A propos de ce document

Ce document a été construit automatiquement à partir de la documentation HTML disponible sur <u>java.sun.com</u>. Il est périodiquement remis à jour, lorsque de nouvelles versions des documentations originales deviennent disponibles. Pour télécharger librement et gratuitement ces mises à jour, mais aussi d'autres documents Java au format HTMLHelp ou WinHelp, visitez le site de Franck Allimant :

http://www.confluent.fr/javadoc (en Français)

http://www.confluent.fr/javadoc/indexe.html (en Anglais)

Bonne utilisation!

Contact

Les remarques et les suggestions sont les bienvenues. Merci de les faire parvenir à <u>Franck</u> Allimant.

Notes

La documentation HTML originale est disponible sur <u>java.sun.com</u>. Merci de la consulter si vous avez des incertitudes à propos les informations présentées dans ce document.

La version HTMLHelp de cette documentation diffère par quelques points de la version HTML, à cause de certaines limitations de HTMLHelp.

1. Applets Java

Les applets Java 1 fonctionnent correctement dans HTMLHelp, pourvu que les classes Java ne soient pas embarquées dans le fichier d'aide (chm) lui-même. Elles sont stockées dans le répertoire *applets*, qui se trouve au même endroit que le fichier d'aide. Ce répertoire n'est pas nécessaire au fonctionnement de l'aide : s'il est absent, les applets ne seront pas affichées.

Le plugin Java n'est pas supporté par HTMLHelp. Les applets nécessitant une machine virtuelle Java 2 ne peuvent pas être exécutées.

2. Fichiers nécessitant un plugin

Le support des plugins n'est pas complet dans HTMLHelp. En fonction de la configuration de votre machine, certains problèmes peuvent survenir lors de la visualisation de ces fichiers, en particulier les documents PDF.

3. Occurrences multiples d'un mot-clef de l'index

Si un mot clef de l'index pointe vers plusieurs documents, une page listant toutes les occurrences trouvées est affichée. Cette page fournit le titre des documents dans lequel le mot clef apparaît, et s'il s'agit d'une méthode, les paramètres de cette méthode. Ce mode d'affichage très détaillé a été préféré à la boîte de dialogue "Rubriques Trouvées" proposées en standard par HTMLHelp, pour permettre une navigation plus efficace et plus rapide dans un grand nombre de documents.

Merci à...

a.do Consulting, pour offrir gratuitement un parser HTML écrit en Java aussi compact et efficace.

Doug Kramer, de Sun, qui a autorisé la distribution des documentations Java pour WinHelp et HTMLHelp.

Distribution

Ce logiciel est gratuit. Il ne peut être diffusé sur Internet ou dupliqué que dans des buts non commerciaux, pour usage privé exclusivement. Toute distribution par CD-ROM est soumise à une autorisation préalable [contact].

Toutes les restrictions de distribution imposées par Sun à la documentation originale sont applicables au présent document. Consultez <u>cette page</u> (en anglais) pour en prendre connaissance.

Copyright © 1997-2002, Franck Allimant

About this document

This document is built from the HTML documentations available at <u>java.sun.com</u>. It is regularly updated, when new versions of original documentations become available. To download updates and many other WinHelp and HTMLHelp Java documentations for free, visit <u>Franck Allimant</u>'s web site:

http://www.confluent.fr/javadoc/indexe.html (in English)

http://www.confluent.fr/javadoc (in French)

Contact

Comments and suggestions are welcome. Feel free to send them to Franck Allimant.

Notes

The original HTML documentation is available at <u>java.sun.com</u>. Please refer to this original documentation if you're not confident with the present document.

Due to some HTMLHelp limitations, there are some differences between this help file and the original HTML version of the documentation.

1. Java Applets

In most cases, Java 1 applets are running inside the HTMLHelp viewer, provided they're not embedded in the help (chm) file. They are stored in the *applets* directory, located at the same level as the help file. This folder is not required for browsing help, but if it is missing, applets will not be displayed.

Java 2 applets require the Java Plugin, which is not supported by the HTMLHelp viewer. These applets will not be displayed.

2. Documents that require a plugin

Plugins are not fully supported by the HTMLHelp viewer. Depending on the configuration of your machine, some problems may occur when viewing these files, for example PDF documents.

3. Multiple occurrences of an index keyword

If an index keyword points to more than one document, a page that lists all found occurrences is displayed. This page shows the title of every document in which the keyword

appears, and if this keyword is a method, the parameters of this method. This detailed display has been preferred to the standard HTMLHelp "Topics Found" dialog box, to provide a faster and more efficient browsing of a large number of documents.

Thanks to...

<u>a.do Consulting</u>, for HtmlStreamTokenizer™, a compact, efficient and free HTML parser written in Java.

Doug Kramer at Sun, who authorized the distribution of this document.

Distribution

This software is free. Permission to use, copy, and distribute this software for non-commercial purposes via the Internet without fee is hereby granted. Permission is not granted for any distribution in the form of CD-ROM [contact].

