tr>
<tr>
<td>A string based interface to configure package components.</td>
<td></td>
</tr>
<tr>
<td><strong>TriggeringEventEvaluator</strong></td>
<td></td>
</tr>
<tr>
<td>Implementations of this interface allow certain appenders to decide when to perform an appender specific action.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j.spi used by org.apache.log4j.nt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LoggingEvent</strong></td>
</tr>
<tr>
<td>The internal representation of logging events.</td>
</tr>
<tr>
<td><strong>OptionHandler</strong></td>
</tr>
<tr>
<td>A string based interface to configure package components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j.spi used by org.apache.log4j.or</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RendererSupport</strong></td>
</tr>
</tbody>
</table>
### Classes in `org.apache.log4j.spi` used by `org.apache.log4j.performance`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LoggingEvent</code></td>
<td>The internal representation of logging events.</td>
</tr>
<tr>
<td><code>OptionHandler</code></td>
<td>A string based interface to configure package components.</td>
</tr>
</tbody>
</table>

### Classes in `org.apache.log4j.spi` used by `org.apache.log4j.spi`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppenderAttachable</code></td>
<td>Interface for attaching appenders to objects.</td>
</tr>
<tr>
<td><code>Filter</code></td>
<td>Users should extend this class to implement customized logging event filtering.</td>
</tr>
<tr>
<td><code>HierarchyEventListener</code></td>
<td>Listen to events occurring within a <code>Hierarchy</code>.</td>
</tr>
<tr>
<td><code>LocationInfo</code></td>
<td>The internal representation of caller location information.</td>
</tr>
<tr>
<td><code>LoggerFactory</code></td>
<td>Implement this interface to create new instances of Logger or a sub-class of Logger.</td>
</tr>
<tr>
<td><code>LoggerRepository</code></td>
<td>A <code>LoggerRepository</code> is used to create and retrieve Loggers.</td>
</tr>
<tr>
<td><code>LoggingEvent</code></td>
<td>The internal representation of logging events.</td>
</tr>
<tr>
<td><code>OptionHandler</code></td>
<td>A string based interface to configure package components.</td>
</tr>
<tr>
<td><code>RepositorySelector</code></td>
<td>The <code>LogManager</code> uses one (and only one) <code>RepositorySelector</code> implementation to select the <code>LoggerRepository</code> for a particular</td>
</tr>
<tr>
<td><strong>application context.</strong></td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ThrowableInformation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ThrowableInformation is log4j's internal representation of throwables.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Classes in</strong> org.apache.log4j.spi <strong>used by</strong> org.apache.log4j.varia</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configurator</strong></td>
</tr>
<tr>
<td>Implemented by classes capable of configuring log4j using a URL.</td>
</tr>
<tr>
<td>** ErrorHandler**</td>
</tr>
<tr>
<td>Appenders may delegate their error handling to ErrorHandler.</td>
</tr>
<tr>
<td><strong>Filter</strong></td>
</tr>
<tr>
<td>Users should extend this class to implement customized logging event filtering.</td>
</tr>
<tr>
<td><strong>LoggerRepository</strong></td>
</tr>
<tr>
<td>A LoggerRepository is used to create and retrieve Loggers.</td>
</tr>
<tr>
<td><strong>LoggingEvent</strong></td>
</tr>
<tr>
<td>The internal representation of logging events.</td>
</tr>
<tr>
<td><strong>OptionHandler</strong></td>
</tr>
<tr>
<td>A string based interface to configure package components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Classes in</strong> org.apache.log4j.spi <strong>used by</strong> org.apache.log4j.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configurator</strong></td>
</tr>
<tr>
<td>Implemented by classes capable of configuring log4j using a URL.</td>
</tr>
<tr>
<td><strong>LoggerRepository</strong></td>
</tr>
<tr>
<td>A LoggerRepository is used to create and retrieve Loggers.</td>
</tr>
<tr>
<td><strong>LoggingEvent</strong></td>
</tr>
<tr>
<td>The internal representation of logging events.</td>
</tr>
</tbody>
</table>
OptionHandler

A string based interface to configure package components.
public interface AppenderAttachable

Interface for attaching appenders to objects.

Since: 
0.9.1
Author: 
Ceki Gülcü

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAppender(Appender newAppender)</td>
<td>Add an appender.</td>
</tr>
<tr>
<td>Enumeration getAllAppenders()</td>
<td>Get all previously added appenders as an Enumeration.</td>
</tr>
<tr>
<td>Appender getAppender(String name)</td>
<td>Get an appender by name.</td>
</tr>
<tr>
<td>boolean isAttached(Appender appender)</td>
<td>Returns true if the specified appender is in list of attached, false otherwise.</td>
</tr>
<tr>
<td>void removeAllAppenders()</td>
<td>Remove all previously added appenders.</td>
</tr>
<tr>
<td>void removeAppender(Appender appender)</td>
<td>Remove the appender passed as parameter from the list of appenders.</td>
</tr>
<tr>
<td>void removeAppender(String name)</td>
<td>Remove the appender with the name passed as parameter from the list of appenders.</td>
</tr>
</tbody>
</table>
Method Detail

addAppender

```java
public void addAppender(Appender newAppender)
```

Add an appender.

getAllAppenders

```java
public Enumeration getAllAppenders()
```

Get all previously added appenders as an Enumeration.

getAppender

```java
public Appender getAppender(String name)
```

Get an appender by name.

isAttached

```java
public boolean isAttached(Appender appender)
```

Returns true if the specified appender is in list of attached, false otherwise.

Since: 1.2

removeAllAppenders

```java
public void removeAllAppenders()
```
Remove all previously added appenders.

**removeAppender**

```java
public void removeAppender(Appender appender)
```

Remove the appender passed as parameter from the list of appenders.

**removeAppender**

```java
public void removeAppender(String name)
```

Remove the appender with the name passed as parameter from the list of appenders.

Copyright 2000-2005 Apache Software Foundation.
Interface Configurator

All Known Implementing Classes:
- DefaultLF5Configurator
- PropertyConfigurator
- ReloadingPropertyConfigurator
- DOMConfigurator

public interface Configurator

Implemented by classes capable of configuring log4j using a URL.

Since: 1.0
Author: Anders Kristensen

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td>INHERITED</td>
<td>Special level value signifying inherited behaviour.</td>
</tr>
<tr>
<td>static String</td>
<td>NULL</td>
<td>Special level signifying inherited behaviour, same as INHERITED.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>doConfigure(URL url, LoggerRepository repository)</td>
<td>Interpret a resource pointed by a URL and set up log4j accordingly.</td>
</tr>
</tbody>
</table>

Field Detail

INHERITED
public static final String INHERITED

Special level value signifying inherited behaviour. The current value of this string constant is inherited. NULL is a synonym.

---

NULL

public static final String NULL

Special level signifying inherited behaviour, same as INHERITED. The current value of this string constant is null.

---

Method Detail

doConfigure

public void doConfigure(URL url, LoggerRepository repository)

Interpret a resource pointed by a URL and set up log4j accordingly. The configuration is done relative to the hierarchy parameter.

Parameters:
url - The URL to parse
repository - The hierarchy to operation upon.
<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: INNER | FIELD | CONSTR | METHOD  |

FRAMES  | NO FRAMES  |
DETAIL: FIELD | CONSTR | METHOD  |
public interface ErrorCode

This interface defines commonly encountered error codes.

Since:
0.9.0

Author:
Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>static int</th>
<th>ADDRESS_PARSE_FAILURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
<td>CLOSE_FAILURE</td>
</tr>
<tr>
<td>static int</td>
<td>FILE_OPEN_FAILURE</td>
</tr>
<tr>
<td>static int</td>
<td>FLUSH_FAILURE</td>
</tr>
<tr>
<td>static int</td>
<td>GENERIC_FAILURE</td>
</tr>
<tr>
<td>static int</td>
<td>MISSING_LAYOUT</td>
</tr>
<tr>
<td>static int</td>
<td>WRITE_FAILURE</td>
</tr>
</tbody>
</table>

Field Detail

GENERIC_FAILURE
public static final int GENERIC_FAILURE

WRITE_FAILURE

public static final int WRITE_FAILURE

FLUSH_FAILURE

public static final int FLUSH_FAILURE

CLOSE_FAILURE

public static final int CLOSE_FAILURE

FILE_OPEN_FAILURE

public static final int FILE_OPEN_FAILURE

MISSING_LAYOUT

public static final int MISSING_LAYOUT

ADDRESS_PARSE_FAILURE

public static final int ADDRESS_PARSE_FAILURE

Copyright 2000-2005 Apache Software Foundation.
public interface `ErrorHandler` extends `OptionHandler`

Appenders may delegate their error handling to `ErrorHandler`

Error handling is a particularly tedious to get right because by definition errors are hard to predict and to reproduce.

Please take the time to contact the author in case you discover that errors are not properly handled. You are most welcome to suggest new error handling policies or criticize existing policies.

**Author:**
Ceki Gülcü

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>error(String message)</code></td>
</tr>
<tr>
<td>This method is normally used to just print the error message passed as a parameter.</td>
</tr>
<tr>
<td>void <code>error(String message, Exception e, int errorCode)</code></td>
</tr>
<tr>
<td>Equivalent to the <code>error(String, Exception, int, LoggingEvent event)</code> with the the event parameter set to null.</td>
</tr>
<tr>
<td>void <code>error(String message, Exception e, int errorCode, LoggingEvent event)</code></td>
</tr>
<tr>
<td>This method is invoked to handle the error.</td>
</tr>
<tr>
<td>void <code>setAppender(Appender appender)</code></td>
</tr>
</tbody>
</table>
Set the appender for which errors are handled.

```java
void setBackupAppender(Appender appender)
```
Set the appender to falkback upon in case of failure.

```java
void setLogger(Logger logger)
```
Add a reference to a logger to which the failing appender might be attached to.

Methods inherited from interface org.apache.log4j.spi.OptionHandler

```java
activateOptions
```

## Method Detail

### setLogger

```java
class MyHandler implements OptionHandler {
    public void setLogger(Logger logger) {
        // Add a reference to a logger to which the failing appender might be attached to. The failing appender will be searched and replaced only in the loggers you add through this method.
        // Parameters:
        // logger - One of the loggers that will be searched for the failing appender in view of replacement.
        // Since:
        // 1.2
    }
}
```

### error

```java
public void error(String message, Exception e, int errorCode)
```
Equivalent to the `error(String, Exception, int, LoggingEvent event)` with the event parameter set to null.
error

public void error(String message)

This method is normally used to just print the error message passed as a parameter.

error

public void error(String message,
               Exception e,
               int errorCode,
               LoggingEvent event)

This method is invoked to handle the error.

Parameters:

message - The message associated with the error.
e - The Exception that was thrown when the error occurred.
errorCode - The error code associated with the error.
event - The logging event that the failing appender is asked to log.

Since: 1.2

setAppender

public void setAppender(Appender appender)

Set the appender for which errors are handled. This method is usually called when the error handler is configured.

Since: 1.2

setBackupAppender

public void setBackupAppender(Appender appender)
Set the appender to fallback upon in case of failure.

**Since:**

1.2
public interface HierarchyEventListener

Listen to events occurring within a Hierarchy.

Since: 1.2
Author: Ceki Gülcü

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAppenderEvent (Category cat, Appender appender)</td>
<td></td>
</tr>
<tr>
<td>void removeAppenderEvent (Category cat, Appender appender)</td>
<td></td>
</tr>
</tbody>
</table>

Method Detail

addAppenderEvent

public void addAppenderEvent (Category cat, Appender appender)

removeAppenderEvent

public void removeAppenderEvent (Category cat, Appender appender)
Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>SUMMARY:</th>
<th>INNER</th>
<th>FIELD</th>
<th>CONSTR</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETAIL:</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FRAMES</th>
<th>NO FRAMES</th>
</tr>
</thead>
</table>

public interface LoggerFactory

Implement this interface to create new instances of Logger or a sub-class of Logger.

See examples/subclass/MyLogger.java for an example.

Since:
    version 0.8.5

Author:
    Ceki Gülcü

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoggerFactory</td>
</tr>
<tr>
<td>makeNewLoggerInstance(String name)</td>
</tr>
</tbody>
</table>

Method Detail

makeNewLoggerInstance

public Logger makeNewLoggerInstance(String name)
public interface LoggerRepository

A LoggerRepository is used to create and retrieve Loggers. The relation between loggers in a repository depends on the repository but typically loggers are arranged in a named hierarchy.

In addition to the creational methods, a LoggerRepository can be queried for existing loggers, can act as a point of registry for events related to loggers.

Since: 1.2
Author: Ceki Gülcü

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addHierarchyEventListener(HierarchyEventListener listener)</td>
<td>Add a HierarchyEventListener event to the repository.</td>
</tr>
<tr>
<td>void emitNoAppenderWarning(Category cat)</td>
<td></td>
</tr>
<tr>
<td>String exists(String name)</td>
<td></td>
</tr>
<tr>
<td>void fireAddAppenderEvent(Category logger, Appender appender)</td>
<td></td>
</tr>
<tr>
<td>Enumeration getCurrentCategories()</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>Enumeration getCurrentLoggers()</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Logger.getLogger(String name)</code></td>
<td>Returns a named logger.</td>
</tr>
<tr>
<td><code>Logger.getLogger(String name, LoggerFactory factory)</code></td>
<td>Returns a named logger using a factory.</td>
</tr>
<tr>
<td><code>Logger.getRootLogger()</code></td>
<td>Returns the root logger.</td>
</tr>
<tr>
<td><code>Level.getThreshold()</code></td>
<td>Get the repository-wide threshold.</td>
</tr>
<tr>
<td><code>boolean isDisabled(int level)</code></td>
<td>Returns whether this repository is disabled for a given level.</td>
</tr>
<tr>
<td><code>void resetConfiguration()</code></td>
<td>Reset the configuration.</td>
</tr>
<tr>
<td><code>void setThreshold(Level level)</code></td>
<td>Set the repository-wide threshold.</td>
</tr>
<tr>
<td><code>void setThreshold(String val)</code></td>
<td>Another form of <code>setThreshold(Level)</code> accepting a string parameter instead of a Level.</td>
</tr>
<tr>
<td><code>void shutdown()</code></td>
<td></td>
</tr>
</tbody>
</table>

**Method Detail**

**addHierarchyEventListener**

```java
public void addHierarchyEventListener(HierarchyEventListener listener)
```

Add a `HierarchyEventListener` event to the repository.

**isDisabled**

```java
public boolean isDisabled(int level)
```

Returns whether this repository is disabled for a given level. The
answer depends on the repository threshold and the level parameter. See also setThreshold(org.apache.log4j.Level) method.

---

**setThreshold**

public void setThreshold(Level level)

Set the repository-wide threshold. All logging requests below the threshold are immediately dropped. By default, the threshold is set to Level.ALL which has the lowest possible rank.

---

**setThreshold**

public void setThreshold(String val)

Another form of setThreshold(Level) accepting a string parameter instead of a Level.

---

**emitNoAppenderWarning**

public void emitNoAppenderWarning(Category cat)

---

**getThreshold**

public Level getThreshold()

Get the repository-wide threshold. See setThreshold(Level) for an explanation.

---

**getLogger**

public Logger getLogger(String name)
getLogger

public Logger getLogger(String name, LoggerFactory factory)

getRootLogger

public Logger getRootLogger()

exists

public Logger exists(String name)

shutdown

public void shutdown()

gGetCurrentLoggers

public Enumeration getCurrentLoggers()

gGetCurrentCategories

public Enumeration getCurrentCategories()

   Deprecated. Please use getCurrentLoggers() instead.

fireAddAppenderEvent

public void fireAddAppenderEvent(Category logger, Appender appender)
resetConfiguration

public void resetConfiguration()
Interface OptionHandler

All Known Subinterfaces:
   ErrorHandler

All Known Implementing Classes:
   AppenderSkeleton, Filter, Layout

public interface OptionHandler

A string based interface to configure package components.

Since: 0.8.1
Author: Ceki Gülcü, Anders Kristensen

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activateOptions</td>
<td>void activateOptions()</td>
<td>Activate the options that were previously set with calls to option setters.</td>
</tr>
</tbody>
</table>

Method Detail

activateOptions

public void activateOptions()

Activate the options that were previously set with calls to option setters.

This allows to defer activation of the options until all options have been set. This is required for components which have related options
that remain ambiguous until all are set.

For example, the FileAppender has the File and Append options both of which are ambiguous until the other is also set.

Copyright 2000-2005 Apache Software Foundation.
## Interface RendererSupport

### All Known Implementing Classes:
- [Hierarchy](#)

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>RendererMap</td>
</tr>
<tr>
<td><strong>getRendererMap</strong>()</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td><strong>setRenderer</strong>(Class renderedClass, ObjectRenderer renderer)</td>
</tr>
</tbody>
</table>

### Method Detail

**getRendererMap**

```java
public RendererMap getRendererMap()
```

**setRenderer**

```java
public void setRenderer(Class renderedClass, ObjectRenderer renderer)
```
public interface RepositorySelector

The LogManager uses one (and only one) RepositorySelector implementation to select the LoggerRepository for a particular application context.

It is the responsibility of the RepositorySelector implementation to track the application context. Log4j makes no assumptions about the application context or on its management.

See also LogManager.

Since: 1.2
Author: Ceki Gülcü

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getLoggerRepository()</td>
<td>Returns a LoggerRepository depending on the context.</td>
</tr>
</tbody>
</table>

Method Detail

getLoggerRepository

public LoggerRepository getLoggerRepository()
Returns a LoggerRepository depending on the context. Implementors must make sure that a valid (non-null) LoggerRepository is returned.
public interface TriggeringEventEvaluator

Implementations of this interface allow certain appenders to decide when to perform an appender specific action.

For example the SMTPAppender sends an email when the isTriggeringEvent(org.apache.log4j.spi.LoggingEvent) method returns true and adds the event to an internal buffer when the returned result is false.

Since: version 1.0

Author: Ceki Gülcü

### Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>isTriggeringEvent(LoggingEvent event)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Is this the triggering event?</td>
</tr>
</tbody>
</table>

### Method Detail

isTriggeringEvent

public boolean isTriggeringEvent(LoggingEvent event)

  Is this the triggering event?
Copyright 2000-2005 Apache Software Foundation.
**Class DefaultRepositorySelector**

java.lang.Object  
  +--org.apache.log4j.spi.DefaultRepositorySelector

All Implemented Interfaces:  
RepositorySelector

public class DefaultRepositorySelector  
extends Object  
implements RepositorySelector

---

**Constructor Summary**

| DefaultRepositorySelector | (LoggerRepository repository) |

---

**Method Summary**

<table>
<thead>
<tr>
<th>LoggerRepository</th>
<th>getRepository()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a LoggerRepository depending on the context.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object  
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

**Constructor Detail**

DefaultRepositorySelector

public DefaultRepositorySelector(LoggerRepository repository)
Method Detail

getLoggerRepository

public LoggerRepository getLoggerRepository()

Description copied from interface: RepositorySelector

Returns a LoggerRepository depending on the context. Implementors must make sure that a valid (non-null) LoggerRepository is returned.

Specified by:

getLoggerRepository in interface RepositorySelector

Overview Package Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONST | METHOD
DETAIL: FIELD | CONST | METHOD

Copyright 2000-2005 Apache Software Foundation.
org.apache.log4j.spi Class Filter

java.lang.Object

+--- org.apache.log4j.spi.Filter

All Implemented Interfaces:
  OptionHandler

Direct Known Subclasses:
  DenyAllFilter, LevelMatchFilter, LevelRangeFilter, StringMatchFilter

public abstract class Filter
  extends Object
  implements OptionHandler

Users should extend this class to implement customized logging event filtering. Note that Category and AppenderSkeleton, the parent class of all standard appenders, have built-in filtering rules. It is suggested that you first use and understand the built-in rules before rushing to write your own custom filters.

This abstract class assumes and also imposes that filters be organized in a linear chain. The decide(LoggingEvent) method of each filter is called sequentially, in the order of their addition to the chain.

The decide(LoggingEvent) method must return one of the integer constants DENY, NEUTRAL or ACCEPT.

If the value DENY is returned, then the log event is dropped immediately without consulting with the remaining filters.

If the value NEUTRAL is returned, then the next filter in the chain is consulted. If there are no more filters in the chain, then the log event is logged. Thus, in the presence of no filters, the default behaviour is to log all logging events.

If the value ACCEPT is returned, then the log event is logged without
consulting the remaining filters.

The philosophy of log4j filters is largely inspired from the Linux ipchains.

Note that filtering is only supported by the DOMConfigurator. The PropertyConfigurator does not support filters.

Since: 0.9.0
Author: Ceki Gülcü

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static int</strong> ACCEPT</td>
</tr>
<tr>
<td><strong>static int</strong> DENY</td>
</tr>
<tr>
<td><strong>static int</strong> NEUTRAL</td>
</tr>
<tr>
<td><strong>next</strong> Filter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Filter()</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void activateOptions()</strong></td>
</tr>
<tr>
<td><strong>abstract decide(LoggingEvent event)</strong></td>
</tr>
</tbody>
</table>
If the decision is **DENY**, then the event will be dropped.

<table>
<thead>
<tr>
<th>Filter</th>
<th>getNext()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Return the pointer to the next filter;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setNext(Filter next)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the next filter pointer.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.**Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

**Field Detail**

next

**public** Filter next

** Deprecated. As of 1.2.12, use getNext() and setNext(org.apache.log4j.spi.Filter) instead**

Points to the next filter in the filter chain.

---

**DENY**

**public** static final int **DENY**

The log event must be dropped immediately without consulting with the remaining filters, if any, in the chain.

---

**NEUTRAL**

**public** static final int **NEUTRAL**

This filter is neutral with respect to the log event. The remaining filters, if any, should be consulted for a final decision.
ACCEPT

public static final int ACCEPT

    The log event must be logged immediately without consulting with the remaining filters, if any, in the chain.

Constructor Detail

Filter

public Filter()

Method Detail

activateOptions

public void activateOptions()  

    Usually filters options become active when set. We provide a default do-nothing implementation for convenience.  

    Specified by: activateOptions in interface OptionHandler

decide

public abstract int decide(LoggingEvent event)

    If the decision is DENY, then the event will be dropped. If the decision is NEUTRAL, then the next filter, if any, will be invoked. If the decision is ACCEPT then the event will be logged without consulting with other filters in the chain.

Parameters:
    event - The LoggingEvent to decide upon.
Returns:

decision The decision of the filter.

---

**setNext**

public void setNext(Filter next)

Set the next filter pointer.

---

**getNext**

public Filter getNext()

Return the pointer to the next filter;

---

Copyright 2000-2005 Apache Software Foundation.
public class LocationInfo extends Object implements Serializable

The internal representation of caller location information.

Since: 0.8.3

See Also: Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>String</th>
<th>fullInfo</th>
<th>All available caller information, in the format fully.qualified.className.of.caller.methodName(Filename.java:line)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>String</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When location information is not available the constant NA is returned.</td>
</tr>
</tbody>
</table>

Constructor Summary

LocationInfo(Throwable t, String fqnOfCallingClass)

Instantiate location information based on a Throwable.

Method Summary
String **getClassName()**
Return the fully qualified class name of the caller making the logging request.

String **getFileName()**
Return the file name of the caller.

String **getLineNumber()**
Returns the line number of the caller.

String **getMethodName()**
Returns the method name of the caller.

Methods inherited from class java.lang.**Object**
close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

**fullInfo**

```
public String fullInfo
All available caller information, in the format
fully.qualified.classname.of.caller.methodName(Filename.java:line)
```

**NA**

```
public static final String NA
When location information is not available the constant NA is returned. Current value of this string constant is ?.
```

Constructor Detail

**LocationInfo**
public LocationInfo(Throwable t, String fqnOfCallingClass)

Instantiate location information based on a Throwable. We expect the Throwable t, to be in the format

java.lang.Throwable
...
at org.apache.log4j.PatternLayout.format(PatternLayout.java:413)
at org.apache.log4j.FileAppender.doAppend(FileAppender.java:183)
at org.apache.log4j.Category.callAppenders(Category.java:131)
at org.apache.log4j.Category.log(Category.java:512)
at callers.fully.qualified.className.methodName(FileName.java:74)
...

However, we can also deal with JIT compilers that "lose" the location information, especially between the parentheses.

**Method Detail**

**getClassName**

public String getClassName()

Return the fully qualified class name of the caller making the logging request.

**getFileName**

public String getFileName()

Return the file name of the caller.

This information is not always available.

**getLineNumber**

public String getLineNumber()
Returns the line number of the caller.

This information is not always available.

---

**getMethodName**

```java
public String getMethodName()
```

Returns the method name of the caller.

---

Copyright 2000-2005 Apache Software Foundation.
public class LoggingEvent
extends Object
implements Serializable

The internal representation of logging events. When an affirmative
decision is made to log then a LoggingEvent instance is created. This
instance is passed around to the different log4j components.

This class is of concern to those wishing to extend log4j.

Since: 0.8.2
Author: Ceki Gülcü, James P. Cakalic
See Also: Serialized Form

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>categoryName</td>
<td>String</td>
<td>Deprecated. This field will be marked as private in future releases. Please do not access it directly. Use the getLogName() method instead.</td>
</tr>
<tr>
<td>fqnOfCategoryClass</td>
<td>String</td>
<td>Fully qualified name of the calling category class.</td>
</tr>
<tr>
<td>level</td>
<td>Priority</td>
<td>Deprecated. This field will be marked as private in future releases.</td>
</tr>
</tbody>
</table>
releases. Please do not access it directly. Use the `getLevel()` method instead.

| `long timeStamp` | The number of milliseconds elapsed from 1/1/1970 until logging event was created. |

## Constructor Summary

**LoggingEvent** *(String fqnOfCategoryClass, Category logger, long timeStamp, Priority level, Object message, Throwable throwable)*

- Instantiate a LoggingEvent from the supplied parameters.

**LoggingEvent** *(String fqnOfCategoryClass, Category logger, Priority level, Object message, Throwable throwable)*

- Instantiate a LoggingEvent from the supplied parameters.

## Method Summary

<table>
<thead>
<tr>
<th>LocationInfo</th>
<th><code>getLevel()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level</strong></td>
<td>Return the level of this event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LocationInfo</th>
<th><code>getLocatinInformation()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>String</strong></td>
<td>Set the location information for this logging event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>getLoggerName()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>String</strong></td>
<td>Return the name of the logger.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object</th>
<th><code>getMDC(String key)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td>Returns the the context corresponding to the key parameter.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><code>getMDCCopy()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong></td>
<td>Obtain a copy of this thread's MDC prior to serialization or asynchronous logging.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object</th>
<th><code>getMessage()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td>Return the message for this logging event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>getNDC()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>String</strong></td>
<td>This method returns the NDC for this event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>getRenderedMessage()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>String</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Field Detail

#### fqnOfCategoryClass

`public final transient String fqnOfCategoryClass`

Fully qualified name of the calling category class.

#### categoryName

`public final String categoryName`

**Deprecated.** This field will be marked as private in future releases. Please do not access it directly. Use the `getLoggerName()` method instead.

The category (logger) name.
**level**

public transient `Priority` level

**Deprecation.** This field will be marked as private in future releases. Please do not access it directly. Use the `getLevel()` method instead.

Level of logging event. Level cannot be serializable because it is a flyweight. Due to its special serialization it cannot be declared final either.

This field should not be accessed directly. You should use the `getLevel()` method instead.

**timeStamp**

public final long `timeStamp`

The number of milliseconds elapsed from 1/1/1970 until logging event was created.

### Constructor Detail

**LoggingEvent**

public `LoggingEvent`(`String` fqnOfCategoryClass, `Category` logger, `Priority` level, `Object` message, `Throwable` throwable)

Instantiate a `LoggingEvent` from the supplied parameters.

Except `timeStamp` all the other fields of `LoggingEvent` are filled when actually needed.

**Parameters:**

- `logger` - The logger generating this event.
- `level` - The level of this event.
message - The message of this event.
throwable - The throwable of this event.

LoggingEvent

public LoggingEvent(String fqnOfCategoryClass, Category logger, long timeStam, Priority level, Object message, Throwable throwable)

Instantiate a LoggingEvent from the supplied parameters.

Except timeStamp all the other fields of LoggingEvent are filled when actually needed.

Parameters:
  logger - The logger generating this event.
  timeStamp - the timestamp of this logging event
  level - The level of this event.
  message - The message of this event.
  throwable - The throwable of this event.

Method Detail

getLocationInformation

public LocationInfo getLocationInformation()

Set the location information for this logging event. The collected information is cached for future use.

getLevel

public Level getLevel()

Return the level of this event. Use this form instead of directly
accessing the level field.

getLoggerName

public String getLoggerName()

Return the name of the logger. Use this form instead of directly accessing the categoryName field.

getMessage

public Object getMessage()

Return the message for this logging event.

Before serialization, the returned object is the message passed by the user to generate the logging event. After serialization, the returned value equals the String form of the message possibly after object rendering.

Since:
1.1

getNDC

public String getNDC()

This method returns the NDC for this event. It will return the correct content even if the event was generated in a different thread or even on a different machine. The NDC.get() method should never be called directly.

getMDC

public Object getMDC(String key)
Returns the context corresponding to the key parameter. If there is a local MDC copy, possibly because we are in a logging server or running inside AsyncAppender, then we search for the key in MDC copy, if a value is found it is returned. Otherwise, if the search in MDC copy returns a null result, then the current thread's MDC is used.

Note that both the local MDC copy and the current thread's MDC are searched.

---

**getMDCCopy**

default: public void getMDCCopy()

Obtain a copy of this thread's MDC prior to serialization or asynchronous logging.

---

**getRenderedMessage**

default: public String getRenderedMessage()

---

**getStartTime**

default: public static long getStartTime()

Returns the time when the application started, in milliseconds elapsed since 01.01.1970.

---

**getThreadName**

default: public String getThreadName()

---

**getThrowableInformation**

default: public ThrowableInformation getThrowableInformation()
Returns the throwable information contained within this event. May be null if there is no such information.

Note that the Throwable object contained within a ThrowableInformation does not survive serialization.

Since:
1.1

getThrowableStrRep

public String[] getThrowableStrRep()

Return this event's throwable's string[] representation.
Class RootCategory

java.lang.Object
    +- org.apache.log4j.Category
        +- org.apache.log4j.Logger
            +- org.apache.log4j.spi.RootCategory

All Implemented Interfaces:
    AppenderAttachable

Deprecated. Replaced by RootLogger.

public final class RootCategory
extends Logger

Fields inherited from class org.apache.log4j.Category
additive, level, name, parent, repository, resourceBundle

Constructor Summary

RootCategory(Level level)
    Deprecated. The root category names itself as "root".

Method Summary

Level getChainedLevel()
    Deprecated. Return the assigned level value without walking the category hierarchy.

void setLevel(Level level)
    Deprecated. Setting a null value to the level of the root category may have catastrophic results.

void setPriority(Level level)
    Deprecated.
**Constructor Detail**

**RootCategory**

public `RootCategory`(**Level** level)

**Deprecated.**

The root category names itself as "root". However, the root category cannot be retrieved by name.

**Method Detail**

**getChainedLevel**

public final `Level` getChainedLevel()
Deprecated.
Return the assigned level value without walking the category hierarchy.

**setLevel**

```java
public final void setLevel(Level level)
```

**Deprecated.**
Setting a null value to the level of the root category may have catastrophic results. We prevent this here.

**Overrides:**
`setLevel` in class `Category`

**Since:**
0.8.3

**setPriority**

```java
public final void setPriority(Level level)
```

**Deprecated.**
org.apache.log4j.spi Class RootLogger

java.lang.Object
|  +-- org.apache.log4j.Category
|  |  +-- org.apache.log4j.Logger
|  |  |  +-- org.apache.log4j.spi.RootLogger

All Implemented Interfaces:
   AppenderAttachable

public final class RootLogger
extends Logger

RootLogger sits at the top of the logger hierarchy. It is a regular logger except that it provides several guarantees.

First, it cannot be assigned a null level. Second, since root logger cannot have a parent, the getChainedLevel() method always returns the value of the level field without walking the hierarchy.

Author:
   Ceki Gülcü

Fields inherited from class org.apache.log4j.Category
additive, level, name, parent, repository, resourceBundle

Constructor Summary

RootLogger(<Level> level)
   The root logger names itself as "root".

Method Summary
### Constructor Detail

**RootLogger**

```java
public RootLogger( Level level )
```

The root logger names itself as "root". However, the root logger cannot be retrieved by name.

### Method Detail

**getChainedLevel**

- **Return the assigned level value without walking the logger hierarchy.**

**setLevel**

- **Setting a null value to the level of the root logger may have catastrophic results.**
getChainedLevel

public final Level getChainedLevel()

Return the assigned level value without walking the logger hierarchy.

setLevel

public final void setLevel(Level level)

Setting a null value to the level of the root logger may have catastrophic results. We prevent this here.

Overrides:
    setLevel in class Category

Since:
    0.8.3
Class ThrowableInformation

All Implemented Interfaces:
    Serializable

public class ThrowableInformation extends Object
    implements Serializable

ThrowableInformation is log4j's internal representation of throwables. It essentially consists of a string array, called 'rep', where the first element, that is rep[0], represents the string representation of the throwable (i.e. the value you get when you do throwable.toString()) and subsequent elements correspond the stack trace with the top most entry of the stack corresponding to the second entry of the 'rep' array that is rep[1].

Author:
    Ceki Gülcü

See Also:
    Serialized Form

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThrowableInformation(Throwables throwable)</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throwable getThrowable()</td>
</tr>
<tr>
<td>String[] getThrowableStrRep()</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructor Detail

ThrowableInformation

public ThrowableInformation(Throwables throwable)

Method Detail

getThrowable

public Throwables getThrowable()

getThrowableStrRep

public String[] getThrowableStrRep()
Package org.apache.log4j.varia

Contains various appenders, filters and other odds and ends.

See:   [Description](#)

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DenyAllFilter</strong></td>
</tr>
<tr>
<td><strong>ExternallyRolledFileAppender</strong></td>
</tr>
<tr>
<td><strong>FallbackErrorHandler</strong></td>
</tr>
<tr>
<td><strong>LevelMatchFilter</strong></td>
</tr>
<tr>
<td><strong>LevelRangeFilter</strong></td>
</tr>
<tr>
<td><strong>NullAppender</strong></td>
</tr>
<tr>
<td><strong>ReloadingPropertyConfigurator</strong></td>
</tr>
<tr>
<td><strong>Roller</strong></td>
</tr>
<tr>
<td><strong>StringMatchFilter</strong></td>
</tr>
</tbody>
</table>
Package org.apache.log4j.varia Description

Contains various appenders, filters and other odds and ends.

Last modified: Tue Mar 21 20:28:14 MET 2000

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.varia

Package Hierarchies:

All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.**AppenderSkeleton** (implements org.apache.log4j.**Appender**, org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.varia.**NullAppender**
    - class org.apache.log4j.**WriterAppender**
      - class org.apache.log4j.**FileAppender**
        - class org.apache.log4j.**RollingFileAppender**
    - class org.apache.log4j.varia.**ExternallyRolledFileAppender**
  - class org.apache.log4j.varia.**FallbackErrorHandler** (implements org.apache.log4j.spi.**ErrorHandler**)
  - class org.apache.log4j.spi.**Filter** (implements org.apache.log4j.spi.**OptionHandler**)
    - class org.apache.log4j.varia.**DenyAllFilter**
    - class org.apache.log4j.varia.**LevelMatchFilter**
    - class org.apache.log4j.varia.**LevelRangeFilter**
    - class org.apache.log4j.varia.**StringMatchFilter**
  - class org.apache.log4j.varia.**ReloadingPropertyConfigurator** (implements org.apache.log4j.spi.**Configurator**)
  - class org.apache.log4j.varia.**Roller**
Uses of Package
org.apache.log4j.varia

Packages that use org.apache.log4j.varia

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
</tbody>
</table>

Classes in org.apache.log4j.varia used by org.apache.log4j.varia

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NullAppender</td>
<td>A NullAppender merely exists, it never outputs a message to any device.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
public class DenyAllFilter extends Filter

This filter drops all logging events.

You can add this filter to the end of a filter chain to switch from the default "accept all unless instructed otherwise" filtering behaviour to a "deny all unless instructed otherwise" behaviour.

Since:
0.9.0

Author:
Ceki Gülcü
Always returns the integer constant `Filter.DENY` regardless of the `LoggingEvent` parameter.

```java
String[] getOptionStrings()
```

**Deprecated.** We now use JavaBeans introspection to configure components. Options strings are no longer needed.

```java
void setOption(String key, String value)
```

**Deprecated.** Use the setter method for the option directly instead of the generic `setOption` method.

Methods inherited from class `org.apache.log4j.spi.Filter`:
- `activateOptions`, `getNext`, `setNext`

Methods inherited from class `java.lang.Object`:
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Constructor Detail

**DenyAllFilter**

```java
public DenyAllFilter()
```

### Method Detail

**getOptionStrings**

```java
public String[] getOptionStrings()
```

**Deprecated.** We now use JavaBeans introspection to configure components. Options strings are no longer needed.

Returns `null` as there are no options.
setOption

public void setOption(String key, String value)

**Deprecated.** Use the setter method for the option directly instead of the generic setOption method.

No options to set.

decide

public int decide(LoggingEvent event)

Always returns the integer constant Filter.DENY regardless of the LoggingEvent parameter.

**Overrides:**

decide in class Filter

**Parameters:**

event - The LoggingEvent to filter.

**Returns:**

Always returns Filter.DENY.
public class **ExternallyRolledFileAppender**
extends **RollingFileAppender**

This appender listens on a socket on the port specified by the **Port** property for a "RollOver" message. When such a message is received, the underlying log file is rolled over and an acknowledgment message is sent back to the process initiating the roll over.

This method of triggering roll over has the advantage of being operating system independent, fast and reliable.

A simple application **Roller** is provided to initiate the roll over.

Note that the initiator is not authenticated. Anyone can trigger a rollover. In production environments, it is recommended that you add some form of protection to prevent undesired rollovers.

**Since:**
version 0.9.0

**Author:**
Ceki Gülcü
Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String OK</td>
<td>The string constant sent to acknowledge a roll over.</td>
</tr>
<tr>
<td>static String ROLL_OVER</td>
<td>The string constant sent to initiate a roll over.</td>
</tr>
</tbody>
</table>

Fields inherited from class org.apache.log4j.RollingFileAppender
maxBackupIndex, maxFileSize

Fields inherited from class org.apache.log4j.FileAppender
bufferedIO, bufferSize, fileAppend, fileName

Fields inherited from class org.apache.log4j.WriterAppender
encoding, immediateFlush, qw

Fields inherited from class org.apache.log4j.AppenderSkeleton
closed, errorHandler, headFilter, layout, name, tailFilter, threshold

Constructor Summary

ExternallyRolledFileAppender()  
The default constructor does nothing but calls its super-class constructor.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void activateOptions()</td>
<td>Start listening on the port specified by a preceding call to setPort(int).</td>
</tr>
<tr>
<td>int getPort()</td>
<td>Returns value of the Port option.</td>
</tr>
<tr>
<td>void setPort(int port)</td>
<td>The Port property is used for setting the port for listening to</td>
</tr>
</tbody>
</table>
Methods inherited from class org.apache.log4j.RollingFileAppender

getMaxBackupIndex, getMaximumFileSize, rollOver, setFile, setMaxBackupIndex, setMaxFileSize, setMaximumFileSize, setQWForFiles, subAppend

Methods inherited from class org.apache.log4j.FileAppender

closeFile, getAppend, getBufferedIO, getBufferSize, getFile, reset, setAppend, setBufferedIO, setBufferSize, setFile

Methods inherited from class org.apache.log4j.WriterAppender

append, checkEntryConditions, close, closeWriter, createWriter, getEncoding, getImmediateFlush, requiresLayout, setEncoding, setErrorHandler, setImmediateFlush, setWriter, writeFooter, writeHeader

Methods inherited from class org.apache.log4j.AppenderSkeleton

addFilter, clearFilters, doAppend, finalize, getErrorHandler, getFilter, getFirstFilter, getLayout, getName, getThreshold, isAsSevereAsThreshold, setLayout, setName, setThreshold

Methods inherited from class java.lang.Object

close, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

ROLL_OVER

public static final String ROLL_OVER

The string constant sent to initiate a roll over. Current value of this string constant is RollOver.
OK

public static final String OK

The string constant sent to acknowledge a roll over. Current value of this string constant is OK.

Constructor Detail

ExternallyRolledFileAppender

public ExternallyRolledFileAppender()

The default constructor does nothing but calls its super-class constructor.