Sun's documentation redistribution policy is applicable to this document. Please see $\underline{\text{this page}}$ for details.

Copyright © 1997-2002, Franck Allimant

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Serialized Form

Package javax.servlet

Class <u>javax.servlet.GenericServlet</u> implements Serializable

Class <u>javax.servlet.ServletContextAttributeEvent</u> implements Serializable

Serialized Fields

name

java.lang.String name

value

java.lang.Object value

Class <u>javax.servlet.ServletContextEvent</u> implements Serializable

Class <u>javax.servlet.ServletException</u> implements Serializable

Serialized Fields

rootCause

java.lang.Throwable rootCause

Class <u>javax.servlet.UnavailableException</u> implements Serializable

Serialized Fields

permanent

boolean permanent

seconds

int seconds

servlet

Servlet servlet

Package javax.servlet.http

Class <u>javax.servlet.http.HttpServlet</u> implements Serializable

Class javax.servlet.http.HttpSessionBindingEvent implements Serializable

Serialized Fields

name

java.lang.String name

value

java.lang.Object value

Class <u>javax.servlet.http.HttpSessionEvent</u> implements Serializable

Overview Package Class Tree Deprecated Index Help

REV NEXT FRAMES NO FRAME

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

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 <u>Overview</u> 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

- Inner Class Summary
- Field Summary
- Constructor Summary
- Method Summary
- Field Detail
- Constructor Detail
- Method Detail

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

Tree (Class Hierarchy)

There is a <u>Class Hierarchy</u> 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 <u>Deprecated API</u> 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 <u>Index</u> 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.

Overview Package Class Tree Deprecated Index Help

REV NEXT FRAMES NO FRAMES

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Hierarchy For All Packages

Package Hierarchies:

javax.servlet, javax.servlet.http

Class Hierarchy

- class java.lang.Object
 - class javax.servlet.http.<u>Cookie</u> (implements java.lang.Cloneable)
 - class java.util.EventObject (implements java.io.Serializable)
 - class javax.servlet.http.<u>HttpSessionEvent</u>
 - class javax.servlet.http.<u>HttpSessionBindingEvent</u>
 - class javax.servlet.<u>ServletContextEvent</u>
 - class javax.servlet.ServletContextAttributeEvent
 - class javax.servlet.<u>GenericServlet</u> (implements java.io.Serializable, javax.servlet.<u>Servlet</u>, javax.servlet.<u>ServletConfig</u>)
 - class javax.servlet.http.<u>HttpServlet</u> (implements java.io.Serializable)
 - class javax.servlet.http.<u>HttpUtils</u>
 - class java.io.InputStream
 - class javax.servlet.ServletInputStream
 - class java.io.OutputStream
 - class javax.servlet.<u>ServletOutputStream</u>
 - class javax.servlet.<u>ServletRequestWrapper</u> (implements javax.servlet.<u>ServletRequest</u>)
 - class javax.servlet.http.<u>HttpServletRequestWrapper</u> (implements javax.servlet.http.<u>HttpServletRequest</u>)
 - class javax.servlet.<u>ServletResponseWrapper</u> (implements javax.servlet.<u>ServletResponse</u>)
 - class javax.servlet.http.<u>HttpServletResponseWrapper</u> (implements javax.servlet.http.<u>HttpServletResponse</u>)
 - class java.lang.Throwable (implements java.io.Serializable)
 - class java.lang.Exception
 - class javax.servlet.<u>ServletException</u>
 - class javax.servlet. Unavailable Exception

Interface Hierarchy

- interface java.util.EventListener
 - interface javax.servlet.http.<u>HttpSessionActivationListener</u>
 - interface javax.servlet.http.<u>HttpSessionAttributeListener</u>
 - interface javax.servlet.http.<u>HttpSessionBindingListener</u>
 - interface javax.servlet.http.<u>HttpSessionListener</u>
 - interface javax.servlet.<u>ServletContextAttributeListener</u>
 - interface javax.servlet.<u>ServletContextListener</u>
- interface javax.servlet.Filter
- interface javax.servlet.<u>FilterChain</u>
- interface javax.servlet.FilterConfig
- interface javax.servlet.http.<u>HttpSession</u>
- interface javax.servlet.http.<u>HttpSessionContext</u>
- interface javax.servlet.<u>RequestDispatcher</u>
- interface javax.servlet.Servlet
- interface javax.servlet.<u>ServletConfig</u>
- interface javax.servlet.ServletContext
- interface javax.servlet.ServletRequest
 - interface javax.servlet.http.<u>HttpServletRequest</u>
- interface javax.servlet.ServletResponse
 - interface javax.servlet.http.<u>HttpServletResponse</u>
- interface javax.servlet.<u>SingleThreadModel</u>

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Overview Package Class Tree Deprecated Index Help

REV NEXT FRAMES NO FRAMES

ABCDEFGHIJLPRSUV

```
addCookie(Cookie) - Method in class
javax.servlet.http.HttpServletResponseWrapper
    The default behavior of this method is to call addCookie(Cookie cookie) on
     the wrapped response object.
addCookie(Cookie) - Method in interface
javax.servlet.http.<u>HttpServletResponse</u>
     Adds the specified cookie to the response.
addDateHeader(String, long) - Method in class
javax.servlet.http.<u>HttpServletResponseWrapper</u>
    The default behavior of this method is to call addDateHeader(String name,
    long date) on the wrapped response object.
addDateHeader(String, long) - Method in interface
javax.servlet.http.HttpServletResponse
     Adds a response header with the given name and date-value.
addHeader(String, String) - Method in class
javax.servlet.http.<u>HttpServletResponseWrapper</u>
     The default behavior of this method is to return addHeader(String name,
     String value) on the wrapped response object.
addHeader(String, String) - Method in interface
javax.servlet.http.<u>HttpServletResponse</u>
     Adds a response header with the given name and value.
addIntHeader(String, int) - Method in class
javax.servlet.http.HttpServletResponseWrapper
    The default behavior of this method is to call addIntHeader(String name, int
    value) on the wrapped response object.
addIntHeader(String, int) - Method in interface
javax.servlet.http.<u>HttpServletResponse</u>
     Adds a response header with the given name and integer value.
attributeAdded(HttpSessionBindingEvent) - Method in interface
javax.servlet.http.<u>HttpSessionAttributeListener</u>
     Notification that an attribute has been added to a session.
attributeAdded(ServletContextAttributeEvent) - Method in interface
```

Notification that a new attribute was added to the servlet context. $\underline{ \text{attributeRemoved(HttpSessionBindingEvent)}} \text{ - Method in interface}$

javax.servlet.ServletContextAttributeListener

javax.servlet.http.<u>HttpSessionAttributeListener</u>

Notification that an attribute has been removed from a session.

 $\underline{attributeRemoved(ServletContextAttributeEvent)} - Method in interface \\ javax.servlet.\underline{ServletContextAttributeListener}$

Notification that an existing attribute has been remved from the servlet context.

 $\frac{attributeReplaced(HttpSessionBindingEvent)}{javax.servlet.http.} - Method in interface \\ javax.servlet.http. \\ \frac{HttpSessionAttributeListener}{javax.servlet.http.} - Method in interface \\ \frac{1}{3} + \frac{1}$

Notification that an attribute has been replaced in a session. <u>attributeReplaced(ServletContextAttributeEvent)</u> - Method in interface javax.servlet.<u>ServletContextAttributeListener</u>

Notification that an attribute on the servlet context has been replaced.

BASIC_AUTH - Static variable in interface javax.servlet.http.<u>HttpServletRequest</u>
String identifier for Basic authentication.

CLIENT CERT AUTH - Static variable in interface

javax.servlet.http.<u>HttpServletRequest</u>

String identifier for Basic authentication.

clone() - Method in class javax.servlet.http.Cookie

Overrides the standard java.lang.Object.clone method to return a copy of this cookie.

containsHeader(String) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to call containsHeader(String name) on the wrapped response object.

containsHeader(String) - Method in interface

javax.servlet.http.<u>HttpServletResponse</u>

Returns a boolean indicating whether the named response header has already been set.

contextDestroyed(ServletContextEvent) - Method in interface javax.servlet.ServletContextListener

Notification that the servlet context is about to be shut down.

 $\underline{contextInitialized(ServletContextEvent)} - Method in interface$

javax.servlet.ServletContextListener

Notification that the web application is ready to process requests.

<u>Cookie</u> - class javax.servlet.http.<u>Cookie</u>.

Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server.

Cookie(String, String) - Constructor for class javax.servlet.http.<u>Cookie</u>
Constructs a cookie with a specified name and value.

destroy() - Method in interface javax.servlet.<u>Filter</u>

Called by the web container to indicate to a filter that it is being taken out of service.

destroy() - Method in interface javax.servlet.Servlet

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.

destroy() - Method in class javax.servlet.GenericServlet

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.

DIGEST AUTH - Static variable in interface

 $javax. servlet. http. \underline{HttpServletRequest}$

String identifier for Basic authentication.

doDelete(HttpServletRequest, HttpServletResponse) - Method in class javax.servlet.http.HttpServlet

Called by the server (via the service method) to allow a servlet to handle a DELETE request.

doFilter(ServletRequest, ServletResponse) - Method in interface javax.servlet.<u>FilterChain</u>

Causes the next filter in the chain to be invoked, or if the calling filter is the last filter in the chain, causes the resource at the end of the chain to be invoked.

doFilter(ServletRequest, ServletResponse, FilterChain) - Method in interface javax.servlet.<u>Filter</u>

The doFilter method of the Filter is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.

doGet(HttpServletRequest, HttpServletResponse) - Method in class javax.servlet.http.HttpServlet