Method Detail

setPort

public void setPort(int port)

The Port [roperty is used for setting the port for listening to external roll over messages.

getPort

public int getPort()

Returns value of the Port option.

activateOptions

public void activateOptions()

Start listening on the port specified by a preceding call to
**setPort(int).**

**Overrides:**

activateOptions in class FileAppender
<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: INNER</td>
<td>FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>
Class FallbackErrorHandler

public class FallbackErrorHandler extends Object
implements ErrorHandler

The FallbackErrorHandler implements the ErrorHandler interface such that a secondary appender may be specified. This secondary appender takes over if the primary appender fails for whatever reason.

The error message is printed on System.err, and logged in the new secondary appender.

Author:
Ceki Gücü

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>FallbackErrorHandler()</td>
<td>void activateOptions()</td>
</tr>
<tr>
<td></td>
<td>No options to activate.</td>
</tr>
<tr>
<td></td>
<td>void error(String message)</td>
</tr>
<tr>
<td></td>
<td>Print a the error message passed as parameter on System.err.</td>
</tr>
<tr>
<td>Method</td>
<td>Signature</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>void</td>
<td><code>error(String message, Exception e, int errorCode)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>error(String message, Exception e, int errorCode, LoggingEvent event)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setAppender(Appender primary)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setBackupAppender(Appender backup)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setLogger(Logger logger)</code></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object:
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

**FallbackErrorHandler**

public `FallbackErrorHandler()`

### Method Detail

**setLogger**

public void `setLogger(Logger logger)`

*Adds* the logger passed as parameter to the list of loggers that we need to search for in case of appender failure.

**Specified by:**

setLogger in interface ErrorHandler
Following copied from interface: org.apache.log4j.spi.ErrorHandler

Parameters:
logger - One of the loggers that will be searched for the failing appender in view of replacement.

activateOptions

public void activateOptions()

No options to activate.
Specified by:
activateOptions in interface OptionHandler

error

public void error(String message, Exception e, int errorCode)

Prints the message and the stack trace of the exception on System.err.
Specified by:
error in interface ErrorHandler

error

public void error(String message, Exception e, int errorCode, LoggingEvent event)

Prints the message and the stack trace of the exception on System.err.
Specified by:
error in interface ErrorHandler

Following copied from interface: org.apache.log4j.spi.ErrorHandler

Parameters:
message - The message associated with the error.
e - The Exception that was thrown when the error occurred.
errorCode - The error code associated with the error.
event - The logging event that the failing appender is asked to log.

---

**error**

```java
public void error(String message)
```

Print a the error message passed as parameter on System.err.

**Specified by:**

`error` in interface `ErrorHandler`

---

**setAppender**

```java
public void setAppender(Appender primary)
```

The appender to which this error handler is attached.

**Specified by:**

`setAppender` in interface `ErrorHandler`

---

**setBackupAppender**

```java
public void setBackupAppender(Appender backup)
```

Set the backup appender.

**Specified by:**

`setBackupAppender` in interface `ErrorHandler`

---

Copyright 2000-2005 Apache Software Foundation.
```
org.apache.log4j.varia Class LevelMatchFilter

java.lang.Object
   +-- org.apache.log4j.spi.Filter
       +-- org.apache.log4j.varia.LevelMatchFilter

All Implemented Interfaces:
   OptionHandler

public class LevelMatchFilter
extends Filter

This is a very simple filter based on level matching.

The filter admits two options LevelToMatch and AcceptOnMatch. If there is an exact match between the value of the LevelToMatch option and the level of the LoggingEvent, then the decide(org.apache.log4j.spi.LoggingEvent) method returns Filter.ACCEPT in case the AcceptOnMatch option value is set to true, if it is false then Filter.DENY is returned. If there is no match, Filter.NEUTRAL is returned.

Since: 1.2
Author: Ceki Gülcü
```

<table>
<thead>
<tr>
<th>Fields inherited from class org.apache.log4j.spi.Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT, DENY, NEUTRAL, next</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>LevelMatchFilter()</td>
</tr>
</tbody>
</table>
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int decide(LoggingEvent event)</code></td>
<td>Return the decision of this filter.</td>
</tr>
<tr>
<td><code>boolean getAcceptOnMatch()</code></td>
<td></td>
</tr>
<tr>
<td><code>String getLevelToMatch()</code></td>
<td></td>
</tr>
<tr>
<td><code>void setAcceptOnMatch(boolean acceptOnMatch)</code></td>
<td></td>
</tr>
<tr>
<td><code>void setLevelToMatch(String level)</code></td>
<td></td>
</tr>
</tbody>
</table>

### Methods inherited from class `org.apache.log4j.spi.Filter`<br>`activateOptions`, `getNext`, `setNext`

### Methods inherited from class `java.lang.Object`<br>`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
getLevelToMatch

public String getLevelToMatch()

setAcceptOnMatch

public void setAcceptOnMatch(boolean acceptOnMatch)

getAcceptOnMatch

public boolean getAcceptOnMatch()

decide

public int decide(LoggingEvent event)

Return the decision of this filter. Returns Filter.NEUTRAL if the LevelToMatch option is not set or if there is not match. Otherwise, if there is a match, then the returned decision is Filter.ACCEPT if the AcceptOnMatch property is set to true. The returned decision is Filter.DENY if the AcceptOnMatch property is set to false.

Overrides:
decide in class Filter
Following copied from class: org.apache.log4j.spi.Filter

Parameters:
event - The LoggingEvent to decide upon.

Returns:
decision The decision of the filter.
Class LevelRangeFilter

All Implemented Interfaces:
   OptionHandler

public class LevelRangeFilter
extends Filter

This is a very simple filter based on level matching, which can be used to reject messages with priorities outside a certain range.

The filter admits three options LevelMin, LevelMax and AcceptOnMatch.

If the level of the LoggingEvent is not between Min and Max (inclusive), then Filter.DENY is returned.

If the Logging event level is within the specified range, then if AcceptOnMatch is true, Filter.ACCEPT is returned, and if AcceptOnMatch is false, Filter.NEUTRAL is returned.

If LevelMin is not defined, then there is no minimum acceptable level (ie a level is never rejected for being too "low"/unimportant). If LevelMax is not defined, then there is no maximum acceptable level (ie a level is never rejected for being too "high"/important).

Refer to the setThreshold method available to all appenders extending AppenderSkeleton for a more convenient way to filter out events by level.

Author:
   Simon Kitching, based on code by Ceki Gülcü
### Constructor Summary

**LevelRangeFilter()**

### Method Summary

<table>
<thead>
<tr>
<th>return type</th>
<th>method name</th>
<th>parameters</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><strong>decide</strong> (LoggingEvent event)</td>
<td></td>
<td>Return the decision of this filter.</td>
</tr>
<tr>
<td>boolean</td>
<td><strong>getAcceptOnMatch</strong></td>
<td></td>
<td>Get the value of the AcceptOnMatch option.</td>
</tr>
<tr>
<td>Level</td>
<td><strong>getLevelMax</strong></td>
<td></td>
<td>Get the value of the LevelMax option.</td>
</tr>
<tr>
<td>Level</td>
<td><strong>getLevelMin</strong></td>
<td></td>
<td>Get the value of the LevelMin option.</td>
</tr>
<tr>
<td>void</td>
<td><strong>setAcceptOnMatch</strong></td>
<td>boolean</td>
<td>Set the AcceptOnMatch option.</td>
</tr>
<tr>
<td>void</td>
<td><strong>setLevelMax</strong> (Level levelMax)</td>
<td></td>
<td>Set the LevelMax option.</td>
</tr>
<tr>
<td>void</td>
<td><strong>setLevelMin</strong> (Level levelMin)</td>
<td></td>
<td>Set the LevelMin option.</td>
</tr>
</tbody>
</table>

### Methods inherited from class org.apache.log4j.spi.Filter

**activateOptions**, **getNext**, **setNext**

### Methods inherited from class java.lang.Object


### Constructor Detail
LevelRangeFilter

public LevelRangeFilter()

### Method Detail

**decide**

public int decide(LoggingEvent event)

Return the decision of this filter.

**Overrides:**
- `decide` in class Filter

**Parameters:**
- event - The LoggingEvent to decide upon.

**Returns:**
- decision The decision of the filter.

---

**getLevelMax**

public Level getLevelMax()

Get the value of the LevelMax option.

---

**getLevelMin**

public Level getLevelMin()

Get the value of the LevelMin option.

---

**getAcceptOnMatch**

public boolean getAcceptOnMatch()
Get the value of the `AcceptOnMatch` option.

---

**setLevelMax**

public void `setLevelMax`(`Level` levelMax)

Set the `LevelMax` option.

---

**setLevelMin**

public void `setLevelMin`(`Level` levelMin)

Set the `LevelMin` option.

---

**setAcceptOnMatch**

public void `setAcceptOnMatch`(`boolean` acceptOnMatch)

Set the `AcceptOnMatch` option.

---

Copyright 2000-2005 Apache Software Foundation.
nullappender

All Implemented Interfaces:
   Appender, OptionHandler

public class NullAppender
extends AppenderSkeleton

A NullAppender merely exists, it never outputs a message to any device.

Author:
  Ceki Gülc"
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void releaseResources()</code></td>
<td>Release any resources allocated within the appender such as file handles, network connections, etc.</td>
</tr>
<tr>
<td><code>void doAppend(LoggingEvent event)</code></td>
<td>Does not do anything.</td>
</tr>
<tr>
<td><code>NullAppender getInstance()</code></td>
<td>Whenever you can, use this method to retrieve an instance instead of instantiating a new one with <code>new</code>.</td>
</tr>
<tr>
<td><code>boolean requiresLayout()</code></td>
<td>NullAppenders do not need a layout.</td>
</tr>
</tbody>
</table>

### Methods inherited from class `org.apache.log4j.AppenderSkeleton`
- `addFilter`, `clearFilters`, `finalize`, `getErrorHandler`, `getFilter`, `getFirstFilter`, `getLayout`, `getName`, `getThreshold`, `isAsSevereAsThreshold`, `setErrorHandler`, `setLayout`, `setName`, `setThreshold`

### Methods inherited from class `java.lang.Object`
- `clone`, `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Constructor Detail

#### NullAppender

`public NullAppender()`

### Method Detail

#### activateOptions

`public void activateOptions()`

There are no options to activate.

**Overrides:**

`activateOptions` in class `AppenderSkeleton`
getInstance

public NullAppender getInstance()

Whenever you can, use this method to retrieve an instance instead of instantiating a new one with new.

close

public void close()

Description copied from interface: Appender
Release any resources allocated within the appender such as file handles, network connections, etc.

It is a programming error to append to a closed appender.

doAppend

public void doAppend(LoggingEvent event)

Does not do anything.
Overrides:
doAppend in class AppenderSkeleton

append

protected void append(LoggingEvent event)

Does not do anything.
Overrides:
append in class AppenderSkeleton

requiresLayout
public boolean requiresLayout()

NullAppenders do not need a layout.
Class

ReloadingPropertyConfigurator

java.lang.Object

+-org.apache.log4j.varia.ReloadingPropertyConfigurator

All Implemented Interfaces:

Configurator

public class ReloadingPropertyConfigurator extends Object

extends Object

implements Configurator

Fields inherited from interface org.apache.log4j.spi.Configurator

INHERITED, NULL

Constructor Summary

ReloadingPropertyConfigurator() 

Method Summary

void doConfigure(URL url, LoggerRepository repository)

Interpret a resource pointed by a URL and set up log4j accordingly.

Methods inherited from class java.lang.Object

class, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
Constructor Detail

ReloadingPropertyConfigurator

public ReloadablePropertyConfigurator()

Method Detail

doConfigure

public void doConfigure(URL url, LoggerRepository repository)

Description copied from interface: Configurator
Interpret a resource pointed by a URL and set up log4j accordingly. The configuration is done relative to the hierarchy parameter.

Specified by:
doConfigure in interface Configurator

Following copied from interface: org.apache.log4j.spi.Configurator

Parameters:
url - The URL to parse
repository - The hierarchy to operation upon.

Copyright 2000-2005 Apache Software Foundation.
public class Roller
extends Object

A simple application to send roll over messages to a potentially remote ExternallyRolledFileAppender.

It takes two arguments, the host_name and port_number where the ExternallyRolledFileAppender is listening.

Since:
version 0.9.0

Author:
Ceki Gülcü

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><code>main(String[] argv)</code></td>
<td>Send a &quot;RollOver&quot; message to ExternallyRolledFileAppender on host and port.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

main
public static void main(String[] argv)

Send a "RollOver" message to ExternallyRolledFileAppender on host and port.

Copyright 2000-2005 Apache Software Foundation.
public class StringMatchFilter
extends Filter

This is a very simple filter based on string matching.

The filter admits two options StringToMatch and AcceptOnMatch. If there is a match between the value of the StringToMatch option and the message of the LoggingEvent, then the decide(LoggingEvent) method returns Filter.ACCEPT if the AcceptOnMatch option value is true, if it is false then Filter.DENY is returned. If there is no match, Filter.NEUTRAL is returned.

Since:
0.9.0

Author:
Ceki Gülcü

Field Summary

<table>
<thead>
<tr>
<th>static String</th>
<th>ACCEPT_ON_MATCH_OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String</th>
<th>STRING_TO_MATCH_OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Options are now handled using the</td>
</tr>
</tbody>
</table>
JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

Fields inherited from class org.apache.log4j.spi.Filter
ACCEPT, DENY, NEUTRAL, next

Constructor Summary

StringMatchFilter()

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decide(LoggingEvent event)</td>
<td>Returns Filter.NEUTRAL if there is no string match.</td>
</tr>
<tr>
<td>getAcceptOnMatch()</td>
<td></td>
</tr>
<tr>
<td>getOptionStrings()</td>
<td>Deprecated. We now use JavaBeans introspection to configure components. Options strings are no longer needed.</td>
</tr>
<tr>
<td>getStringToMatch()</td>
<td></td>
</tr>
<tr>
<td>setAcceptOnMatch(boolean acceptOnMatch)</td>
<td></td>
</tr>
<tr>
<td>setOption(String key, String value)</td>
<td>Deprecated. Use the setter method for the option directly instead of the generic setoption method.</td>
</tr>
<tr>
<td>setStringToMatch(String s)</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class org.apache.log4j.spi.Filter
activateOptions, getNext, setNext

Methods inherited from class java.lang.Object
clon, equals, finalize, getClass, hashCode, notify, notifyAll,
Field Detail

STRING_TO_MATCH_OPTION

public static final String STRING_TO_MATCH_OPTION

    Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

ACCEPT_ON_MATCH_OPTION

public static final String ACCEPT_ON_MATCH_OPTION

    Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

Constructor Detail

StringMatchFilter

public StringMatchFilter()

Method Detail

getOptionStrings

public String[] getOptionStrings()

    Deprecated. We now use JavaBeans introspection to configure components. Options strings are no longer needed.
setOption

public void setOption(String key, String value)

Deprecated. Use the setter method for the option directly instead of the generic setOption method.

setStringToMatch

public void setStringToMatch(String s)

getStringToMatch

public String getStringToMatch()

setAcceptOnMatch

public void setAcceptOnMatch(boolean acceptOnMatch)

getAcceptOnMatch

public boolean getAcceptOnMatch()

decide

public int decide(LoggingEvent event)

    Returns Filter.NEUTRAL is there is no string match.
    Overrides:
        decide in class Filter
    Following copied from class: org.apache.log4j.spi.Filter
    Parameters:
The LoggingEvent to decide upon.

Returns:

decision The decision of the filter.
## Package org.apache.log4j.xml

XML based components.

See: [Description](#)

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMConfigurator</td>
<td>Use this class to initialize the log4j environment using a DOM tree.</td>
</tr>
<tr>
<td>Log4jEntityResolver</td>
<td>An <a href="#">EntityResolver</a> specifically designed to return log4j.dtd which is embedded within the log4j jar file.</td>
</tr>
<tr>
<td>SAXErrorHandler</td>
<td></td>
</tr>
<tr>
<td>XMLLayout</td>
<td>The output of the XMLLayout consists of a series of log4j:event elements as defined in the log4j.dtd.</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.xml Description

XML based components.

Last modified: Mon Mar 27 21:17:13 MDT 2000

Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package org.apache.log4j.xml

Package Hierarchies:
   All Packages
Class Hierarchy

○ class java.lang.\texttt{Object}
  ○ class org.apache.log4j.xml.\texttt{DOMConfigurator} (implements org.apache.log4j.spi.\texttt{Configurator})
  ○ class org.apache.log4j.\texttt{Layout} (implements org.apache.log4j.spi.\texttt{OptionHandler})
    ○ class org.apache.log4j.xml.\texttt{XMLLayout}
  ○ class org.apache.log4j.xml.\texttt{Log4jEntityResolver} (implements org.xml.sax.\texttt{EntityResolver})
  ○ class org.apache.log4j.xml.\texttt{SAXErrorHandler} (implements org.xml.sax.\texttt{ErrorHandler})

Copyright 2000-2005 Apache Software Foundation.
Uses of Package
org.apache.log4j.xml

No usage of org.apache.log4j.xml

Copyright 2000-2005 Apache Software Foundation.
Class DOMConfigurator

All Implemented Interfaces:
  Configurator

public class DOMConfigurator extends Object implements Configurator

Use this class to initialize the log4j environment using a DOM tree.

The DTD is specified in log4j.dtd.

Sometimes it is useful to see how log4j is reading configuration files. You can enable log4j internal logging by defining the log4j.debug variable on the java command line. Alternatively, set the debug attribute in the log4j:configuration element. As in

<log4j:configuration debug="true" xmlns:log4j="http://jakarta.apache.org/log4j">
  ...
</log4j:configuration>

There are sample XML files included in the package.

Since:
  0.8.3

Author:
  Christopher Taylor, Ceki Gülcü, Anders Kristensen

Fields inherited from interface org.apache.log4j.spi.Configurator

INHERITED, NULL
## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DOMConfigurator()</code></td>
<td>No argument constructor.</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>configure(Element element)</code></td>
<td>Configure log4j using a configuration element as defined in the log4j.dtd.</td>
</tr>
<tr>
<td><code>configure(String filename)</code></td>
<td>A static version of <code>doConfigure(String, LoggerRepository)</code>.</td>
</tr>
<tr>
<td><code>configure(URL url)</code></td>
<td>A static version of <code>doConfigure(URL, LoggerRepository)</code>.</td>
</tr>
<tr>
<td><code>configureAndWatch(String configFilename)</code></td>
<td>Like <code>configureAndWatch(String, long)</code> except that the default delay as defined by <code>FileWatchdog.DEFAULT_DELAY</code> is used.</td>
</tr>
<tr>
<td><code>configureAndWatch(String configFilename, long delay)</code></td>
<td>Read the configuration file <code>configFilename</code> if it exists.</td>
</tr>
<tr>
<td><code>doConfigure(Element element, LoggerRepository repository)</code></td>
<td>Configure by taking in an DOM element.</td>
</tr>
<tr>
<td><code>doConfigure(InputSource inputSource, LoggerRepository repository)</code></td>
<td>Configure log4j by reading in a log4j.dtd compliant XML configuration file.</td>
</tr>
<tr>
<td><code>doConfigure(Reader reader, LoggerRepository repository)</code></td>
<td>Configure log4j by reading in a log4j.dtd compliant XML configuration file.</td>
</tr>
<tr>
<td><code>doConfigure(String filename, LoggerRepository repository)</code></td>
<td>Configure log4j by reading in a log4j.dtd compliant XML configuration file.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>void doConfigure(URL url, LoggerRepository repository)</code></td>
<td>Interpret a resource pointed by a URL and set up log4j accordingly.</td>
</tr>
<tr>
<td><code>protected Appender findAppenderByName(Document doc, String appenderName)</code></td>
<td>Used internally to parse appenders by IDREF name.</td>
</tr>
<tr>
<td><code>protected Appender findAppenderByReference(Element appenderRef)</code></td>
<td>Used internally to parse appenders by IDREF element.</td>
</tr>
<tr>
<td><code>protected void parse(Element element)</code></td>
<td>Used internally to configure the log4j framework by parsing a DOM tree of XML elements based on log4j.dtd.</td>
</tr>
<tr>
<td><code>protected Appender parseAppender(Element appenderElement)</code></td>
<td>Used internally to parse an appender element.</td>
</tr>
<tr>
<td><code>protected void parseCategory(Element loggerElement)</code></td>
<td>Used internally to parse a category element.</td>
</tr>
<tr>
<td><code>protected void parseCategoryFactory(Element factoryElement)</code></td>
<td>Used internally to parse the category factory element.</td>
</tr>
<tr>
<td><code>protected void parseChildrenOfLoggerElement(Element catElement, Logger cat, boolean isRoot)</code></td>
<td>Used internally to parse the children of a category element.</td>
</tr>
<tr>
<td><code>protected void parseErrorHandler(Element element, Appender appender)</code></td>
<td>Used internally to parse an ErrorHandler element.</td>
</tr>
<tr>
<td><code>protected void parseFilters(Element element, Appender appender)</code></td>
<td>Used internally to parse a filter element.</td>
</tr>
<tr>
<td><code>protected void parseLayout(Element layout_element)</code></td>
<td>Used internally to parse a layout element.</td>
</tr>
<tr>
<td><code>protected void parseLevel(Element element, Logger logger, boolean isRoot)</code></td>
<td>Used internally to parse a level element.</td>
</tr>
<tr>
<td><code>protected void parseRenderer(Element element)</code></td>
<td></td>
</tr>
<tr>
<td><code>protected void parseRoot(Element rootElement)</code></td>
<td>Used internally to parse the root category element.</td>
</tr>
<tr>
<td><code>protected void setParameter(Element elem, PropertySetter propSetter)</code></td>
<td></td>
</tr>
</tbody>
</table>
protected `String` `subst(String value)`

**Methods inherited from class java.lang.`Object`**
`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`  

**Constructor Detail**

**DOMConfigurator**

public `DOMConfigurator()`

No argument constructor.

**Method Detail**

**findAppenderByName**

protected `Appender` `findAppenderByName(Document doc, String appenderName)`

Used internally to parse appenders by IDREF name.

**findAppenderByReference**

protected `Appender` `findAppenderByReference(Element appenderRef)`

Used internally to parse appenders by IDREF element.

**parseAppender**

protected `Appender` `parseAppender(Element appenderElement)`
Used internally to parse an appender element.

**parseErrorHandler**

protected void `parseErrorHandler(Element element, Appender appender)`

Used internally to parse an `ErrorHandler` element.

**parseFilters**

protected void `parseFilters(Element element, Appender appender)`

Used internally to parse a filter element.

**parseCategory**

protected void `parseCategory(Element loggerElement)`

Used internally to parse an `Category` element.

**parseCategoryFactory**

protected void `parseCategoryFactory(Element factoryElement)`

Used internally to parse the category factory element.

**parseRoot**

protected void `parseRoot(Element rootElement)`

Used internally to parse the root category element.
parseChildrenOfLoggerElement

protected void parseChildrenOfLoggerElement(Element catElement, Logger cat, boolean isRoot)

Used internally to parse the children of a category element.

parseLayout

protected Layout parseLayout(Element layout_element)

Used internally to parse a layout element.

parseRenderer

protected void parseRenderer(Element element)

parseLevel

protected void parseLevel(Element element, Logger logger, boolean isRoot)

Used internally to parse a level element.

setParameter

protected void setParameter(Element elem, PropertySetter propSetter)

configure

public static void configure(Element element)

Configure log4j using a configuration element as defined in the
configureAndWatch

public static void configureAndWatch(String configFilename)

Like configureAndWatch(String, long) except that the default delay as defined by FileWatchdog.DEFAULT_DELAY is used.

**Parameters:**
- configFilename - A log4j configuration file in XML format.

configureAndWatch

public static void configureAndWatch(String configFilename, long delay)

Read the configuration file configFilename if it exists. Moreover, a thread will be created that will periodically check if configFilename has been created or modified. The period is determined by the delay argument. If a change or file creation is detected, then configFilename is read to configure log4j.

**Parameters:**
- configFilename - A log4j configuration file in XML format.
- delay - The delay in milliseconds to wait between each check.

doConfigure

public void doConfigure(String filename, LoggerRepository repository)

doConfigure

public void doConfigure(URL url, LoggerRepository repository)

**Description copied from interface:** Configurator
Interpret a resource pointed by a URL and set up log4j accordingly. The configuration is done relative to the hierarchy parameter. **Specified by:**  
`doConfigure` in interface `Configurator`  
Following copied from interface: `org.apache.log4j.spi.Configurator`  
**Parameters:**  
- `url` - The URL to parse  
- `repository` - The hierarchy to operation upon.

---

```java
public void doConfigure(InputStream inputStream, LoggerRepository repository) throws FactoryConfigurationError
```

Configure log4j by reading in a log4j.dtd compliant XML configuration file.

---

```java
public void doConfigure(Reader reader, LoggerRepository repository) throws FactoryConfigurationError
```

Configure log4j by reading in a log4j.dtd compliant XML configuration file.

---

```java
protected void doConfigure(InputSource inputSource, LoggerRepository repository) throws FactoryConfigurationError
```

Configure log4j by reading in a log4j.dtd compliant XML configuration file.
**doConfigure**

public void **doConfigure**(Element element, LoggerRepository repository)

Configure by taking in an DOM element.

---

**configure**

public static void **configure**(String filename)

throws FactoryConfigurationError

A static version of **doConfigure**(String, LoggerRepository).

---

**configure**

public static void **configure**(URL url)

throws FactoryConfigurationError

A static version of **doConfigure**(URL, LoggerRepository).

---

**parse**

protected void **parse**(Element element)

Used internally to configure the log4j framework by parsing a DOM tree of XML elements based on log4j.dtd.

---

**subst**

protected String **subst**(String value)
Copyright 2000-2005 Apache Software Foundation.
public class Log4jEntityResolver
extends Object
implements EntityResolver

An EntityResolver specifically designed to return log4j.dtd which is embedded within the log4j jar file.

Author:
    Paul Austin

---

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log4jEntityResolver()</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolveEntity</td>
<td>(String publicId, String systemId)</td>
</tr>
</tbody>
</table>

Method Summary

Methods inherited from class java.lang.Object

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</td>
</tr>
</tbody>
</table>

Constructor Detail
Log4jEntityResolver

public Log4jEntityResolver()

### Method Detail

**resolveEntity**

public **InputSource** resolveEntity(**String** publicId, **String** systemId)

Specified by:

resolveEntity in interface EntityResolver

Copyright 2000-2005 Apache Software Foundation.
<table>
<thead>
<tr>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
<th>Log4j 1.2.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: INNER</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
</tr>
</tbody>
</table>
org.apache.log4j.xml  Class SAXErrorHandler

java.lang.Object
  +--org.apache.log4j.xml.SAXErrorHandler

All Implemented Interfaces:
  ErrorHandler

public class SAXErrorHandler
  extends Object
  implements ErrorHandler

Constructor Summary

SAXErrorHandler()  

Method Summary

void error(SAXParseException ex)

void fatalError(SAXParseException ex)

void warning(SAXParseException ex)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail


SAXErrorHandler

public SAXErrorHandler()

Method Detail

error

public void error(SAXParseException ex)

    Specified by:
        error in interface ErrorHandler

fatalError

public void fatalError(SAXParseException ex)

    Specified by:
        fatalError in interface ErrorHandler

warning

public void warning(SAXParseException ex)

    Specified by:
        warning in interface ErrorHandler

Copyright 2000-2005 Apache Software Foundation.
public class XMLLayout extends Layout

The output of the XMLLayout consists of a series of log4j:event elements as defined in the log4j.dtd. It does not output a complete well-formed XML file. The output is designed to be included as an external entity in a separate file to form a correct XML file.

For example, if abc is the name of the file where the XMLLayout output goes, then a well-formed XML file would be:

```xml
<?xml version="1.0" ?>
```

This approach enforces the independence of the XMLLayout and the appender where it is embedded.

The version attribute helps components to correctly interpret output generated by XMLLayout. The value of this attribute should be "1.1" for output generated by log4j versions prior to log4j 1.2 (final release) and "1.2" for release 1.2 and later.

Since: 0.9.0
Fields inherited from class org.apache.log4j.Layout
LINE_SEP, LINE_SEP_LEN

Constructor Summary

XMLLayout()  

Method Summary

void activateOptions()  
No options to activate.

String format(LoggingEvent event)  
Formats a LoggingEvent in conformance with the log4j.dtd.

boolean getLocationInfo()  
Returns the current value of the LocationInfo option.

boolean ignoresThrowable()  
The XMLLayout prints and does not ignore exceptions.

void setLocationInfo(boolean flag)  
The LocationInfo option takes a boolean value.

Methods inherited from class org.apache.log4j.Layout
getRequestContentType, getFooter, getHeader

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail
XMLLayout

public XMLLayout()

Method Detail

setLocationInfo

public void setLocationInfo(boolean flag)

The LocationInfo option takes a boolean value. By default, it is set to false which means there will be no location information output by this layout. If the option is set to true, then the file name and line number of the statement at the origin of the log statement will be output.

If you are embedding this layout within an SMTPAppender then make sure to set the LocationInfo option of that appender as well.

getLocationInfo

public boolean getLocationInfo()

Returns the current value of the LocationInfo option.

activateOptions

public void activateOptions()

No options to activate.

format

public String format(LoggingEvent event)

Formats a LoggingEvent in conformance with the log4j.dtd.
Overrides:

format in class Layout

gignoresThrowable

public boolean ignoresThrowable()

The XMLLayout prints and does not ignore exceptions. Hence the return value false.

Overrides:
ignoresThrowable in class Layout

Copyright 2000-2005 Apache Software Foundation.
Package org.apache.log4j.xml.examples

Example usage of log4j with XML (including source code).

See: Description

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReportParserError</strong></td>
</tr>
<tr>
<td><strong>XMLSample</strong></td>
</tr>
</tbody>
</table>
Package org.apache.log4j.xml.examples

Description

Example usage of log4j with XML (including source code).

This package's shows how log4j can be used with an XML configuration file.

See source code of XMLSample.java showing how to configure with an XML file. Sample XML files sample1.xml, sample2.xml, sample3.xml, sample4.xml, sample5.xml are provided.


Copyright 2000-2005 Apache Software Foundation.
Hierarchy For Package
org.apache.log4j.xml.examples

Package Hierarchies:

All Packages
Class Hierarchy

- class java.lang.**Object**
  - class org.apache.log4j.xml.examples.**ReportParserError**
    (implements org.xml.sax.**ErrorHandler**)
  - class org.apache.log4j.xml.examples.**XMLSample**
Uses of Package
org.apache.log4j.xml.examples

No usage of org.apache.log4j.xml.examples
<table>
<thead>
<tr>
<th>Overview</th>
<th>Package</th>
<th>Use</th>
<th>Tree</th>
<th>Deprecated</th>
<th>Index</th>
<th>Help</th>
<th>Log4j 1.2.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREV CLASS</td>
<td>NEXT CLASS</td>
<td>SUMMARY: INNER</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
</tr>
<tr>
<td>SUMMARY: INNER</td>
<td>FIELD</td>
<td>CONSTR</td>
<td>METHOD</td>
<td>FRAMES</td>
<td>NO FRAMES</td>
<td>DETAIL: FIELD</td>
<td>CONSTR</td>
</tr>
</tbody>
</table>
Class ReportParserError

java.lang.Object
+--org.apache.log4j.xml.examples.ReportParserError

All Implemented Interfaces:
   ErrorHandler

public class ReportParserError extends Object implements ErrorHandler

This class is needed for validating a log4j.dtd derived XML file.

Since: 0.8.3
Author: Joe Kesselman

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReportParserError()</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void error(SAXParseException e)</td>
</tr>
<tr>
<td>void fatalError(SAXParseException e)</td>
</tr>
<tr>
<td>void warning(SAXParseException e)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
Constructor Detail

ReportParserError

public ReportParserError()

Method Detail

warning

public void warning(SAXParseException e)

   Specified by:
      warning in interface ErrorHandler

error

public void error(SAXParseException e)

   Specified by:
      error in interface ErrorHandler

fatalError

public void fatalError(SAXParseException e)

   Specified by:
      fatalError in interface ErrorHandler
public class XMLSample
extends Object

This example code shows how to read an XML based configuration file using a DOM parser.

Sample XML files sample1.xml and sample2.xml are provided.

Note that the log4j.dtd is not in the local directory. It is found by the class loader.

Author:
Ceki Gülcü
Constructor Detail

XMLSample

public XMLSample()

Method Detail

main

public static void main(String[] argv)
# Deprecated API

## Deprecated Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.apache.log4j.spi.RootCategory</code></td>
<td><code>RootLogger</code></td>
</tr>
</tbody>
</table>

## Deprecated Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.apache.log4j.varia.StringMatchFilter.ACCEPT_ON_MATCH_OPTION</code></td>
<td>Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
<tr>
<td><code>org.apache.log4j.spi.LoggingEvent.categoryName</code></td>
<td>This field will be marked as private in future releases. Please do not access it directly. Use the <code>LoggingEvent.getLoggerName()</code> method instead.</td>
</tr>
<tr>
<td><code>org.apache.log4j.LogManager.CONFIGURATOR_CLASS_KEY</code></td>
<td>This variable is for internal use only. It will become private in future versions.</td>
</tr>
<tr>
<td><code>org.apache.log4j.helpers.DateLayout.DATE_FORMAT_OPTION</code></td>
<td>Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
<tr>
<td><code>org.apache.log4j LogManager.DEFAULT_CONFIGURATION_FILE</code></td>
<td>This variable is for internal use only. It will become package protected in future versions.</td>
</tr>
<tr>
<td><code>org.apache.log4j LogManager.DEFAULT_CONFIGURATION_KEY</code></td>
<td>This variable is for internal use only. It will become private in future versions.</td>
</tr>
<tr>
<td><code>org.apache.log4j LogManager.DEFAULT_INIT_OVERRIDE_KEY</code></td>
<td>This variable is for internal use only. It will become private in future versions.</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>org.apache.log4j.Priority.ERROR</td>
<td>Use <code>Level.ERROR</code> instead.</td>
</tr>
<tr>
<td>org.apache.log4j.spi.LoggingEvent.level</td>
<td>This field will be marked as private in future releases. Please do not access it directly. Use the <code>LoggingEvent.getLevel()</code> method instead.</td>
</tr>
<tr>
<td>org.apache.log4j.html.layout.LOCATION_INFO_OPTION</td>
<td>Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
<tr>
<td>org.apache.log4j.varia.StringMatchFilter.STRING_TO_MATCH_OPTION</td>
<td>Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers.DateLayout.TIMEZONE_OPTION</td>
<td>Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.</td>
</tr>
<tr>
<td>org.apache.log4j.Priority.WARN</td>
<td>Use <code>Level.WARN</code> instead.</td>
</tr>
</tbody>
</table>

### Deprecated Methods

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.Priority.getAllPossiblePriorities()</td>
<td>This method will be removed with no replacement.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getChainedPriority()</td>
<td>Please use the the <code>Category.getEffectiveLevel()</code> method instead.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getCurrentCategories()</td>
<td></td>
</tr>
</tbody>
</table>

---

**Deprecation Notice**

Please consider using the recommended methods and constants instead of the deprecated ones. For more information, refer to the latest documentation or release notes.
<table>
<thead>
<tr>
<th>Method</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogManager.getCurrentLoggers()</td>
<td>Please use LogManager.getCurrentLoggers() instead.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getDefaultHierarchy()</td>
<td>Please use LogManager.getLoggerRepository() instead.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getHierarchy()</td>
<td>Please use Category.getLoggerRepository() instead.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getInstance(Class)</td>
<td>Please make sure to use Logger.getLogger(Class) instead.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getInstance(String)</td>
<td>Make sure to use Logger.getLogger(String) instead.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers.DateLayout.getOptionStrings()</td>
<td>Use the setter method for the option directly instead of the generic setOption method.</td>
</tr>
<tr>
<td>org.apache.log4j.varia.DenyAllFilter.getOptionStrings()</td>
<td>We now use JavaBeans introspection to configure components. Options strings are no longer needed.</td>
</tr>
<tr>
<td>org.apache.log4j.varia.StringMatchFilter.getOptionStrings()</td>
<td>We now use JavaBeans introspection to configure components. Options strings are no longer needed.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getPriority()</td>
<td>Please use Category.getLevel() instead.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers.Loader.getResource(String, Class)</td>
<td>as of 1.2.</td>
</tr>
<tr>
<td>org.apache.log4j.Category.getRoot()</td>
<td>Please use Logger.getRootLogger() instead.</td>
</tr>
<tr>
<td>org.apache.log4j.Hierarchy.overrideAsString(String)</td>
<td>Deprecated with no replacement.</td>
</tr>
<tr>
<td>org.apache.log4j.Hierarchy.setDisableOverride(String)</td>
<td>Deprecated with no replacement.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers.DateLayout.setOption(String, String)</td>
<td>Use the setter method for the option directly instead of the generic setOption method.</td>
</tr>
<tr>
<td>org.apache.log4j.varia.DenyAllFilter.setOption(String, String)</td>
<td>Use the setter method for the option directly instead of the</td>
</tr>
<tr>
<td>Method</td>
<td>Alternative Method</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>org.apache.log4j.varia.StringMatchFilter.setOption(String, String)</code></td>
<td>Use the setter method for the option directly instead of the generic <code>setOption</code> method.</td>
</tr>
<tr>
<td><code>org.apache.log4j.Category.shutdown()</code></td>
<td>Please use <code>LogManager.shutdown()</code> instead.</td>
</tr>
<tr>
<td><code>org.apache.log4j.Priority.toPriority(int)</code></td>
<td>Please use the <code>Level.toLevel(int)</code> method instead.</td>
</tr>
<tr>
<td><code>org.apache.log4j.Priority.toPriority(int, Priority)</code></td>
<td>Please use the <code>Level.toLevel(int, Level)</code> method instead.</td>
</tr>
<tr>
<td><code>org.apache.log4j.Priority.toPriority(String)</code></td>
<td>Please use the <code>Level.toLevel(String)</code> method instead.</td>
</tr>
<tr>
<td><code>org.apache.log4j.Priority.toPriority(String, Priority)</code></td>
<td>Please use the <code>Level.toLevel(String, Level)</code> method instead.</td>
</tr>
</tbody>
</table>
ABS_TIME_DATE_FORMAT - Static variable in class org.apache.log4j.helpers.AbsoluteTimeDateFormat
    String constant used to specify AbsoluteTimeDateFormat in layouts.

AbsoluteTimeDateFormat - class org.apache.log4j.helpers.AbsoluteTimeDateFormat.
    Formats a Date in the format "HH:mm:ss,SSS" for example, "15:49:37,459".

AbsoluteTimeDateFormat() - Constructor for class org.apache.log4j.helpers.AbsoluteTimeDateFormat

AbsoluteTimeDateFormat(TimeZone) - Constructor for class org.apache.log4j.helpers.AbsoluteTimeDateFormat


AbstractDynamicMBean() - Constructor for class org.apache.log4j.jmx.AbstractDynamicMBean

ACCEPT - Static variable in class org.apache.log4j.spi.Filter
    The log event must be logged immediately without consulting with the remaining filters, if any, in the chain.