Called by the server (via the service method) to allow a servlet to handle a GET request.

doHead(HttpServletRequest, HttpServletResponse) - Method in class javax.servlet.http.HttpServlet

Receives an HTTP HEAD request from the protected service method and handles the request.

doOptions(HttpServletRequest, HttpServletResponse) - Method in class

javax.servlet.http.<u>HttpServlet</u>

Called by the server (via the service method) to allow a servlet to handle a OPTIONS request.

doPost(HttpServletRequest, HttpServletResponse) - Method in class javax.servlet.http.HttpServlet

Called by the server (via the service method) to allow a servlet to handle a POST request.

doPut(HttpServletRequest, HttpServletResponse) - Method in class javax.servlet.http.<u>HttpServlet</u>

Called by the server (via the service method) to allow a servlet to handle a PUT request.

doTrace(HttpServletRequest, HttpServletResponse) - Method in class javax.servlet.http.HttpServlet

Called by the server (via the service method) to allow a servlet to handle a TRACE request.

encodeRedirectUrl(String) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to return encodeRedirectUrl(String url) on the wrapped response object.

encodeRedirectUrl(String) - Method in interface

javax.servlet.http.<u>HttpServletResponse</u>

Deprecated. As of version 2.1, use encodeRedirectURL(String url) instead **encodeRedirectURL(String)** - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to return encodeRedirectURL(String url) on the wrapped response object.

encodeRedirectURL(String) - Method in interface

javax.servlet.http.<u>HttpServletResponse</u>

Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged.

encodeUrl(String) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to call encodeUrl(String url) on the wrapped response object.

encodeUrl(String) - Method in interface javax.servlet.http.<u>HttpServletResponse</u>

Deprecated. As of version 2.1, use encodeURL(String url) instead

encodeURL(String) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to call encodeURL(String url) on the wrapped response object.

encodeURL(String) - Method in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged.

<u>Filter</u> - interface javax.servlet.<u>Filter</u>.

A filter is an object than perform filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both.

Filters perform filtering in the doFilter method.

<u>FilterChain</u> - interface javax.servlet.<u>FilterChain</u>.

A FilterChain is an object provided by the servlet container to the developer giving a view into the invocation chain of a filtered request for a resource.

<u>FilterConfig</u> - interface javax.servlet.<u>FilterConfig</u>.

A filter configuration object used by a servlet container used to pass information to a filter during initialization.

flushBuffer() - Method in interface javax.servlet.ServletResponse

Forces any content in the buffer to be written to the client.

flushBuffer() - Method in class javax.servlet.<u>ServletResponseWrapper</u>

The default behavior of this method is to call flushBuffer() on the wrapped response object.

FORM_AUTH - Static variable in interface

javax.servlet.http.HttpServletRequest

String identifier for Basic authentication.

forward(ServletRequest, ServletResponse) - Method in interface javax.servlet.<u>RequestDispatcher</u>

Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

- <u>GenericServlet</u> class javax.servlet.<u>GenericServlet</u>.
 - Defines a generic, protocol-independent servlet.
- **GenericServlet()** Constructor for class javax.servlet.<u>GenericServlet</u> Does nothing.
- **getAttribute(String)** Method in interface javax.servlet.<u>ServletContext</u>
 Returns the servlet container attribute with the given name, or null if there is no attribute by that name.
- **getAttribute(String)** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to call getAttribute(String name) on the wrapped request object.
- **getAttribute(String)** Method in interface javax.servlet. ServletRequest Returns the value of the named attribute as an Object, or null if no attribute of the given name exists.
- **getAttribute(String)** Method in interface javax.servlet.http.<u>HttpSession</u>
 Returns the object bound with the specified name in this session, or null if no object is bound under the name.
- getAttributeNames() Method in interface javax.servlet.ServletContext
 Returns an Enumeration containing the attribute names available within
 this servlet context.
- getAttributeNames() Method in class javax.servlet.ServletRequestWrapper
 The default behavior of this method is to return getAttributeNames() on the
 wrapped request object.
- getAttributeNames() Method in interface javax.servlet.ServletRequest
 Returns an Enumeration containing the names of the attributes available to
 this request.
- **getAttributeNames()** Method in interface javax.servlet.http.<u>HttpSession</u>
 Returns an Enumeration of String objects containing the names of all the objects bound to this session.
- **getAuthType()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns the name of the authentication scheme used to protect the servlet.
- **getAuthType()** Method in class javax.servlet.http.<u>HttpServletRequestWrapper</u>
 The default behavior of this method is to return getAuthType() on the wrapped request object.
- **getBufferSize()** Method in interface javax.servlet.<u>ServletResponse</u>
 Returns the actual buffer size used for the response.

- **getBufferSize()** Method in class javax.servlet.<u>ServletResponseWrapper</u>
 The default behavior of this method is to return getBufferSize() on the wrapped response object.
- **getCharacterEncoding()** Method in interface javax.servlet.<u>ServletResponse</u>
 Returns the name of the charset used for the MIME body sent in this response.
- getCharacterEncoding() Method in class
- javax.servlet.<u>ServletRequestWrapper</u>
 - The default behavior of this method is to return getCharacterEncoding() on the wrapped request object.
- getCharacterEncoding() Method in interface javax.servlet.ServletRequest
 Returns the name of the character encoding used in the body of this request.
 getCharacterEncoding() Method in class
- javax.servlet.<u>ServletResponseWrapper</u>
 - The default behavior of this method is to return getCharacterEncoding() on the wrapped response object.
- **getComment()** Method in class javax.servlet.http.<u>Cookie</u>
 Returns the comment describing the purpose of this cookie, or null if the cookie has no comment.
- **getContentLength()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getContentLength() on the wrapped request object.
- **getContentLength()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known.
- **getContentType()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getContentType() on the wrapped request object.
- **getContentType()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the MIME type of the body of the request, or null if the type is not known.
- getContext(String) Method in interface javax.servlet.ServletContext
 Returns a ServletContext object that corresponds to a specified URL on
 the server.
- **getContextPath()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns the portion of the request URI that indicates the context of the request.
- getContextPath() Method in class
 javax.servlet.http.HttpServletRequestWrapper

The default behavior of this method is to return getContextPath() on the wrapped request object.

getCookies() - Method in interface javax.servlet.http.<u>HttpServletRequest</u>
Returns an array containing all of the Cookie objects the client sent with this request.

getCookies() - Method in class javax.servlet.http.<u>HttpServletRequestWrapper</u>
The default behavior of this method is to return getCookies() on the wrapped request object.

getCreationTime() - Method in interface javax.servlet.http.<u>HttpSession</u>
Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.

getDateHeader(String) - Method in interface

javax.servlet.http.<u>HttpServletRequest</u>

Returns the value of the specified request header as a long value that represents a Date object.

getDateHeader(String) - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return getDateHeader(String name) on the wrapped request object.

getDomain() - Method in class javax.servlet.http.Cookie

Returns the domain name set for this cookie.

getFilterName() - Method in interface javax.servlet.<u>FilterConfig</u>

Returns the filter-name of this filter as defined in the deployment descriptor.

getHeader(String) - Method in interface javax.servlet.http.<u>HttpServletRequest</u>
Returns the value of the specified request header as a String.

getHeader(String) - Method in class

 $javax. servlet. http. \underline{HttpServletRequestWrapper}$

The default behavior of this method is to return getHeader(String name) on the wrapped request object.

getHeaderNames() - Method in interface javax.servlet.http.<u>HttpServletRequest</u>
Returns an enumeration of all the header names this request contains.

getHeaderNames() - Method in class

 $javax. servlet. http. \underline{HttpServletRequestWrapper}$

The default behavior of this method is to return getHeaderNames() on the wrapped request object.

getHeaders(String) - Method in interface javax.servlet.http.<u>HttpServletRequest</u>
Returns all the values of the specified request header as an Enumeration of String objects.

getHeaders(String) - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

of this API.

- The default behavior of this method is to return getHeaders(String name) on the wrapped request object.
- **getId()** Method in interface javax.servlet.http.<u>HttpSession</u>
 Returns a string containing the unique identifier assigned to this session.
- **getIds()** Method in interface javax.servlet.http.<u>HttpSessionContext</u> **Deprecated.** As of Java Servlet API 2.1 with no replacement. This method must return an empty Enumeration and will be removed in a future version
- **getInitParameter(String)** Method in interface javax.servlet.<u>FilterConfig</u>
 Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.
- **getInitParameter(String)** Method in interface javax.servlet.<u>ServletConfig</u>
 Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.
- **getInitParameter(String)** Method in interface javax.servlet.<u>ServletContext</u>
 Returns a String containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.
- **getInitParameter(String)** Method in class javax.servlet.<u>GenericServlet</u>
 Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.
- getInitParameterNames() Method in interface javax.servlet.FilterConfig
 Returns the names of the servlet's initialization parameters as an
 Enumeration of String objects, or an empty Enumeration if the servlet has
 no initialization parameters.
- getInitParameterNames() Method in interface javax.servlet.ServletConfig
 Returns the names of the servlet's initialization parameters as an
 Enumeration of String objects, or an empty Enumeration if the servlet has
 no initialization parameters.
- getInitParameterNames() Method in interface javax.servlet.ServletContext
 Returns the names of the context's initialization parameters as an
 Enumeration of String objects, or an empty Enumeration if the context has
 no initialization parameters.
- getInitParameterNames() Method in class javax.servlet.GenericServlet
 Returns the names of the servlet's initialization parameters as an
 Enumeration of String objects, or an empty Enumeration if the servlet has
 no initialization parameters.
- **getInputStream()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getInputStream() on the

wrapped request object.

getInputStream() - Method in interface javax.servlet.<u>ServletRequest</u>
Retrieves the body of the request as binary data using a
<u>ServletInputStream</u>.

getIntHeader(String) - Method in interface

javax.servlet.http.<u>HttpServletRequest</u>

Returns the value of the specified request header as an int.

getIntHeader(String) - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return getIntHeader(String name) on the wrapped request object.

getLastAccessedTime() - Method in interface javax.servlet.http.<u>HttpSession</u>
Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container recieved the request.

getLastModified(HttpServletRequest) - Method in class javax.servlet.http.<u>HttpServlet</u>

Returns the time the HttpServletRequest object was last modified, in milliseconds since midnight January 1, 1970 GMT.