ACCEPT_ON_MATCH_OPTION - Static variable in class org.apache.log4j.varia.StringMatchFilter
    Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

activate() - Method in class org.apache.log4j.config.PropertySetter

activateOptions() - Method in class org.apache.log4j.SimpleLayout
    Does not do anything as options become effective

activateOptions() - Method in class org.apache.log4j.PatternLayout

activateOptions() - Method in class org.apache.log4j.HTMLLayout
    No options to activate.
activateOptions() - Method in class org.apache.log4j.AppenderSkeleton
Derived appenders should override this method if option structure requires it.

activateOptions() - Method in class org.apache.log4j.WriterAppender
Does nothing.

activateOptions() - Method in class org.apache.log4j.FileAppender
If the value of File is not null, then FileAppender.setFile(java.lang.String) is called with the values of File and Append properties.

activateOptions() - Method in class org.apache.log4j.ConsoleAppender
Prepares the appender for use.

activateOptions() - Method in class org.apache.log4j.DailyRollingFileAppender

activateOptions() - Method in class org.apache.log4j.helpers.DateLayout

activateOptions() - Method in class org.apache.log4j.helpers.OnlyOnceErrorHandler
No options to activate.

activateOptions() - Method in class org.apache.log4j.net.SocketHubAppender
Set up the socket server on the specified port.

activateOptions() - Method in class org.apache.log4j.net.SyslogAppender
This method returns immediately as options are activated when they are set.

activateOptions() - Method in class org.apache.log4j.net.SMTPAppender
Activate the specified options, such as the smtp host, the recipient, from, etc.

activateOptions() - Method in class org.apache.log4j.net.JMSAppender
Options are activated and become effective only after calling this method.

activateOptions() - Method in class org.apache.log4j.net.TelnetAppender
all of the options have been set, create the socket handler and wait for connections.

activateOptions() - Method in class
org.apache.log4j.net.SocketAppender
    Connect to the specified RemoteHost and Port.
activateOptions() - Method in class
org.apache.log4j.nt.NTEventLogAppender

activateOptions() - Method in class org.apache.log4j.spi.Filter
    Usually filters options become active when set.
activateOptions() - Method in interface
org.apache.log4j.spi.OptionHandler
    Activate the options that were previously set with calls to option setters.
activateOptions() - Method in class org.apache.log4j.varia.NullAppender
    There are no options to activate.
activateOptions() - Method in class org.apache.log4j.varia.ExternallyRolledFileAppender
    Start listening on the port specified by a preceding call to
    ExternallyRolledFileAppender.setPort(int).
activateOptions() - Method in class org.apache.log4j.varia.FallbackErrorHandler
    No options to activate.
activateOptions() - Method in class org.apache.log4j.xml.XMLOutput
    No options to activate.
add(LoggingEvent) - Method in class
org.apache.log4j.helpers.CyclicBuffer
    Add an event as the last event in the buffer.
addAppender(Appender) - Method in class org.apache.log4j.Category
    Add newAppender to the list of appenders of this Category instance.
addAppender(Appender) - Method in class
org.apache.log4j.AsyncAppender
    Add appender.
addAppender(Appender) - Method in class
org.apache.log4j.helpers.AppenderAttachableImpl
    Attach an appender.
addAppender(Appender) - Method in interface
org.apache.log4j.spi.AppenderAttachable
    Add an appender.
addAppenderEvent(Category, Appender) - Method in class
org.apache.log4j.jmx.HierarchyDynamicMBean
addAppenderEvent(Category, Appender) - Method in interface org.apache.log4j.spi.HierarchyEventListener

addConverter(PatternConverter) - Method in class org.apache.log4j.helpers.PatternParser

addFilter(Filter) - Method in class org.apache.log4j.AppenderSkeleton
  Add a filter to the end of the filter list.
addFilter(Filter) - Method in interface org.apache.log4j.Appender
  Add a filter to the end of the filter list.
addHierarchyEventListener(HierarchyEventListener) - Method in class org.apache.log4j.Hierarchy

addHierarchyEventListener(HierarchyEventListener) - Method in interface org.apache.log4j.spi.LoggerRepository
  Add a HierarchyEventListener event to the repository.
additive - Variable in class org.apache.log4j.Category
  Additivity is set to true by default, that is children inherit the appenders of their ancestors by default.
addLoggerMBean(String) - Method in class org.apache.log4j.jmx.HierarchyDynamicMBean

addNotificationListener(NotificationListener, NotificationFilter, Object) - Method in class org.apache.log4j.jmx.HierarchyDynamicMBean

addRenderer(Class, ObjectRenderer) - Method in class org.apache.log4j.Hierarchy
  Add an object renderer for a specific class.
addRenderer(RendererSupport, String, String) - Static method in class org.apache.log4j.or.RendererMap
  Add a renderer to a hierarchy passed as parameter.
ADDRESS_PARSE_FAILURE - Static variable in interface org.apache.log4j.spi.ErrorCode

addressMessage(Message) - Method in class org.apache.log4j.net.SMTPAppender
  Address message.
Agent - class org.apache.log4j.jmx.Agent.
**Agent()** - Constructor for class org.apache.log4j.jmx.Agent

**ALL** - Static variable in class org.apache.log4j.Level
   The ALL has the lowest possible rank and is intended to turn on all logging.

**ALL_INT** - Static variable in class org.apache.log4j.Priority

**append(LoggingEvent)** - Method in class org.apache.log4j.AppenderSkeleton
   Subclasses of AppenderSkeleton should implement this method to perform actual logging.

**append(LoggingEvent)** - Method in class org.apache.log4j.WriterAppender
   This method is called by the AppenderSkeleton.doAppend(org.apache.log4j.spi.LoggingEvent) method.

**append(LoggingEvent)** - Method in class org.apache.log4j.AsyncAppender
   {@inheritDoc}

**append(LoggingEvent)** - Method in class org.apache.log4j.jdbc.JDBCAppender
   Adds the event to the buffer.

**append(LoggingEvent)** - Method in class org.apache.log4j.lf5.LF5Appender
   Appends a LoggingEvent record to the LF5Appender.

**append(LoggingEvent)** - Method in class org.apache.log4j.net.SocketHubAppender
   Append an event to all of current connections.

**append(LoggingEvent)** - Method in class org.apache.log4j.net.SyslogAppender

**append(LoggingEvent)** - Method in class org.apache.log4j.net.SMTPAppender
   Perform SMTPAppender specific appending actions, mainly adding the event to a cyclic buffer and checking if the event triggers an e-mail to be sent.

**append(LoggingEvent)** - Method in class org.apache.log4j.net.jms.JMSAppender
   This method called by
AppenderSkeleton.doAppend(org.apache.log4j.spi.LoggingEvent) method to do most of the real appending work.

**append(LoggingEvent)** - Method in class org.apache.log4j.net.TelnetAppender
Handles a log event.

**append(LoggingEvent)** - Method in class org.apache.log4j.net.SocketAppender

**append(LoggingEvent)** - Method in class org.apache.log4j.nt.NTEventLogAppender

**append(LoggingEvent)** - Method in class org.apache.log4j.performance.NullAppender

**append(LoggingEvent)** - Method in class org.apache.log4j.varia.NullAppender
- Does not do anything.

- Implement this interface for your own strategies for outputting log statements.

- Interface for attaching appenders to objects.

- A straightforward implementation of the AppenderAttachable interface.

**AppenderAttachableImpl()** - Constructor for class org.apache.log4j.helpers.AppenderAttachableImpl


**AppenderDynamicMBean(Appender)** - Constructor for class org.apache.log4j.jmx.AppenderDynamicMBean

- AppenderFinalizer has a single method that will finalize resources associated with a LogBrokerMonitor in the event that the LF5Appen
class is destroyed, and the class loader is garbage collected.

**AppenderFinalizer(LogBrokerMonitor)** - Constructor for class org.apache.log4j.lf5.AppenderFinalizer

**appendList** - Variable in class org.apache.log4j.helpers.AppenderAttachableImpl
   Array of appenders.

**appendNames** - Variable in class org.apache.log4j.config.PropertyPrinter

**AppenderSkeleton** - class org.apache.log4j.AppenderSkeleton
   Abstract superclass of the other appenders in the package.

**AppenderSkeleton()** - Constructor for class org.apache.log4j.AppenderSkeleton

**appendEscapingCDATA(StringBuffer, String)** - Static method in class org.apache.log4j.helpers.Transform
   Ensures that embedded CDEnd strings (]]>) are handled properly within message, NDC and throwable tag text.

   Call the doAppend method on all attached appenders.

**assertLog(boolean, String)** - Method in class org.apache.log4j.Category
   If assertion parameter is false, then logs msg as an error statement.

**AsyncAppender** - class org.apache.log4j.AsyncAppender
   The AsyncAppender lets users log events asynchronously.

**AsyncAppender()** - Constructor for class org.apache.log4j.AsyncAppender
   Create new instance.

**AttributesRenderer** - class org.apache.log4j.or.sax.AttributesRenderer
   Render org.xml.sax.Attributes objects.

**AttributesRenderer()** - Constructor for class org.apache.log4j.or.sax.AttributesRenderer
**BasicConfigurator** - class org.apache.log4j.BasicConfigurator.
Use this class to quickly configure the package.

**BasicConfigurator()** - Constructor for class org.apache.log4j.BasicConfigurator

**BoundedFIFO** - class org.apache.log4j.helpers.BoundedFIFO.
BoundedFIFO serves as the bounded first-in-first-out buffer heavily used by the AsyncAppender.

**BoundedFIFO(int)** - Constructor for class org.apache.log4j.helpers.BoundedFIFO
Instantiate a new BoundedFIFO with a maximum size passed as argument.

**buf** - Variable in class org.apache.log4j.TTCCLayout

**BUF_SIZE** - Variable in class org.apache.log4j.PatternLayout

**BUF_SIZE** - Variable in class org.apache.log4j.HTMLLayout

**buffer** - Variable in class org.apache.log4j.jdbc.JDBCAppender
ArrayList holding the buffer of Logging Events.

**bufferedIO** - Variable in class org.apache.log4j.FileAppender
Do we do bufferedIO?

**bufferSize** - Variable in class org.apache.log4j.FileAppender
Determines the size of IO buffer be.

**bufferSize** - Variable in class org.apache.log4j.jdbc.JDBCAppender
size of LoggingEvent buffer before writing to the database.
callAppenders(LoggingEvent) - Method in class org.apache.log4j.Category
   Call the appenders in the hierarchy starting at this.
capitalize(String) - Static method in class org.apache.log4j.config.PropertyPrinter

Category - class org.apache.log4j.Category
   This class has been deprecated and replaced by the Logger subclass.
Category(String) - Constructor for class org.apache.log4j.Category
   This constructor created a new Category instance and sets its name.
categoryName - Variable in class org.apache.log4j.spi.LoggingEvent
   Deprecated. This field will be marked as private in future releases.
   Please do not access it directly. Use the
   LoggingEvent.getLoggerName() method instead.
cb - Variable in class org.apache.log4j.net.SMTPAppender

checkAndConfigure() - Method in class org.apache.log4j.helpers.FileWatchdog

checkEntryConditions() - Method in class org.apache.log4j.WriterAppender
   This method determines if there is a sense in attempting to append.
checkEntryConditions() - Method in class org.apache.log4j.net.SMTPAppender
   This method determines if there is a sense in attempting to append.
checkEntryConditions() - Method in class org.apache.log4j.net.JMSAppender

childValue(Object) - Method in class org.apache.log4j.helpers.ThreadLocalMap

cleanUp() - Method in class org.apache.log4j.net.SocketHubAppender
   Release the underlying ServerMonitor thread, and drop the
   connections to all connected remote servers.
**cleanUp()** - Method in class org.apache.log4j.net.SocketAppender
Drop the connection to the remote host and release the underlying connector thread if it has been created.

**clear()** - Method in class org.apache.log4j.Hierarchy
This call will clear all logger definitions from the internal hashtable.

**clear()** - Static method in class org.apache.log4j.NDC
Clear any nested diagnostic information if any.

**clear()** - Method in class org.apache.log4j.or.RendererMap

**clearFilters()** - Method in class org.apache.log4j.AppenderSkeleton
Clear the filters chain.

**clearFilters()** - Method in interface org.apache.log4j.Appender
Clear the list of filters by removing all the filters in it.

**cloneStack()** - Static method in class org.apache.log4j.NDC
Clone the diagnostic context for the current thread.

**CLOSE_FAILURE** - Static variable in interface org.apache.log4j.spi.ErrorCode

**close()** - Method in class org.apache.log4j.WriterAppender
Close this appender instance.

**close()** - Method in interface org.apache.log4j.Appender
Release any resources allocated within the appender such as file handles, network connections, etc.

**close()** - Method in class org.apache.log4j.AsyncAppender
Close this AsyncAppender by interrupting the dispatcher thread which will process all pending events before exiting.

**close()** - Method in class org.apache.log4j.helpers.SyslogWriter

**close()** - Method in class org.apache.log4j.jdbc.JDBCAppender
Closes the appender, flushing the buffer first then closing the default connection if it is open.

**close()** - Method in class org.apache.log4j.lf5.LF5Appender
This method is an empty implementation of the close() method inherited from the org.apache.log4j.Appender interface.

**close()** - Method in class org.apache.log4j.net.SocketHubAppender
Close this appender.

**close()** - Method in class org.apache.log4j.net.SyslogAppender
Release any resources held by this SyslogAppender.

**close()** - Method in class org.apache.log4j.net.SMTPAppender
**close()** - Method in class org.apache.log4j.net. **JMSAppender**
  Close this JMSAppender.
**close()** - Method in class org.apache.log4j.net. **TelnetAppender**
  Shuts down the appender.
**close()** - Method in class org.apache.log4j.net. **SocketAppender**
  Close this appender.
**close()** - Method in class org.apache.log4j.nt. **NTEventLogAppender**
**close()** - Method in class org.apache.log4j.performance. **NullAppender**
**close()** - Method in class org.apache.log4j.performance. **NOPWriter**
**close()** - Method in class org.apache.log4j.varia. **NullAppender**

**closeConnection(Connection)** - Method in class org.apache.log4j.jdbc. **JDBCAppender**
  Override this to return the connection to a pool, or to clean up the resource.
**closed** - Variable in class org.apache.log4j. **AppenderSkeleton**
  Is this appender closed?
**closeFile()** - Method in class org.apache.log4j. **FileAppender**
  Closes the previously opened file.
**closeWriter()** - Method in class org.apache.log4j. **WriterAppender**
  Close the underlying **Writer**.
**closeWriter()** - Method in class org.apache.log4j. **ConsoleAppender**
  {@inheritDoc}

**concatenateArrays(String[], String[])** - Static method in class org.apache.log4j.helpers. **OptionConverter**

**CONFIG** - Static variable in class org.apache.log4j.lf5. **LogLevel**

**CONFIG_DEBUG_KEY** - Static variable in class org.apache.log4j.helpers. **LogLog**
  **Deprecated. Use LogLog.DEBUG_KEY instead.**
**Configurator** - interface org.apache.log4j.spi. **Configurator**
  Implemented by classes capable of configuring log4j using a URL.
**CONFIGURATOR_CLASS_KEY** - Static variable in class org.apache.log4j. **LogManager**
**Deprecated.** This variable is for internal use only. It will become private in future versions.

**configure()** - Static method in class org.apache.log4j.BasicConfigurator
Add a **ConsoleAppender** that uses **PatternLayout** using the **PatternLayout.TTCC_CONVERSION_PATTERN** and prints to System.out to the root category.

**configure()** - Static method in class org.apache.log4j.lf5.DefaultLF5Configurator
This method configures the LF5Appender using a default configuration file.

**configure(Appender)** - Static method in class org.apache.log4j.BasicConfigurator
Add appender to the root category.

**configure(Element)** - Static method in class org.apache.log4j.xml.DOMConfigurator
Configure log4j using a configuration element as defined in the log4j.dtd.

**configure(Properties)** - Static method in class org.apache.log4j.PropertyConfigurator
Read configuration options from properties.

**configure(String)** - Static method in class org.apache.log4j.PropertyConfigurator
A static version of **DOMConfigurator.doConfigure(String, LoggerRepository)**.

**configure(URL)** - Static method in class org.apache.log4j.PropertyConfigurator
Read configuration options from url configURL.

**configure(URL)** - Static method in class org.apache.log4j.xml.DOMConfigurator
A static version of **DOMConfigurator.doConfigure(URL, LoggerRepository)**.

**configureAndWatch(String)** - Static method in class org.apache.log4j.PropertyConfigurator
Like **PropertyConfigurator.configureAndWatch(String, long)** except that the default delay as defined by **FileWatchdog.DEFAULT_DELAY** is used.
configureAndWatch(String) - Static method in class org.apache.log4j.xml.DOMConfigurator
   Like DOMConfigurator.configureAndWatch(String, long) except that the default delay as defined by FileWatchdog.DEFAULT_DELAY is used.
configureAndWatch(String, long) - Static method in class org.apache.log4j.PropertyConfigurator
   Read the configuration file configFilename if it exists.
configureAndWatch(String, long) - Static method in class org.apache.log4j.xml.DOMConfigurator
   Read the configuration file configFilename if it exists.
configureLoggerFactory(Properties) - Method in class org.apache.log4j.PropertyConfigurator
   Check the provided Properties object for a LoggerFactory entry specified by PropertyConfigurator.LOGGER_FACTORY_KEY.
connection - Variable in class org.apache.log4j.jdbc.JDBCAppender
   Connection used by default.
ConsoleAppender() - Constructor for class org.apache.log4j.ConsoleAppender
   Creates a configured appender.
ConsoleAppender(Layout) - Constructor for class org.apache.log4j.ConsoleAppender
   Creates a configured appender.
ConsoleAppender(Layout, String) - Constructor for class org.apache.log4j.ConsoleAppender
   Creates a configured appender.
convert(LoggingEvent) - Method in class org.apache.log4j.helpers.PatternConverter
   Derived pattern converters must override this method in order to convert conversion specifiers in the correct way.
convertArg(String, Class) - Method in class org.apache.log4j.config.PropertySetter
   Convert val a String parameter to an object of a given type.
convertSpecialChars(String) - Static method in class org.apache.log4j.helpers.OptionConverter

count - Variable in class org.apache.log4j.helpers.CountingQuietWriter
Counts the number of bytes written.

**CountingQuietWriter(Writer, ErrorHandler)** - Constructor for class org.apache.log4j.helpers.CountingQuietWriter

**createPatternParser(String)** - Method in class org.apache.log4j.PatternLayout
Returns PatternParser used to parse the conversion string.

**createSession()** - Method in class org.apache.log4j.net.SMTPAppender
Create mail session.

**createWriter(OutputStream)** - Method in class org.apache.log4j.WriterAppender
Returns an OutputStreamWriter when passed an OutputStream.

**currentLiteral** - Variable in class org.apache.log4j.helpers.PatternParser

**CyclicBuffer** - class org.apache.log4j.helpers.CyclicBuffer.
CyclicBuffer is used by other appenders to hold LoggingEvents for immediate or differed display.

**CyclicBuffer(int)** - Constructor for class org.apache.log4j.helpers.CyclicBuffer
Instantiates a new CyclicBuffer of at most maxSize events.
DailyRollingFileAppender extends FileAppender so that the underlying file is rolled over at a user chosen frequency.

**DailyRollingFileAppender()** - Constructor for class org.apache.log4j.DailyRollingFileAppender
The default constructor does nothing.

**DailyRollingFileAppender(Layout, String, String)** - Constructor for class org.apache.log4j.DailyRollingFileAppender
Instantiate a DailyRollingFileAppender and open the file designated by filename.

**databasePassword** - Variable in class org.apache.log4j.jdbc.JDBCAppender
User to use for default connection handling

**databaseURL** - Variable in class org.apache.log4j.jdbc.JDBCAppender
URL of the DB for default connection handling

**databaseUser** - Variable in class org.apache.log4j.jdbc.JDBCAppender
User to connect as for default connection handling

**date** - Variable in class org.apache.log4j.helpers.DateLayout

**DATE_AND_TIME_DATE_FORMAT** - Static variable in class org.apache.log4j.helpers.AbsoluteTimeDateFormat
String constant used to specify DateTimeDateFormat in layouts.

**DATE_FORMAT_OPTION** - Static variable in class org.apache.log4j.helpers.DateLayout
Depreciated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

**dateFormat** - Variable in class org.apache.log4j.helpers.DateLayout

**dateFormat(StringBuffer, LoggingEvent)** - Method in class org.apache.log4j.helpers.DateLayout

**DateLayout** - class org.apache.log4j.helpers.DateLayout
This abstract layout takes care of all the date related options and
formatting work.

**DateLayout** - Constructor for class org.apache.log4j.helpers.DateLayout

**DateTimeDateFormat** - class org.apache.log4j.helpers.DateTimeDateFormat.

Formats a `Date` in the format "dd MMM yyyy HH:mm:ss,SSS" for example, "06 Nov 1994 15:49:37,459".

**DateTimeDateFormat** - Constructor for class org.apache.log4j.helpers.DateTimeDateFormat

**DateTimeDateFormat(TimeZone)** - Constructor for class org.apache.log4j.helpers.DateTimeDateFormat

**DEBUG** - Static variable in class org.apache.log4j.Priority

*Deprecated.* Use `Level.DEBUG` instead.

**DEBUG** - Static variable in class org.apache.log4j.Level

The `DEBUG` Level designates fine-grained informational events that are most useful to debug an application.

**DEBUG** - Static variable in class org.apache.log4j.lf5.LogLevel

**DEBUG_INT** - Static variable in class org.apache.log4j.Priority

**DEBUG_KEY** - Static variable in class org.apache.log4j.helpers.LogLog

Defining this value makes log4j print log4j-internal debug statements to `System.out`.

**debug(Object)** - Method in class org.apache.log4j.Category

Log a message object with the `DEBUG` level.

**debug(Object, Throwable)** - Method in class org.apache.log4j.Category

Log a message object with the `DEBUG` level including the stack trace of the `Throwables` passed as parameter.

**debug(String)** - Static method in class org.apache.log4j.helpers.LogLog

This method is used to output log4j internal debug statements.

**debug(String, Throwable)** - Static method in class org.apache.log4j.helpers.LogLog

This method is used to output log4j internal debug statements.

**debugEnabled** - Static variable in class org.apache.log4j.helpers.LogLog

**decide(LoggingEvent)** - Method in class org.apache.log4j.spi.Filter
If the decision is **DENY**, then the event will be dropped.

**decide(LoggingEvent)** - Method in class org.apache.log4j.varia.DenyAllFilter

- Always returns the integer constant **Filter.DENY** regardless of the **LoggingEvent** parameter.

**decide(LoggingEvent)** - Method in class org.apache.log4j.varia.StringMatchFilter

- Returns **Filter.NEUTRAL** if there is no string match.

**decide(LoggingEvent)** - Method in class org.apache.log4j.varia.LevelMatchFilter

- Return the decision of this filter.

**decide(LoggingEvent)** - Method in class org.apache.log4j.varia.LevelRangeFilter

- Return the decision of this filter.

**DEFAULT_BUFFER_SIZE** - Static variable in class org.apache.log4j.AsyncAppender

- The default buffer size is set to 128 events.

**DEFAULT_CONFIGURATION_FILE** - Static variable in class org.apache.log4j.LogManager

- **Deprecated.** *This variable is for internal use only. It will become package protected in future versions.*

**DEFAULT_CONFIGURATION_KEY** - Static variable in class org.apache.log4j.LogManager

- **Deprecated.** *This variable is for internal use only. It will become private in future versions.*

**DEFAULT_CONVERSION_PATTERN** - Static variable in class org.apache.log4j.PatternLayout

- Default pattern string for log output.

**DEFAULT_DELAY** - Static variable in class org.apache.log4j.helpers.FileWatchdog

- The default delay between every file modification check, set to 60 seconds.

**DEFAULT_INIT_OVERRIDE_KEY** - Static variable in class org.apache.log4j.LogManager

- **Deprecated.** *This variable is for internal use only. It will become private in future versions.*

**DefaultLF5Configurator** - class org.apache.log4j lf5.DefaultLF5Configurator

- The **DefaultLF5Configurator** provides a default configuration for the
**LF5Appender.**


**DefaultRepositorySelector(LoggerRepository)** - Constructor for class org.apache.log4j.spi.DefaultRepositorySelector

**delay** - Variable in class org.apache.log4j.helpers.FileWatchdog
   The delay to observe between every check.

**DENY** - Static variable in class org.apache.log4j.spi.Filter
   The log event must be dropped immediately without consulting with the remaining filters, if any, in the chain.

**DenyAllFilter** - class org.apache.log4j.varia.DenyAllFilter
   This filter drops all logging events.

**DenyAllFilter()** - Constructor for class org.apache.log4j.varia.DenyAllFilter

**doAppend(LoggingEvent)** - Method in class org.apache.log4j.AppenderSkeleton
   This method performs threshold checks and invokes filters before delegating actual logging to the subclasses specific AppenderSkeleton.append(org.apache.log4j.spi.LoggingEvent)
   method.

   Log in Appender specific way.


   Does not do anything.

**doCapitalize** - Variable in class org.apache.log4j.config.PropertyPrinter

**doConfigure(Element, LoggerRepository)** - Method in class org.apache.log4j.xml.DOMConfigurator
   Configure by taking in an DOM element.

**doConfigure(InputSource, LoggerRepository)** - Method in class org.apache.log4j.xml.DOMConfigurator
Configure log4j by reading in a log4j.dtd compliant XML configuration file.

**doConfigure(InputStream, LoggerRepository)** - Method in class org.apache.log4j.xml.DOMConfigurator
Configure log4j by reading in a log4j.dtd compliant XML configuration file.

Read configuration options from properties.

**doConfigure(Reader, LoggerRepository)** - Method in class org.apache.log4j.xml.DOMConfigurator
Configure log4j by reading in a log4j.dtd compliant XML configuration file.

**doConfigure(String, LoggerRepository)** - Method in class org.apache.log4j.PropertyConfigurator
Read configuration from a file.

**doConfigure(String, LoggerRepository)** - Method in class org.apache.log4j.xml.DOMConfigurator

**doConfigure(URL, LoggerRepository)** - Method in class org.apache.log4j.PropertyConfigurator
Read configuration options from url configURL.

**doConfigure(URL, LoggerRepository)** - Method in class org.apache.log4j.lf5.DefaultLF5Configurator
This is a dummy method that will throw an IllegalStateException if used.

Interpret a resource pointed by a URL and set up log4j accordingly.

**doConfigure(URL, LoggerRepository)** - Method in class org.apache.log4j.varia.ReloadingPropertyConfigurator

**doConfigure(URL, LoggerRepository)** - Method in class org.apache.log4j.xml.DOMConfigurator

**DOMConfigurator** - class org.apache.log4j.xml.DOMConfigurator
Use this class to initialize the log4j environment using a DOM tree.

**DOMConfigurator()** - Constructor for class org.apache.log4j.xml.DOMConfigurator
No argument constructor.


**doRender(Object)** - Method in interface org.apache.log4j.or.ObjectRenderer

Render the object passed as parameter as a String.

**doRender(Object)** - Method in class org.apache.log4j.or.ThreadGroupRenderer

Render a ThreadGroup object similar to the way that the ThreadGroup.list() method output information.

**doRender(Object)** - Method in class org.apache.log4j.or.jms.MessageRenderer

Render a Message.

**doRender(Object)** - Method in class org.apache.log4j.or.sax.AttributesRenderer

Render a Attributes.
emitNoAppenderWarning(Class) - Method in class org.apache.log4j.Hierarchy

emitNoAppenderWarning(Category) - Method in interface org.apache.log4j.spi.LoggerRepository

encoding - Variable in class org.apache.log4j.WriterAppender
   The encoding to use when writing.

encompasses(LogLevel) - Method in class org.apache.log4j.lf5.LogLevel
   Returns true if the level supplied is encompassed by this level.
encompasses(Object) - Method in class org.apache.log4j.Priority
   Two priorities are equal if their level fields are equal.
encompasses(Object) - Method in class org.apache.log4j.lf5.LogLevel

ERROR - Static variable in class org.apache.log4j.Priority
   Deprecated. Use Level.ERROR instead.
ERROR - Static variable in class org.apache.log4j.Priority
   The ERROR level designates error events that might still allow the application to continue running.
ERROR - Static variable in class org.apache.log4j.lf5.LogLevel

ERROR_INT - Static variable in class org.apache.log4j.Priority

class - Method in class org.apache.log4j.Category
   Log a message object with the ERROR Level.
class(Object, Throwable) - Method in class org.apache.log4j.Category
   Log a message object with the ERROR level including the stack trace of the Throwable t passed as parameter.
class(SAXParseException) - Method in class org.apache.log4j.xml.SAXErrorHandler
**error(SAXParseException)** - Method in class
org.apache.log4j.xml.examples.ReportParserError

**error(String)** - Static method in class org.apache.log4j.helpers.LogLog
This method is used to output log4j internal error statements.

**error(String)** - Method in class
org.apache.log4j.helpers.OnlyOnceErrorHandler
Print a the error message passed as parameter on System.err.

**error(String)** - Method in interface org.apache.log4j.spi.ErrorHandler
This method is normally used to just print the error message passed as a parameter.

**error(String)** - Method in class
org.apache.log4j.variaFallbackErrorHandler
Print a the error message passed as parameter on System.err.

**error(String, Exception, int)** - Method in class
org.apache.log4j.helpers.OnlyOnceErrorHandler
Prints the message and the stack trace of the exception on System.err.

**error(String, Exception, int)** - Method in interface
org.apache.log4j.spi.ErrorHandler
Equivalent to the ErrorHandler.error(String, Exception, int, LoggingEvent event) with the the event parameter set to null.

**error(String, Exception, int)** - Method in class
org.apache.log4j.variaFallbackErrorHandler
Prints the message and the stack trace of the exception on System.err.

**error(String, Exception, int, LoggingEvent)** - Method in class
org.apache.log4j.helpers.OnlyOnceErrorHandler
Prints the message and the stack trace of the exception on System.err.

**error(String, Exception, int, LoggingEvent)** - Method in interface
org.apache.log4j.spi.ErrorHandler
This method is invoked to handle the error.

**error(String, Exception, int, LoggingEvent)** - Method in class
org.apache.log4j.variaFallbackErrorHandler
Prints the message and the stack trace of the exception on System.err.

**error(String, Throwable)** - Static method in class
org.apache.log4j.helpers.LogLog
This method is used to output log4j internal error statements.

This interface defines commonly encountered error codes.

**errorHandler** - Variable in class org.apache.log4j.AppenderSkeleton
It is assumed and enforced that errorHandler is never null.

**errorHandler** - Variable in class org.apache.log4j.helpers.QuietWriter

Appenders may delegate their error handling to ErrorHandlers.

**escapeTags(String)** - Static method in class org.apache.log4j.helpers.Transform
This method takes a string which may contain HTML tags (ie, `<b>`, `<table>`, etc) and replaces any `'<` and `>'` characters with respective predefined entity references.

**evaluator** - Variable in class org.apache.log4j.net.SMTPAppender

**execute(String)** - Method in class org.apache.log4j.jdbc.JDBCAppender
Override this to provide an alertnate method of getting connections (such as caching).

**exists(String)** - Static method in class org.apache.log4j.Category
Deprecated. Please use LogManager.exists(java.lang.String) instead.

**exists(String)** - Static method in class org.apache.log4j.LogManager

**exists(String)** - Method in interface org.apache.log4j.spi.LoggerRepository
Check if the named logger exists in the hierarchy.

**ExternallyRolledFileAppender** - class org.apache.log4j.variaExternallyRolledFileAppender.
This appender listens on a socket on the port specified by the Port property for a "RollOver" message.

**ExternallyRolledFileAppender()** - Constructor for class org.apache.log4j.variaExternallyRolledFileAppender.
The default constructor does nothing but calls its super-class constructor.

**extractOption()** - Method in class
org.apache.log4j.helpers.PatternParser

extractPrecisionOption() - Method in class
org.apache.log4j.helpers.PatternParser

The option is expected to be in decimal and positive.
FACILITY_OI - Static variable in class org.apache.log4j.net.SyslogAppender

FallbackErrorHandler - class org.apache.log4j.varia.FallbackErrorHandler. The FallbackErrorHandler implements the ErrorHandler interface such that a secondary appender may be specified.

FallbackErrorHandler() - Constructor for class org.apache.log4j.varia.FallbackErrorHandler

FATAL - Static variable in class org.apache.log4j.Priority

Deprecated. Use Level.FATAL instead.

FATAL - Static variable in class org.apache.log4j.Level

The FATAL level designates very severe error events that will presumably lead the application to abort.

FATAL - Static variable in class org.apache.log4j.lf5.LogLevel

FATAL_INT - Static variable in class org.apache.log4j.Priority

fatal(Object) - Method in class org.apache.log4j.Category

Log a message object with the FATAL Level.

fatal(Object, Throwable) - Method in class org.apache.log4j.Category

Log a message object with the FATAL level including the stack trace of the Throwable t passed as parameter.

fatalError(SAXParseException) - Method in class org.apache.log4j.xml.SAXErrorHandler

FILE_OPEN_FAILURE - Static variable in interface org.apache.log4j.spi.ErrorCode

fileAppend - Variable in class org.apache.log4j.FileAppender

Controls file truncation.
**FileAppender** - class org.apache.log4j.FileAppender.
FileAppender appends log events to a file.

**FileAppender()** - Constructor for class org.apache.log4j.FileAppender
The default constructor does not do anything.

**FileAppender(Layout, String)** - Constructor for class org.apache.log4j.FileAppender
Instantiate a FileAppender and open the file designated by `filename`.

**FileAppender(Layout, String, boolean)** - Constructor for class org.apache.log4j.FileAppender
Instantiate a FileAppender and open the file designated by `filename`.

**FileAppender(Layout, String, boolean, boolean, int)** - Constructor for class org.apache.log4j.FileAppender
Instantiate a FileAppender and open the file designated by `filename`.

`filename` - Variable in class org.apache.log4j.helpers.FileWatchdog
The name of the file to observe for changes.

`fileName` - Variable in class org.apache.log4j.FileAppender
The name of the log file.

**FileWatchdog** - class org.apache.log4j.helpers.FileWatchdog
Check every now and then that a certain file has not changed.

**FileWatchdog(String)** - Constructor for class org.apache.log4j.helpers.FileWatchdog

**Filter** - class org.apache.log4j.spi.Filter.
Users should extend this class to implement customized logging event filtering.

**Filter()** - Constructor for class org.apache.log4j.spi.Filter

**finalize()** - Method in class org.apache.log4j.AppenderSkeleton
Finalize this appender by calling the derived class' `close` method.

**finalize()** - Method in class org.apache.log4j.jdbc.JDBCAppender
closes the appender before disposal

**finalize()** - Method in class org.apache.log4j.lf5.AppenderFinalizer

**finalize()** - Method in class org.apache.log4j.net.TelnetAppender.SocketHandler
make sure we close all network connections when this handler is destroyed.

**finalize()** - Method in class org.apache.log4j.nt.NTEventLogAppender
finalizeConverter(char) - Method in class org.apache.log4j.helpers.PatternParser

findAndRender(Object) - Method in class org.apache.log4j.or.RendererMap
   Find the appropriate renderer for the class type of the o parameter.

findAndSubst(String, Properties) - Static method in class org.apache.log4j.helpers.OptionConverter
   Find the value corresponding to key in props.

findAppenderByName(Document, String) - Method in class org.apache.log4j.xml.DOMConfigurator
   Used internally to parse appenders by IDREF name.

findAppenderByReference(Element) - Method in class org.apache.log4j.xml.DOMConfigurator
   Used internally to parse appenders by IDREF element.

FINE - Static variable in class org.apache.log4j.lf5.LogLevel

FINER - Static variable in class org.apache.log4j.lf5.LogLevel

FINEST - Static variable in class org.apache.log4j.lf5.LogLevel

fireAddAppenderEvent(Category, Appender) - Method in class org.apache.log4j.Hierarchy

fireAddAppenderEvent(Category, Appender) - Method in interface org.apache.log4j.spi.LoggerRepository

FLUSH_FAILURE - Static variable in interface org.apache.log4j.spi.ErrorCode

flush() - Method in class org.apache.log4j.helpers.QuietWriter

flush() - Method in class org.apache.log4j.helpers.SyslogWriter

flush() - Method in class org.apache.log4j.performance.NOPWriter

flushBuffer() - Method in class org.apache.log4j.jdbc.JDBCAppender
   loops through the buffer of LoggingEvents, gets a sql string from getLogStatement() and sends it to execute().
**forcedLog(String, Priority, Object, Throwable)** - Method in class org.apache.log4j.Category

This method creates a new logging event and logs the event without further checks.

**format(Date, StringBuffer, FieldPosition)** - Method in class org.apache.log4j.helpers.AbsoluteTimeDateFormat

Appends to `sbuf` the time in the format "HH:mm:ss,SSS" for example, "15:49:37,459"

**format(Date, StringBuffer, FieldPosition)** - Method in class org.apache.log4j.helpers.ISO8601DateFormat

Appends a date in the format "YYYY-mm-dd HH:mm:ss,SSS" to `sbuf`.

**format(Date, StringBuffer, FieldPosition)** - Method in class org.apache.log4j.helpers.DateTimeDateFormat

Appends to `sbuf` the date in the format "dd MMM yyyy HH:mm:ss,SSS" for example, "06 Nov 1994 08:49:37,459".

**format(Date, StringBuffer, FieldPosition)** - Method in class org.apache.log4j.helpers.RelativeTimeDateFormat

Appends to `sbuf` the number of milliseconds elapsed since the start of the application.

**format(LoggingEvent)** - Method in class org.apache.log4j.Layout

Implement this method to create your own layout format.

**format(LoggingEvent)** - Method in class org.apache.log4j.SimpleLayout

Returns the log statement in a format consisting of the `level`, followed by " - " and then the `message`.

**format(LoggingEvent)** - Method in class org.apache.log4j.TTCCLayout

In addition to the level of the statement and message, the returned byte array includes time, thread, category and `NDC` information.

**format(LoggingEvent)** - Method in class org.apache.log4j.PatternLayout

Produces a formatted string as specified by the conversion pattern.

**format(LoggingEvent)** - Method in class org.apache.log4j.HTMLLayout

**format(LoggingEvent)** - Method in class org.apache.log4j.xml.XMLLayout

Formats a `LoggingEvent` in conformance with the log4j.dtd.

**format(StringBuffer, LoggingEvent)** - Method in class org.apache.log4j.helpers.PatternConverter

A template method for formatting in a converter specific way.

**formattingInfo** - Variable in class
org.apache.log4j.helpers.PatternParser

**FormattingInfo** - class **org.apache.log4j.helpers.FormattingInfo**
FormattingInfo instances contain the information obtained when parsing formatting modifiers in conversion modifiers.

**FormattingInfo()** - Constructor for class **org.apache.log4j.helpers.FormattingInfo**

**foundProperty(Object, String, String, Object)** - Method in interface **org.apache.log4j.config.PropertyGetter.PropertyCallback**

**foundProperty(Object, String, String, Object)** - Method in class **org.apache.log4j.config.PropertyPrinter**

**fqnOfCategoryClass** - Variable in class **org.apache.log4j.spi.LoggingEvent**
Fully qualified name of the calling category class.

**fullInfo** - Variable in class **org.apache.log4j.spi.LocationInfo**
All available caller information, in the format
fully.qualified.classname.of.caller.methodName(Filename.java:line)
**genAppName()** - Method in class org.apache.log4j.config.PropertyPrinter

**GENERIC_FAILURE** - Static variable in interface org.apache.log4j.spi.ErrorCode

**get()** - Static method in class org.apache.log4j.NDC
  Never use this method directly, use the LoggingEvent.getNDC() method instead.

**get()** - Method in class org.apache.log4j.helpers.CyclicBuffer
  Get the oldest (first) element in the buffer.

**get()** - Method in class org.apache.log4j.helpers.BoundedFIFO
  Get the first element in the buffer.

**get(Class)** - Method in class org.apache.log4j.or.RendererMap
  Search the parents of clazz for a renderer.

**get(int)** - Method in class org.apache.log4j.helpers.CyclicBuffer
  Get the i-th oldest event currently in the buffer.

**get(Object)** - Method in class org.apache.log4j.or.RendererMap
  Syntactic sugar method that calls RendererMap.get(Class) with the class of the object parameter.

**get(String)** - Static method in class org.apache.log4j.MDC
  Get the context identified by the key parameter.

**getAcceptOnMatch()** - Method in class org.apache.log4j.varia.StringMatchFilter

**getAcceptOnMatch()** - Method in class org.apache.log4j.varia.LevelMatchFilter
  Get the value of the AcceptOnMatch option.

**getAdditivity()** - Method in class org.apache.log4j.Category
  Get the additivity flag for this Category instance.

**getAllAppenders()** - Method in class org.apache.log4j.Category
  Get the appenders contained in this category as an Enumeration.
getAllAppenders() - Method in class org.apache.log4j.AsyncAppender
Get iterator over attached appenders.

getAllAppenders() - Method in class
org.apache.log4j.helpers.AppenderAttachableImpl
Get all attached appenders as an Enumeration.

getAllAppenders() - Method in interface
org.apache.log4j.spi.AppenderAttachable
Get all previously added appenders as an Enumeration.

getAllDefaultLevels() - Static method in class
org.apache.log4j.lf5.LogLevel

getAllPossiblePriorities() - Static method in class
org.apache.log4j.Priority
Deprecated. This method will be removed with no replacement.

getAppend() - Method in class org.apache.log4j.FileAppender
Returns the value of the Append option.

getAppender(String) - Method in class org.apache.log4j.Category
Look for the appender named as name.

getAppender(String) - Method in class
org.apache.log4j.AsyncAppender
Get appender by name.

getAppender(String) - Method in class
org.apache.log4j.helpers.AppenderAttachableImpl
Look for an attached appender named as name.

getAppender(String) - Method in interface
org.apache.log4j.spi.AppenderAttachable
Get an appender by name.

getAttribute(String) - Method in class
org.apache.log4j.jmx.LoggerDynamicMBean

getAttribute(String) - Method in class
org.apache.log4j.jmx.LayoutDynamicMBean

getAttribute(String) - Method in class
org.apache.log4j.jmx.AppenderDynamicMBean

getAttribute(String) - Method in class
org.apache.log4j.jmx.HierarchyDynamicMBean
**getAttributes(String[])** - Method in class org.apache.log4j.mmx.**AbstractDynamicMBean**

Enables the to get the values of several attributes of the Dynamic MBean.