- **getLocale()** Method in interface javax.servlet.<u>ServletResponse</u>
 Returns the locale assigned to the response.
- getLocale() Method in class javax.servlet.ServletRequestWrapper
 The default behavior of this method is to return getLocale() on the wrapped
 request object.
- **getLocale()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the preferred Locale that the client will accept content in, based on the Accept-Language header.
- **getLocale()** Method in class javax.servlet.<u>ServletResponseWrapper</u>

 The default behavior of this method is to return getLocale() on the wrapped response object.
- **getLocales()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getLocales() on the wrapped request object.
- **getLocales()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns an Enumeration of Locale objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header.
- **getMajorVersion()** Method in interface javax.servlet.<u>ServletContext</u>

 Returns the major version of the Java Servlet API that this servlet container

- supports.
- getMaxAge() Method in class javax.servlet.http.Cookie
 - Returns the maximum age of the cookie, specified in seconds, By default, -1 indicating the cookie will persist until browser shutdown.
- **getMaxInactiveInterval()** Method in interface javax.servlet.http.<u>HttpSession</u>
 Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
- **getMethod()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.
- **getMethod()** Method in class javax.servlet.http.<u>HttpServletRequestWrapper</u>
 The default behavior of this method is to return getMethod() on the wrapped request object.
- **getMimeType(String)** Method in interface javax.servlet.<u>ServletContext</u>
 Returns the MIME type of the specified file, or null if the MIME type is not known.
- **getMinorVersion()** Method in interface javax.servlet.<u>ServletContext</u>
 Returns the minor version of the Servlet API that this servlet container supports.
- **getName()** Method in class javax.servlet.<u>ServletContextAttributeEvent</u>
 Return the name of the attribute that changed on the ServletContext.
- **getName()** Method in class javax.servlet.http.<u>HttpSessionBindingEvent</u>
 Returns the name with which the attribute is bound to or unbound from the session.
- **getName()** Method in class javax.servlet.http.<u>Cookie</u>
 Returns the name of the cookie.
- **getNamedDispatcher(String)** Method in interface javax.servlet.ServletContext
 - Returns a <u>RequestDispatcher</u> object that acts as a wrapper for the named servlet.
- **getOutputStream()** Method in interface javax.servlet.<u>ServletResponse</u>
 Returns a <u>ServletOutputStream</u> suitable for writing binary data in the response.
- **getOutputStream()** Method in class javax.servlet.<u>ServletResponseWrapper</u>
 The default behavior of this method is to return getOutputStream() on the wrapped response object.
- **getParameter(String)** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getParameter(String name) on the wrapped request object.

- **getParameter(String)** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the value of a request parameter as a String, or null if the parameter does not exist.
- **getParameterMap()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getParameterMap() on the wrapped request object.
- **getParameterMap()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns a java.util.Map of the parameters of this request.
- **getParameterNames()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getParameterNames() on the wrapped request object.
- **getParameterNames()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns an Enumeration of String objects containing the names of the parameters contained in this request.
- **getParameterValues(String)** Method in class
- javax.servlet.<u>ServletRequestWrapper</u>
 - The default behavior of this method is to return getParameterValues(String name) on the wrapped request object.
- **getParameterValues(String)** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist.
- getPath() Method in class javax.servlet.http.Cookie
 - Returns the path on the server to which the browser returns this cookie.
- **getPathInfo()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns any extra path information associated with the URL the client sent when it made this request.
- **getPathInfo()** Method in class javax.servlet.http.<u>HttpServletRequestWrapper</u>
 The default behavior of this method is to return getPathInfo() on the wrapped request object.
- **getPathTranslated()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns any extra path information after the servlet name but before the query string, and translates it to a real path.
- getPathTranslated() Method in class
- javax.servlet.http.<u>HttpServletRequestWrapper</u>
 - The default behavior of this method is to return getPathTranslated() on the wrapped request object.
- **getProtocol()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getProtocol() on the wrapped request object.