**getBcc()** - Method in class org.apache.log4j.net.**SMTPAppender**

Get the bcc recipient addresses.

**getBlocking()** - Method in class org.apache.log4j.**AsyncAppender**

Gets whether appender should block calling thread when buffer is full.

**getBufferedIO()** - Method in class org.apache.log4j.**FileAppender**

Get the value of the **BufferedIO** option.

**getBufferSize()** - Method in class org.apache.log4j.**FileAppender**

Get the size of the IO buffer.

**getBufferSize()** - Method in class org.apache.log4j.**AsyncAppender**

Gets the current buffer size.

**getBufferSize()** - Method in class org.apache.log4j.jdbc.**JDBCAppender**

**getBufferSize()** - Method in class org.apache.log4j.net.**SMTPAppender**

Returns value of the **BufferSize** option.

**getCategory()** - Method in class org.apache.log4j.lf5.**LogRecord**

Get the category associated with this LogRecord.

**getCategoryPrefixing()** - Method in class org.apache.log4j.**TTCCLayout**

Returns value of the **CategoryPrefixing** option.

**getCc()** - Method in class org.apache.log4j.net.**SMTPAppender**

Get the cc recipient addresses.

**getChainedLevel()** - Method in class org.apache.log4j.spi.**RootLogger**

Return the assigned level value without walking the logger hierarchy.

**getChainedLevel()** - Method in class org.apache.log4j.spi.**RootCategory**

Deprecated. Return the assigned level value without walking the category hierarchy.

**getChainedPriority()** - Method in class org.apache.log4j.**Category**

Deprecated. Please use the the **Category.getEffectiveLevel()** method instead.

**getClassName()** - Method in class org.apache.log4j.spi.**LocationInfo**

Return the fully qualified class name of the caller making the logging request.

**getConnection()** - Method in class org.apache.log4j.jdbc.**JDBCAppender**

Override this to link with your connection pooling system.

**getContentType()** - Method in class org.apache.log4j.**Layout**
Returns the content type output by this layout.

**getContentType()** - Method in class org.apache.log4j.HTMLLayout
Returns the content type output by this layout, i.e "text/html".

**getContext()** - Static method in class org.apache.log4j.MDC
Get the current thread's MDC as a hashtable.

**getContextPrinting()** - Method in class org.apache.log4j.TTCCLayout
Returns value of the **ContextPrinting** option.

**getConversionPattern()** - Method in class org.apache.log4j.PatternLayout
Returns the value of the **ConversionPattern** option.

**getCount()** - Method in class org.apache.log4j.helpers.CountingQuietWriter

**getCurrentCategories()** - Static method in class org.apache.log4j.Category
Deprecated. Please use LogManager.getCurrentLoggers() instead.

**getCurrentCategories()** - Method in class org.apache.log4j.Hierarchy
Deprecated. Please use Hierarchy.getCurrentLoggers() instead.

**getCurrentCategories()** - Method in class org.apache.log4j.spi.LoggerRepository
Deprecated.

**getCurrentLoggers()** - Static method in class org.apache.log4j.LogManager

**getCurrentLoggers()** - Method in class org.apache.log4j.Hierarchy
Returns all the currently defined categories in this hierarchy as an Enumeration.

**getCurrentLoggers()** - Method in class org.apache.log4j.spi.LoggerRepository

**getDateFormat()** - Method in class org.apache.log4j.helpers.DateLayout
Returns value of the **DateFormat** option.

**getDatePattern()** - Method in class org.apache.log4j.DailyRollingFileAppender
Returns the value of the **DatePattern** option.
org.apache.log4j.lf5.LF5Appender

**getDefaultMonitorHeight()** - Static method in class org.apache.log4j.lf5.LF5Appender

**getDefaultMonitorWidth()** - Static method in class org.apache.log4j.lf5.LF5Appender

**getDefaultRenderer()** - Method in class org.apache.log4j.or.RendererMap

**getDepth()** - Static method in class org.apache.log4j.NDC
   Get the current nesting depth of this diagnostic context.

**getEffectiveLevel()** - Method in class org.apache.log4j.Category
   Starting from this category, search the category hierarchy for a non-null level and return it.

**getEncoding()** - Method in class org.apache.log4j.WriterAppender

**getErrorHandler()** - Method in class org.apache.log4j.AppenderSkeleton
   Return the currently set ErrorHandler for this Appender.

**getErrorHandler()** - Method in interface org.apache.log4j.Appender
   Returns the ErrorHandler for this appender.

**getEvaluatorClass()** - Method in class org.apache.log4j.net.SMTPAppender
   Returns value of the EvaluatorClass option.

**getFacility()** - Method in class org.apache.log4j.net.SyslogAppender
   Returns the value of the Facility option.

**getFacility(String)** - Static method in class org.apache.log4j.net.SyslogAppender
   Returns the integer value corresponding to the named syslog facility, or -1 if it couldn't be recognized.

**getFacilityPrinting()** - Method in class org.apache.log4j.net.SyslogAppender
   Returns the value of the FacilityPrinting option.

**getFacilityString(int)** - Static method in class org.apache.log4j.net.SyslogAppender
   Returns the specified syslog facility as a lower-case String, e.g.

**getFile()** - Method in class org.apache.log4j.FileAppender
   Returns the value of the File option.
**getFileName()** - Method in class org.apache.log4j.spi.LocationInfo
Return the file name of the caller.

**getFilter()** - Method in class org.apache.log4j.AppenderSkeleton
Returns the head Filter.

**getFilter()** - Method in interface org.apache.log4j.Appender
Returns the head Filter.

**getFirstFilter()** - Method in class org.apache.log4j.AppenderSkeleton
Return the first filter in the filter chain for this Appender.

**getFollow()** - Method in class org.apache.log4j.ConsoleAppender
Gets whether the appender honors reassignments of System.out or System.err made after configuration.

**getFooter()** - Method in class org.apache.log4j.Layout
Returns the footer for the layout format.

**getFooter()** - Method in class org.apache.log4j.HTMLLayout
Returns the appropriate HTML footers.

**getFrom()** - Method in class org.apache.log4j.net.SMTPAppender
Returns value of the From option.

**getHeader()** - Method in class org.apache.log4j.Layout
Returns the header for the layout format.

**getHeader()** - Method in class org.apache.log4j.HTMLLayout
Returns appropriate HTML headers.

**getHierarchy()** - Method in class org.apache.log4j.Category
Deprecated. Please use Category.getLoggerRepository() instead.

**getImmediateFlush()** - Method in class org.apache.log4j.WriterAppender
Returns value of the ImmediateFlush option.

**getInitialContextFactoryName()** - Method in class org.apache.log4j.net.JMSAppender
Returns the value of the InitialContextFactoryName option.

**getInstance()** - Static method in class org.apache.log4j.helpers.NullEnumeration
Whenever you can, use this method to retrieve an instance instead of instantiating a new one with new.

**getInstance(Class)** - Static method in class org.apache.log4j.Category
Deprecated. Please make sure to use Logger.getLogger(Class) instead.

**getInstance(String)** - Static method in class org.apache.log4j.Category
Deprecated. Make sure to use `Logger.getLogger(String)` instead.

**getJdk14Levels()** - Static method in class `org.apache.log4j.lf5.LogLevel`
- Returns the Jdk14 levels.

**getLabel()** - Method in class `org.apache.log4j.lf5.LogLevel`
- Return the Label of the LogLevel.

**getLayout()** - Method in class `org.apache.log4j.AppenderSkeleton`
- Returns the layout of this appender.

**getLayout()** - Method in interface `org.apache.log4j.Appender`
- Returns this appenders layout.

**getLevel()** - Method in class `org.apache.log4j.Category`
- Returns the assigned `Level`, if any, for this Category.

**getLevel()** - Method in class `org.apache.log4j.lf5.LogRecord`
- Get the level of this LogRecord.

**getLevel()** - Method in class `org.apache.log4j.spi.LoggingEvent`
- Return the level of this event.

**getLevelMax()** - Method in class `org.apache.log4j.varia.LevelRangeFilter`
- Get the value of the `LevelMax` option.

**getLevelMin()** - Method in class `org.apache.log4j.varia.LevelRangeFilter`
- Get the value of the `LevelMin` option.

**getLevelToMatch()** - Method in class `org.apache.log4j.varia.LevelMatchFilter`
- Method for matching levels.

**getLineNumber()** - Method in class `org.apache.log4j.spi.LocationInfo`
- Returns the line number of the caller.

**getLocation()** - Method in class `org.apache.log4j.lf5.LogRecord`
- Get the location in code where this LogRecord originated.

**getLocationInfo()** - Method in class `org.apache.log4j.HTMLLayout`
- Returns the current value of the `LocationInfo` option.

**getLocationInfo()** - Method in class `org.apache.log4j.AsyncAppender`
- Gets whether the location of the logging request call should be captured.

**getLocationInfo()** - Method in class `org.apache.log4j.net.SocketHubAppender`
- Returns value of the `LocationInfo` option.

**getLocationInfo()** - Method in class `org.apache.log4j.net.SMTPAppender`
- Returns value of the `LocationInfo` option.

**getLocationInfo()** - Method in class `org.apache.log4j.net.JMSAppender`
- Returns value of the `LocationInfo` property which determines...
whether location (stack) info is sent to the remote subscriber.

**getLocationInfo()** - Method in class org.apache.log4j.net.SocketAppender
Returns value of the **LocationInfo** option.

**getLocationInfo()** - Method in class org.apache.log4j.xml.XMLLayout
Returns the current value of the **LocationInfo** option.

**getLocationInformation()** - Method in class org.apache.log4j.spi.LoggingEvent
Set the location information for this logging event.

**getLog4JLevels()** - Static method in class org.apache.log4j.lf5.LogLevel

**getLogBrokerMonitor()** - Method in class org.apache.log4j.lf5.LF5Appender

**getLogger()** - Method in class org.apache.log4j.jmx.AbstractDynamicMBean

**getLogger()** - Method in class org.apache.log4j.jmx.LoggerDynamicMBean

**getLogger()** - Method in class org.apache.log4j.jmx.LayoutDynamicMBean

**getLogger()** - Method in class org.apache.log4j.jmx.AppenderDynamicMBean

**getLogger()** - Method in class org.apache.log4j.jmx.HierarchyDynamicMBean

**getLogger(Class)** - Static method in class org.apache.log4j LogManager
Retrieve the appropriate **Logger** instance.

**getLogger(Class)** - Static method in class org.apache.log4j.Logger
Shorthand for getLogger(clazz.getName()).

**getLogger(String)** - Static method in class org.apache.log4j LogManager
Retrieve the appropriate **Logger** instance.

**getLogger(String)** - Static method in class org.apache.log4j.Logger
Retrieve a logger named according to the value of the name parameter.
**getLogger(String)** - Method in class org.apache.log4j.Hierarchy
Return a new logger instance named as the first parameter using the default factory.

**getLogger(String)** - Method in interface org.apache.log4j.spi.LoggerRepository

**getLogger(String, LoggerFactory)** - Static method in class org.apache.log4j.LogManager
Retrieve the appropriate Logger instance.

**getLogger(String, LoggerFactory)** - Static method in class org.apache.log4j.Hierarchy
Return a new logger instance named as the first parameter using the factory.

**getLogger(String, LoggerFactory)** - Method in interface org.apache.log4j.spi.LoggerRepository

**getLoggerName()** - Method in class org.apache.log4j.spi.LoggingEvent
Return the name of the logger.

**getLoggerRepository()** - Method in class org.apache.log4j.Category
Return the the LoggerRepository where this Category is attached.

**getLoggerRepository()** - Static method in class org.apache.log4j.spi.DefaultRepositorySelector

**getLoggerRepository()** - Method in interface org.apache.log4j.spi.RepositorySelector
Returns a LoggerRepository depending on the context.

**getLogLevelColorMap()** - Static method in class org.apache.log4j.lf5.LogLevel

**getLogStatement(LoggingEvent)** - Method in class
org.apache.log4j.jdbc.JDBCAppender
By default getLogStatement sends the event to the required Layout object.

getMaxBackupIndex() - Method in class org.apache.log4j.RollingFileAppender
Returns the value of the MaxBackupIndex option.

getMaximumFileSize() - Method in class org.apache.log4j.RollingFileAppender
Get the maximum size that the output file is allowed to reach before being rolled over to backup files.

getMaxSize() - Method in class org.apache.log4j.helpers.CyclicBuffer
Get the maximum size of the buffer.

getMBeanInfo() - Method in class org.apache.log4j.jmx.LoggerDynamicMBean

getMBeanInfo() - Method in class org.apache.log4j.jmx.LayoutDynamicMBean

getMBeanInfo() - Method in class org.apache.log4j.jmx.AppenderDynamicMBean

getMBeanInfo() - Method in class org.apache.log4j.jmx.HierarchyDynamicMBean

getMDC(String) - Method in class org.apache.log4j.spi.LoggingEvent
Returns the context corresponding to the key parameter.

getMDCCopy() - Method in class org.apache.log4j.spi.LoggingEvent
Obtain a copy of this thread's MDC prior to serialization or asynchronous logging.

getMessage() - Method in class org.apache.log4j.config.PropertySetterException
Returns descriptive text on the cause of this exception.

getMessage() - Method in class org.apache.log4j.lf5.LogRecord
Get the message associated with this LogRecord.

getMessage() - Method in class org.apache.log4j.spi.LoggingEvent
Return the message for this logging event.

getMethodName() - Method in class org.apache.log4j.spi.LocationInfo
Returns the method name of the caller.

**getMillis()** - Method in class org.apache.log4j.lf5.LogRecord
Get the event time of this record in milliseconds from 1970.

**getName()** - Method in class org.apache.log4j.Category
Return the category name.

**getName()** - Method in class org.apache.log4j.AppenderSkeleton
Returns the name of this FileAppender.

**getName()** - Method in class org.apache.log4j.Appender
Get the name of this appender.

**getNDC()** - Method in class org.apache.log4j.lf5.LogRecord
Get the NDC (nested diagnostic context) for this record.

**getNDC()** - Method in class org.apache.log4j.spi.LoggingEvent
This method returns the NDC for this event.

**getNext()** - Method in class org.apache.log4j.spi.Filter
Return the pointer to the next filter;

**getNextId()** - Static method in class org.apache.log4j.lf5.LogRecord

**getNotificationInfo()** - Method in class org.apache.log4j.jmx.HierarchyDynamicMBean

**getOptionStrings()** - Method in class org.apache.log4j.helpers.DateLayout
**Deprecated.** Use the setter method for the option directly instead of the generic setOption method.

**getOptionStrings()** - Method in class org.apache.log4j.varia.DenyAllFilter
**Deprecated.** We now use JavaBeans introspection to configure components. Options strings are no longer needed.

**getOptionStrings()** - Method in class org.apache.log4j.varia.StringMatchFilter
**Deprecated.** We now use JavaBeans introspection to configure components. Options strings are no longer needed.

**getParent()** - Method in class org.apache.log4j.Category
Returns the parent of this category.

**getPassword()** - Method in class org.apache.log4j.jdbc.JDBCAppender

**getPassword()** - Method in class org.apache.log4j.net.JMSAppender

**getPort()** - Method in class org.apache.log4j.net.SocketHubAppender
Returns value of the Port option.

**getPort()** - Method in class `org.apache.log4j.net.TelnetAppender`

**getPort()** - Method in class `org.apache.log4j.net.SocketAppender`

Returns value of the Port option.

**getPort()** - Method in class
`org.apache.log4j.variaExternallyRolledFileAppender`

Returns value of the Port option.

**getPrecedence()** - Method in class `org.apache.log4j.lf5.LogLevel`

**getPriority()** - Method in class `org.apache.log4j.Category`

*Deprecated.* Please use `Category.getLevel()` instead.

**getProperties(Object, PropertyGetter.PropertyCallback, String)** - Static method in class `org.apache.log4j.config.PropertyGetter`

**getProperties(PropertyGetter.PropertyCallback, String)** - Method in class
`org.apache.log4j.config.PropertyGetter`

**getPropertyDescriptor(String)** - Method in class
`org.apache.log4j.config.PropertySetter`

**getProviderURL()** - Method in class `org.apache.log4j.net.JMSAppender`

**getReconnectionDelay()** - Method in class
`org.apache.log4j.net.SocketAppender`

Returns value of the ReconnectionDelay option.

**getRemoteHost()** - Method in class
`org.apache.log4j.net.SocketAppender`

Returns value of the RemoteHost option.

**getRenderedMessage()** - Method in class
`org.apache.log4j.spi.LoggingEvent`

**getRendererMap()** - Method in class `org.apache.log4j.spi.Hierarchy`

Get the renderer map for this hierarchy.

**getRendererMap()** - Method in interface
`org.apache.log4j.spi.RendererSupport`

**getResource(String)** - Static method in class
`org.apache.log4j.helpers.Loader`
This method will search for resource in different places.

**get_resource(String, Class)** - Static method in class org.apache.log4j.helpers.Loader

*Deprecated. as of 1.2.*

**get_resourceBundle()** - Method in class org.apache.log4j.Category

Return the inherited ResourceBundle for this category.

**get_resourceBundleString(String)** - Method in class org.apache.log4j.Category

Returns the string resource corresponding to key in this category's inherited resource bundle.

**get_root()** - Static method in class org.apache.log4j.Category

*Deprecated. Please use Logger.getRootLogger() instead.*

**get_rootLogger()** - Static method in class org.apache.log4j LogManager

Retrieve the appropriate root logger.

**get_rootLogger()** - Static method in class org.apache.log4j.Logger

Return the root logger for the current logger repository.

**get_rootLogger()** - Method in class org.apache.log4j.Hierarchy

Get the root of this hierarchy.

**get_rootLogger()** - Method in interface org.apache.log4j.spi.LoggerRepository

**get_screenHeight()** - Static method in class org.apache.log4j.lf5.LF5Appender

**get_screenWidth()** - Static method in class org.apache.log4j.lf5.LF5Appender

**get_securityCredentials()** - Method in class org.apache.log4j.net.JMSAppender

**get_securityPrincipalName()** - Method in class org.apache.log4j.net.JMSAppender

**get_sequenceNumber()** - Method in class org.apache.log4j.lf5.LogRecord

Get the sequence number associated with this LogRecord.

**get_SMTPDebug()** - Method in class org.apache.log4j.net.SMTPAppender

Get SMTP debug.
getSMTPHost() - Method in class org.apache.log4j.net.SMTPAppender
   Returns value of the SMTPHost option.
getSMTPPassword() - Method in class org.apache.log4j.net.SMTPAppender
   Get SMTP password.
getSMTPUsername() - Method in class org.apache.log4j.net.SMTPAppender
   Get SMTP user name.
getSource() - Method in class org.apache.log4j.nt.NTEventLogAppender
getSql() - Method in class org.apache.log4j.jdbc.JDBCAppender
   Returns pre-formatted statement eg: insert into LogTable (msg) values ("%m")
getStartTime() - Static method in class org.apache.log4j.spi.LoggingEvent
   Returns the time when the application started, in milliseconds elapsed since 01.01.1970.
getStringToMatch() - Method in class org.apache.log4j.varia.StringMatchFilter
getSubject() - Method in class org.apache.log4j.net.SMTPAppender
   Returns value of the Subject option.
getSyslogEquivalent() - Method in class org.apache.log4j.Priority
   Return the syslog equivalent of this priority as an integer.
getSyslogHost() - Method in class org.apache.log4j.net.SyslogAppender
   Returns the value of the SyslogHost option.
getSystemProperty(String, String) - Static method in class org.apache.log4j.helpers.OptionConverter
   Very similar to System.getProperty except that the SecurityException is hidden.
getTarget() - Method in class org.apache.log4j.ConsoleAppender
   Returns the current value of the Target property.
getThreadDescription() - Method in class org.apache.log4j.lf5.LogRecord
   Get the thread description associated with this LogRecord.
getThreadName() - Method in class org.apache.log4j.spi.LoggingEvent
getThreadPrinting() - Method in class org.apache.log4j.TTCCLayout
   Returns value of the ThreadPrinting option.
**getThreshold()** - Method in class org.apache.log4j.AppenderSkeleton
  Returns this appenders threshold level.

**getThreshold()** - Method in class org.apache.log4j.Hierarchy
  Returns a Level representation of the enable state.

**getThreshold()** - Method in interface org.apache.log4j.spi.LoggerRepository
  Get the repository-wide threshold.

**getThrowable()** - Method in class org.apache.log4j.spi.ThrowableInformation

**getThrowableInformation()** - Method in class org.apache.log4j.spi.LoggingEvent
  Returns the throwable information contained within this event.

**getThrowableStrRep()** - Method in class org.apache.log4j.spi.ThrowableInformation
  Return this event's throwable's string[] representaion.

**getThrown()** - Method in class org.apache.log4j.lf5.LogRecord
  Get the Throwable associated with this LogRecord.

**getThrownStackTrace()** - Method in class org.apache.log4j.lf5.LogRecord
  Get the stack trace in a String-based format for the associated Throwable of this LogRecord.

**getTimeZone()** - Method in class org.apache.log4j.helpers.DateLayout
  Returns value of the TimeZone option.

**getTitle()** - Method in class org.apache.log4j.HTMLLayout
  Returns the current value of the Title option.

**getTo()** - Method in class org.apache.log4j.net.SMTPAppender
  Returns value of the To option.

**getTopicBindingName()** - Method in class org.apache.log4j.net.JMSAppender
  Returns the value of the TopicBindingName option.

**getTopicConnection()** - Method in class org.apache.log4j.net.JMSAppender
  Returns the TopicConnection used for this appender.

**getTopicConnectionFactoryBindingName()** - Method in class org.apache.log4j.net.JMSAppender
Returns the value of the `TopicConnectionFactoryBindingName` option.

**getTopicPublisher()** - Method in class `org.apache.log4j.net.JMSAppender`

Returns the TopicPublisher used for this appender.

**getTopicSession()** - Method in class `org.apache.log4j.net.JMSAppender`

Returns the TopicSession used for this appender.

**getURL()** - Method in class `org.apache.log4j.jdbc.JDBCApender`

**getUser()** - Method in class `org.apache.log4j.jdbc.JDBCApender`

**getUserName()** - Method in class `org.apache.log4j.net.JMSAppender`
handleNotification(Notification, Object) - Method in class org.apache.log4j.jmx.LoggerDynamicMBean

hashCode() - Method in class org.apache.log4j.lf5.LogLevel

hasMoreElements() - Method in class org.apache.log4j.helpers.NullEnumeration

hasThrown() - Method in class org.apache.log4j.lf5.LogRecord

headFilter - Variable in class org.apache.log4j.AppenderSkeleton
  The first filter in the filter chain.

  This class is specialized in retrieving loggers by name and also maintaining the logger hierarchy.

Hierarchy(Logger) - Constructor for class org.apache.log4j.Hierarchy.
  Create a new logger hierarchy.


HierarchyDynamicMBean() - Constructor for class org.apache.log4j.jmx.HierarchyDynamicMBean

  Listen to events occuring within a Hierarchy.

HTMLLayout - class org.apache.log4j.HTMLLayout.
  This layout outputs events in a HTML table.

HTMLLayout() - Constructor for class org.apache.log4j.HTMLLayout
**i** - Variable in class org.apache.log4j.helpers.PatternParser

**ignoresThrowable()** - Method in class org.apache.log4j.Layout
If the layout handles the throwable object contained within LoggingEvent, then the layout should return false.

**ignoresThrowable()** - Method in class org.apache.log4j.SimpleLayout
The SimpleLayout does not handle the throwable contained within LoggingEvents.

**ignoresThrowable()** - Method in class org.apache.log4j.TTCCLayout
The TTCCLayout does not handle the throwable contained within LoggingEvents.

**ignoresThrowable()** - Method in class org.apache.log4j.PatternLayout
The PatternLayout does not handle the throwable contained within LoggingEvents.

**ignoresThrowable()** - Method in class org.apache.log4j.HTMLLayout
The HTML layout handles the throwable contained in logging events.

**ignoresThrowable()** - Method in class org.apache.log4j.xml.XMLLayout
The XMLLayout prints and does not ignore exceptions.

**immediateFlush** - Variable in class org.apache.log4j.WriterAppender
Immediate flush means that the underlying writer or output stream will be flushed at the end of each append operation.

**INFO** - Static variable in class org.apache.log4j.Priority
Deprecated. *Use Level.INFO instead.*

**INFO** - Static variable in class org.apache.log4j.Level
The INFO level designates informational messages that highlight the progress of the application at coarse-grained level.

**INFO** - Static variable in class org.apache.log4j.lf5.LogLevel

**INFO_INT** - Static variable in class org.apache.log4j.Priority

**info(Object)** - Method in class org.apache.log4j.Category
Log a message object with the INFO Level.

**info(Object, Throwable)** - Method in class org.apache.log4j.Category
Log a message object with the INFO level including the stack trace of the Throwable t passed as parameter.
**inherit(Stack)** - Static method in class org.apache.log4j.NDC
Inherit the diagnostic context of another thread.

**INHERITED** - Static variable in interface org.apache.log4j.spi.Configurator
Special level value signifying inherited behaviour.

**instantiateByClassName(String, Class, Object)** - Static method in class org.apache.log4j.helpers.OptionConverter
Instantiate an object given a class name.

**instantiateByKey(Properties, String, Class, Object)** - Static method in class org.apache.log4j.helpers.OptionConverter

**introspect()** - Method in class org.apache.log4j.config.PropertySetter
Uses JavaBeans Introspector to compute setters of object to be configured.

**invoke(String, Object[], String[])** - Method in class org.apache.log4j.jmx.LoggerDynamicMBean

**invoke(String, Object[], String[])** - Method in class org.apache.log4j.jmx.LayoutDynamicMBean

**invoke(String, Object[], String[])** - Method in class org.apache.log4j.jmx.AppenderDynamicMBean

**invoke(String, Object[], String[])** - Method in class org.apache.log4j.jmx.HierarchyDynamicMBean

**isAsSevereAsThreshold(Priority)** - Method in class org.apache.log4j.AppenderSkeleton
Check whether the message level is below the appender's threshold.

**isAttached(Appender)** - Method in class org.apache.log4j.Category
Is the appender passed as parameter attached to this category?

**isAttached(Appender)** - Method in class org.apache.log4j.AsyncAppender
Determines if specified appender is attached.

**isAttached(Appender)** - Method in class org.apache.log4j.helpers.AppenderAttachableImpl
Returns true if the specified appender is in the list of attached appenders, false otherwise.

**isAttached(Appender)** - Method in interface
org.apache.log4j.spi.AppenderAttachable

Returns true if the specified appender is in list of attached, false otherwise.

isDebugEnabled() - Method in class org.apache.log4j.Category

Check whether this category is enabled for the DEBUG Level.

isDisabled(int) - Method in class org.apache.log4j.Hierarchy

This method will return true if this repository is disabled for level object passed as parameter and false otherwise.

isDisabled(int) - Method in interface org.apache.log4j.spi.LoggerRepository

Returns whether this repository is disabled for a given level.

isEnabledFor(Priority) - Method in class org.apache.log4j.Category

Check whether this category is enabled for a given Level passed as parameter.

isFatal() - Method in class org.apache.log4j.lf5.LogRecord

isFull() - Method in class org.apache.log4j.helpers.BoundedFIFO

Return true if the buffer is full, that is, whether the number of elements in the buffer equals the buffer size.

isGenAppName(String) - Method in class org.apache.log4j.config.PropertyPrinter

Returns true if the specified appender name is considered to have been generated, that is, if it is of the form A[0-9]+.

isGreaterOrEqual(Priority) - Method in class org.apache.log4j.Priority

Returns true if this level has a higher or equal level than the level passed as argument, false otherwise.

isHandledType(Class) - Method in class org.apache.log4j.config.PropertyGetter

isInfoEnabled() - Method in class org.apache.log4j.Category

Check whether this category is enabled for the info Level.

isJava1() - Static method in class org.apache.log4j.helpers.Loader

Are we running under JDK 1.x?

ISO8601_DATE_FORMAT - Static variable in class org.apache.log4j.helpers.AbsoluteTimeDateFormat

String constant used to specify ISO8601DateFormat in layouts.

ISO8601DateFormat - class org.apache.log4j.helpers.ISO8601DateFormat

Formats a Date in the format "yyyy-MM-dd HH:mm:ss,SSS" for
example "1999-11-27 15:49:37,459".

ISO8601DateFormat() - Constructor for class org.apache.log4j.helpers.ISO8601DateFormat

ISO8601DateFormat(TimeZone) - Constructor for class org.apache.log4j.helpers.ISO8601DateFormat

isSevereLevel() - Method in class org.apache.log4j.lf5.LogRecord
Abstract method.

isSevereLevel() - Method in class org.apache.log4j.lf5.Log4JLogRecord
Determines which Priority levels will be displayed in colored font when the LogMonitorAppender renders this log message.

isTraceEnabled() - Method in class org.apache.log4j.Logger
Check whether this category is enabled for the TRACE Level.

isTriggeringEvent(LoggingEvent) - Method in interface org.apache.log4j.spi.TriggeringEventEvaluator
Is this the triggering event?
**JDBCAppender** - class org.apache.log4j.jdbc.$\texttt{JDBCAppender}.$

WARNING: This version of JDBCAppender is very likely to be completely replaced in the future.

**JDBCAppender()** - Constructor for class org.apache.log4j.jdbc.$\texttt{JDBCAppender}$

**JMSAppender** - class org.apache.log4j.net.$\texttt{JMSAppender}$.

A simple appender that publishes events to a JMS Topic.

**JMSAppender()** - Constructor for class org.apache.log4j.net.$\texttt{JMSAppender}$

**JMSSink** - class org.apache.log4j.net.$\texttt{JMSSink}$.

A simple application that consumes logging events sent by a $\texttt{JMSAppender}$.

**JMSSink(String, String, String, String)** - Constructor for class org.apache.log4j.net.$\texttt{JMSSink}$
**log**

`l7dlog(Priority, String, Object[], Throwable)` - Method in class org.apache.log4j.Category

Log a localized and parameterized message.

`l7dlog(Priority, String, Throwable)` - Method in class org.apache.log4j.Category

Log a localized message.

**layout** - Variable in class org.apache.log4j.AppenderSkeleton

The layout variable does not need to be set if the appender implementation has its own layout.

**Layout** - class org.apache.log4j.Layout

Extend this abstract class to create your own log layout format.

**Layout()** - Constructor for class org.apache.log4j.Layout


**LayoutDynamicMBean(Layout)** - Constructor for class org.apache.log4j.jmx.LayoutDynamicMBean

**layoutNames** - Variable in class org.apache.log4j.config.PropertyPrinter

**length()** - Method in class org.apache.log4j.helpers.CyclicBuffer

Get the number of elements in the buffer.

**length()** - Method in class org.apache.log4j.helpers.BoundedFIFO

Get the number of elements in the buffer.

**level** - Variable in class org.apache.log4j.Category

The assigned level of this category.

**level** - Variable in class org.apache.log4j.spi.LoggingEvent

*Deprecated.* This field will be marked as private in future releases. Please do not access it directly. Use the `LoggingEvent[level]` method instead.

**Level** - class org.apache.log4j.Level

Defines the minimum set of levels recognized by the system, that is OFF, FATAL, ERROR, WARN, INFO, DEBUG and ALL.

**Level(int, String, int)** - Constructor for class org.apache.log4j.Level
Instantiate a Level object.

**LevelMatchFilter** - class org.apache.log4j.varia.LevelMatchFilter. This is a very simple filter based on level matching.

**LevelMatchFilter()** - Constructor for class org.apache.log4j.varia.LevelMatchFilter

**LevelRangeFilter** - class org.apache.log4j.varia.LevelRangeFilter. This is a very simple filter based on level matching, which can be used to reject messages with priorities outside a certain range.

**LevelRangeFilter()** - Constructor for class org.apache.log4j.varia.LevelRangeFilter

**LF5Appender** - class org.apache.log4j.lf5.LF5Appender. LF5Appender logs events to a swing based logging console.

**LF5Appender()** - Constructor for class org.apache.log4j.lf5.LF5Appender

**LF5Appender(LogBrokerMonitor)** - Constructor for class org.apache.log4j.lf5.LF5Appender

**LINE_SEP** - Static variable in class org.apache.log4j.Layout

**LINE_SEP_LEN** - Static variable in class org.apache.log4j.Layout

**ListVsVector** - class org.apache.log4j.performance.ListVsVector. Compares the performance of looping through a list versus a Vector.

**ListVsVector()** - Constructor for class org.apache.log4j.performance.ListVsVector

**loadClass(String)** - Static method in class org.apache.log4j.helpers.Loader

If running under JDK 1.2 load the specified class using the Thread contextClassLoader if that fails try Class.forName.

**Loader** - class org.apache.log4j.helpers.Loader.

Load resources (or images) from various sources.

**Loader()** - Constructor for class org.apache.log4j.helpers.Loader

**LOCATION_INFO_OPTION** - Static variable in class
org.apache.log4j.\*HTMLLayout

**Deprecated.** Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

**LocationInfo** - class org.apache.log4j.spi.LocationInfo. The internal representation of caller location information.

**LocationInfo(Throwable, String)** - Constructor for class org.apache.log4j.spi.LocationInfo

- Instantiate location information based on a Throwable.

**LOG_AUTH** - Static variable in class org.apache.log4j.net.SyslogAppender

- security/authorization messages

**LOG_AUTHPRIV** - Static variable in class org.apache.log4j.net.SyslogAppender

- security/authorization messages (private)

**LOG_CRON** - Static variable in class org.apache.log4j.net.SyslogAppender

- clock daemon

**LOG_DAEMON** - Static variable in class org.apache.log4j.net.SyslogAppender

- System daemons

**LOG_FTP** - Static variable in class org.apache.log4j.net.SyslogAppender

- ftp daemon

**LOG_KERN** - Static variable in class org.apache.log4j.net.SyslogAppender

- Kernel messages

**LOG_LOCAL0** - Static variable in class org.apache.log4j.net.SyslogAppender

- reserved for local use

**LOG_LOCAL1** - Static variable in class org.apache.log4j.net.SyslogAppender

- reserved for local use

**LOG_LOCAL2** - Static variable in class org.apache.log4j.net.SyslogAppender

- reserved for local use

**LOG_LOCAL3** - Static variable in class org.apache.log4j.net.SyslogAppender

- reserved for local use

**LOG_LOCAL4** - Static variable in class
org.apache.log4j.net.SyslogAppender
reserved for local use
LOG_LOCAL5 - Static variable in class
org.apache.log4j.net.SyslogAppender
reserved for local use
LOG_LOCAL6 - Static variable in class
org.apache.log4j.net.SyslogAppender
reserved for local use
LOG_LOCAL7 - Static variable in class
org.apache.log4j.net.SyslogAppender
reserved for local use
LOG_LPR - Static variable in class org.apache.log4j.net.SyslogAppender
line printer subsystem
LOG_MAIL - Static variable in class
org.apache.log4j.net.SyslogAppender
Mail system
LOG_NEWS - Static variable in class
org.apache.log4j.net.SyslogAppender
network news subsystem
LOG_SYSLOG - Static variable in class
org.apache.log4j.net.SyslogAppender
messages generated internally by syslogd
LOG_USER - Static variable in class
org.apache.log4j.net.SyslogAppender
Random user-level messages
LOG_UUCP - Static variable in class
org.apache.log4j.net.SyslogAppender
UUCP subsystem
log(Priority, Object) - Method in class org.apache.log4j.Category
This generic form is intended to be used by wrappers.
log(Priority, Object, Throwable) - Method in class
org.apache.log4j.Category
This generic form is intended to be used by wrappers.
log(String, Priority, Object, Throwable) - Method in class
org.apache.log4j.Category
This is the most generic printing method.
An EntityResolver specifically designed to return log4j.dtd which is
embedded within the log4j jar file.
**Log4jEntityResolver()** - Constructor for class `org.apache.log4j.xml.Log4jEntityResolver`

**Log4JLogRecord** - class `org.apache.log4j.lf5.Log4JLogRecord`
A Log4JLogRecord encapsulates the details of your log4j LoggingEvent in a format usable by the LogBrokerMonitor.

**Log4JLogRecord()** - Constructor for class `org.apache.log4j.lf5.Log4JLogRecord`
Constructs an instance of a Log4JLogRecord.

**Logger** - class `org.apache.log4j.Logger`
This is the central class in the log4j package.

**LOGGER_FACTORY_KEY** - Static variable in class `org.apache.log4j.PropertyConfigurator`
Key for specifying the LoggerFactory.

**Logger(String)** - Constructor for class `org.apache.log4j.Logger`

**LoggerDynamicMBean** - class `org.apache.log4j.jmx.LoggerDynamicMBean`

**LoggerDynamicMBean(Logger)** - Constructor for class `org.apache.log4j.jmx.LoggerDynamicMBean`

**loggerFactory** - Variable in class `org.apache.log4j.PropertyConfigurator`

**LoggerFactory** - interface `org.apache.log4j.spi.LoggerFactory`
Implement this interface to create new instances of Logger or a sub-class of Logger.

**LoggerRepository** - interface `org.apache.log4j.spi.LoggerRepository`
A LoggerRepository is used to create and retrieve Loggers.

**LoggingEvent** - class `org.apache.log4j.spi.LoggingEvent`
The internal representation of logging events.

**LoggingEvent(String, Category, long, Priority, Object, Throwable)** - Constructor for class `org.apache.log4j.spi.LoggingEvent`
Instantiate a LoggingEvent from the supplied parameters.

**LoggingEvent(String, Category, Priority, Object, Throwable)** - Constructor for class `org.apache.log4j.spi.LoggingEvent`
Instantiate a LoggingEvent from the supplied parameters.

**LogLevel** - class `org.apache.log4j.lf5.LogLevel`
The LogLevel class defines a set of standard logging levels.
**LogLevel(String, int)** - Constructor for class org.apache.log4j.lf5LogLevel

**LogLevelFormatException** - exception org.apache.log4j.lf5LogLevelFormatException

Thrown to indicate that the client has attempted to convert a string to one the LogLevel types, but the string does not have the appropriate format.

**LogLevelFormatException(String)** - Constructor for class org.apache.log4j.lf5LogLevelFormatException

**LogLog** - class org.apache.log4j.helpersLogLog

This class used to output log statements from within the log4j package.

**LogLog()** - Constructor for class org.apache.log4j.helpersLogLog

**LogManager** - class org.apache.log4j LogManager

Use the LogManager class to retrieve Logger instances or to operate on the current LogManagerRepository.

**LogManager()** - Constructor for class org.apache.log4j LogManager

**LogRecord** - class org.apache.log4j.lf5LogRecord

**LogRecord()** - Constructor for class org.apache.log4j.lf5LogRecord

**LogRecordFilter** - interface org.apache.log4j.lf5LogRecordFilter

An interface for classes which filters LogRecords.

**lookup(Context, String)** - Method in class org.apache.log4j.netJMSAppender

**lookup(Context, String)** - Static method in class org.apache.log4j.netJMSSink
Main - class org.apache.log4j.chainsaw.Main
  The main application.
main(String[]) - Static method in class org.apache.log4j.chainsaw.Main
  The main method.
main(String[]) - Static method in class org.apache.log4j.config.PropertyPrinter
main(String[]) - Static method in class org.apache.log4j.lf5.LF5Appender
main(String[]) - Static method in class org.apache.log4j.lf5.StartLogFactor5
  Main - starts a an instance of the LogFactor5 console and configures
  the console settings.
main(String[]) - Static method in class org.apache.log4j.net.SocketServer
main(String[]) - Static method in class org.apache.log4j.net.SimpleSocketServer
main(String[]) - Static method in class org.apache.log4j.net.JMSSink
main(String[]) - Static method in class org.apache.log4j.performance.ListVsVector
main(String[]) - Static method in class org.apache.log4j.performance.SystemTime
main(String[]) - Static method in class org.apache.log4j.performance.NewVsSetLen
main(String[]) - Static method in class org.apache.log4j.varia.Roller
  Send a "RollOver" message to ExternallyRolledFileAppender on
  host and port.
main(String[]) - Static method in class org.apache.log4j.xml.examples.XMLSample
**makeNewLoggerInstance(String)** - Method in interface org.apache.log4j.spi.LoggerFactory

**MAX_CAPACITY** - Variable in class org.apache.log4j.PatternLayout

**MAX_CAPACITY** - Variable in class org.apache.log4j.HTMLLayout

**maxBackupIndex** - Variable in class org.apache.log4j.RollingFileAppender
   There is one backup file by default.

**maxFileSize** - Variable in class org.apache.log4j.RollingFileAppender
   The default maximum file size is 10MB.

**MDC** - class org.apache.log4j.MDC.
   The MDC class is similar to the NDC class except that it is based on a map instead of a stack.

**MessageRenderer** - class org.apache.log4j.or.jms.MessageRenderer.
   Render javax.jms.Message objects.

**MessageRenderer()** - Constructor for class org.apache.log4j.or.jms.MessageRenderer

**MISSING_LAYOUT** - Static variable in interface org.apache.log4j.spi.ErrorCode

**msg** - Variable in class org.apache.log4j.net.SMTPAppender
**NA** - Static variable in class org.apache.log4j.spi.LocationInfo
When location information is not available the constant NA is returned.

**name** - Variable in class org.apache.log4j.spi.Category
The name of this category.

**name** - Variable in class org.apache.log4j.spi.AppenderSkeleton
Appenders are named.

**NDC** - class org.apache.log4j.spi.NDC.
The NDC class implements *nested diagnostic contexts* as defined by Neil Harrison in the article "Patterns for Logging Diagnostic Messages" part of the book "*Pattern Languages of Program Design 3*" edited by Martin et al.

**NEUTRAL** - Static variable in class org.apache.log4j.spi.Filter
This filter is neutral with respect to the log event.

This program compares the cost of creating a new StringBuffer and converting it to a String versus keeping the same StringBuffer, setting its size to zero and then converting it to String.

**NewVsSetLen()** - Constructor for class org.apache.log4j.performance.NewVsSetLen

**next** - Variable in class org.apache.log4j.spi.Filter
*Deprecated. As of 1.2.12, use Filter.getNext() and Filter.setNext(org.apache.log4j.spi.Filter) instead*

**nextElement()** - Method in class org.apache.log4j.spi.NullEnumeration

**NOPWriter** - class org.apache.log4j.performance.NOPWriter.
Extends Writer with methods that return immediately without doing anything.

**NOPWriter()** - Constructor for class org.apache.log4j.performance.NOPWriter
**NTEventLogAppender** - class org.apache.log4j.nt.<u>NTEventLogAppender</u>
   Append to the NT event log system.

**NTEventLogAppender()** - Constructor for class org.apache.log4j.nt.<u>NTEventLogAppender</u>

**NTEventLogAppender(Layout)** - Constructor for class org.apache.log4j.nt.<u>NTEventLogAppender</u>

**NTEventLogAppender(String)** - Constructor for class org.apache.log4j.nt.<u>NTEventLogAppender</u>

**NTEventLogAppender(String, Layout)** - Constructor for class org.apache.log4j.nt.<u>NTEventLogAppender</u>

**NTEventLogAppender(String, String)** - Constructor for class org.apache.log4j.nt.<u>NTEventLogAppender</u>

**NTEventLogAppender(String, String, Layout)** - Constructor for class org.apache.log4j.nt.<u>NTEventLogAppender</u>

**NULL** - Static variable in interface org.apache.log4j.spi.<u>Configurator</u>
   Special level signifying inherited behaviour, same as <u>Configurator.INHERITED</u>.

**NULL_ARG** - Static variable in class org.apache.log4j.config.<u>PropertyGetter</u>

**NULL_DATE_FORMAT** - Static variable in class org.apache.log4j.helpers.<u>DateLayout</u>
   String constant designating no time information.

**NullAppender** - class org.apache.log4j.performance.<u>NullAppender</u>
   A bogus appender which calls the format method of its layout object but does not write the result anywhere.

**NullAppender()** - Constructor for class org.apache.log4j.performance.<u>NullAppender</u>
   A NullAppender merely exists, it never outputs a message to any device.
**NullAppender()** - Constructor for class org.apache.log4j.varia.*NullAppender*

**NullAppender(Layout)** - Constructor for class org.apache.log4j.performance.*NullAppender*


**numAppenders** - Variable in class org.apache.log4j.config.*PropertyPrinter*
obj - Variable in class org.apache.log4j.config.PropertySetter

obj - Variable in class org.apache.log4j.config.PropertyGetter

Implement this interface in order to render objects as strings.

OFF - Static variable in class org.apache.log4j.Level
The OFF has the highest possible rank and is intended to turn off logging.

OFF_INT - Static variable in class org.apache.log4j.Priority

OK - Static variable in class org.apache.log4j.varia.ExternallyRolledFileAppender
The string constant sent to acknowledge a roll over.

OnlyOnceErrorHandler - class org.apache.log4j.helpers.OnlyOnceErrorHandler.
The OnlyOnceErrorHandler implements log4j's default error handling policy which consists of emitting a message for the first error in an appender and ignoring all following errors.

OnlyOnceErrorHandler() - Constructor for class org.apache.log4j.helpers.OnlyOnceErrorHandler

onMessage(Message) - Method in class org.apache.log4j.net.JMSSink

OptionConverter - class org.apache.log4j.helpers.OptionConverter.
A convenience class to convert property values to specific types.

OptionHandler - interface org.apache.log4j.spi.OptionHandler.
A string based interface to configure package components.

org.apache.log4j - package org.apache.log4j
The main log4j package.

org.apache.log4j.chainsaw - package org.apache.log4j.chainsaw
Chainsaw is a GUI log viewer and filter for the log4j package.

org.apache.log4j.config - package org.apache.log4j.config
Package used in getting/setting component properties.

org.apache.log4j.helpers - package org.apache.log4j.log4j
This package is used internally.

**org.apache.log4j.jdbc** - package org.apache.log4j.jdbc
The JDBCAppender provides for sending log events to a database.

**org.apache.log4j.jmx** - package org.apache.log4j.jmx
This package lets you manage log4j settings using JMX.

**org.apache.log4j.lf5** - package org.apache.log4j.lf5

**org.apache.log4j.net** - package org.apache.log4j.net
Package for remote logging.

**org.apache.log4j.nt** - package org.apache.log4j.nt
Package for NT event logging.

**org.apache.log4j.or** - package org.apache.log4j.or
ObjectRenders are responsible for rendering messages depending on their class type.

**org.apache.log4j.or.jms** - package org.apache.log4j.or.jms
This package contains the MessageRenderer which renders objects of type javax.jms.Message.

**org.apache.log4j.or.sax** - package org.apache.log4j.or.sax
This package contains the AttributesRenderer which renders objects of class org.xml.sax.Attributes.

**org.apache.log4j.performance** - package org.apache.log4j.performance
Package to measure the performance of the different log4j components.

**org.apache.log4j.spi** - package org.apache.log4j.spi
Contains part of the System Programming Interface (SPI) needed to extend log4j.

**org.apache.log4j.varia** - package org.apache.log4j.varia
Contains various appenders, filters and other odds and ends.

**org.apache.log4j.xml** - package org.apache.log4j.xml
XML based components.

**org.apache.log4j.xml.examples** - package org.apache.log4j.xml.examples
Example usage of log4j with XML (including source code).

**out** - Variable in class org.apache.log4j.config.PropertyPrinter

**overrideAsNeeded(String)** - Method in class org.apache.log4j.Hierarchy
Deprecated. *Deprecated with no replacement.*
parent - Variable in class org.apache.log4j.Category
   The parent of this category.
parse() - Method in class org.apache.log4j.helpers.PatternParser
parse(Element) - Method in class org.apache.log4j.xml.DOMConfigurator
   Used internally to configure the log4j framework by parsing a DOM
tree of XML elements based on log4j.dtd.
parse(String, ParsePosition) - Method in class org.apache.log4j.helpers.AbsoluteTimeDateFormat
   This method does not do anything but return null.
parse(String, ParsePosition) - Method in class org.apache.log4j.helpers.ISO8601DateFormat
   This method does not do anything but return null.
parse(String, ParsePosition) - Method in class org.apache.log4j.helpers.DateTimeDateFormat
   This method does not do anything but return null.
parse(String, ParsePosition) - Method in class org.apache.log4j.helpers.RelativeTimeDateFormat
   This method does not do anything but return null.
parseAppender(Element) - Method in class org.apache.log4j.xml.DOMConfigurator
   Used internally to parse an appender element.
parseCategory(Element) - Method in class org.apache.log4j.xml.DOMConfigurator
   Used internally to parse an category element.
parseCategoryFactory(Element) - Method in class org.apache.log4j.xml.DOMConfigurator
   Used internally to parse the category factory element.
parseCatsAndRenderers(Properties, LoggerRepository) - Method in class org.apache.log4j.PropertyConfigurator
   Parse non-root elements, such non-root categories and renderers.
parseChildrenOfLoggerElement(Element, Logger, boolean) - Method in class org.apache.log4j.xml.DOMConfigurator
   Used internally to parse the children of a category element.
**parseErrorHandler(Element, Appender)** - Method in class org.apache.log4j.xml.DOMConfigurator
Used internally to parse an ErrorHandler element.

**parseFilters(Element, Appender)** - Method in class org.apache.log4j.xml.DOMConfigurator
Used internally to parse a filter element.

**parseLayout(Element)** - Method in class org.apache.log4j.xml.DOMConfigurator
Used internally to parse a layout element.

**parseLevel(Element, Logger, boolean)** - Method in class org.apache.log4j.xml.DOMConfigurator
Used internally to parse a level element.

**parseRenderer(Element)** - Method in class org.apache.log4j.xml.DOMConfigurator

**parseRoot(Element)** - Method in class org.apache.log4j.xml.DOMConfigurator
Used internally to parse the root category element.

**passes(LogRecord)** - Method in interface org.apache.log4j.lf5.LogRecordFilter

**passes(LogRecord)** - Method in class org.apache.log4j.lf5.PassingLogRecordFilter

**PassingLogRecordFilter** - class org.apache.log4j.lf5.PassingLogRecordFilter.
An implementation of LogRecordFilter which always returns true.

**PassingLogRecordFilter()** - Constructor for class org.apache.log4j.lf5.PassingLogRecordFilter

**pattern** - Variable in class org.apache.log4j.helpers.PatternParser

PatternConverter is an abstract class that provides the formatting functionality that derived classes need.

**PatternConverter()** - Constructor for class org.apache.log4j.helpers.PatternConverter

**PatternConverter(FormattingInfo)** - Constructor for class
org.apache.log4j.helpers.**PatternConverter**

**PatternLayout** - class org.apache.log4j.**PatternLayout**.
A flexible layout configurable with pattern string.
**PatternLayout()** - Constructor for class org.apache.log4j.**PatternLayout**
Constructs a PatternLayout using the DEFAULT_LAYOUT_PATTERN.
**PatternLayout(String)** - Constructor for class org.apache.log4j.**PatternLayout**
Constructs a PatternLayout using the supplied conversion pattern.
**patternLength** - Variable in class org.apache.log4j.helpers.**PatternParser**

**PatternParser** - class org.apache.log4j.helpers.**PatternParser**.
Most of the work of the **PatternLayout** class is delegated to the PatternParser class.
**PatternParser(String)** - Constructor for class org.apache.log4j.helpers.**PatternParser**

peek() - Static method in class org.apache.log4j.**NDC**
Looks at the last diagnostic context at the top of this NDC without removing it.
pop() - Static method in class org.apache.log4j.**NDC**
Clients should call this method before leaving a diagnostic context.
**PORT_PROP_NAME** - Static variable in class org.apache.log4j.chainsaw.**Main**
name of property for port name
**pos** - Variable in class org.apache.log4j.helpers.**DateLayout**

postDeregister() - Method in class org.apache.log4j.jmx.**AbstractDynamicMBean**

postRegister(Boolean) - Method in class org.apache.log4j.jmx.**AbstractDynamicMBean**

postRegister(Boolean) - Method in class org.apache.log4j.jmx.**LoggerDynamicMBean**

postRegister(Boolean) - Method in class org.apache.log4j.jmx.**HierarchyDynamicMBean**
**preDeregister()** - Method in class org.apache.log4j.jmx.AbstractDynamicMBean

**preRegister(MBeanServer, ObjectName)** - Method in class org.apache.log4j.jmx.AbstractDynamicMBean

**preRegister(MBeanServer, ObjectName)** - Method in class org.apache.log4j.jmx.AppenderDynamicMBean

**print(PrintWriter)** - Method in class org.apache.log4j.config.PropertyPrinter
  
  Prints the configuration of the default log4j hierarchy as a Java properties file on the specified Writer.

**printOptions(PrintWriter, Logger)** - Method in class org.apache.log4j.config.PropertyPrinter

**printOptions(PrintWriter, Object, String)** - Method in class org.apache.log4j.config.PropertyPrinter

**Priority** - class org.apache.log4j.Priority
  
  Refrain from using this class directly, use the Level class instead.

**Priority()** - Constructor for class org.apache.log4j.Priority
  
  Default constructor for deserialization.

**Priority(int, String, int)** - Constructor for class org.apache.log4j.Priority
  
  Instantiate a level object.

**PropertyConfigurator** - class org.apache.log4j.PropertyConfigurator
  
  Allows the configuration of log4j from an external file.

**PropertyConfigurator()** - Constructor for class org.apache.log4j.PropertyConfigurator

**PropertyGetter** - class org.apache.log4j.config.PropertyGetter
  
  Used for inferring configuration information for a log4j’s component.


**PropertyGetter(Object)** - Constructor for class org.apache.log4j.config.PropertyGetter
  
  Create a new PropertyGetter for the specified Object.
**PropertyPrinter** - class org.apache.log4j.config.PropertyPrinter
   Prints the configuration of the log4j default hierarchy (which needs to be auto-initialized) as a properties file on a PrintWriter.

**PropertyPrinter(PrintWriter)** - Constructor for class org.apache.log4j.config.PropertyPrinter

**PropertyPrinter(PrintWriter, boolean)** - Constructor for class org.apache.log4j.config.PropertyPrinter

**PropertySetter** - class org.apache.log4j.config.PropertySetter
   General purpose Object property setter.

**PropertySetter(Object)** - Constructor for class org.apache.log4j.config.PropertySetter
   Create a new PropertySetter for the specified Object.

   Thrown when an error is encountered whilst attempting to set a property using the PropertySetter utility class.

**PropertySetterException(String)** - Constructor for class org.apache.log4j.config.PropertySetterException

**PropertySetterException(Throwable)** - Constructor for class org.apache.log4j.config.PropertySetterException

**props** - Variable in class org.apache.log4j.config.PropertySetter

**props** - Variable in class org.apache.log4j.config.PropertyGetter

**push(String)** - Static method in class org.apache.log4j.NDC
   Push new diagnostic context information for the current thread.

**put(Class, ObjectRenderer)** - Method in class org.apache.log4j.or.RendererMap
   Register an ObjectRenderer for clazz.

   Place a LoggingEvent in the buffer.

**put(String, Object)** - Static method in class org.apache.log4j.MDC
   Put a context value (the o parameter) as identified with the key parameter into the current thread's context map.
QuietWriter - class org.apache.log4j.helpers.QuietWriter. QuietWriter does not throw exceptions when things go wrong.

QuietWriter(Writer, ErrorHandler) - Constructor for class org.apache.log4j.helpers.QuietWriter

qw - Variable in class org.apache.log4j.LoggerAppender. This is the quietWriter where we will write to.
register(List) - Static method in class org.apache.log4j.lf5.LogLevel

register(LogLevel) - Static method in class org.apache.log4j.lf5.LogLevel
  Registers a used defined LogLevel.

register(LogLevel[]) - Static method in class org.apache.log4j.lf5.LogLevel

registry - Variable in class org.apache.log4j.PropertyConfigurator
  Used internally to keep track of configured appenders.

RELATIVE_TIME_DATE_FORMAT - Static variable in class org.apache.log4j.helpers.DateLayout
  String constant designating relative time.

RelativeTimeDateFormat - class org.apache.log4j.helpers.RelativeTimeDateFormat.
  Formats a Date by printing the number of milliseconds elapsed since
  construction of the format.

RelativeTimeDateFormat() - Constructor for class org.apache.log4j.helpers.RelativeTimeDateFormat


ReloadingPropertyConfigurator() - Constructor for class org.apache.log4j.varia.ReloadingPropertyConfigurator

remove() - Static method in class org.apache.log4j.NDC
  Remove the diagnostic context for this thread.

remove(String) - Static method in class org.apache.log4j.MDC
  Remove the the context identified by the key parameter.

removeAllAppenders() - Method in class org.apache.log4j.Category
  Remove all previously added appenders from this Category instance.

removeAllAppenders() - Method in class org.apache.log4j.AsyncAppender
Removes and closes all attached appenders. 
**removeAllAppenders()** - Method in class 
org.apache.log4j.helpers.AppenderAttachableImpl
Remove and close all previously attached appenders. 
**removeAllAppenders()** - Method in interface 
org.apache.log4j.spi.AppenderAttachable
Remove all previously added appenders. 
**removeAppender(Appender)** - Method in class 
org.apache.log4j.Category
Remove the appender passed as parameter form the list of appenders. 
**removeAppender(Appender)** - Method in class 
org.apache.log4j.AsyncAppender
Removes an appender. 
**removeAppender(Appender)** - Method in class 
org.apache.log4j.helpers.AppenderAttachableImpl
Remove the appender passed as parameter form the list of attached appenders. 
**removeAppender(Appender)** - Method in interface 
org.apache.log4j.spi.AppenderAttachable
Remove the appender passed as parameter from the list of appenders. 
**removeAppender(String)** - Method in class org.apache.log4j.Category
Remove the appender with the name passed as parameter form the list of appenders. 
**removeAppender(String)** - Method in class 
org.apache.log4j.AsyncAppender
Remove appender by name. 
**removeAppender(String)** - Method in class 
org.apache.log4j.helpers.AppenderAttachableImpl
Remove the appender with the name passed as parameter form the list of appenders. 
**removeAppender(String)** - Method in interface 
org.apache.log4j.spi.AppenderAttachable
Remove the appender with the name passed as parameter from the list of appenders. 
**removeAppenderEvent(Category, Appender)** - Method in class 
org.apache.log4j.jmx.HierarchyDynamicMBean

**removeNotificationListener**(NotificationListener) - Method in class org.apache.log4j.jmx.HierarchyDynamicMBean

**removes** - Variable in class org.apache.log4j.jdbc.JDBCAppender
   Helper object for clearing out the buffer

**RendererMap** - class org.apache.log4j.or<RendererMap>
   Map class objects to an ObjectRenderer.

**RendererMap()** - Constructor for class org.apache.log4j.or<RendererMap>

**RendererSupport** - interface org.apache.log4j.spi<RendererSupport>

   This class is needed for validating a log4j.dtd derived XML file.


**repository** - Variable in class org.apache.log4j.Category

**RepositorySelector** - interface org.apache.log4j.spi<RepositorySelector>
   The LogManager uses one (and only one) RepositorySelector implementation to select the LoggerRepository for a particular application context.

**requiresLayout()** - Method in class org.apache.log4j.WriterAppender
   The WriterAppender requires a layout.

**requiresLayout()** - Method in interface org.apache.log4j.Appender
   Configurators call this method to determine if the appender requires a layout.

**requiresLayout()** - Method in class org.apache.log4j.lf5.LF5Appender
   Returns a value that indicates whether this appender requires a Layout.
requiresLayout() - Method in class org.apache.log4j.net.SocketHubAppender
The SocketHubAppender does not use a layout.

requiresLayout() - Method in class org.apache.log4j.net.SyslogAppender
The SyslogAppender requires a layout.

requiresLayout() - Method in class org.apache.log4j.net.SMTPAppender
The SMTPAppender requires a layout.

requiresLayout() - Method in class org.apache.log4j.net.JMSAppender
The JMSAppender sends serialized events and consequently does not require a layout.

requiresLayout() - Method in class org.apache.log4j.net.TelnetAppender
This appender requires a layout to format the text to the attached client(s).

requiresLayout() - Method in class org.apache.log4j.net.SocketAppender
The SocketAppender does not use a layout.

requiresLayout() - Method in class org.apache.log4j.nt.NTEventLogAppender
The NTEventLogAppender requires a layout.

This is a bogus appender but it still uses a layout.

requiresLayout() - Method in class org.apache.log4j.varia.NullAppender
NullAppenders do not need a layout.

reset() - Method in class org.apache.log4j.WriterAppender
Clear internal references to the writer and other variables.

reset() - Method in class org.apache.log4j.FileAppender
Close any previously opened file and call the parent's reset.

reset() - Method in class org.apache.log4j.lf5.PassingLogRecordFilter
Does nothing.

resetConfiguration() - Static method in class org.apache.log4j LogManager

resetConfiguration() - Method in class org.apache.log4j.Hierarchy
Reset all values contained in this hierarchy instance to their default.

resetConfiguration() - Static method in class org.apache.log4j.BasicConfigurator
Reset the default hierarchy to its default.
resetConfiguration() - Method in interface org.apache.log4j.spi.LoggerRepository

resetLogLevelColorMap() - Static method in class org.apache.log4j.lf5.LogLevel

resetSequenceNumber() - Static method in class org.apache.log4j.lf5.LogRecord
  Resets that sequence number to 0.

resize(int) - Method in class org.apache.log4j.helpers.CyclicBuffer
  Resize the cyclic buffer to newSize.

resize(int) - Method in class org.apache.log4j.helpers.BoundedFIFO
  Resize the buffer to a new size.

resolveEntity(String, String) - Method in class org.apache.log4j.xml.Log4jEntityResolver

resourceBundle - Variable in class org.apache.log4j.Category

ROLL_OVER - Static variable in class org.apache.log4j.varia.ExternallyRolledFileAppender
  The string constant sent to initiate a roll over.

Roller - class org.apache.log4j.varia.Roller
  A simple application to send roll over messages to a potentially remote ExternallyRolledFileAppender.

RollingFileAppender - class org.apache.log4j.RollingFileAppender
  RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.

RollingFileAppender() - Constructor for class org.apache.log4j.RollingFileAppender
  The default constructor simply calls its parents constructor.

RollingFileAppender(Layout, String) - Constructor for class org.apache.log4j.RollingFileAppender
  Instantiate a FileAppender and open the file designated by filename.

RollingFileAppender(Layout, String, boolean) - Constructor for class org.apache.log4j.RollingFileAppender
  Instantiate a RollingFileAppender and open the file designated by filename.

rollOver() - Method in class org.apache.log4j.RollingFileAppender
  Implements the usual roll over behaviour.
*Deprecated. Replaced by RootLogger.*

**RootCategory(Level)** - Constructor for class org.apache.log4j.spi.RootCategory  
*Deprecated. The root category names itself as "root".*

**rootCause** - Variable in class org.apache.log4j.config.PropertySetterException

RootLogger sits at the top of the logger hierarchy.

**RootLogger(Level)** - Constructor for class org.apache.log4j.spi.RootLogger  
The root logger names itself as "root".

**run()** - Method in class org.apache.log4j.helpers.FileWatchdog

**run()** - Method in class org.apache.log4j.net.SocketNode

**run()** - Method in class org.apache.log4j.net.TelnetAppender.SocketHandler  
Continually accepts client connections.
s - Static variable in class org.apache.log4j.performance. NullAppender

SAXErrorHandler - class org.apache.log4j.xml. SAXErrorHandler.

SAXErrorHandler() - Constructor for class org.apache.log4j.xml. SAXErrorHandler

selectAndConfigure(URL, String, LoggerRepository) - Static method in class org.apache.log4j.helpers. OptionConverter
  Configure log4j given a URL.

send(String) - Method in class org.apache.log4j.net. TelnetAppender.SocketHandler
  sends a message to each of the clients in telnet-friendly output.

sendBuffer() - Method in class org.apache.log4j.net. SMTPAppender
  Send the contents of the cyclic buffer as an e-mail message.

setAcceptOnMatch(boolean) - Method in class org.apache.log4j.varia. StringMatchFilter

setAcceptOnMatch(boolean) - Method in class org.apache.log4j.varia. LevelMatchFilter

setAcceptOnMatch(boolean) - Method in class org.apache.log4j.varia. LevelRangeFilter
  Set the AcceptOnMatch option.

setAdditivity(boolean) - Method in class org.apache.log4j. Category
  Set the additivity flag for this Category instance.

setAppend(boolean) - Method in class org.apache.log4j.FileAppender
  The Append option takes a boolean value.

setAppender(Appender) - Method in class org.apache.log4j.helpers. OnlyOnceErrorHandler
  Does not do anything.

setAppender(Appender) - Method in interface org.apache.log4j.spi. ErrorHandler
  Set the appender for which errors are handled.

setAppender(Appender) - Method in class
org.apache.log4j.varia.FallbackErrorHandler
  The appender to which this error handler is attached.

setAttribute(Attribute) - Method in class
org.apache.log4j.jmx.LoggerDynamicMBean

setAttribute(Attribute) - Method in class
org.apache.log4j.jmx.LayoutDynamicMBean

setAttribute(Attribute) - Method in class
org.apache.log4j.jmx.AppenderDynamicMBean

setAttribute(Attribute) - Method in class
org.apache.log4j.jmx.HierarchyDynamicMBean

setAttributes(AttributeList) - Method in class
org.apache.log4j.jmx.AbstractDynamicMBean
  Sets the values of several attributes of the Dynamic MBean, and returns the list of attributes that have been set.

setBackupAppender(Appender) - Method in class
org.apache.log4j.helpers.OnlyOnceErrorHandler
  Does not do anything.

setBackupAppender(Appender) - Method in interface
org.apache.log4j.spi.ErrorHandler
  Set the appender to falkback upon in case of failure.

setBackupAppender(Appender) - Method in class
org.apache.log4j.varia.FallbackErrorHandler
  Set the backup appender.

setBcc(String) - Method in class org.apache.log4j.net.SMTPAppender
  Set the bcc recipient addresses.

setBlocking(boolean) - Method in class
org.apache.log4j.AsyncAppender
  Sets whether appender should wait if there is no space available in the event buffer or immediately return.

setBufferedIO(boolean) - Method in class
org.apache.log4j.FileAppender
  The BufferedIO option takes a boolean value.

setSize(int) - Method in class org.apache.log4j.FileAppender
  Set the size of the IO buffer.

setSize(int) - Method in class org.apache.log4j.AsyncAppender
Sets the number of messages allowed in the event buffer before the calling thread is blocked (if blocking is true) or until messages are summarized and discarded.

**setBufferSize(int)** - Method in class org.apache.log4j.jdbc.JDBCAppender

**setBufferSize(int)** - Method in class org.apache.log4j.net.SMTPAppender

The **BufferSize** option takes a positive integer representing the maximum number of logging events to collect in a cyclic buffer.

**setCallSystemExitOnClose(boolean)** - Method in class org.apache.log4j.lf5.LF5Appender

This method is used to set the property that controls whether the LogBrokerMonitor is hidden or closed when a user exits the monitor.

**setCategory(String)** - Method in class org.apache.log4j.lf5.LogRecord

Set the category associated with this LogRecord.

**setCategoryPrefixing(boolean)** - Method in class org.apache.log4j.TTCCLayout

The **CategoryPrefixing** option specifies whether the Category name is part of log output or not.

**setCc(String)** - Method in class org.apache.log4j.net.SMTPAppender

Set the cc recipient addresses.

**setContextPrinting(boolean)** - Method in class org.apache.log4j.TTCCLayout

The **ContextPrinting** option specifies log output will include the nested context information belonging to the current thread.

**setConversionPattern(String)** - Method in class org.apache.log4j.PatternLayout

Set the **ConversionPattern** option.

**setCount(long)** - Method in class org.apache.log4j.helpers.CountingQuietWriter

Sets the **DateFormat** used to format time and date in the zone determined by timeZone.

**setDateFormat(DateFormat, TimeZone)** - Method in class org.apache.log4j.helpers.DateLayout

Sets the **DateFormat** used to format time and date in the zone determined by timeZone.

**setDateFormat(String)** - Method in class org.apache.log4j.helpers.DateLayout

The value of the **DateFormat** option should be either an argument to
the constructor of SimpleDateFormat or one of the strings "NULL", "RELATIVE", "ABSOLUTE", "DATE" or "ISO8601.

**setDateFormat(String, TimeZone)** - Method in class org.apache.log4j.helpers.DateLayout

Sets the DateFormat used to format date and time in the time zone determined by timeZone parameter.

**setDatePattern(String)** - Method in class org.apache.log4j.DailyRollingFileAppender

The DatePattern takes a string in the same format as expected by SimpleDateFormat.

**setDelay(long)** - Method in class org.apache.log4j.helpers.FileWatchdog

Set the delay to observe between each check of the file changes.

**setDisableOverride(String)** - Method in class org.apache.log4j.Hierarchy

Deprecated. Deprecated with no replacement.

**setDriver(String)** - Method in class org.apache.log4j.jdbc.JDBCAppender

Ensures that the given driver class has been loaded for sql connection creation.

**setEncoding(String)** - Method in class org.apache.log4j.WriterAppender

**setErrorHandler(ErrorHandler)** - Method in class org.apache.log4j.AppenderSkeleton

Set the ErrorHandler for this Appender.

**setErrorHandler(ErrorHandler)** - Method in class org.apache.log4j.WriterAppender

Set the ErrorHandler for this WriterAppender and also the underlying QuietWriter if any.

**setErrorHandler(ErrorHandler)** - Method in class org.apache.log4j.Appender

Set the ErrorHandler for this appender.

**setEvaluatorClass(String)** - Method in class org.apache.log4j.net.SMTPAppender

The EvaluatorClass option takes a string value representing the name of the class implementing the TriggeringEventEvaluator interface.
**setFacility(String)** - Method in class org.apache.log4j.net.SyslogAppender
Set the syslog facility.

**setFacilityPrinting(boolean)** - Method in class org.apache.log4j.net.SyslogAppender
If the **FacilityPrinting** option is set to true, the printed message will include the facility name of the application.

**setFile(String)** - Method in class org.apache.log4j.FileAppender
The **File** property takes a string value which should be the name of the file to append to.

**setFile(String, boolean, boolean, int)** - Method in class org.apache.log4j.FileAppender
Sets and *opens* the file where the log output will go.

**setFile(String, boolean, boolean, int)** - Method in class org.apache.log4j.RollingFileAppender

**setFollow(boolean)** - Method in class org.apache.log4j.ConsoleAppender
Sets whether the appender honors reassignments of System.out or System.err made after configuration.

**setFrom(String)** - Method in class org.apache.log4j.net.SMTPAppender
The **From** option takes a string value which should be a e-mail address of the sender.

**setImmediateFlush(boolean)** - Method in class org.apache.log4j.WriterAppender
If the **ImmediateFlush** option is set to true, the appender will flush at the end of each write.

**setInitialContextFactoryName(String)** - Method in class org.apache.log4j.net.JMSAppender
Setting the **InitialContextFactoryName** method will cause this JMSAppender instance to use the `InitialContext.InitialContext(Hashtable)` method instead of the no-argument constructor.

**setInternalDebugging(boolean)** - Static method in class org.apache.log4j.helpers.LogLog
Allows to enable/disable log4j internal logging.

**setLayout(Layout)** - Method in class org.apache.log4j.AppenderSkeleton
Set the layout for this appender.
**setLayout(Layout)** - Method in interface org.apache.log4j.Appender
Set the Layout for this appender.

**setLevel(int)** - Method in class org.apache.log4j.helpers.SyslogQuietWriter

**setLevel(Level)** - Method in class org.apache.log4j.Category
Set the level of this Category.

**setLevel(Level)** - Method in class org.apache.log4j.spi.RootLogger
Setting a null value to the level of the root logger may have catastrophic results.

**setLevel(Level)** - Method in class org.apache.log4j.spi.RootCategory
**Deprecated.** Setting a null value to the level of the root category may have catastrophic results.

**setLevel(LogLevel)** - Method in class org.apache.log4j.lf5.LogRecord
Set the level of this LogRecord.

**setLevelMax(Level)** - Method in class org.apache.log4j.varia.LevelRangeFilter
Set the LevelMax option.

**setLevelMin(Level)** - Method in class org.apache.log4j.varia.LevelRangeFilter
Set the LevelMin option.

**setLevelToMatch(String)** - Method in class org.apache.log4j.varia.LevelMatchFilter

**setLocation(String)** - Method in class org.apache.log4j.lf5.LogRecord
Set the location in code where this LogRecord originated.

**setLocationInfo(boolean)** - Method in class org.apache.log4j.HTMLLayout
The LocationInfo option takes a boolean value.

**setLocationInfo(boolean)** - Method in class org.apache.log4j.AsyncAppender
The LocationInfo option takes a boolean value.

**setLocationInfo(boolean)** - Method in class org.apache.log4j.net.SocketHubAppender
The LocationInfo option takes a boolean value.

**setLocationInfo(boolean)** - Method in class org.apache.log4j.net.SMTPAppender
The LocationInfo option takes a boolean value.
org.apache.log4j.net.JMSAppender
   If true, the information sent to the remote subscriber will include caller's location information.

setLocationInfo(boolean) - Method in class
org.apache.log4j.net.SocketAppender
   The LocationInfo option takes a boolean value.

setLocationInfo(boolean) - Method in class
org.apache.log4j.xml.XMLLayout
   The LocationInfo option takes a boolean value.

setLogger(Logger) - Method in class
org.apache.log4j.helpers.OnlyOnceErrorHandler
   Does not do anything.

setLogger(Logger) - Method in interface
org.apache.log4j.spi.ErrorHandler
   Add a reference to a logger to which the failing appender might be attached to.

setLogger(Logger) - Method in class
org.apache.log4j.varia.FallbackErrorHandler
   Adds the logger passed as parameter to the list of loggers that we need to search for in case of appender failure.

setLogLevelColorMap(LogLevel, Color) - Method in class
org.apache.log4j.lf5.LogLevel

setMaxBackupIndex(int) - Method in class
org.apache.log4j.RollingFileAppender
   Set the maximum number of backup files to keep around.

setMaxDepth(int) - Static method in class org.apache.log4j.NDC
   Set maximum depth of this diagnostic context.

setMaxFileSize(String) - Method in class
org.apache.log4j.RollingFileAppender
   Set the maximum size that the output file is allowed to reach before being rolled over to backup files.

setMaximumFileSize(long) - Method in class
org.apache.log4j.RollingFileAppender
   Set the maximum size that the output file is allowed to reach before being rolled over to backup files.

setMaxNumberOfRecords(int) - Method in class
org.apache.log4j.lf5.LF5Appender
**setMessage(String)** - Method in class org.apache.log4j.lf5.LogRecord
    Set the message associated with this LogRecord.

**setMillis(long)** - Method in class org.apache.log4j.lf5.LogRecord
    Set the event time of this record.

**setName(String)** - Method in class org.apache.log4j.AppenderSkeleton
    Set the name of this Appender.

**setName(String)** - Method in interface org.apache.log4j.Appender
    Set the name of this appender.

**setNDC(String)** - Method in class org.apache.log4j.lf5.LogRecord
    Set the NDC (nested diagnostic context) for this record.

**setNext(Filter)** - Method in class org.apache.log4j.spi.Filter
    Set the next filter pointer.

**setOption(String, String)** - Method in class org.apache.log4j.helpers.DateLayout
    Deprecated. *Use the setter method for the option directly instead of the generic setOption method.*

**setOption(String, String)** - Method in class org.apache.log4j.varia.DenyAllFilter
    Deprecated. *Use the setter method for the option directly instead of the generic setOption method.*

**setOption(String, String)** - Method in class org.apache.log4j.varia.StringMatchFilter
    Deprecated. *Use the setter method for the option directly instead of the generic setOption method.*

**setParameter(Element, PropertySetter)** - Method in class org.apache.log4j.xml.DOMConfigurator

**setPassword(String)** - Method in class org.apache.log4j.jdbc.JDBCAppender

**setPassword(String)** - Method in class org.apache.log4j.net.JMSAppender
    The password to use when creating a topic session.

**setPort(int)** - Method in class org.apache.log4j.net.SocketHubAppender
    The Port option takes a positive integer representing the port where the server is waiting for connections.

**setPort(int)** - Method in class org.apache.log4j.net.TelnetAppender

**setPort(int)** - Method in class org.apache.log4j.net.SocketAppender
The **Port** option takes a positive integer representing the port where the server is waiting for connections.

**setPort(int)** - Method in class org.apache.log4j.varia.**ExternallyRolledFileAppender**

The **Port** property is used for setting the port for listening to external roll over messages.

**setPriority(Level)** - Method in class org.apache.log4j.spi.**RootCategory**

Deprecated.

**setPriority(Priority)** - Method in class org.apache.log4j.**Category**


**setProperties(Object, Properties, String)** - Static method in class org.apache.log4j.config.**PropertySetter**

Set the properties of an object passed as a parameter in one go.

**setProperties(Properties, String)** - Method in class org.apache.log4j.config.**PropertySetter**

Set the properites for the object that match the prefix passed as parameter.

**setProperty(PropertyDescriptor, String, String)** - Method in class org.apache.log4j.config.**PropertySetter**

Set the named property given a **PropertyDescriptor**.

**setProperty(String, String)** - Method in class org.apache.log4j.config.**PropertySetter**

Set a property on this PropertySetter's Object.

**setProviderURL(String)** - Method in class org.apache.log4j.net.**JMSAppender**

**setQuietMode(boolean)** - Static method in class org.apache.log4j.helpers.**LogLog**

In quite mode no LogLog generates strictly no output, not even for errors.

**setQWForFiles(Writer)** - Method in class org.apache.log4j.**FileAppender**

Sets the quiet writer being used.

**setQWForFiles(Writer)** - Method in class org.apache.log4j.**RollingFileAppender**

**setReconnectionDelay(int)** - Method in class org.apache.log4j.net.**SocketAppender**

The **ReconnectionDelay** option takes a positive integer
representing the number of milliseconds to wait between each failed connection attempt to the server.

**setRemoteHost(String)** - Method in class org.apache.log4j.net.SocketAppender

The **RemoteHost** option takes a string value which should be the host name of the server where a **SocketNode** is running.

**setRenderer(Class, ObjectRenderer)** - Method in class org.apache.log4j.Hierarchy

Used by subclasses to add a renderer to the hierarchy passed as parameter.

**setRenderer(Class, ObjectRenderer)** - Method in interface org.apache.log4j.spi.RendererSupport

**setRepositorySelector(RepositorySelector, Object)** - Static method in class org.apache.log4j.LogManager

Sets LoggerFactory but only if the correct **guard** is passed as parameter.

**setResourceBundle(ResourceBundle)** - Method in class org.apache.log4j.Category

Set the resource bundle to be used with localized logging methods **Category.17dlog(Priority, String, Throwable)** and **Category.17dlog(Priority, String, Object[], Throwable)**.

**setSecurityCredentials(String)** - Method in class org.apache.log4j.net.JMSAppender

**setSecurityPrincipalName(String)** - Method in class org.apache.log4j.net.JMSAppender

**setSequenceNumber(long)** - Method in class org.apache.log4j.lf5.LogRecord

Set the sequence number associated with this LogRecord.

**setSMTPDebug(boolean)** - Method in class org.apache.log4j.net.SMTPAppender

Setting the **SmtpDebug** option to true will cause the mail session to log its server interaction to stdout.

**setSMTPHost(String)** - Method in class org.apache.log4j.net.SMTPAppender

The **SMTPHost** option takes a string value which should be a the host name of the SMTP server that will send the e-mail message.
**setSMTPPassword(String)** - Method in class org.apache.log4j.net.SMTPAppender

The **SmtpPassword** option takes a string value which should be the password required to authenticate against the mail server.

**setSMTPUsername(String)** - Method in class org.apache.log4j.net.SMTPAppender

The **SmtpUsername** option takes a string value which should be the username required to authenticate against the mail server.

**setSource(String)** - Method in class org.apache.log4j.nt.NTEventLogAppender

The **Source** option which names the source of the event.

**setSql(String)** - Method in class org.apache.log4j.jdbc.JDBCAppender

**setStringToMatch(String)** - Method in class org.apache.log4j.varia.StringMatchFilter

**setSubject(String)** - Method in class org.apache.log4j.net.SMTPAppender

The **Subject** option takes a string value which should be a the subject of the e-mail message.

**setSyslogFacility(int)** - Method in class org.apache.log4j.helpers.SyslogQuietWriter

**setSyslogHost(String)** - Method in class org.apache.log4j.net.SyslogAppender

The **SyslogHost** option is the name of the the syslog host where log output should go.

**setTarget(String)** - Method in class org.apache.log4j.ConsoleAppender

Sets the value of the **Target** option.

**setThreadDescription(String)** - Method in class org.apache.log4j.lf5.LogRecord

Set the thread description associated with this LogRecord.