- **getProtocol()** Method in interface javax.servlet.ServletRequest
 - Returns the name and version of the protocol the request uses in the form *protocol/majorVersion.minorVersion*, for example, HTTP/1.1.
- **getQueryString()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns the query string that is contained in the request URL after the path.
- getQueryString() Method in class
- $javax.servlet.http. \underline{HttpServletRequestWrapper}$
 - The default behavior of this method is to return getQueryString() on the wrapped request object.
- getReader() Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getReader() on the wrapped
 request object.
- **getReader()** Method in interface javax.servlet.<u>ServletRequest</u>

 Retrieves the body of the request as character data using a BufferedReader.
- **getRealPath(String)** Method in interface javax.servlet.<u>ServletContext</u>
 Returns a String containing the real path for a given virtual path.
- **getRealPath(String)** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getRealPath(String path) on the wrapped request object.
- **getRealPath(String)** Method in interface javax.servlet.<u>ServletRequest</u> **Deprecated.** As of Version 2.1 of the Java Servlet API, use
 <u>ServletContext.getRealPath(java.lang.String)</u> instead.
- **getRemoteAddr()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getRemoteAddr() on the wrapped request object.
- **getRemoteAddr()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the Internet Protocol (IP) address of the client that sent the request.
- **getRemoteHost()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getRemoteHost() on the wrapped request object.
- **getRemoteHost()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the fully qualified name of the client that sent the request.
- **getRemoteUser()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated.
- getRemoteUser() Method in class
- javax.servlet.http.<u>HttpServletRequestWrapper</u>
 - The default behavior of this method is to return getRemoteUser() on the wrapped request object.

getRequest() - Method in class javax.servlet.ServletRequestWrapper

Return the wrapped request object.

getRequestDispatcher(String) - Method in interface

javax.servlet.ServletContext

Returns a <u>RequestDispatcher</u> object that acts as a wrapper for the resource located at the given path.

getRequestDispatcher(String) - Method in class

javax.servlet.<u>ServletRequestWrapper</u>

The default behavior of this method is to return

getRequestDispatcher(String path) on the wrapped request object.

getRequestDispatcher(String) - Method in interface

javax.servlet.ServletRequest

Returns a <u>RequestDispatcher</u> object that acts as a wrapper for the resource located at the given path.

getRequestedSessionId() - Method in interface

javax.servlet.http.<u>HttpServletRequest</u>

Returns the session ID specified by the client.

getRequestedSessionId() - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return getRequestedSessionId() on the wrapped request object.

getRequestURI() - Method in interface javax.servlet.http.<u>HttpServletRequest</u>
Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.

getRequestURI() - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return getRequestURI() on the wrapped request object.

getRequestURL() - Method in interface javax.servlet.http.<u>HttpServletRequest</u>
Reconstructs the URL the client used to make the request.

getRequestURL() - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return getRequestURL() on the wrapped request object.

getRequestURL(HttpServletRequest) - Static method in class javax.servlet.http.HttpUtils

Deprecated. Reconstructs the URL the client used to make the request, using information in the HttpServletRequest object.

getResource(String) - Method in interface javax.servlet.ServletContext

- Returns a URL to the resource that is mapped to a specified path.
- **getResourceAsStream(String)** Method in interface
- javax.servlet.<u>ServletContext</u>
- Returns the resource located at the named path as an InputStream object.
- **getResourcePaths(String)** Method in interface javax.servlet.<u>ServletContext</u>
 Returns a directory-like listing of all the paths to resources within the web application whose longest sub-path matches the supplied path argument.
- **getResponse()** Method in class javax.servlet.<u>ServletResponseWrapper</u>
 Return the wrapped ServletResponse object.
- **getRootCause()** Method in class javax.servlet.<u>ServletException</u>
 Returns the exception that caused this servlet exception.
- **getScheme()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getScheme() on the wrapped request object.
- getScheme() Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the name of the scheme used to make this request, for example,
 http, https, or ftp.
- **getSecure()** Method in class javax.servlet.http.<u>Cookie</u>

 Returns true if the browser is sending cookies only over a secure protocol, or false if the browser can send cookies using any protocol.
- getServerInfo() Method in interface javax.servlet.ServletContext
 Returns the name and version of the servlet container on which the servlet
 is running.
- **getServerName()** Method in class javax.servlet.<u>ServletRequestWrapper</u>
 The default behavior of this method is to return getServerName() on the wrapped request object.
- **getServerName()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the host name of the server that received the request.
- getServerPort() Method in class javax.servlet.ServletRequestWrapper
 The default behavior of this method is to return getServerPort() on the
 wrapped request object.
- **getServerPort()** Method in interface javax.servlet.<u>ServletRequest</u>
 Returns the port number on which this request was received.
- **getServlet()** Method in class javax.servlet.<u>UnavailableException</u> **Deprecated.** *As of Java Servlet API 2.2, with no replacement. Returns the servlet that is reporting its unavailability.*
- **getServlet(String)** Method in interface javax.servlet.<u>ServletContext</u> **Deprecated.** *As of Java Servlet API 2.1*, *with no direct replacement.*

This method was originally defined to retrieve a servlet from a ServletContext. In this version, this method always returns null and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

In lieu of this method, servlets can share information using the ServletContext class and can perform shared business logic by invoking methods on common non-servlet classes.