**setThreadPrinting(boolean)** - Method in class org.apache.log4j.TTCCLayout

The **ThreadPrinting** option specifies whether the name of the current thread is part of log output or not.

**setThreshold(Level)** - Method in class org.apache.log4j.Hierarchy

Enable logging for logging requests with level 1 or higher.

**setThreshold(Level)** - Method in interface
org.apache.log4j.spi.LoggerRepository
    Set the repository-wide threshold.

**setThreshold(Priority)** - Method in class org.apache.log4j.AppenderSkeleton
    Set the threshold level.

**setThreshold(String)** - Method in class org.apache.log4j.Hierarchy
    The string form of `Hierarchy.setThreshold(Level)`.

**setThreshold(String)** - Method in interface org.apache.log4j.spi.LoggerRepository
    Another form of `LoggerRepository.setThreshold(Level)` accepting a string parameter instead of a `Level`.

**setThrown(Throwable)** - Method in class org.apache.log4j.lf5.LogRecord
    Set the Throwable associated with this LogRecord.

**setThrownStackTrace(String)** - Method in class org.apache.log4j.lf5.LogRecord
    Set the ThrownStackTrace for the log record.

**setThrownStackTrace(ThrowableInformation)** - Method in class org.apache.log4j.lf5.Log4JLogRecord
    Set stack trace information associated with this Log4JLogRecord.

**setTimeZone(String)** - Method in class org.apache.log4j.helpers.DateLayout
    The `TimeZoneID` option is a time zone ID string in the format expected by the `TimeZone.getTimeZone(java.lang.String)` method.

**setTitle(String)** - Method in class org.apache.log4j.HTMLLayout
    The `Title` option takes a String value.

**setTo(String)** - Method in class org.apache.log4j.net.SMTPAppender
    The `To` option takes a string value which should be a comma separated list of e-mail address of the recipients.

**setTopicBindingName(String)** - Method in class org.apache.log4j.net.JMSAppender
    The `TopicBindingName` option takes a string value.

**setTopicConnectionFactoryBindingName(String)** - Method in class org.apache.log4j.net.JMSAppender
    The `TopicConnectionFactoryBindingName` option takes a string value.

**setURL(String)** - Method in class org.apache.log4j.jdbc.JDBCAppender

**setURLPkgPrefixes(String)** - Method in class
org.apache.log4j.net.**JMSAppender**

**setUser(String)** - Method in class org.apache.log4j.jdbc.**JDBCAppender**

**setUserName(String)** - Method in class org.apache.log4j.net.**JMSAppender**

The user name to use when creating a topic session.

**setWriter(Writer)** - Method in class org.apache.log4j.**WriterAppender**

Sets the Writer where the log output will go.

**SEVERE** - Static variable in class org.apache.log4j.net.**LogLevel**

**shutdown()** - Static method in class org.apache.log4j.**Category**

*Deprecated. Please use LogManager.shutdown() instead.*

**shutdown()** - Static method in class org.apache.log4j.**LogManager**

**shutdown()** - Method in class org.apache.log4j.**Hierarchy**

Shutting down a hierarchy will safely close and remove all appenders in all categories including the root logger.

**shutdown()** - Method in interface org.apache.log4j.spl.**LoggerRepository**

**SimpleLayout** - class org.apache.log4j.**SimpleLayout**.

SimpleLayout consists of the level of the log statement, followed by " - " and then the log message itself.

**SimpleLayout()** - Constructor for class org.apache.log4j.**SimpleLayout**

**SimpleSocketServer** - class org.apache.log4j.net.**SimpleSocketServer**.

A simple **SocketNode** based server.

**SimpleSocketServer()** - Constructor for class org.apache.log4j.net.**SimpleSocketServer**

**SMTPAppender** - class org.apache.log4j.net.**SMTPAppender**.

Send an e-mail when a specific logging event occurs, typically on errors or fatal errors.

**SMTPAppender()** - Constructor for class org.apache.log4j.net.**SMTPAppender**

The default constructor will instantiate the appender with a **TriggeringEventEvaluator** that will trigger on events with level ERROR or higher.

**SMTPAppender(TriggeringEventEvaluator)** - Constructor for class
org.apache.log4j.net.SMTPAppender
Use evaluator passed as parameter as the TriggeringEventEvaluator for this SMTPAppender.

SocketAppender - class org.apache.log4j.net.SocketAppender.
Sends LoggingEvent objects to a remote a log server, usually a SocketNode.

SocketAppender() - Constructor for class org.apache.log4j.net.SocketAppender

SocketAppender(InetAddress, int) - Constructor for class org.apache.log4j.net.SocketAppender
Connects to remote server at address and port.

SocketAppender(String, int) - Constructor for class org.apache.log4j.net.SocketAppender
Connects to remote server at host and port.

SocketHubAppender - class org.apache.log4j.net.SocketHubAppender.
Sends LoggingEvent objects to a set of remote log servers, usually a SocketNodes.

SocketHubAppender() - Constructor for class org.apache.log4j.net.SocketHubAppender

SocketHubAppender(int) - Constructor for class org.apache.log4j.net.SocketHubAppender
Connects to remote server at address and port.

Read LoggingEvent objects sent from a remote client using Sockets (TCP).

SocketNode(Socket, LoggerRepository) - Constructor for class org.apache.log4j.net.SocketNode

SocketServer - class org.apache.log4j.net.SocketServer.
A SocketNode based server that uses a different hierarchy for each client.

SocketServer(File) - Constructor for class org.apache.log4j.net.SocketServer

spacePad(StringBuffer, int) - Method in class org.apache.log4j.helpers.PatternConverter
Fast space padding method.
**sqlStatement** - Variable in class org.apache.log4j.jdbc.JDBCAppender
Stores the string given to the pattern layout for conversion into a SQL statement, eg: insert into LogTable (Thread, Class, Message) values ("%t", "%c", "%m").

**start()** - Method in class org.apache.log4j.jmx.Agent

Starts an instance of the LogFactor5 console for off-line viewing.

**StartLogFactor5()** - Constructor for class org.apache.log4j.lf5.StartLogFactor5

**startTime** - Variable in class org.apache.log4j.helpers.RelativeTimeDateFormat

**STRING_TO_MATCH_OPTION** - Static variable in class org.apache.log4j.varia.StringMatchFilter

  Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

**StringMatchFilter** - class org.apache.log4j.varia.StringMatchFilter
This is a very simple filter based on string matching.

**StringMatchFilter()** - Constructor for class org.apache.log4j.varia.StringMatchFilter

**subAppend(LoggingEvent)** - Method in class org.apache.log4j.WriterAppender
Actual writing occurs here.

**subAppend(LoggingEvent)** - Method in class org.apache.log4j.DailyRollingFileAppender
This method differentiates DailyRollingFileAppender from its super class.

**subAppend(LoggingEvent)** - Method in class org.apache.log4j.RollingFileAppender
This method differentiates RollingFileAppender from its super class.

**subst(String)** - Method in class org.apache.log4j.xml.DOMConfigurator
Perform variable substitution in string val from the values of keys
found in the system properties.

**SYSLOG_HOST_OI** - Static variable in class org.apache.log4j.net. **SyslogAppender**

**SyslogAppender** - class org.apache.log4j.net. **SyslogAppender**. Use SyslogAppender to send log messages to a remote syslog daemon.

**SyslogAppender()** - Constructor for class org.apache.log4j.net. **SyslogAppender**

**SyslogAppender(Layout, int)** - Constructor for class org.apache.log4j.net. **SyslogAppender**

**SyslogAppender(Layout, String, int)** - Constructor for class org.apache.log4j.net. **SyslogAppender**

**SyslogQuietWriter** - class org.apache.log4j.helpers. **SyslogQuietWriter**. SyslogQuietWriter extends QuietWriter by prepending the syslog level code before each printed String.

**SyslogQuietWriter(Writer, int, ErrorHandler)** - Constructor for class org.apache.log4j.helpers. **SyslogQuietWriter**

**SyslogWriter** - class org.apache.log4j.helpers. **SyslogWriter**. SyslogWriter is a wrapper around the java.net.DatagramSocket class so that it behaves like a java.io.Writer.

**SyslogWriter(String)** - Constructor for class org.apache.log4j.helpers. **SyslogWriter**

Constructs a new instance of SyslogWriter.

**SYSTEM_ERR** - Static variable in class org.apache.log4j. **ConsoleAppender**

**SYSTEM_OUT** - Static variable in class org.apache.log4j. **ConsoleAppender**

**SystemTime** - class org.apache.log4j.performance. **SystemTime**. Measures the time required to make a System.currentTimeMillis() and Thread.currentThread().getName() calls.

**SystemTime()** - Constructor for class org.apache.log4j.performance. **SystemTime**
t - Variable in class org.apache.log4j.performance.NullAppender

tailFilter - Variable in class org.apache.log4j.AppenderSkeleton. The last filter in the filter chain.
target - Variable in class org.apache.log4j.ConsoleAppender

TelnetAppender - class org.apache.log4j.net.TelnetAppender. The TelnetAppender is a log4j appender that specializes in writing to a read-only socket.

TelnetAppender.SocketHandler - class org.apache.log4j.net.TelnetAppender.SocketHandler. The SocketHandler class is used to accept connections from clients.

TelnetAppender.SocketHandler(TelnetAppender, int) - Constructor for class org.apache.log4j.net.TelnetAppender.SocketHandler

TelnetAppender() - Constructor for class org.apache.log4j.net.TelnetAppender

ThreadGroupRenderer - class org.apache.log4j.or.ThreadGroupRenderer. Render ThreadGroup objects in a format similar to the information output by the ThreadGroup.list() method.

ThreadGroupRenderer() - Constructor for class org.apache.log4j.or.ThreadGroupRenderer

ThreadLocalMap - class org.apache.log4j.helpers.ThreadLocalMap. ThreadLocalMap extends InheritableThreadLocal to bequeath a copy of the hashtable of the MDC of the parent thread.

ThreadLocalMap() - Constructor for class org.apache.log4j.helpers.ThreadLocalMap

threshold - Variable in class org.apache.log4j.AppenderSkeleton. There is no level threshold filtering by default.

ThrowableInformation - class org.apache.log4j.spi ThrowableInformation.
ThrowableInformation is log4j's internal representation of throwables.

**ThrowableInformation(Throwable)** - Constructor for class org.apache.log4j.spi.ThrowableInformation

**timeStamp** - Variable in class org.apache.log4j.spi.LoggingEvent
   The number of milliseconds elapsed from 1/1/1970 until logging event was created.

**TIMEZONE_OPTION** - Static variable in class org.apache.log4j.spi.
   Deprecated. Options are now handled using the JavaBeans paradigm. This constant is not longer needed and will be removed in the near term.

**TITLE_OPTION** - Static variable in class org.apache.log4j.
   A string constant used in naming the option for setting the the HTML document title.

**toBoolean(String, boolean)** - Static method in class org.apache.log4j.helpers.
   If value is "true", then true is returned.

**toFileSize(String, long)** - Static method in class org.apache.log4j.helpers.

**toInt()** - Method in class org.apache.log4j.
   Returns the integer representation of this level.

**toInt(String, int)** - Static method in class org.apache.log4j.helpers.

**toLevel(int)** - Static method in class org.apache.log4j.
   Convert an integer passed as argument to a level.

**toLevel(int, Level)** - Static method in class org.apache.log4j.
   Convert an integer passed as argument to a level.

**toLevel(String)** - Static method in class org.apache.log4j.
   Convert the string passed as argument to a level.

**toLevel(String, Level)** - Static method in class org.apache.log4j.
   Convert the string passed as argument to a level.

**toLevel(String, Level)** - Static method in class org.apache.log4j.helpers.
   Converts a standard or custom priority level to a Level object.

**toPriority(int)** - Static method in class org.apache.log4j.
   Deprecated. Please use the **Level.toLevel(int)** method instead.
**toPriority(int, Priority)** - Static method in class org.apache.log4j.Priority

Deprecated. Please use the `Level.toLevel(int, Level)` method instead.

**toPriority(String)** - Static method in class org.apache.log4j.Priority

Deprecated. Please use the `Level.toLevel(String)` method instead.

**toPriority(String, Priority)** - Static method in class org.apache.log4j.Priority

Deprecated. Please use the `Level.toLevel(String, Level)` method instead.

**toString()** - Method in class org.apache.log4j.Priority

Returns the string representation of this priority.

**toString()** - Method in class org.apache.log4j.lf5.LogLevel

**TRACE** - Static variable in class org.apache.log4j.Level

The TRACE Level designates finer-grained informational events than the DEBUG.

**TRACE_INT** - Static variable in class org.apache.log4j.Level

TRACE level integer value.

**trace(Object)** - Method in class org.apache.log4j.Logger

Log a message object with the TRACE level.

**trace(Object, Throwable)** - Method in class org.apache.log4j.Logger

Log a message object with the TRACE level including the stack trace of the Throwable passed as parameter.

**Transform** - class org.apache.log4j.helpers.Transform

Utility class for transforming strings.

**Transform()** - Constructor for class org.apache.log4j.helpers.Transform

**TriggeringEventEvaluator** - interface

org.apache.log4j.spi.TriggeringEventEvaluator

Implementations of this interface allow certain appenders to decide when to perform an appender specific action.

**TTCC_CONVERSION_PATTERN** - Static variable in class org.apache.log4j.PatternLayout

A conversion pattern equivalent to the TTCCCLayout.

**TTCCLayout** - class org.apache.log4j.TTCCLayout

TTCC layout format consists of time, thread, category and nested diagnostic context information, hence the name.
**TTCCLayout()** - Constructor for class org.apache.log4j.TTCCLayout

Instantiate a TTCCLayout object with RelativeTimeDateFormat as the date formatter in the local time zone.

**TTCCLayout(String)** - Constructor for class org.apache.log4j.TTCCLayout

Instantiate a TTCCLayout object using the local time zone.
valueOf(String) - Static method in class org.apache.log4j.lf5LogLevel
Convert a log level label into a LogLevel object.
**WARN** - Static variable in class `org.apache.log4j.Priority`
    Deprecated. Use `Level.WARN` instead.

**WARN** - Static variable in class `org.apache.log4j.Level`
    The `WARN` level designates potentially harmful situations.

**WARN** - Static variable in class `org.apache.log4j.lf5.LogLevel`

**WARN_INT** - Static variable in class `org.apache.log4j.Priority`

**warn(Object)** - Method in class `org.apache.log4j.Category`
    Log a message object with the `WARN` Level.

**warn(Object, Throwable)** - Method in class `org.apache.log4j.Category`
    Log a message with the `WARN` level including the stack trace of the `Throwable` passed as parameter.

**warn(String)** - Static method in class `org.apache.log4j.helpers.LogLog`
    This method is used to output log4j internal warning statements.

**warn(String, Throwable)** - Static method in class `org.apache.log4j.helpers.LogLog`
    This method is used to output log4j internal warnings.

**WARNING** - Static variable in class `org.apache.log4j.lf5.LogLevel`

**warning(SAXParseException)** - Method in class `org.apache.log4j.xml.SAXErrorHandler`

**warning(SAXParseException)** - Method in class `org.apache.log4j.xml.examples.ReportParserError`

**wasEmpty()** - Method in class `org.apache.log4j.helpers.BoundedFIFO`
    Returns `true` if there is just one element in the buffer.

**wasFull()** - Method in class `org.apache.log4j.helpers.BoundedFIFO`
    Returns `true` if the number of elements in the buffer plus 1 equals the maximum buffer size, returns `false` otherwise.

**WRITE_FAILURE** - Static variable in interface `org.apache.log4j.spi.ErrorCode`

**write(char[])** - Method in class `org.apache.log4j.performance.NOPWriter`
**write(char[], int, int)** - Method in class org.apache.log4j.helpers.SyslogWriter

**write(char[], int, int)** - Method in class org.apache.log4j.performance.NOPWriter

**write(int)** - Method in class org.apache.log4j.performance.NOPWriter

**write(String)** - Method in class org.apache.log4j.helpers.QuietWriter

**write(String)** - Method in class org.apache.log4j.helpers.CountingQuietWriter

**write(String)** - Method in class org.apache.log4j.helpers.SyslogQuietWriter

**write(String)** - Method in class org.apache.log4j.performance.NOPWriter

**write(String, int, int)** - Method in class org.apache.log4j.performance.NOPWriter

**writeFooter()** - Method in class org.apache.log4j.WriterAppender
  Write a footer as produced by the embedded layout's Layout.getFooter() method.

**writeHeader()** - Method in class org.apache.log4j.WriterAppender
  Write a header as produced by the embedded layout's Layout.getHeader() method.

**WriterAppender** - class org.apache.log4j.WriterAppender
  WriterAppender appends log events to a Writer or an OutputStream depending on the user's choice.

**WriterAppender()** - Constructor for class org.apache.log4j.WriterAppender
  This default constructor does nothing.

**WriterAppender(Layout, OutputStream)** - Constructor for class org.apache.log4j.WriterAppender
  Instantiate a WriterAppender and set the output destination to a new
OutputStreamWriter initialized with os as its OutputStream.

WriterAppender(Layout, Writer) - Constructor for class org.apache.log4j.WriterAppender

Instantiate a WriterAppender and set the output destination to writer.
**X**

**XMLLayout** - class org.apache.log4j.xml.XMLLayout.
The output of the XMLLayout consists of a series of log4j:event elements as defined in the log4j.dtd.

**XMLLayout()** - Constructor for class org.apache.log4j.xml.XMLLayout

**XMLSample** - class org.apache.log4j.xml.examples.XMLSample.
This example code shows how to read an XML based configuration file using a DOM parser.

**XMLSample()** - Constructor for class org.apache.log4j.xml.examples.XMLSample
_category - Variable in class org.apache.log4j.lf5.LogRecord

defaultLogMonitor - Static variable in class org.apache.log4j.lf5.LF5Appender

defaultMonitor - Variable in class org.apache.log4j.lf5.AppenderFinalizer

finalizer - Static variable in class org.apache.log4j.lf5.LF5Appender

label - Variable in class org.apache.log4j.lf5.LogLevel

level - Variable in class org.apache.log4j.lf5.LogRecord

location - Variable in class org.apache.log4j.lf5.LogRecord

logMonitor - Variable in class org.apache.log4j.lf5.LF5Appender

message - Variable in class org.apache.log4j.lf5.LogRecord

millis - Variable in class org.apache.log4j.lf5.LogRecord

ndc - Variable in class org.apache.log4j.lf5.LogRecord

precedence - Variable in class org.apache.log4j.lf5.LogLevel

seqCount - Static variable in class org.apache.log4j.lf5.LogRecord

sequenceNumber - Variable in class org.apache.log4j.lf5.LogRecord

thread - Variable in class org.apache.log4j.lf5.LogRecord

thrown - Variable in class org.apache.log4j.lf5.LogRecord

thrownStackTrace - Variable in class org.apache.log4j.lf5.LogRecord
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
- Exceptions
- Errors

Class/Interface

Each class, interface, inner class and inner 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
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.

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.

This help file applies to API documentation generated using the standard doclet.

Copyright 2000-2005 Apache Software Foundation.
All Classes

AbsoluteTimeDateFormat
AbstractDynamicMBean
Agent
Appender
AppenderAttribute
AppenderAttributesResolver
AppenderSkeleton
AsyncAppender
AttributesRenderer
BasicConfigurator
BoundedFIFO
Category
Configurator
ConsoleAppender
CountingQuietWriter
CyclicBuffer
DailyRollingFileAppender
DateLayout
DateTimeDateFormat
DefaultLF5Configurator
DefaultRepositorySelector
DenyAllFilter
DOMConfigurator
ErrorMessage
ErrorHandler
ExternallyRolledFileAppender
FallbackErrorHandler
FileAppender
Fracture
FRAMES
NO FRAMES
Next Page
Overview
Package
Use
Tree
Deprecated
Index
PREV
EXCEPTION
NEXT
Use
Tree
Deprecated
Index
PREV
EXCEPTION
NEXT
Use
Tree
Deprecated
Index
PREV
EXCEPTION
NEXT
Uses of Interface
org.apache.log4j.Appender

<table>
<thead>
<tr>
<th>Packages that use Appender</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.jdbc</td>
<td>The JDBCAppender provides for sending log events to a database.</td>
</tr>
<tr>
<td>org.apache.log4j.jmx</td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
<tr>
<td>org.apache.log4j.lf5</td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for NT event logging.</td>
</tr>
<tr>
<td>org.apache.log4j.nt</td>
<td>Package to measure the performance of the different log4j components.</td>
</tr>
<tr>
<td>org.apache.log4j.performance</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

Uses of Appender in org.apache.log4j

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j that implement Appender</th>
</tr>
</thead>
<tbody>
<tr>
<td>class AppenderSkeleton</td>
</tr>
<tr>
<td>class AsyncAppender</td>
</tr>
</tbody>
</table>
### Class: `ConsoleAppender`
ConsoleAppender appends log events to `System.out` or `System.err` using a layout specified by the user.

### Class: `DailyRollingFileAppender`
DailyRollingFileAppender extends `FileAppender` so that the underlying file is rolled over at a user chosen frequency.

### Class: `FileAppender`
FileAppender appends log events to a file.

### Class: `RollingFileAppender`
RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.

### Class: `WriterAppender`
WriterAppender appends log events to a `Writer` or an `OutputStream` depending on the user's choice.

### Methods in `org.apache.log4j` that return `Appender`

<table>
<thead>
<tr>
<th>Appender</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Category</code></td>
<td><code>getAppender(String name)</code></td>
<td>Look for the appender named as <code>name</code>.</td>
</tr>
<tr>
<td><code>AsyncAppender</code></td>
<td><code>getAppender(String name)</code></td>
<td>Get appender by name.</td>
</tr>
</tbody>
</table>

### Methods in `org.apache.log4j` with parameters of type `Appender`

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>addAppender(Appender newAppender)</code></td>
<td>Add <code>newAppender</code> to the list of appenders of this <code>Category</code> instance.</td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>isAttached(Appender appender)</code></td>
<td>Is the appender passed as parameter attached to this <code>category</code>?</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>removeAppender(Appender appender)</code></td>
<td>Remove the appender passed as parameter form the list of appenders.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>fireAddAppenderEvent(Category logger, Appender appender)</code></td>
<td></td>
</tr>
</tbody>
</table>
static void BasicConfigurator.configure(Appender appender)
Add appender to the root category.

void AsyncAppender.addAppender(Appender newAppender)
Add appender.

boolean AsyncAppender.isAttached(Appender appender)
Determines if specified appender is attached.

void AsyncAppender.removeAppender(Appender appender)
Removes an appender.

Uses of Appender in org.apache.log4j.helpers

Methods in org.apache.log4j.helpers that return Appender

<table>
<thead>
<tr>
<th>Appender</th>
<th>AppenderAttachableImpl.getAppender(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Look for an attached appender named as name.</td>
</tr>
</tbody>
</table>

Methods in org.apache.log4j.helpers with parameters of type Appender

<table>
<thead>
<tr>
<th>void</th>
<th>AppenderAttachableImpl.addAppender(Appender newAppender)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attach an appender.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>AppenderAttachableImpl.isAttached(Appender appender)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns true if the specified appender is in the list of</td>
</tr>
<tr>
<td></td>
<td>attached appenders, false otherwise.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>AppenderAttachableImpl.removeAppender(Appender appender)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Remove the appender passed as parameter form the list</td>
</tr>
<tr>
<td></td>
<td>of attached appenders.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>OnlyOnceErrorHandler.setAppender(Appender appender)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Does not do anything.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>OnlyOnceErrorHandler.setBackupAppender(Appender appender)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Does not do anything.</td>
</tr>
</tbody>
</table>
### Uses of **Appender** in **org.apache.log4j.jdbc**

**Classes in** `org.apache.log4j.jdbc` **that implement** `Appender`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBCAppender</td>
<td>WARNING: This version of JDBCAppender is very likely to be completely replaced in the future.</td>
</tr>
</tbody>
</table>

### Uses of **Appender** in **org.apache.log4j.jmx**

**Methods in** `org.apache.log4j.jmx` **with parameters of type** `Appender`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HierarchyDynamicMBean.addAppenderEvent</code></td>
<td><code>Category logger, Appender appender</code></td>
</tr>
<tr>
<td><code>HierarchyDynamicMBean.removeAppenderEvent</code></td>
<td><code>Category cat, Appender appender</code></td>
</tr>
</tbody>
</table>

**Constructors in** `org.apache.log4j.jmx` **with parameters of type** `Appender`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppenderDynamicMBean(Appender appender)</code></td>
<td></td>
</tr>
</tbody>
</table>

### Uses of **Appender** in **org.apache.log4j.lf5**

**Classes in** `org.apache.log4j.lf5` **that implement** `Appender`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF5Appender</td>
<td>LF5Appender logs events to a swing based logging console.</td>
</tr>
</tbody>
</table>
# Uses of `Appender` in `org.apache.log4j.net`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JMSAppender</strong></td>
<td>A simple appender that publishes events to a JMS Topic.</td>
</tr>
<tr>
<td><strong>SMTPAppender</strong></td>
<td>Send an e-mail when a specific logging event occurs, typically on errors or fatal errors.</td>
</tr>
<tr>
<td><strong>SocketAppender</strong></td>
<td>Sends <code>LoggingEvent</code> objects to a remote a log server, usually a <code>SocketAddress</code>.</td>
</tr>
<tr>
<td><strong>SocketHubAppender</strong></td>
<td>Sends <code>LoggingEvent</code> objects to a set of remote log servers, usually a <code>SocketAddress</code>.</td>
</tr>
<tr>
<td><strong>SyslogAppender</strong></td>
<td>Use SyslogAppender to send log messages to a remote syslog daemon.</td>
</tr>
<tr>
<td><strong>TelnetAppender</strong></td>
<td>The TelnetAppender is a log4j appender that specializes in writing to a read-only socket.</td>
</tr>
</tbody>
</table>

# Uses of `Appender` in `org.apache.log4j.nt`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NTEventLogAppender</strong></td>
<td>Append to the NT event log system.</td>
</tr>
</tbody>
</table>
# Uses of `Appender` in `org.apache.log4j.performance`

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NullAppender</code></td>
<td>A bogus appender which calls the format method of its layout object but does not write the result anywhere.</td>
</tr>
</tbody>
</table>

# Uses of `Appender` in `org.apache.log4j.spi`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppenderAttachable.getAppender(String name)</code></td>
<td>Get an appender by name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppenderAttachable.addAppender(Appender newAppender)</code></td>
<td>Add an appender.</td>
</tr>
<tr>
<td><code>AppenderAttachable.isAttached(Appender appender)</code></td>
<td>Returns <code>true</code> if the specified appender is in list of attached, <code>false</code> otherwise.</td>
</tr>
<tr>
<td><code>AppenderAttachable.removeAppender(Appender appender)</code></td>
<td>Remove the appender passed as parameter from the list of appenders.</td>
</tr>
<tr>
<td><code>ErrorHandler.setAppender(Appender appender)</code></td>
<td>Set the appender for which errors are handled.</td>
</tr>
<tr>
<td><code>ErrorHandler.setBackupAppender(Appender appender)</code></td>
<td>Set the appender to falkback upon in case of failure.</td>
</tr>
<tr>
<td><code>LoggerRepository.fireAddAppenderEvent(Category logger, Appender appender)</code></td>
<td></td>
</tr>
</tbody>
</table>
void HierarchyEventListener.addAppenderEvent(Category cat, Appender appender)

void HierarchyEventListener.removeAppenderEvent(Category cat, Appender appender)

Uses of Appender in org.apache.log4j.varia

Classes in org.apache.log4j.varia that implement Appender

<table>
<thead>
<tr>
<th>class</th>
<th>ExternallyRolledFileAppender</th>
</tr>
</thead>
</table>
|       | This appender listens on a socket on the port specified by the Port property for a "RollOver" message.

Methods in org.apache.log4j.varia with parameters of type Appender

<table>
<thead>
<tr>
<th>void</th>
<th>FallbackErrorHandler.setAppender(Appender primary)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The appender to which this error handler is attached.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>FallbackErrorHandler.setBackupAppender(Appender backup)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the backup appender.</td>
</tr>
</tbody>
</table>

Uses of Appender in org.apache.log4j.xml

Methods in org.apache.log4j.xml that return Appender

<table>
<thead>
<tr>
<th>protected Appender</th>
<th>DOMConfigurator.findAppenderByReference(Element appendRef)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Used internally to parse appenders by IDREF element.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected Appender</th>
<th>DOMConfigurator.findAppenderByName(Document doc, String appendName)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Used internally to parse appenders by IDREF name.</td>
</tr>
</tbody>
</table>
**Appender**

Used internally to parse an appender element.

Methods in `org.apache.log4j.xml` with parameters of type `Appender`

<table>
<thead>
<tr>
<th>Method Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DOMConfigurator.parseErrorHandler(Element element, Appender appender)</code></td>
<td>Used internally to parse an <code>ErrorHandler</code> element.</td>
</tr>
<tr>
<td><code>DOMConfigurator.parseFilters(Element element, Appender appender)</code></td>
<td>Used internally to parse a filter element.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.AppenderSkeleton

<table>
<thead>
<tr>
<th>Packages that use AppenderSkeleton</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
</tr>
<tr>
<td>org.apache.log4j.jdbc</td>
</tr>
<tr>
<td>org.apache.log4j.lf5</td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
</tr>
<tr>
<td>org.apache.log4j.nt</td>
</tr>
<tr>
<td>org.apache.log4j.performance</td>
</tr>
</tbody>
</table>

Uses of AppenderSkeleton in org.apache.log4j

<table>
<thead>
<tr>
<th>Subclasses of AppenderSkeleton in org.apache.log4j</th>
</tr>
</thead>
<tbody>
<tr>
<td>class AsyncAppender</td>
</tr>
<tr>
<td>The AsyncAppender lets users log events asynchronously.</td>
</tr>
<tr>
<td>class ConsoleAppender</td>
</tr>
<tr>
<td>ConsoleAppender appends log events to System.out or System.err using a layout specified by the user.</td>
</tr>
<tr>
<td>class DailyRollingFileAppender</td>
</tr>
<tr>
<td>DailyRollingFileAppender extends FileAppender so that the underlying file is rolled over at a user chosen frequency.</td>
</tr>
<tr>
<td>class FileAppender</td>
</tr>
<tr>
<td>FileAppender appends log events to a file.</td>
</tr>
</tbody>
</table>
### Uses of AppenderSkeleton in org.apache.log4j.jdbc

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RollingFileAppender</td>
<td>RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.</td>
</tr>
<tr>
<td>WriterAppender</td>
<td>WriterAppender appends log events to a Writer or an OutputStream depending on the user's choice.</td>
</tr>
</tbody>
</table>

### Subclasses of AppenderSkeleton in org.apache.log4j.jdbc

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBCAppender</td>
<td>WARNING: This version of JDBCAppender is very likely to be completely replaced in the future.</td>
</tr>
</tbody>
</table>

### Uses of AppenderSkeleton in org.apache.log4j.lf5

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF5Appender</td>
<td>LF5Appender logs events to a swing based logging console.</td>
</tr>
</tbody>
</table>

### Subclasses of AppenderSkeleton in org.apache.log4j.lf5

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBCAppender</td>
<td>WARNING: This version of JDBCAppender is very likely to be completely replaced in the future.</td>
</tr>
</tbody>
</table>

### Uses of AppenderSkeleton in org.apache.log4j.net

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF5Appender</td>
<td>LF5Appender logs events to a swing based logging console.</td>
</tr>
</tbody>
</table>

### Subclasses of AppenderSkeleton in org.apache.log4j.net

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
</table>
**class JMSAppender**
A simple appender that publishes events to a JMS Topic.

**class SMTPAppender**
Send an e-mail when a specific logging event occurs, typically on errors or fatal errors.

**class SocketAppender**
Sends `<LoggingEvent>` objects to a remote log server, usually a `<SocketNode>`.

**class SocketHubAppender**
Sends `<LoggingEvent>` objects to a set of remote log servers, usually a `<SocketNodes>`.

**class SyslogAppender**
Use SyslogAppender to send log messages to a remote syslog daemon.

**class TelnetAppender**
The TelnetAppender is a log4j appender that specializes in writing to a read-only socket.

---

**Uses of `<AppenderSkeleton>` in `<org.apache.log4j.nt>`**

---

**Subclasses of `<AppenderSkeleton>` in `<org.apache.log4j.nt>`**

<table>
<thead>
<tr>
<th>class</th>
<th><code>&lt;NTEventLogAppender&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Append to the NT event log system.</td>
</tr>
</tbody>
</table>

---

**Uses of `<AppenderSkeleton>` in `<org.apache.log4j.performance>`**

---

**Subclasses of `<AppenderSkeleton>` in `<org.apache.log4j.performance>`**
NullAppender

A bogus appender which calls the format method of its layout object but does not write the result anywhere.

Uses of AppenderSkeleton in org.apache.log4j.varia

ExternallyRolledFileAppender

This appender listens on a socket on the port specified by the Port property for a "RollOver" message.

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.Category

<table>
<thead>
<tr>
<th>Packages that use <strong>Category</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
</tr>
<tr>
<td>org.apache.log4j.jmx</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
</tr>
</tbody>
</table>

Uses of **Category** in **org.apache.log4j**

<table>
<thead>
<tr>
<th>Subclasses of <strong>Category</strong> in <strong>org.apache.log4j</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>class <strong>Logger</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>This is the central class in the log4j package.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields in <strong>org.apache.log4j</strong> declared as <strong>Category</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>protected <strong>Category</strong> parent</td>
</tr>
<tr>
<td>The parent of this category.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in <strong>org.apache.log4j</strong> that return <strong>Category</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>Category</strong> getInstance(String name)</td>
</tr>
<tr>
<td><strong>Deprecated. Make sure to use Logger.getLogger(String) instead.</strong></td>
</tr>
<tr>
<td>static <strong>Category</strong> getInstance(Class clazz)</td>
</tr>
<tr>
<td><strong>Deprecated. Please make sure to use Logger.getLogger(Class) instead.</strong></td>
</tr>
<tr>
<td><strong>Category</strong> getParent()</td>
</tr>
</tbody>
</table>
Returns the parent of this category.

<table>
<thead>
<tr>
<th>static Category.getRoot()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deprecated.</strong> Please use Logger.getRootLogger() instead.</td>
</tr>
</tbody>
</table>

Methods in `org.apache.log4j` with parameters of type `Category`

- `void Hierarchy.emitNoAppenderWarning(Category cat)`
- `void Hierarchy.fireAddAppenderEvent(Category logger, Appender appender)`

**Uses of `Category` in `org.apache.log4j.jmx`**

Methods in `org.apache.log4j.jmx` with parameters of type `Category`

- `void HierarchyDynamicMBean.addAppenderEvent(Category logger, Appender appender)`
- `void HierarchyDynamicMBean.removeAppenderEvent(Category cat, Appender appender)`

**Uses of `Category` in `org.apache.log4j.spi`**

Subclasses of `Category` in `org.apache.log4j.spi`

<table>
<thead>
<tr>
<th>class</th>
<th>RootCategory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deprecated.</strong> Replaced by <code>RootLogger</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
<th>RootLogger</th>
</tr>
</thead>
<tbody>
<tr>
<td>RootLogger sits at the top of the logger hierarchy.</td>
<td></td>
</tr>
</tbody>
</table>
### Methods in `org.apache.log4j.spi` with parameters of type `Category`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void LoggerRepository.emitNoAppenderWarning(Category cat)</code></td>
<td></td>
</tr>
<tr>
<td><code>void LoggerRepository.fireAddAppenderEvent(Category logger, Appender appender)</code></td>
<td></td>
</tr>
<tr>
<td><code>void HierarchyEventListener.addAppenderEvent(Category cat, Appender appender)</code></td>
<td></td>
</tr>
<tr>
<td><code>void HierarchyEventListener.removeAppenderEvent(Category cat, Appender appender)</code></td>
<td></td>
</tr>
</tbody>
</table>

### Constructors in `org.apache.log4j.spi` with parameters of type `Category`

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LoggingEvent(String fqcnOfCategoryClass, Category logger, Priority level, Object message, Throwable throwable)</code></td>
<td>Instantiate a LoggingEvent from the supplied parameters.</td>
</tr>
<tr>
<td><code>LoggingEvent(String fqcnOfCategoryClass, Category logger, long timeStamp, Priority level, Object message, Throwable throwable)</code></td>
<td>Instantiate a LoggingEvent from the supplied parameters.</td>
</tr>
</tbody>
</table>
# Uses of Class

---

**org.apache.log4j.FileAppender**

## Packages that use **FileAppender**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.apache.log4j</strong></td>
<td>The main log4j package.</td>
</tr>
<tr>
<td><strong>org.apache.log4j.varia</strong></td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
</tbody>
</table>

---

## Uses of **FileAppender** in **org.apache.log4j**

## Subclasses of **FileAppender** in **org.apache.log4j**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DailyRollingFileAppender</strong></td>
<td>DailyRollingFileAppender extends <strong>FileAppender</strong> so that the underlying file is rolled over at a user chosen frequency.</td>
</tr>
<tr>
<td><strong>RollingFileAppender</strong></td>
<td>RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.</td>
</tr>
</tbody>
</table>

## Uses of **FileAppender** in **org.apache.log4j.varia**

## Subclasses of **FileAppender** in **org.apache.log4j.varia**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ExternallyRolledFileAppender</strong></td>
<td>This appender listens on a socket on the port specified by the <strong>Port</strong> property for a &quot;RollOver&quot; message.</td>
</tr>
</tbody>
</table>
Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.Layout

Packages that use Layout

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.jmx</td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td>org.apache.log4j.nt</td>
<td>Package for NT event logging.</td>
</tr>
<tr>
<td>org.apache.log4j.performance</td>
<td>Package to measure the performance of the different log4j components.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

Uses of Layout in org.apache.log4j

Subclasses of Layout in org.apache.log4j

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLLayout</td>
<td>This layout outputs events in a HTML table.</td>
</tr>
<tr>
<td>PatternLayout</td>
<td>A flexible layout configurable with pattern string.</td>
</tr>
<tr>
<td>SimpleLayout</td>
<td>SimpleLayout consists of the level of the log statement, followed by &quot; - &quot; and then the log message itself.</td>
</tr>
<tr>
<td>TTCCLayout</td>
<td>TTCC layout format consists of time, thread, category and nested diagnostic context information, hence the name.</td>
</tr>
</tbody>
</table>
Fields in `org.apache.log4j` declared as `Layout`:

| protected Layout AppenderSkeleton.layout | The layout variable does not need to be set if the appender implementation has its own layout. |

Methods in `org.apache.log4j` that return `Layout`:

| Layout AppenderSkeleton.getLayout() | Returns the layout of this appender. |
| Layout Appender.getLayout() | Returns this appenders layout. |

Methods in `org.apache.log4j` with parameters of type `Layout`:

| void AppenderSkeleton.setLayout(Layout layout) | Set the layout for this appender. |
| void Appender.setLayout(Layout layout) | Set the `Layout` for this appender. |

Constructors in `org.apache.log4j` with parameters of type `Layout`:

| WriterAppender(Layout layout, OutputStream os) | Instantiate a WriterAppender and set the output destination to a new `OutputStreamWriter` initialized with `os` as its `OutputStream`. |
| WriterAppender(Layout layout, Writer writer) | Instantiate a WriterAppender and set the output destination to `writer`. |
| FileAppender(Layout layout, String filename, boolean append, boolean bufferedIO, int bufferSize) | Instantiate a `FileAppender` and open the file designated by `filename`. |
| FileAppender(Layout layout, String filename, boolean append) | Instantiate a `FileAppender` and open the file designated by `filename`. |
| FileAppender(Layout layout, String filename) | |
Instantiate a FileAppender and open the file designated by filename.

**ConsoleAppender**(Layout layout)

Creates a configured appender.

**ConsoleAppender**(Layout layout, String target)

Creates a configured appender.

**DailyRollingFileAppender**(Layout layout, String filename, String datePattern)

Instantiate a DailyRollingFileAppender and open the file designated by filename.

**RollingFileAppender**(Layout layout, String filename, boolean append)

Instantiate a RollingFileAppender and open the file designated by filename.

**RollingFileAppender**(Layout layout, String filename)

Instantiate a FileAppender and open the file designated by filename.

---

**Uses of Layout in org.apache.log4j.helpers**

**Subclasses of Layout in org.apache.log4j.helpers**

<table>
<thead>
<tr>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DateLayout</strong></td>
<td>This abstract layout takes care of all the date related options and formatting work.</td>
</tr>
</tbody>
</table>

**Uses of Layout in org.apache.log4j.jmx**

**Constructors in org.apache.log4j.jmx with parameters of type Layout**

**LayoutDynamicMBean**(Layout layout)
Uses of **Layout** in **org.apache.log4j.net**

<table>
<thead>
<tr>
<th>Constructors in <strong>org.apache.log4j.net</strong> with parameters of type <strong>Layout</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SyslogAppender</strong>(Layout layout, int syslogFacility)</td>
</tr>
<tr>
<td><strong>SyslogAppender</strong>(Layout layout, String syslogHost, int syslogFacility)</td>
</tr>
</tbody>
</table>

Uses of **Layout** in **org.apache.log4j.nt**

<table>
<thead>
<tr>
<th>Constructors in <strong>org.apache.log4j.nt</strong> with parameters of type <strong>Layout</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NTEventLogAppender</strong>(Layout layout)</td>
</tr>
<tr>
<td><strong>NTEventLogAppender</strong>(String source, Layout layout)</td>
</tr>
<tr>
<td><strong>NTEventLogAppender</strong>(String server, String source, Layout layout)</td>
</tr>
</tbody>
</table>

Uses of **Layout** in **org.apache.log4j.performance**

<table>
<thead>
<tr>
<th>Constructors in <strong>org.apache.log4j.performance</strong> with parameters of type <strong>Layout</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NullAppender</strong>(Layout layout)</td>
</tr>
</tbody>
</table>
Uses of **Layout** in **org.apache.log4j.xml**

### Subclasses of **Layout** in **org.apache.log4j.xml**

<table>
<thead>
<tr>
<th>class</th>
<th>XMLLayout</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The output of the XMLLayout consists of a series of log4j:event elements as defined in the log4j.dtd.</td>
</tr>
</tbody>
</table>

### Methods in **org.apache.log4j.xml** that return **Layout**

<table>
<thead>
<tr>
<th>protected Layout</th>
<th>DOMConfigurator.parseLayout(Element layout_element)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Used internally to parse a layout element.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.Level

Packages that use Level

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
</tbody>
</table>

Uses of Level in org.apache.log4j

Fields in org.apache.log4j declared as Level

<table>
<thead>
<tr>
<th>level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category.</td>
<td>The assigned level of this category.</td>
</tr>
<tr>
<td>OFF</td>
<td>The OFF has the highest possible rank and is intended to turn off logging.</td>
</tr>
<tr>
<td>FATAL</td>
<td>The FATAL level designates very severe error events that will presumably lead the application to abort.</td>
</tr>
<tr>
<td>ERROR</td>
<td>The ERROR level designates error events that might still allow the application to continue running.</td>
</tr>
<tr>
<td>WARN</td>
<td>The WARN level designates potentially harmful situations.</td>
</tr>
<tr>
<td>INFO</td>
<td>The INFO level designates informational messages that highlight the progress of the application at coarse-grained</td>
</tr>
</tbody>
</table>
The DEBUG Level designates fine-grained informational events that are most useful to debug an application.

The TRACE Level designates finer-grained informational events than the DEBUG Level.

The ALL has the lowest possible rank and is intended to turn on all logging.

Methods in org.apache.log4j that return Level

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category.getEffectiveLevel()</td>
<td>Starting from this category, search the category hierarchy for a non-null level and return it.</td>
</tr>
<tr>
<td>Category.getLevel()</td>
<td>Returns the assigned Level, if any, for this Category.</td>
</tr>
<tr>
<td>Category.getPriority()</td>
<td>Deprecated. Please use Category.getLevel() instead.</td>
</tr>
<tr>
<td>Level.toLevel(String sArg)</td>
<td>Convert the string passed as argument to a level.</td>
</tr>
<tr>
<td>Level.toLevel(int val)</td>
<td>Convert an integer passed as argument to a level.</td>
</tr>
<tr>
<td>Level.toLevel(int val, Level defaultLevel)</td>
<td>Convert an integer passed as argument to a level.</td>
</tr>
<tr>
<td>Level.toLevel(String sArg, Level defaultLevel)</td>
<td>Convert the string passed as argument to a level.</td>
</tr>
<tr>
<td>Hierarchy.getThreshold()</td>
<td>Returns a Level representation of the enable state.</td>
</tr>
</tbody>
</table>

Methods in org.apache.log4j with parameters of type Level

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Category.setLevel(Level level)</td>
<td>Set the level of this Category.</td>
</tr>
</tbody>
</table>
static Level toLevel(int val, Level defaultLevel)
    Convert an integer passed as argument to a level.

static Level toLevel(String sArg, Level defaultLevel)
    Convert the string passed as argument to a level.

Hierarchy.setThreshold(Level l)
    Enable logging for logging requests with level 1 or higher.

Uses of Level in org.apache.log4j.helpers

Methods in org.apache.log4j.helpers that return Level

static Level OptionConverter.toLevel(String value, Level defaultValue)
    Converts a standard or custom priority level to a Level object.

Methods in org.apache.log4j.helpers with parameters of type Level

static Level OptionConverter.toLevel(String value, Level defaultValue)
    Converts a standard or custom priority level to a Level object.

Uses of Level in org.apache.log4j.spi

Methods in org.apache.log4j.spi that return Level

Level RootLogger.getChainedLevel()
    Return the assigned level value without walking the logger hierarchy.

Level LoggerRepository.getThreshold()
    Get the repository-wide threshold.

RootCategory.getChainedLevel()
Level

**Deprecated.** Return the assigned level value without walking the category hierarchy.

level

```
LoggingEvent.Level.getLevel()
```

Return the level of this event.

---

**Methods in** [org.apache.log4j.spi](https://logging.apache.org/log4j/2.x/apidocs/index.html) **with parameters of type** Level

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void RootLogger.setLevel(Level level)</td>
<td>Setting a null value to the level of the root logger may have catastrophic results.</td>
</tr>
<tr>
<td>void LoggerRepository.setThreshold(Level level)</td>
<td>Set the repository-wide threshold.</td>
</tr>
<tr>
<td>void RootCategory.setLevel(Level level)</td>
<td><strong>Deprecated.</strong> Setting a null value to the level of the root category may have catastrophic results.</td>
</tr>
<tr>
<td>void RootCategory.setPriority(Level level)</td>
<td><strong>Deprecated.</strong></td>
</tr>
</tbody>
</table>

---

**Constructors in** [org.apache.log4j.spi](https://logging.apache.org/log4j/2.x/apidocs/index.html) **with parameters of type** Level

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RootLogger(Level level)</td>
<td>The root logger names itself as &quot;root&quot;.</td>
</tr>
<tr>
<td>RootCategory(Level level)</td>
<td><strong>Deprecated.</strong> The root category names itself as &quot;root&quot;.</td>
</tr>
</tbody>
</table>

---

**Uses of** Level in [org.apache.log4j.varia](https://logging.apache.org/log4j/2.x/apidocs/index.html)

---

**Methods in** [org.apache.log4j.varia](https://logging.apache.org/log4j/2.x/apidocs/index.html) **that return** Level

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level LevelRangeFilter.getLevelMax()</td>
<td>Get the value of the LevelMax option.</td>
</tr>
<tr>
<td>Level LevelRangeFilter.getLevelMin()</td>
<td>Get the value of the LevelMin option.</td>
</tr>
</tbody>
</table>
## Methods in `org.apache.log4j.varia` with parameters of type `Level`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void LevelRangeFilter.setLevelMax(Level levelMax)</code></td>
<td>Set the LevelMax option.</td>
</tr>
<tr>
<td><code>void LevelRangeFilter.setLevelMin(Level levelMin)</code></td>
<td>Set the LevelMin option.</td>
</tr>
</tbody>
</table>
# Uses of Class

## org.apache.log4j.Logger

<table>
<thead>
<tr>
<th>Packages that use Logger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.config</td>
<td>Package used in getting/setting component properties.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.jmx</td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

## Uses of Logger in org.apache.log4j

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j that return Logger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Logger Category.exists(String name)</td>
<td><strong>Deprecated. Please use LogManager.exists(java.lang.String) instead.</strong></td>
</tr>
<tr>
<td>static Logger LogManager.getRootLogger()</td>
<td>Retrieve the appropriate root logger.</td>
</tr>
<tr>
<td>static Logger LogManager.getLogger(String name)</td>
<td>Retrieve the appropriate Logger instance.</td>
</tr>
<tr>
<td>static Logger LogManager.getLogger(Class clazz)</td>
<td>Retrieve the appropriate Logger instance.</td>
</tr>
<tr>
<td>static Logger LogManager.getLogger(String name, LoggerFactory factory)</td>
<td>Retrieve the appropriate Logger instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Signature</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>static Logger</td>
<td>exists(String name)</td>
</tr>
<tr>
<td>static Logger</td>
<td>getLogger(String name)</td>
</tr>
<tr>
<td>static Logger</td>
<td>getLogger(Class clazz)</td>
</tr>
<tr>
<td>static Logger</td>
<td>getLogger(String name, LoggerFactory factory)</td>
</tr>
<tr>
<td>static Logger</td>
<td>getLogger(String name, LoggerFactory factory)</td>
</tr>
<tr>
<td>static Logger</td>
<td>getRootLogger()</td>
</tr>
<tr>
<td>Logger</td>
<td>exists(String name)</td>
</tr>
<tr>
<td>Logger</td>
<td>getLogger(String name)</td>
</tr>
<tr>
<td>Logger</td>
<td>getLogger(String name, LoggerFactory factory)</td>
</tr>
<tr>
<td>Logger</td>
<td>getRootLogger()</td>
</tr>
</tbody>
</table>

Constructors in org.apache.log4j with parameters of type Logger

Hierarchy(Logger root)

Create a new logger hierarchy.

**Uses of Logger in org.apache.log4j.config**

Methods in org.apache.log4j.config with parameters of type Logger
protected void PropertyPrinter.\(\text{printOptions}\) (PrintWriter out, Logger cat)

## Uses of **Logger** in **org.apache.log4j.helpers**

### Methods in **org.apache.log4j.helpers** with parameters of type **Logger**

<table>
<thead>
<tr>
<th>void</th>
<th>OnlyOnceErrorHandler.(\text{setLogger}) (Logger logger)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Does not do anything.</td>
</tr>
</tbody>
</table>

## Uses of **Logger** in **org.apache.log4j.jmx**

### Methods in **org.apache.log4j.jmx** that return **Logger**

<table>
<thead>
<tr>
<th>protected abstract Logger</th>
<th>AbstractDynamicMBean.(\text{getLogger}) ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Logger</td>
<td>LoggerDynamicMBean.(\text{getLogger}) ()</td>
</tr>
<tr>
<td>protected Logger</td>
<td>LayoutDynamicMBean.(\text{getLogger}) ()</td>
</tr>
<tr>
<td>protected Logger</td>
<td>AppenderDynamicMBean.(\text{getLogger}) ()</td>
</tr>
<tr>
<td>protected Logger</td>
<td>HierarchyDynamicMBean.(\text{getLogger}) ()</td>
</tr>
</tbody>
</table>

### Constructors in **org.apache.log4j.jmx** with parameters of type **Logger**

| LoggerDynamicMBean (Logger logger) |
# Uses of Logger in org.apache.log4j.spi

## Subclasses of Logger in org.apache.log4j.spi

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RootCategory</td>
<td>Deprecated. Replaced by RootLogger.</td>
</tr>
<tr>
<td>RootLogger</td>
<td>RootLogger sits at the top of the logger hierarchy.</td>
</tr>
</tbody>
</table>

## Methods in org.apache.log4j.spi that return Logger

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoggerRepository.getLogger(String name)</td>
<td></td>
</tr>
<tr>
<td>LoggerRepository.getLogger(String name, LoggerFactory factory)</td>
<td></td>
</tr>
<tr>
<td>LoggerRepository.getRootLogger()</td>
<td></td>
</tr>
<tr>
<td>LoggerRepository.exists(String name)</td>
<td></td>
</tr>
<tr>
<td>LoggerFactory.makeNewLoggerInstance(String name)</td>
<td></td>
</tr>
</tbody>
</table>

## Methods in org.apache.log4j.spi with parameters of type Logger

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorHandler.setLogger(Logger logger)</td>
<td>Add a reference to a logger to which the failing appender might be attached to.</td>
</tr>
</tbody>
</table>

---

# Uses of Logger in org.apache.log4j.varia

## Methods in org.apache.log4j.varia with parameters of type Logger

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
```java
void FallbackErrorHandler.setLogger(Logger logger)

  Adds the logger passed as parameter to the list of loggers that we need to search for in case of appender failure.
```

### Uses of `Logger` in `org.apache.log4j.xml`

<table>
<thead>
<tr>
<th>Methods in <code>org.apache.log4j.xml</code> with parameters of type <code>Logger</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protected void DOMConfigurator.parseChildrenOfLoggerElement</strong> <em>(Element catElem, Logger cat, boolean isRoot)</em></td>
</tr>
<tr>
<td><strong>protected void DOMConfigurator.parseLevel</strong> <em>(Element element, Logger logger, boolean isRoot)</em></td>
</tr>
</tbody>
</table>
# Uses of Class
## org.apache.log4j.Priority

<table>
<thead>
<tr>
<th>Packages that use <strong>Priority</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.apache.log4j</strong></td>
</tr>
<tr>
<td><strong>org.apache.log4j.spi</strong></td>
</tr>
</tbody>
</table>

## Uses of **Priority** in **org.apache.log4j**

## Subclasses of **Priority** in **org.apache.log4j**

| class | **Level** | Defines the minimum set of levels recognized by the system, that is OFF, FATAL, ERROR, WARN, INFO, DEBUG and ALL. |

## Fields in **org.apache.log4j** declared as **Priority**

<table>
<thead>
<tr>
<th>protected <strong>Priority</strong></th>
<th>AppenderSkeleton. <strong>threshold</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>There is no level threshold filtering by default.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <strong>Priority</strong></th>
<th><strong>FATAL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Use <strong>Level.FATAL</strong> instead.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <strong>Priority</strong></th>
<th><strong>ERROR</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Use <strong>Level.ERROR</strong> instead.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <strong>Priority</strong></th>
<th><strong>WARN</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Use <strong>Level.WARN</strong> instead.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <strong>Priority</strong></th>
<th><strong>INFO</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Use <strong>Level.INFO</strong> instead.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static <strong>Priority</strong></th>
<th><strong>DEBUG</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Use <strong>Level.DEBUG</strong> instead.</td>
</tr>
</tbody>
</table>
Methods in `org.apache.log4j` that return `Priority`

| Priority          | Category.getMethodName()                                      | *Deprecated.* Please use the the Category.getMethodName() method instead.
|-------------------|----------------------------------------------------------------|--------------------------------------------------------------------------
|                   | `getChainedPriority()`                                        |                                                                          |
|                   | `getThreshold()`                                               | Returns this appenders threshold level.                                   |
| static `Priority[]` | `getAllPossiblePriorities()`                                   | *Deprecated.* This method will be removed with no replacement.            |
| static `Priority`  | `toPriority(String sArg)`                                     | *Deprecated.* Please use the Level.toLevel(String) method instead.        |
| static `Priority`  | `toPriority(int val)`                                         | *Deprecated.* Please use the Level.toLevel(int) method instead.           |
| static `Priority`  | `toPriority(int val, Priority defaultPriority)`                | *Deprecated.* Please use the Level.toLevel(int, Level) method instead.    |
| static `Priority`  | `toPriority(String sArg, Priority defaultPriority)`            | *Deprecated.* Please use the Level.toLevel(String, Level) method instead. |

Methods in `org.apache.log4j` with parameters of type `Priority`

| protected void    | Category.getMethodName(String fqcn, Priority level, Object message, Throwable t) | This method creates a new logging event and logs the event without further checks. |
| boolean           | Category.getMethodName(Priority level)                                       | Check whether this category is enabled for a given Level passed as parameter. |
| void              | Category.getMethodName(Priority priority, String key, Throwable t)           | Log a localized message.                                                   |
### Uses of `Priority` in `org.apache.log4j.spi`

<table>
<thead>
<tr>
<th>Fields in <code>org.apache.log4j.spi</code> declared as <code>Priority</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LoggingEvent.level</code></td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void Category.**log**(Priority priority, Object message)</code></td>
<td>This generic form is intended to be used by wrappers.</td>
</tr>
<tr>
<td><code>void Category.**log**(Priority priority, Object message, Throwable t)</code></td>
<td>Log a localized and parameterized message.</td>
</tr>
<tr>
<td><code>void Category.**log**(String callerFQCN, Priority level, Object message, Throwable t)</code></td>
<td>This is the most generic printing method.</td>
</tr>
<tr>
<td><code>boolean AppenderSkeleton.**isAsSevereAsThreshold**(Priority priority)</code></td>
<td>Check whether the message level is below the appender's threshold.</td>
</tr>
<tr>
<td><code>void AppenderSkeleton.**setThreshold**(Priority threshold)</code></td>
<td>Set the threshold level.</td>
</tr>
<tr>
<td><code>boolean Priority.**isGreaterOrEqual**(Priority r)</code></td>
<td>Returns true if this level has a higher or equal level than the level passed as argument, false otherwise.</td>
</tr>
<tr>
<td><code>static Priority Priority.**toPriority**(int val, Priority defaultPriority)</code></td>
<td>Deprecated. Please use the <code>Level.toLevel(int, Level)</code> method instead.</td>
</tr>
<tr>
<td><code>static Priority Priority.**toPriority**(String sArg, Priority defaultPriority)</code></td>
<td>Deprecated. Please use the <code>Level.toLevel(String, Level)</code> method instead.</td>
</tr>
</tbody>
</table>
Priority

**Deprecated.** This field will be marked as private in future releases. Please do not access it directly. Use the `LoggingEvent.getLevel()` method instead.

Constructors in `org.apache.log4j.spi` with parameters of type `Priority`:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>LoggingEvent(String fqncOfCategoryClass, Category logger, Priority level, Object message, Throwable throwable)</code></td>
<td>Instantiate a LoggingEvent from the supplied parameters.</td>
</tr>
<tr>
<td><code>LoggingEvent(String fqncOfCategoryClass, Category logger, long timeStamp, Priority level, Object message, Throwable throwable)</code></td>
<td>Instantiate a LoggingEvent from the supplied parameters.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.WriterAppender

Packages that use **WriterAppender**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
</tbody>
</table>

Uses of **WriterAppender** in **org.apache.log4j**

Subclasses of **WriterAppender** in **org.apache.log4j**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConsoleAppender</strong></td>
<td>ConsoleAppender appends log events to System.out or System.err using a layout specified by the user.</td>
</tr>
<tr>
<td><strong>DailyRollingFileAppender</strong></td>
<td>DailyRollingFileAppender extends <strong>FileAppender</strong> so that the underlying file is rolled over at a user chosen frequency.</td>
</tr>
<tr>
<td><strong>FileAppender</strong></td>
<td>FileAppender appends log events to a file.</td>
</tr>
<tr>
<td><strong>RollingFileAppender</strong></td>
<td>RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.</td>
</tr>
</tbody>
</table>

Uses of **WriterAppender** in **org.apache.log4j.varia**

Subclasses of **WriterAppender** in **org.apache.log4j.varia**
ExternallyRolledFileAppender

This appender listens on a socket on the port specified by the Port property for a "RollOver" message.
Uses of Class
org.apache.log4j.RollingFileAppender

Packages that use RollingFileAppender

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
</tbody>
</table>

Uses of RollingFileAppender in org.apache.log4j.varia

Subclasses of RollingFileAppender in org.apache.log4j.varia

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternallyRolledFileAppender</td>
<td>This appender listens on a socket on the port specified by the Port property for a &quot;RollOver&quot; message.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.AsyncAppender

No usage of org.apache.log4j.AsyncAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.BasicConfigurator

No usage of org.apache.log4j.BasicConfigurator

Copyright 2000-2005 Apache Software Foundation.
# Uses of Class

**org.apache.log4j.ConsoleAppender**

No usage of org.apache.log4j.ConsoleAppender

---

## Copyright

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.DailyRollingFileAppender

No usage of org.apache.log4j.DailyRollingFileAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.Hierarchy

No usage of org.apache.log4j.Hierarchy

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.HTMLLayout

No usage of org.apache.log4j.HTMLLayout

Copyright 2000-2005 Apache Software Foundation.
# Serialized Form

<table>
<thead>
<tr>
<th>Package org.apache.log4j</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class org.apache.log4j.Level implements Serializable</td>
</tr>
</tbody>
</table>

## Serialization Methods

### readObject

```java
class org.apache.log4j.Level {
    private void readObject(ObjectInputStream s) throws IOException, ClassNotFoundException {
        // Custom deserialization of Level.
    }
}
```

### writeObject

```java
class org.apache.log4j.Level {
    private void writeObject(ObjectOutputStream s) throws IOException {
        // Serialize level.
    }
}
```

<table>
<thead>
<tr>
<th>Package org.apache.log4j.chainsaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class org.apache.log4j.chainsaw.Main implements Serializable</td>
</tr>
</tbody>
</table>
Package org.apache.log4j.config

Class
org.apache.log4j.config.PropertySetterException
implements Serializable

Serialized Fields

rootCause

Throwable rootCause

Package org.apache.log4j.helpers

Class
org.apache.log4j.helpers.AbsoluteTimeDateFormat
implements Serializable

Class
org.apache.log4j.helpers.DateTimeDateFormat
implements Serializable

Serialized Fields

shortMonths

String[] shortMonths
Class org.apache.log4j.helpers.ISO8601DateFormat
implements Serializable

Class org.apache.log4j.helpers.RelativeTimeDateFormat
implements Serializable

Serialized Fields

startTime
long startTime

Package org.apache.log4j.lf5

Class org.apache.log4j.lf5.Log4JLogRecord
implements Serializable

Class org.apache.log4j.lf5.LogLevel
implements Serializable

Serialized Fields

_label
String _label

_int _precedence

Class org.apache.log4j.lf5.LogLevelFormatException
implements Serializable

Class org.apache.log4j.lf5.LogRecord
implements Serializable

Serialized Fields

(LogLevel _level

String _message

long _sequenceNumber

long _millis
_category
String _category

_thread
String _thread

ThrownStackTrace
String _thrownStackTrace

Thrown
Throwable _thrown

ndc
String _ndc

_location
String _location

Package org.apache.log4j.spi

Class org.apache.log4j.spi.LocationInfo
implements Serializable
**Serialized Fields**

`fullInfo`

```java
String fullInfo
```

All available caller information, in the format `fully.qualified.classname.of.caller.methodName(Filename.java:line)`

---

**Class**

`org.apache.log4j.spi.LoggingEvent`

implements `Serializable`

---

**Serialization Methods**

**readObject**

```java
private void readObject(ObjectInputStream ois) throws IOException, ClassNotFoundException
```

**writeObject**

```java
private void writeObject(ObjectOutputStream oos) throws IOException
```

---

**Serialized Fields**

`categoryName`

```java
String categoryName
```

**Deprecated. This field will be marked as private in future releases. Please do not access it directly. Use the**
The category (logger) name.

ndc

String ndc

The nested diagnostic context (NDC) of logging event.

mdcCopy

Hashtable mdcCopy

The mapped diagnostic context (MDC) of logging event.

ndcLookupRequired

boolean ndcLookupRequired

Have we tried to do an NDC lookup? If we did, there is no need to do it again. Note that its value is always false when serialized. Thus, a receiving SocketNode will never use it's own (incorrect) NDC. See also writeObject method.

mdcCopyLookupRequired

boolean mdcCopyLookupRequired

Have we tried to do an MDC lookup? If we did, there is no need to do it again. Note that its value is always false when serialized. See also the getMDC and getMDCCopy methods.

renderedMessage
String renderedMessage

The application supplied message rendered through the log4j objet rendering mechanism.

threadName

String threadName

The name of thread in which this logging event was generated.

ThrowableInfo

ThrowableInformation throwableInfo

This variable contains information about this event's throwable

timeStamp

long timeStamp

The number of milliseconds elapsed from 1/1/1970 until logging event was created.

locationInfo

LocationInfo locationInfo

Location information for the caller.

Class

org.apache.log4j.spi.ThrowExceptionInformation

implements Serializable
Serialized Fields

rep

`String[] rep`
No usage of org.apache.log4j.LogManager
Uses of Class
org.apache.log4j.MDC

No usage of org.apache.log4j.MDC

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.NDC

No usage of org.apache.log4j.NDC

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.PatternLayout
Uses of Class org.apache.log4j.PropertyConfigurator

No usage of org.apache.log4j.PropertyConfigurator

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.SimpleLayout

No usage of org.apache.log4j.SimpleLayout

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.TTCCLayout

No usage of org.apache.log4j.TTCCLayout

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.chainsaw.Main
Uses of Interface
org.apache.log4j.config.PropertyGetter.PropertyCallback

Packages that use
PropertyGetter.PropertyCallback

org.apache.log4j.config
Package used in getting/setting component properties.

Uses of PropertyGetter.PropertyCallback in
org.apache.log4j.config

Classes in org.apache.log4j.config that implement
PropertyGetter.PropertyCallback

<table>
<thead>
<tr>
<th>class</th>
<th>PropertyPrinter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prints the configuration of the log4j default hierarchy (which needs to be auto-initialized) as a proprieties file on a PrintWriter.</td>
</tr>
</tbody>
</table>

Methods in org.apache.log4j.config with parameters of type
PropertyGetter.PropertyCallback

| static void | PropertyGetter.getProperties(Object obj, PropertyGetter.PropertyCallback callback, String prefix) |
| void        | PropertyGetter.getProperties(PropertyGetter.PropertyCallback callback, String prefix) |
Copyright 2000-2005 Apache Software Foundation.
Packages that use **PropertySetterException**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.config</td>
<td>Package used in getting/setting component properties.</td>
</tr>
</tbody>
</table>

Uses of **PropertySetterException** in

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.config</td>
<td></td>
</tr>
</tbody>
</table>

Methods in **org.apache.log4j.config** that throw **PropertySetterException**

```java
void PropertySetter.setProperty(PropertyDescriptor prop, String name, String value)
```

Set the named property given a **PropertyDescriptor**.

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.config.PropertySetter

Packages that use PropertySetter

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

Uses of PropertySetter in org.apache.log4j.xml

Methods in org.apache.log4j.xml with parameters of type PropertySetter

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void</td>
<td>DOMConfigurator.setParameter(Element elem, PropertySetter propSetter)</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.config.PropertyGetter

No usage of org.apache.log4j.config.PropertyGetter

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.config.PropertyPrinter

No usage of org.apache.log4j.config.PropertyPrinter

Copyright 2000-2005 Apache Software Foundation.
Packages that use **DateLayout**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
</tbody>
</table>

Uses of **DateLayout** in **org.apache.log4j**

Subclasses of **DateLayout** in **org.apache.log4j**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTCCLayout</td>
<td>TTCC layout format consists of time, thread, category and nested diagnostic context information, hence the name.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.PatternParser

Packages that use PatternParser

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
</tbody>
</table>

Uses of PatternParser in org.apache.log4j

Methods in org.apache.log4j that return PatternParser

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatternLayout.createPatternParser(String pattern)</td>
<td>Returns PatternParser used to parse the conversion string.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.QuietWriter

Packages that use QuietWriter

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
</tbody>
</table>

Uses of QuietWriter in org.apache.log4j

Fields in org.apache.log4j declared as QuietWriter

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected QuietWriter</td>
<td>This is the quietWriter where we will write to.</td>
</tr>
</tbody>
</table>

Uses of QuietWriter in org.apache.log4j.helpers

Subclasses of QuietWriter in org.apache.log4j.helpers

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountingQuietWriter</td>
<td>Counts the number of bytes written.</td>
</tr>
<tr>
<td>SyslogQuietWriter</td>
<td>SyslogQuietWriter extends QuietWriter by prepending the syslog level code before each printed String.</td>
</tr>
</tbody>
</table>
Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.AbsoluteTimeDateFormat

Packages that use AbsoluteTimeDateFormat
org.apache.log4j.helpers This package is used internally.

Uses of AbsoluteTimeDateFormat in org.apache.log4j.helpers

Subclasses of AbsoluteTimeDateFormat in org.apache.log4j.helpers

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateTimeDateFormat</td>
<td>Formats a Date in the format &quot;dd MMM yyyy HH:mm:ss,SSS&quot; for example, &quot;06 Nov 1994 15:49:37,459&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO8601DateFormat</td>
<td>Formats a Date in the format &quot;yyyy-MM-dd HH:mm:ss,SSS&quot; for example &quot;1999-11-27 15:49:37,459&quot;.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.FormattingInfo

Packages that use FormattingInfo

| org.apache.log4j.helpers | This package is used internally. |

Uses of FormattingInfo in org.apache.log4j.helpers

Fields in org.apache.log4j.helpers declared as FormattingInfo

| protected PatternParser formattingInfo |

Constructors in org.apache.log4j.helpers with parameters of type FormattingInfo

| PatternConverter(FormattingInfo fi) |

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.NullEnumeration

Packages that use NullEnumeration

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
</tbody>
</table>

Uses of NullEnumeration in org.apache.log4j.helpers

Methods in org.apache.log4j.helpers that return NullEnumeration

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>NullEnumeration.getInstance()</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
## Uses of Class

**org.apache.log4j.helpers.PatternConverter**

### Packages that use `PatternConverter`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
</tbody>
</table>

### Uses of `PatternConverter` in `org.apache.log4j.helpers`

### Fields in `org.apache.log4j.helpers` declared as `PatternConverter`

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatternConverter</td>
<td>PatternConverter.next</td>
</tr>
</tbody>
</table>

### Methods in `org.apache.log4j.helpers` that return `PatternConverter`

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatternParser.parse()</td>
<td>PatternConverter</td>
</tr>
</tbody>
</table>

### Methods in `org.apache.log4j.helpers` with parameters of type `PatternConverter`

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void PatternParser.addConverter(PatternConverter pc)</td>
<td></td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
# Uses of Class

**org.apache.log4j.helpers.CyclicBuffer**

## Packages that use CyclicBuffer

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for remote logging.</td>
</tr>
</tbody>
</table>

## Uses of CyclicBuffer in org.apache.log4j.net

### Fields in org.apache.log4j.net declared as CyclicBuffer

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected CyclicBuffer</td>
</tr>
<tr>
<td>SMTPAppender.cb</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.AppenderAttachableImpl

No usage of org.apache.log4j.helpers.AppenderAttachableImpl

Copyright 2000-2005 Apache Software Foundation.
Uses of Class org.apache.log4j.helpers.BoundedFIFO

No usage of org.apache.log4j.helpers.BoundedFIFO

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.CountingQuietWriter

No usage of org.apache.log4j.helpers.CountingQuietWriter

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.DateTimeDateFormat

No usage of org.apache.log4j.helpers.DateTimeDateFormat

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.FileWatchdog

No usage of org.apache.log4j.helpers.FileWatchdog

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.ISO8601DateFormat

No usage of org.apache.log4j.helpers.ISO8601DateFormat

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.Loader

No usage of org.apache.log4j.helpers.Loader

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.LogLog

No usage of org.apache.log4j.helpers.LogLog

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.OnlyOnceErrorHandler

No usage of org.apache.log4j.helpers.OnlyOnceErrorHandler

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.OptionConverter

No usage of org.apache.log4j.helpers.OptionConverter

---

Copyright 2000-2005 Apache Software Foundation.
Uses of Class org.apache.log4j.helpers.RelativeTimeDateFormat

No usage of org.apache.log4j.helpers.RelativeTimeDateFormat

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.SyslogQuietWriter

No usage of org.apache.log4j.helpers.SyslogQuietWriter

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.helpers.SyslogWriter

No usage of org.apache.log4j.helpers.SyslogWriter

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.helpers.ThreadLocalMap
Uses of Class
org.apache.log4j.helpers.Transform

No usage of org.apache.log4j.helpers.Transform
# Uses of Class

`org.apache.log4j.jdbc.JDBCAppender`

No usage of `org.apache.log4j.jdbc.JDBCAppender`

---

<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 2000-2005 Apache Software Foundation.
# Uses of Class org.apache.log4j.jmx.AbstractDynamicMBean

## Packages that use `AbstractDynamicMBean`

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.jmx</td>
<td>This package lets you manage log4j settings using JMX.</td>
</tr>
</tbody>
</table>

## Uses of `AbstractDynamicMBean` in org.apache.log4j.jmx

## Subclasses of `AbstractDynamicMBean` in org.apache.log4j.jmx

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderDynamicMBean</td>
</tr>
<tr>
<td>HierarchyDynamicMBean</td>
</tr>
<tr>
<td>LayoutDynamicMBean</td>
</tr>
<tr>
<td>LoggerDynamicMBean</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.jmx.Agent

No usage of org.apache.log4j.jmx.Agent

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.jmx.AppenderDynamicMBean

No usage of org.apache.log4j.jmx.AppenderDynamicMBean

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.jmx.HierarchyDynamicMBean

No usage of org.apache.log4j.jmx.HierarchyDynamicMBean

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.jmx.LayoutDynamicMBean
Uses of Class
org.apache.log4j.jmx.LoggerDynamicMBean

No usage of org.apache.log4j.jmx.LoggerDynamicMBean
Uses of Class
org.apache.log4j.lf5.AppenderFinalizer

Packages that use AppenderFinalizer

org.apache.log4j.lf5

Uses of AppenderFinalizer in
org.apache.log4j.lf5

Fields in org.apache.log4j.lf5 declared as AppenderFinalizer

| protected static AppenderFinalizer LF5Appender._finalizer |

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.lf5.LF5Appender

Packages that use LF5Appender

| org.apache.log4j.lf5 |

Uses of LF5Appender in org.apache.log4j.lf5

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j.lf5 with parameters of type LF5Appender</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean LF5Appender.equals(LF5Appender compareTo)</td>
</tr>
<tr>
<td>The equals method compares two LF5Appenders and determines whether they are equal.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
# Uses of Class org.apache.log4j.lf5.LogLevel

## Packages that use LogLevel

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.lf5</td>
</tr>
</tbody>
</table>

## Uses of LogLevel in org.apache.log4j.lf5

<table>
<thead>
<tr>
<th>Field</th>
<th>LogLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>static LogLevel</td>
<td>LogLevel.FATAL</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.ERROR</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.WARN</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.INFO</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.DEBUG</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.SEVERE</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.WARNING</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.CONFIG</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.FINE</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.FINER</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.FINEST</td>
</tr>
<tr>
<td>protected LogRecord._level</td>
<td></td>
</tr>
<tr>
<td>Methods in org.apache.log4j.lf5 that return LogLevel</td>
<td></td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.valueOf(String level) Convert a log level label into a LogLevel object.</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.register(LogLevel logLevel) Registers a used defined LogLevel.</td>
</tr>
<tr>
<td>LogRecord.getLevel()</td>
<td>Get the level of this LogRecord.</td>
</tr>
<tr>
<td>Methods in org.apache.log4j.lf5 with parameters of type LogLevel</td>
<td></td>
</tr>
<tr>
<td>boolean LogLevel</td>
<td>LogLevel.encompasses(LogLevel level) Returns true if the level supplied is encompassed by this level.</td>
</tr>
<tr>
<td>static LogLevel</td>
<td>LogLevel.register(LogLevel logLevel) Registers a used defined LogLevel.</td>
</tr>
<tr>
<td>static void LogLevel.register(LogLevel[])</td>
<td>LogLevel.register(LogLevel[]) Registers a used defined LogLevel.</td>
</tr>
<tr>
<td>void LogLevel.setLogLevelColorMap(LogLevel level, Color color)</td>
<td></td>
</tr>
<tr>
<td>void LogRecord.setLevel(LogLevel level)</td>
<td>Set the level of this LogRecord.</td>
</tr>
</tbody>
</table>
## Uses of Class

**org.apache.log4j.lf5.LogLevelFormatException**

### Packages that use **LogLevelFormatException**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.lf5</td>
<td></td>
</tr>
</tbody>
</table>

### Uses of **LogLevelFormatException** in **org.apache.log4j.lf5**

### Methods in **org.apache.log4j.lf5** that throw **LogLevelFormatException**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static LogLevel.valueOf(String level)</td>
<td>Convert a log level label into a LogLevel object.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
# Uses of Class LogRecord

## Packages that use LogRecord

<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.lf5</td>
</tr>
</tbody>
</table>

## Uses of LogRecord in org.apache.log4j.lf5

### Subclasses of LogRecord in org.apache.log4j.lf5

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log4JLogRecord</td>
<td>A Log4JLogRecord encapsulates the details of your log4j LoggingEvent in a format usable by the LogBrokerMonitor.</td>
</tr>
</tbody>
</table>

### Methods in org.apache.log4j.lf5 with parameters of type LogRecord

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogRecordFilter.passes</td>
<td></td>
</tr>
<tr>
<td>PassingLogRecordFilter.passes</td>
<td></td>
</tr>
</tbody>
</table>

---

Copyright 2000-2005 Apache Software Foundation.
Uses of Interface
org.apache.log4j.lf5.LogRecordFilter

Packages that use LogRecordFilter
org.apache.log4j.lf5

Uses of LogRecordFilter in
org.apache.log4j.lf5

Classes in org.apache.log4j.lf5 that implement LogRecordFilter

<table>
<thead>
<tr>
<th>Class</th>
<th>Use</th>
</tr>
</thead>
</table>
| PassingLogRecordFilter | Class in org.apache.log4j.lf5 that implement LogRecordFilter
An implementation of LogRecordFilter which always returns true.

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.lf5.DefaultLF5Configurator

No usage of org.apache.log4j.lf5.DefaultLF5Configurator

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.lf5.Log4JLogRecord

No usage of org.apache.log4j.lf5.Log4JLogRecord

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.lf5.PassingLogRecordFilter

No usage of org.apache.log4j.lf5.PassingLogRecordFilter

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.lf5.StartLogFactor5

No usage of org.apache.log4j.lf5.StartLogFactor5

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.TelnetAppender

Packages that use **TelnetAppender**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for remote logging.</td>
</tr>
</tbody>
</table>

Uses of **TelnetAppender** in

org.apache.log4j.net

Constructors in [org.apache.log4j.net](http://log4j.apache.org/docs/apidocs/org/apache/log4j/net/package-summary.html) with parameters of type **TelnetAppender**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>TelnetAppender.SocketHandler(int port)</td>
<td></td>
</tr>
</tbody>
</table>
Uses of Class org.apache.log4j.net.JMSAppender

No usage of org.apache.log4j.net.JMSAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.JMSSink

No usage of org.apache.log4j.net.JMSSink

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.SimpleSocketServer

No usage of org.apache.log4j.net.SimpleSocketServer
Uses of Class
org.apache.log4j.net.SMTPAppender

No usage of org.apache.log4j.net.SMTPAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.SocketAppender

No usage of org.apache.log4j.net.SocketAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.SocketHubAppender

No usage of org.apache.log4j.net.SocketHubAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.SocketNode

No usage of org.apache.log4j.net.SocketNode

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.SocketServer

No usage of org.apache.log4j.net.SocketServer

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.net.TelnetAppender.SocketHandler

No usage of org.apache.log4j.net.TelnetAppender.SocketHandler

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.nt.NTEventLogAppender
Uses of Interface
org.apache.log4j.or.ObjectRenderer

Packages that use ObjectRenderer

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.or</td>
<td>ObjectRenderers are responsible for rendering messages depending on their class type.</td>
</tr>
<tr>
<td>org.apache.log4j.or.jms</td>
<td>This package contains the MessageRenderer which renders objects of type javax.jms.Message.</td>
</tr>
<tr>
<td>org.apache.log4j.or.sax</td>
<td>This package contains the AttributesRenderer which renders objects of class org.xml.sax.Attributes.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
</tbody>
</table>

Uses of ObjectRenderer in org.apache.log4j

Methods in org.apache.log4j with parameters of type ObjectRenderer

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addRenderer(Class classToRender, ObjectRenderer or)</td>
<td>Add an object renderer for a specific class.</td>
</tr>
<tr>
<td>void setRenderer(Class renderedClass, ObjectRenderer renderer)</td>
<td>Used by subclasses to add a renderer to the hierarchy passed as parameter.</td>
</tr>
</tbody>
</table>

Uses of ObjectRenderer in
**org.apache.log4j.or**

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j.or that implement ObjectRenderer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ThreadGroupRenderer</strong></td>
</tr>
<tr>
<td>Render ThreadGroup objects in a format similar to the</td>
</tr>
<tr>
<td>information output by the ThreadGroup.list() method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j.or that return ObjectRenderer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ObjectRenderer</strong></td>
</tr>
<tr>
<td>RendererMap.get(Object o)</td>
</tr>
<tr>
<td>Syntactic sugar method that calls RendererMap.get(Class)</td>
</tr>
<tr>
<td>with the class of the object parameter.</td>
</tr>
<tr>
<td><strong>ObjectRenderer</strong></td>
</tr>
<tr>
<td>RendererMap.get(Class clazz)</td>
</tr>
<tr>
<td>Search the parents of clazz for a renderer.</td>
</tr>
<tr>
<td><strong>ObjectRenderer</strong></td>
</tr>
<tr>
<td>RendererMap.getDefaultRenderer()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j.or with parameters of type ObjectRenderer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong> RendererMap.put(Class clazz, ObjectRenderer or)**</td>
</tr>
<tr>
<td>Register an ObjectRenderer for clazz.</td>
</tr>
</tbody>
</table>

**Uses of ObjectRenderer in org.apache.log4j.or.jms**

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j.or.jms that implement ObjectRenderer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MessageRenderer</strong></td>
</tr>
<tr>
<td>Render javax.jms.Message objects.</td>
</tr>
</tbody>
</table>
Uses of **ObjectRenderer** in
```
org.apache.log4j.or.sax
```

**Classes in** `org.apache.log4j.or.sax` **that implement** `ObjectRenderer`

<table>
<thead>
<tr>
<th>class</th>
<th>AttributesRenderer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Render</td>
<td>org.xml.sax.Attributes objects</td>
</tr>
</tbody>
</table>

**Uses of** **ObjectRenderer** **in**
```
org.apache.log4j.spi
```

**Methods in** `org.apache.log4j.spi` **with parameters of type** `ObjectRenderer`

| void | RendererSupport.prototype.setRenderer(Class renderedClass, ObjectRenderer renderer) |

---

Copyright 2000-2005 Apache Software Foundation.
## Uses of Class

**org.apache.log4j.or.RendererMap**

<table>
<thead>
<tr>
<th>Packages that use RendererMap</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
</tbody>
</table>

## Uses of **RendererMap** in **org.apache.log4j**

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j that return RendererMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>RendererMap.getRendererMap()</td>
</tr>
<tr>
<td>Get the renderer map for this hierarchy.</td>
</tr>
</tbody>
</table>

## Uses of **RendererMap** in **org.apache.log4j.spi**

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j.spi that return RendererMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>RendererSupport.getRendererMap()</td>
</tr>
</tbody>
</table>

---

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.or.ThreadGroupRenderer

No usage of org.apache.log4j.or.ThreadGroupRenderer

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.or.jms.MessageRenderer

No usage of org.apache.log4j.or.jms.MessageRenderer
Uses of Class
org.apache.log4j.or.sax.AttributesRenderer

No usage of org.apache.log4j.or.sax.AttributesRenderer

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.performance.ListVsVector

No usage of org.apache.log4j.performance.ListVsVector

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.performance.NewVsSetLen

No usage of org.apache.log4j.performance.NewVsSetLen

Copyright 2000-2005 Apache Software Foundation.
Uses of Class org.apache.log4j.performance.NOPWriter

No usage of org.apache.log4j.performance.NOPWriter

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.performance.NullAppender

No usage of org.apache.log4j.performance.NullAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.performance.SystemTime

No usage of org.apache.log4j.performance.SystemTime

Copyright 2000-2005 Apache Software Foundation.
## Uses of Interface org.apache.log4j.spi.AppenderAttachable

### Packages that use **AppenderAttachable**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.apache.log4j</strong></td>
<td>The main log4j package.</td>
</tr>
<tr>
<td><strong>org.apache.log4j.helpers</strong></td>
<td>This package is used internally.</td>
</tr>
<tr>
<td><strong>org.apache.log4j.spi</strong></td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
</tbody>
</table>

### Uses of **AppenderAttachable** in **org.apache.log4j**

#### Classes in **org.apache.log4j** that implement **AppenderAttachable**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsyncAppender</strong></td>
<td>The AsyncAppender lets users log events asynchronously.</td>
</tr>
<tr>
<td><strong>Category</strong></td>
<td>This class has been deprecated and replaced by the Logger subclass.</td>
</tr>
<tr>
<td><strong>Logger</strong></td>
<td>This is the central class in the log4j package.</td>
</tr>
</tbody>
</table>

### Uses of **AppenderAttachable** in **org.apache.log4j.helpers**

#### Classes in **org.apache.log4j.helpers** that implement **AppenderAttachable**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppenderAttachableImpl</strong></td>
<td></td>
</tr>
</tbody>
</table>
A straightforward implementation of the AppenderAttachable interface.

Uses of AppenderAttachable in org.apache.log4j.spi

<table>
<thead>
<tr>
<th>Class</th>
<th>RootCategory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deprecated. Replaced by RootLogger.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>RootLogger</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RootLogger sits at the top of the logger hierarchy.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
# Uses of Interface

## org.apache.log4j.spi.Configurator

### Packages that use Configurator

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.lf5</td>
<td></td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

### Uses of Configurator in org.apache.log4j

### Classes in org.apache.log4j that implement Configurator

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyConfigurator</td>
<td>Allows the configuration of log4j from an external file.</td>
</tr>
</tbody>
</table>

### Uses of Configurator in org.apache.log4j.lf5

### Classes in org.apache.log4j.lf5 that implement Configurator

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultLF5Configurator</td>
<td>The DefaultLF5Configurator provides a default configuration for the LF5Appender.</td>
</tr>
</tbody>
</table>

### Uses of Configurator in org.apache.log4j.varia
### Classes in `org.apache.log4j.varia` that implement `Configurator`

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReloadingPropertyConfigurator</code></td>
</tr>
</tbody>
</table>

### Uses of `Configurator` in `org.apache.log4j.xml`

### Classes in `org.apache.log4j.xml` that implement `Configurator`

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DOMConfigurator</code></td>
</tr>
</tbody>
</table>

Use this class to initialize the log4j environment using a DOM tree.
Uses of Interface
org.apache.log4j.spi.ErrorHandler

Packages that use **ErrorHandler**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
</tbody>
</table>

Uses of **ErrorHandler** in **org.apache.log4j**

Fields in **org.apache.log4j** declared as **ErrorHandler**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected <strong>ErrorHandler</strong></td>
<td>It is assumed and enforced that errorHandler is never null.</td>
</tr>
</tbody>
</table>

Methods in **org.apache.log4j** that return **ErrorHandler**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderSkeleton.getErrorHandler()</td>
<td>Return the currently set <strong>ErrorHandler</strong> for this Appender.</td>
</tr>
<tr>
<td>Appender.getErrorHandler()</td>
<td>Returns the <strong>ErrorHandler</strong> for this appender.</td>
</tr>
</tbody>
</table>

Methods in **org.apache.log4j** with parameters of type **ErrorHandler**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderSkeleton.setErrorHandler(ErrorHandler eh)</td>
<td>Set the <strong>ErrorHandler</strong> for this Appender.</td>
</tr>
<tr>
<td>WriterAppender.setErrorHandler(ErrorHandler eh)</td>
<td>Set the <strong>ErrorHandler</strong> for this WriterAppender and also the</td>
</tr>
</tbody>
</table>
void Appender.setErrorHandler(ErrorHandler errorHandler)
  Set the ErrorHandler for this appender.

### Uses of ErrorHandler in org.apache.log4j.helpers

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j.helpers that implement ErrorHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnlyOnceErrorHandler</td>
</tr>
<tr>
<td>The OnlyOnceErrorHandler implements log4j's default error handling policy which consists of emitting a message for the first error in an appender and ignoring all following errors.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields in org.apache.log4j.helpers declared as ErrorHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected ErrorHandler QuietWriter.errorHandler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j.helpers with parameters of type ErrorHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>void QuietWriter.setErrorHandler(ErrorHandler eh)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructors in org.apache.log4j.helpers with parameters of type ErrorHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuietWriter(Writer writer, ErrorHandler errorHandler)</td>
</tr>
<tr>
<td>CountingQuietWriter(Writer writer, ErrorHandler eh)</td>
</tr>
<tr>
<td>SyslogQuietWriter(Writer writer, int syslogFacility,</td>
</tr>
</tbody>
</table>
Uses of **ErrorHandler** in org.apache.log4j.varia

Classes in org.apache.log4j.varia that implement **ErrorHandler**

<table>
<thead>
<tr>
<th>class</th>
<th>FallbackErrorHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>The FallbackErrorHandler implements the ErrorHandler interface such that a secondary appender may be specified.</td>
<td></td>
</tr>
</tbody>
</table>
Uses of Class
org.apache.log4j.spi.Filter

Packages that use Filter

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
</tbody>
</table>

Uses of Filter in org.apache.log4j

Fields in org.apache.log4j declared as Filter

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderSkeleton.headFilter</td>
<td>The first filter in the filter chain.</td>
</tr>
<tr>
<td>AppenderSkeleton.tailFilter</td>
<td>The last filter in the filter chain.</td>
</tr>
</tbody>
</table>

Methods in org.apache.log4j that return Filter

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderSkeleton.getFilter()</td>
<td>Returns the head Filter.</td>
</tr>
<tr>
<td>AppenderSkeleton.getFirstFilter()</td>
<td>Return the first filter in the filter chain for this Appender.</td>
</tr>
<tr>
<td>Appender.getFilter()</td>
<td>Returns the head Filter.</td>
</tr>
</tbody>
</table>

Methods in org.apache.log4j with parameters of type Filter

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderSkeleton.addFilter(Filter newFilter)</td>
<td>Add a filter to the filter chain.</td>
</tr>
</tbody>
</table>
void Add a filter to the end of the filter list.

void Appender.\textit{addFilter}(Filter newFilter)
Add a filter to the end of the filter list.

\section*{Uses of Filter in \texttt{org.apache.log4j.spi}}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\texttt{Filter} & \texttt{next} \\
\hline
\multicolumn{2}{|c|}{\textbf{Deprecated}. As of 1.2.12, use \texttt{getNext()} and \texttt{setNext(\texttt{org.apache.log4j.spi.Filter})} instead} \\
\hline
\end{tabular}
\end{table}

\section*{Methods in \texttt{org.apache.log4j.spi} that return Filter}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\texttt{Filter} & \texttt{getNext()} \\
\hline
\multicolumn{2}{|c|}{Return the pointer to the next filter;} \\
\hline
\end{tabular}
\end{table}

\section*{Methods in \texttt{org.apache.log4j.spi} with parameters of type Filter}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\texttt{Filter} & \texttt{setNext(\texttt{Filter} next)} \\
\hline
\multicolumn{2}{|c|}{Set the next filter pointer.} \\
\hline
\end{tabular}
\end{table}

\section*{Uses of Filter in \texttt{org.apache.log4j.varia}}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\texttt{class} & \texttt{DenyAllFilter} \\
\hline
\multicolumn{2}{|c|}{This filter drops all logging events.} \\
\hline
\texttt{class} & \texttt{LevelMatchFilter} \\
\hline
\multicolumn{2}{|c|}{This is a very simple filter based on level matching.} \\
\hline
\texttt{class} & \texttt{LevelRangeFilter} \\
\hline
\end{tabular}
\end{table}
| class StringMatchFilter | This is a very simple filter based on string matching. |
Uses of Interface
org.apache.log4j.spi.HierarchyEventListener

<table>
<thead>
<tr>
<th>Packages that use HierarchyEventListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
</tr>
<tr>
<td>org.apache.log4j.jmx</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
</tr>
</tbody>
</table>

Uses of HierarchyEventListener in org.apache.log4j

Methods in org.apache.log4j with parameters of type HierarchyEventListener:
void Hierarchy.addHierarchyEventListener(HierarchyEventListener listener)

Uses of HierarchyEventListener in org.apache.log4j.jmx

Classes in org.apache.log4j.jmx that implement HierarchyEventListener:
class HierarchyDynamicMBean

Uses of HierarchyEventListener in
Methods in org.apache.log4j.spi with parameters of type HierarchyEventListener

| void LoggerRepository. addHierarchyEventListener(HierarchyEventListener event) |
| Add a HierarchyEventListener event to the repository. |

Copyright 2000-2005 Apache Software Foundation.
Uses of Interface
org.apache.log4j.spi.LoggerFactory

Packages that use LoggerFactory

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
</tbody>
</table>

Uses of LoggerFactory in org.apache.log4j

Fields in org.apache.log4j declared as LoggerFactory

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected LoggerFactory</td>
<td>PropertyConfigurator.LoggerFactory</td>
</tr>
</tbody>
</table>

Methods in org.apache.log4j with parameters of type LoggerFactory

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Logger.getLogger(String name, LoggerFactory factory)</td>
<td>Retrieve the appropriate Logger instance.</td>
</tr>
<tr>
<td>static Logger.getLogger(String name, LoggerFactory factory)</td>
<td>Like Logger.getLogger(String) except that the type of logger instantiated depends on the type returned by the makeNewLoggerInstance(java.lang.String) method of the factory parameter.</td>
</tr>
<tr>
<td>Hierarchy.getLogger(String name, LoggerFactory factory)</td>
<td>Return a new logger instance named as the first parameter using factory.</td>
</tr>
</tbody>
</table>

Uses of LoggerFactory in
Methods in `org.apache.log4j.spi` with parameters of type `LoggerFactory`:

```java
LoggerRepository.getLogger(String name, LoggerFactory factory)
```
Uses of Interface org.apache.log4j.spi.LoggerRepository

<table>
<thead>
<tr>
<th>Packages that use LoggerRepository</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.lf5</td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of LoggerRepository in org.apache.log4j</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Classes in org.apache.log4j that implement LoggerRepository</th>
</tr>
</thead>
<tbody>
<tr>
<td>class Hierarchy This class is specialized in retrieving loggers by name and also maintaining the logger hierarchy.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields in org.apache.log4j declared as LoggerRepository</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected LoggerRepository Category.repository</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j that return LoggerRepository</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static LoggerRepository</td>
<td>Category. <strong>getDefualtHierarchy()</strong>&lt;br&gt;<strong>Deprecated. Please use</strong>&lt;br&gt;LogManager.getLoggerRepository() <strong>instead.</strong></td>
</tr>
<tr>
<td></td>
<td>Category. <strong>getHierarchy()</strong>&lt;br&gt;<strong>Deprecated. Please use</strong>&lt;br&gt;Category.getLoggerRepository() <strong>instead.</strong></td>
</tr>
<tr>
<td></td>
<td>Category. <strong>getLoggerRepository()</strong>&lt;br&gt;Return the the LoggerRepository where this Category is attached.</td>
</tr>
<tr>
<td>static LoggerRepository</td>
<td>LogManager. <strong>getLoggerRepository()</strong></td>
</tr>
</tbody>
</table>

**Methods in org.apache.log4j with parameters of type LoggerRepository**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void PropertyConfigurator. <strong>doConfigure</strong>(String configFileName, LoggerRepository hierarchy)</td>
<td>Read configuration from a file.</td>
</tr>
<tr>
<td>void PropertyConfigurator. <strong>doConfigure</strong>(Properties properties, LoggerRepository hierarchy)</td>
<td>Read configuration options from properties.</td>
</tr>
<tr>
<td>void PropertyConfigurator. <strong>doConfigure</strong>(URL configURL, LoggerRepository hierarchy)</td>
<td>Read configuration options from url configURL.</td>
</tr>
<tr>
<td>protected void PropertyConfigurator. <strong>parseCatsAndRenderers</strong>(Properties props, LoggerRepository hierarchy)</td>
<td>Parse non-root elements, such non-root categories and renderers.</td>
</tr>
</tbody>
</table>

**Uses of LoggerRepository in org.apache.log4j.helpers**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods in org.apache.log4j.helpers with parameters of type LoggerRepository</td>
<td></td>
</tr>
</tbody>
</table>
static void OptionConverter.selectAndConfigure(URL url, String clazz, LoggerRepository hierarchy)

Configure log4j given a URL.

Uses of LoggerRepository in org.apache.log4j.lf5

Methods in org.apache.log4j.lf5 with parameters of type LoggerRepository

void DefaultLF5Configurator.doConfigure(URL configURL, LoggerRepository repository)

This is a dummy method that will throw an IllegalStateException if used.

Uses of LoggerRepository in org.apache.log4j.net

Constructors in org.apache.log4j.net with parameters of type LoggerRepository

SocketNode(Socket socket, LoggerRepository hierarchy)

Uses of LoggerRepository in org.apache.log4j.spi

Methods in org.apache.log4j.spi that return LoggerRepository
DefaultRepositorySelector.getRepository()

Returns a LoggerRepository depending on the context.

Methods in org.apache.log4j.spi with parameters of type LoggerRepository

void Configurator.doConfigure(URL url, LoggerRepository repository)

Interpret a resource pointed by a URL and set up log4j accordingly.

Constructors in org.apache.log4j.spi with parameters of type LoggerRepository

DefaultRepositorySelector(LoggerRepository repository)

Uses of LoggerRepository in org.apache.log4j.varia

Methods in org.apache.log4j.varia with parameters of type LoggerRepository

void ReloadingPropertyConfigurator.doConfigure(URL url, LoggerRepository repository)

Uses of LoggerRepository in org.apache.log4j.xml
Methods in `org.apache.log4j.xml` with parameters of type `LoggerRepository`:

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>DOMConfigurator.doConfigure(String filename, LoggerRepository repository)</code></td>
<td>Configure log4j by reading in a log4j.dtd compliant XML configuration file.</td>
</tr>
<tr>
<td>void</td>
<td><code>DOMConfigurator.doConfigure(URL url, LoggerRepository repository)</code></td>
<td>Configure log4j by reading in a log4j.dtd compliant XML configuration file.</td>
</tr>
<tr>
<td>void</td>
<td><code>DOMConfigurator.doConfigure(Reader reader, LoggerRepository repository)</code></td>
<td>Configure log4j by reading in a log4j.dtd compliant XML configuration file.</td>
</tr>
<tr>
<td>protected void</td>
<td><code>DOMConfigurator.doConfigure(InputSource inputSource, LoggerRepository repository)</code></td>
<td>Configure log4j by reading in a log4j.dtd compliant XML configuration file.</td>
</tr>
<tr>
<td>void</td>
<td><code>DOMConfigurator.doConfigure(Element element, LoggerRepository repository)</code></td>
<td>Configure by taking in an DOM element.</td>
</tr>
</tbody>
</table>

Overview | Package | Class | Tree | Deprecated | Index | Help | Log4j 1.2.14
PREV | NEXT | FRAMES | NO FRAMES

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.spi.LoggingEvent

<table>
<thead>
<tr>
<th>Packages that use LoggingEvent</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.jdbc</td>
<td>The JDBCAppender provides for sending log events to a database.</td>
</tr>
<tr>
<td>org.apache.log4j.lf5</td>
<td></td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td>org.apache.log4j.nt</td>
<td>Package for NT event logging.</td>
</tr>
<tr>
<td>org.apache.log4j.performance</td>
<td>Package to measure the performance of the different log4j components.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

Uses of LoggingEvent in org.apache.log4j

Methods in org.apache.log4j with parameters of type LoggingEvent

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract String Layout.format(LoggingEvent event)</td>
<td>Implement this method to create your own layout format.</td>
</tr>
<tr>
<td>String SimpleLayout.format(LoggingEvent event)</td>
<td>Returns the log statement in a format consisting of the level, followed by &quot; - &quot; and then the message.</td>
</tr>
<tr>
<td>String TTCCLayout.format(LoggingEvent event)</td>
<td></td>
</tr>
</tbody>
</table>
In addition to the level of the statement and message, the returned byte array includes time, thread, category and NDC information.

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td>Category.callAppenders(JavaEvent event)</td>
<td>Call the appenders in the hierarchy starting at this.</td>
</tr>
<tr>
<td>String</td>
<td>PatternLayout.format(JavaEvent event)</td>
<td>Produces a formatted string as specified by the conversion pattern.</td>
</tr>
<tr>
<td>String</td>
<td>HTMLLayout.format(JavaEvent event)</td>
<td></td>
</tr>
<tr>
<td>Protected</td>
<td>AppenderSkeleton.append(JavaEvent event)</td>
<td>Subclasses of AppenderSkeleton should implement this method to perform actual logging.</td>
</tr>
<tr>
<td>Void</td>
<td>AppenderSkeleton.doAppend(JavaEvent event)</td>
<td>This method performs threshold checks and invokes filters before delegating actual logging to the subclasses specific AppenderSkeleton.append(JavaEvent event) method.</td>
</tr>
<tr>
<td>Void</td>
<td>WriterAppender.append(JavaEvent event)</td>
<td>This method is called by the AppenderSkeleton.doAppend(JavaEvent event) method.</td>
</tr>
<tr>
<td>Protected</td>
<td>WriterAppender.subAppend(JavaEvent event)</td>
<td>Actual writing occurs here.</td>
</tr>
<tr>
<td>Void</td>
<td>Appender.doAppend(JavaEvent event)</td>
<td>Log in Appender specific way.</td>
</tr>
<tr>
<td>Protected</td>
<td>DailyRollingFileAppender.subAppend(JavaEvent event)</td>
<td>This method differentiates DailyRollingFileAppender from its super class.</td>
</tr>
<tr>
<td>Protected</td>
<td>RollingFileAppender.subAppend(JavaEvent event)</td>
<td>This method differentiates RollingFileAppender from its super class.</td>
</tr>
<tr>
<td>Void</td>
<td>AsyncAppender.append(JavaEvent event)</td>
<td>{@inheritdoc}</td>
</tr>
</tbody>
</table>
Uses of **LoggingEvent** in
**org.apache.log4j.helpers**

### Methods in **org.apache.log4j.helpers** that return **LoggingEvent**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CyclicBuffer.get(int i)</code></td>
<td>Get the i(^{th}) oldest event currently in the buffer.</td>
</tr>
<tr>
<td><code>CyclicBuffer.get()</code></td>
<td>Get the oldest (first) element in the buffer.</td>
</tr>
<tr>
<td><code>BoundedFIFO.get()</code></td>
<td>Get the first element in the buffer.</td>
</tr>
</tbody>
</table>

### Methods in **org.apache.log4j.helpers** with parameters of type **LoggingEvent**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DateLayout.dateFormat(StringBuffer buf, LoggingEvent event)</code></td>
<td></td>
</tr>
<tr>
<td><code>CyclicBuffer.add(LoggingEvent event)</code></td>
<td>Add an event as the last event in the buffer.</td>
</tr>
<tr>
<td><code>BoundedFIFO.put(LoggingEvent o)</code></td>
<td>Place a LoggingEvent in the buffer.</td>
</tr>
<tr>
<td><code>PatternConverter.convert(LoggingEvent event)</code></td>
<td>Derived pattern converters must override this method in order to convert conversion specifiers in the correct way.</td>
</tr>
<tr>
<td><code>PatternConverter.format(StringBuffer sbuf, LoggingEvent e)</code></td>
<td>A template method for formatting in a converter specific way.</td>
</tr>
<tr>
<td><code>AppenderAttachableImpl.appendLoopOnAppenders(LoggingEvent ev)</code></td>
<td>Call the doAppend method on all attached appenders.</td>
</tr>
<tr>
<td><code>OnlyOnceErrorHandler.error(String message, Exception e, int errorCode, LoggingEvent event)</code></td>
<td>Prints the message and the stack trace of the exception on System.err.</td>
</tr>
</tbody>
</table>
Uses of `LoggingEvent` in `org.apache.log4j.jdbc`

Methods in `org.apache.log4j.jdbc` with parameters of type `LoggingEvent`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void JDBCAppender.append(LoggingEvent event)</code></td>
<td>Adds the event to the buffer.</td>
</tr>
<tr>
<td><code>String JDBCAppender.getLogStatement(LoggingEvent event)</code></td>
<td>By default getLogStatement sends the event to the required Layout object.</td>
</tr>
</tbody>
</table>

Uses of `LoggingEvent` in `org.apache.log4j.lf5`

Methods in `org.apache.log4j.lf5` with parameters of type `LoggingEvent`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void LF5Appender.append(LoggingEvent event)</code></td>
<td>Appends a LoggingEvent record to the LF5Appender.</td>
</tr>
</tbody>
</table>

Uses of `LoggingEvent` in `org.apache.log4j.net`

Methods in `org.apache.log4j.net` with parameters of type `LoggingEvent`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void SocketHubAppender.append(LoggingEvent event)</code></td>
<td>Append an event to all of current connections.</td>
</tr>
<tr>
<td><code>void SyslogAppender.append(LoggingEvent event)</code></td>
<td></td>
</tr>
<tr>
<td><code>void SMTPAppender.append(LoggingEvent event)</code></td>
<td></td>
</tr>
</tbody>
</table>
### Uses of `LoggingEvent` in `org.apache.log4j.nt`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>JMSAppender.append(LoggingEvent event)</code></td>
<td>Handles appending actions to JMS.</td>
</tr>
<tr>
<td><code>TelnetAppender.append(LoggingEvent event)</code></td>
<td>Handles appending actions to Telnet.</td>
</tr>
<tr>
<td><code>SocketAppender.append(LoggingEvent event)</code></td>
<td>Handles appending actions to Socket.</td>
</tr>
<tr>
<td><code>NTEventLogAppender.append(LoggingEvent event)</code></td>
<td>Handles appending actions to NT Event Log.</td>
</tr>
</tbody>
</table>

### Uses of `LoggingEvent` in `org.apache.log4j.performance`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NullAppender.doAppend(LoggingEvent event)</code></td>
<td>Handles appending actions to Null.</td>
</tr>
<tr>
<td><code>NullAppender.append(LoggingEvent event)</code></td>
<td>Handles appending actions to Null.</td>
</tr>
</tbody>
</table>
Uses of *LoggingEvent* in *org.apache.log4j.spi*

<table>
<thead>
<tr>
<th>Methods in <em>org.apache.log4j.spi</em> with parameters of type <em>LoggingEvent</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ErrorHandler</strong>. (\textbf{error}()) (String message, Exception e, int errorCode, LoggingEvent event)</td>
</tr>
<tr>
<td>This method is invoked to handle the error.</td>
</tr>
<tr>
<td><strong>TriggeringEventEvaluator</strong>. (\textbf{isTriggeringEvent}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>Is this the triggering event?</td>
</tr>
<tr>
<td><strong>Filter</strong>. (\textbf{decide}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>If the decision is DENY, then the event will be dropped.</td>
</tr>
</tbody>
</table>

Uses of *LoggingEvent* in *org.apache.log4j.varia*

<table>
<thead>
<tr>
<th>Methods in <em>org.apache.log4j.varia</em> with parameters of type <em>LoggingEvent</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DenyAllFilter</strong>. (\textbf{decide}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>Always returns the integer constant Filter.DENY regardless of the LoggingEvent parameter.</td>
</tr>
<tr>
<td><strong>NullAppender</strong>. (\textbf{doAppend}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>Does not do anything.</td>
</tr>
<tr>
<td><strong>NullAppender</strong>. (\textbf{append}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>Does not do anything.</td>
</tr>
<tr>
<td><strong>StringMatchFilter</strong>. (\textbf{decide}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>Returns Filter.NEUTRAL is there is no string match.</td>
</tr>
<tr>
<td><strong>LevelMatchFilter</strong>. (\textbf{decide}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>Return the decision of this filter.</td>
</tr>
<tr>
<td><strong>LevelRangeFilter</strong>. (\textbf{decide}()) (LoggingEvent event)</td>
</tr>
<tr>
<td>Return the decision of this filter.</td>
</tr>
</tbody>
</table>
void FallbackErrorHandler.error(String message, Exception e, int errorCode, LoggingEvent event)

Prints the message and the stack trace of the exception on System.err.

Uses of **LoggingEvent** in **org.apache.log4j.xml**

Methods in **org.apache.log4j.xml** with parameters of type **LoggingEvent**

**String** XMLLayout.format(LoggingEvent event)

Formats a **LoggingEvent** in conformance with the log4j.dtd.

Copyright 2000-2005 Apache Software Foundation.
Uses of Interface
org.apache.log4j.spi.OptionHandler

### Packages that use **OptionHandler**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.helpers</td>
<td>This package is used internally.</td>
</tr>
<tr>
<td>org.apache.log4j.jdbc</td>
<td>The JDBCAppender provides for sending log events to a database.</td>
</tr>
<tr>
<td>org.apache.log4j.nt</td>
<td>Package for remote logging.</td>
</tr>
<tr>
<td>org.apache.log4j.net</td>
<td>Package for NT event logging.</td>
</tr>
<tr>
<td>org.apache.log4j.performance</td>
<td>Package to measure the performance of the different log4j components.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.varia</td>
<td>Contains various appenders, filters and other odds and ends.</td>
</tr>
<tr>
<td>org.apache.log4j.xml</td>
<td>XML based components.</td>
</tr>
</tbody>
</table>

Uses of **OptionHandler** in **org.apache.log4j**

### Classes in **org.apache.log4j** that implement **OptionHandler**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppenderSkeleton</td>
<td>Abstract superclass of the other appenders in the package.</td>
</tr>
<tr>
<td>AsyncAppender</td>
<td>The AsyncAppender lets users log events asynchronously.</td>
</tr>
<tr>
<td>ConsoleAppender</td>
<td>ConsoleAppender appends log events to System.out or</td>
</tr>
</tbody>
</table>
System.err using a layout specified by the user.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DailyRollingFileAppender</strong></td>
<td>DailyRollingFileAppender extends <strong>FileAppender</strong> so that the underlying file is rolled over at a user chosen frequency.</td>
</tr>
<tr>
<td><strong>FileAppender</strong></td>
<td>FileAppender appends log events to a file.</td>
</tr>
<tr>
<td><strong>HTMLLayout</strong></td>
<td>This layout outputs events in a HTML table.</td>
</tr>
<tr>
<td><strong>Layout</strong></td>
<td>Extend this abstract class to create your own log layout format.</td>
</tr>
<tr>
<td><strong>PatternLayout</strong></td>
<td>A flexible layout configurable with pattern string.</td>
</tr>
<tr>
<td><strong>RollingFileAppender</strong></td>
<td>RollingFileAppender extends FileAppender to backup the log files when they reach a certain size.</td>
</tr>
<tr>
<td><strong>SimpleLayout</strong></td>
<td>SimpleLayout consists of the level of the log statement, followed by &quot; - &quot; and then the log message itself.</td>
</tr>
<tr>
<td><strong>TTCCLayout</strong></td>
<td>TTCC layout format consists of time, thread, category and nested diagnostic context information, hence the name.</td>
</tr>
<tr>
<td><strong>WriterAppender</strong></td>
<td>WriterAppender appends log events to a <strong>Writer</strong> or an <strong>OutputStream</strong> depending on the user's choice.</td>
</tr>
</tbody>
</table>

**Uses of** **OptionHandler** **in**

**org.apache.log4j.helpers**

**Classes in** **org.apache.log4j.helpers** **that implement** **OptionHandler**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DateLayout</strong></td>
<td>This abstract layout takes care of all the date related options</td>
</tr>
</tbody>
</table>
The `OnlyOnceErrorHandler` implements log4j's default error handling policy which consists of emitting a message for the first error in an appender and ignoring all following errors.

### Uses of `OptionHandler` in `org.apache.log4j.jdbc`

Classes in `org.apache.log4j.jdbc` that implement `OptionHandler`:

- **JDBCAppender**
  - WARNING: This version of JDBCAppender is very likely to be completely replaced in the future.

### Uses of `OptionHandler` in `org.apache.log4j.lf5`

Classes in `org.apache.log4j.lf5` that implement `OptionHandler`:

- **LF5Appender**
  - LF5Appender logs events to a swing based logging console.

### Uses of `OptionHandler` in `org.apache.log4j.net`

Classes in `org.apache.log4j.net` that implement `OptionHandler`:

- **JMSAppender**
### Uses of **OptionHandler** in **org.apache.log4j.nt**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NullAppender</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SocketHubAppender</strong></td>
<td>Sends LoggingEvent objects to a set of remote log servers, usually a SocketNodes.</td>
</tr>
<tr>
<td><strong>SMTPAppender</strong></td>
<td>Send an e-mail when a specific logging event occurs, typically on errors or fatal errors.</td>
</tr>
<tr>
<td><strong>SocketAppender</strong></td>
<td>Sends LoggingEvent objects to a remote log server, usually a SocketNode.</td>
</tr>
<tr>
<td><strong>SyslogAppender</strong></td>
<td>Use SyslogAppender to send log messages to a remote syslog daemon.</td>
</tr>
<tr>
<td><strong>TelnetAppender</strong></td>
<td>The TelnetAppender is a log4j appender that specializes in writing to a read-only socket.</td>
</tr>
</tbody>
</table>

### Uses of **OptionHandler** in **org.apache.log4j.performance**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NullAppender</strong></td>
<td></td>
</tr>
</tbody>
</table>

Classes in **org.apache.log4j.nt** that implement **OptionHandler**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NullAppender</strong></td>
<td></td>
</tr>
</tbody>
</table>

Classes in **org.apache.log4j.performance** that implement **OptionHandler**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
</table>
A bogus appender which calls the format method of its layout object but does not write the result anywhere.

**Uses of OptionHandler in org.apache.log4j.spi**

<table>
<thead>
<tr>
<th>Subinterfaces of OptionHandler in org.apache.log4j.spi</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ErrorHandler</strong></td>
</tr>
<tr>
<td>Appenders may delegate their error handling to</td>
</tr>
<tr>
<td>ErrorHandlers.</td>
</tr>
</tbody>
</table>

**Classes in org.apache.log4j.spi that implement OptionHandler**

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Filter</strong></td>
</tr>
<tr>
<td>Users should extend this class to implement customized logging event filtering.</td>
</tr>
</tbody>
</table>

**Uses of OptionHandler in org.apache.log4j.varia**

**Classes in org.apache.log4j.varia that implement OptionHandler**

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DenyAllFilter</strong></td>
</tr>
<tr>
<td>This filter drops all logging events.</td>
</tr>
<tr>
<td><strong>ExternallyRolledFileAppender</strong></td>
</tr>
<tr>
<td>This appender listens on a socket on the port specified by the Port property for a &quot;RollOver&quot; message.</td>
</tr>
<tr>
<td><strong>FallbackErrorHandler</strong></td>
</tr>
<tr>
<td>The FallbackErrorHandler implements the ErrorHandler interface such that a secondary appender may be specified.</td>
</tr>
</tbody>
</table>
**LevelMatchFilter**
This is a very simple filter based on level matching.

**LevelRangeFilter**
This is a very simple filter based on level matching, which can be used to reject messages with priorities outside a certain range.

**StringMatchFilter**
This is a very simple filter based on string matching.

**Uses of** [OptionHandler](http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/xml/OptionHandler.html) **in**

**XMLLayout**
The output of the XMLLayout consists of a series of log4j:event elements as defined in the [log4j.dtd](http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/xml/log4j.dtd).

Copyright 2000-2005 Apache Software Foundation.
# Uses of Interface

**org.apache.log4j.spi.RendererSupport**

## Packages that use **RendererSupport**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.or</td>
<td>ObjectRenders are responsible for rendering messages depending on their class type.</td>
</tr>
</tbody>
</table>

## Uses of **RendererSupport** in **org.apache.log4j**

## Classes in **org.apache.log4j** that implement **RendererSupport**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>This class is specialized in retrieving loggers by name and also maintaining the logger hierarchy.</td>
</tr>
</tbody>
</table>

## Uses of **RendererSupport** in **org.apache.log4j.or**

## Methods in **org.apache.log4j.or** with parameters of type **RendererSupport**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void RendererMap.addRenderer(RendererSupport repository, String renderedClassName, String renderingClassName)</td>
<td>Add a renderer to a hierarchy passed as parameter.</td>
</tr>
</tbody>
</table>
Uses of Interface
org.apache.log4j.spi.RepositorySelector

Packages that use **RepositorySelector**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j</td>
<td>The main log4j package.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
</tbody>
</table>

Uses of **RepositorySelector** in org.apache.log4j

Methods in **org.apache.log4j** with parameters of type **RepositorySelector**

```java
static void LogManager.setRepositorySelector(RepositorySelector selector, Object guard)
Sets LoggerFactory but only if the correct guard is passed as parameter.
```

Uses of **RepositorySelector** in org.apache.log4j.spi

Classes in **org.apache.log4j.spi** that implement **RepositorySelector**

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultRepositorySelector</td>
</tr>
</tbody>
</table>
Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.spi.ThrowablesInformation

Packages that use ThrowableInformation

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.log4j.lf5</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
<tr>
<td>org.apache.log4j.spi</td>
<td></td>
</tr>
</tbody>
</table>

Uses of ThrowableInformation in org.apache.log4j.lf5

Methods in org.apache.log4j.lf5 with parameters of type ThrowableInformation

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void Log4JLogRecord.setThrownStackTrace(ThrowableInformation throwableInfo)</td>
<td>Set stack trace information associated with this Log4JLogRecord.</td>
</tr>
</tbody>
</table>

Uses of ThrowableInformation in org.apache.log4j.spi

Methods in org.apache.log4j.spi that return ThrowableInformation

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoggingEvent.getThrowableInformation()</td>
<td>Returns the throwable information contained within this event.</td>
</tr>
</tbody>
</table>
Copyright 2000-2005 Apache Software Foundation.
### Uses of Interface

**org.apache.log4j.spi.TriggeringEventEvaluator**

<table>
<thead>
<tr>
<th>Packages that use TriggeringEventEvaluator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.apache.log4j.net</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uses of TriggeringEventEvaluator in org.apache.log4j.net</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Fields in org.apache.log4j.net declared as TriggeringEventEvaluator</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected TriggeringEventEvaluator SMTPAppender.evaluator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructors in org.apache.log4j.net with parameters of type TriggeringEventEvaluator</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTPAppender(TriggeringEventEvaluator evaluator)</td>
</tr>
</tbody>
</table>

Use evaluator passed as parameter as the TriggeringEventEvaluator for this SMTPAppender.
Uses of Class
org.apache.log4j.spi.LocationInfo

Packages that use \texttt{LocationInfo}

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{org.apache.log4j.spi}</td>
<td>Contains part of the System Programming Interface (SPI) needed to extend log4j.</td>
</tr>
</tbody>
</table>

Uses of \texttt{LocationInfo} in \texttt{org.apache.log4j.spi}

Methods in \texttt{org.apache.log4j.spi} that return \texttt{LocationInfo}

<table>
<thead>
<tr>
<th>LocationInfo</th>
<th>LoggingEvent.\texttt{getLocationInformation}()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the location information for this logging event.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Interface
org.apache.log4j.spi.ErrorCode

No usage of org.apache.log4j.spi.ErrorCode

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.spi.DefaultRepositorySelector
Uses of Class org.apache.log4j.spi.RootCategory

No usage of org.apache.log4j.spi.RootCategory

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.spi.RootLogger

No usage of org.apache.log4j.spi.RootLogger

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.varia.NullAppender

Packages that use **NullAppender**

| org.apache.log4j.varia | Contains various appenders, filters and other odds and ends. |

Uses of **NullAppender** in
org.apache.log4j.varia

<table>
<thead>
<tr>
<th>Methods in org.apache.log4j.varia that return <strong>NullAppender</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>NullAppender, getInstance()</td>
</tr>
<tr>
<td>Whenever you can, use this method to retrieve an instance instead of instantiating a new one with new.</td>
</tr>
</tbody>
</table>

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.varia.DenyAllFilter

No usage of org.apache.log4j.varia.DenyAllFilter

Copyright 2000-2005 Apache Software Foundation.
Uses of Class org.apache.log4j.varia.ExternallyRolledFileAppender

No usage of org.apache.log4j.varia.ExternallyRolledFileAppender

Copyright 2000-2005 Apache Software Foundation.
Uses of Class org.apache.log4j.varia.FallbackErrorHandler

No usage of org.apache.log4j.varia.FallbackErrorHandler

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.varia.LevelMatchFilter

No usage of org.apache.log4j.varia.LevelMatchFilter

Copyright 2000-2005 Apache Software Foundation.
No usage of org.apache.log4j.varia.LevelRangeFilter
Uses of Class
org.apache.log4j.varia.ReloadPropertyConfigurator

No usage of org.apache.log4j.varia.ReloadPropertyConfigurator

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.varia.Roller

No usage of org.apache.log4j.varia.Roller

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.varia.StringMatchFilter

No usage of org.apache.log4j.varia.StringMatchFilter
Uses of Class
org.apache.log4j.xml.DOMConfigurator

No usage of org.apache.log4j.xml.DOMConfigurator

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.xml.Log4jEntityResolver

No usage of org.apache.log4j.xml.Log4jEntityResolver

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.xml.SAXErrorHandler

No usage of org.apache.log4j.xml.SAXErrorHandler

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.xml.XMLLayout

No usage of org.apache.log4j.xml.XMLLayout

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.xml.examples.ReportParserError

No usage of org.apache.log4j.xml.examples.ReportParserError

Copyright 2000-2005 Apache Software Foundation.
Uses of Class
org.apache.log4j.xml.examples.XMLSample

No usage of org.apache.log4j.xml.examples.XMLSample

Copyright 2000-2005 Apache Software Foundation.