- **getServletConfig()** Method in interface javax.servlet.<u>Servlet</u>
 - Returns a <u>ServletConfig</u> object, which contains initialization and startup parameters for this servlet.
- **getServletConfig()** Method in class javax.servlet.<u>GenericServlet</u>
 Returns this servlet's <u>ServletConfig</u> object.
- getServletContext() Method in interface javax.servlet.FilterConfig

Returns a reference to the <u>ServletContext</u> in which the caller is executing.

- getServletContext() Method in interface javax.servlet.ServletConfig
 - Returns a reference to the <u>ServletContext</u> in which the caller is executing.
- **getServletContext()** Method in class javax.servlet.<u>ServletContextEvent</u>
 Return the ServletContext that changed.
- getServletContext() Method in class javax.servlet.GenericServlet
 Returns a reference to the ServletContext in which this servlet is running.
- **getServletContext()** Method in interface javax.servlet.http.<u>HttpSession</u>
 Returns the ServletContext to which this session belongs.
- **getServletContextName()** Method in interface javax.servlet.<u>ServletContext</u>
 Returns the name of this web application correponding to this
 ServletContext as specified in the deployment descriptor for this web application by the display-name element.
- **getServletInfo()** Method in interface javax.servlet.<u>Servlet</u>
 Returns information about the servlet, such as author, version, and copyright.
- **getServletInfo()** Method in class javax.servlet.<u>GenericServlet</u>
 Returns information about the servlet, such as author, version, and copyright.
- **getServletName()** Method in interface javax.servlet.<u>ServletConfig</u>
 Returns the name of this servlet instance.
- **getServletName()** Method in class javax.servlet.<u>GenericServlet</u> Returns the name of this servlet instance.
- getServletNames() Method in interface javax.servlet.ServletContext

Deprecated. As of Java Servlet API 2.1, with no replacement.

This method was originally defined to return an Enumeration of all the servlet names known to this context. In this version, this method always returns an empty Enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

getServletPath() - Method in interface javax.servlet.http.<u>HttpServletRequest</u>
Returns the part of this request's URL that calls the servlet.

getServletPath() - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return getServletPath() on the wrapped request object.

getServlets() - Method in interface javax.servlet.<u>ServletContext</u> **Deprecated.** *As of Java Servlet API 2.0, with no replacement.*

This method was originally defined to return an Enumeration of all the servlets known to this servlet context. In this version, this method always returns an empty enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

- **getSession()** Method in class javax.servlet.http.<u>HttpSessionEvent</u> Return the session that changed.
- **getSession()** Method in class javax.servlet.http.<u>HttpSessionBindingEvent</u> Return the session that changed.
- **getSession()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns the current session associated with this request, or if the request does not have a session, creates one.
- **getSession()** Method in class javax.servlet.http.<u>HttpServletRequestWrapper</u>
 The default behavior of this method is to return getSession() on the wrapped request object.
- **getSession(boolean)** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns the current HttpSession associated with this request or, if if there is no current session and create is true, returns a new session.

getSession(boolean) - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return getSession(boolean create)

- on the wrapped request object.
- **getSession(String)** Method in interface javax.servlet.http.<u>HttpSessionContext</u> **Deprecated.** As of Java Servlet API 2.1 with no replacement. This method must return null and will be removed in a future version of this API.
- **getSessionContext()** Method in interface javax.servlet.http.<u>HttpSession</u> **Deprecated.** *As of Version 2.1, this method is deprecated and has no replacement. It will be removed in a future version of the Java Servlet API.*
- **getUnavailableSeconds()** Method in class javax.servlet.<u>UnavailableException</u>
 Returns the number of seconds the servlet expects to be temporarily unavailable.
- **getUserPrincipal()** Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 Returns a java.security.Principal object containing the name of the current authenticated user.
- **getUserPrincipal()** Method in class
- javax.servlet.http.<u>HttpServletRequestWrapper</u>
 - The default behavior of this method is to return getUserPrincipal() on the wrapped request object.
- **getValue()** Method in class javax.servlet.<u>ServletContextAttributeEvent</u>
 Returns the value of the attribute that has been added removed or replaced.
- **getValue()** Method in class javax.servlet.http.<u>HttpSessionBindingEvent</u>
 Returns the value of the attribute that has been added, removed or replaced.
- **getValue()** Method in class javax.servlet.http.<u>Cookie</u>
 Returns the value of the cookie.
- **getValue(String)** Method in interface javax.servlet.http.<u>HttpSession</u> **Deprecated.** *As of Version 2.2, this method is replaced by HttpSession.getAttribute(java.lang.String)*.
- **getValueNames()** Method in interface javax.servlet.http.<u>HttpSession</u> **Deprecated.** As of Version 2.2, this method is replaced by

 <u>HttpSession.getAttributeNames()</u>
- **getVersion()** Method in class javax.servlet.http.<u>Cookie</u>
 Returns the version of the protocol this cookie complies with.
- **getWriter()** Method in interface javax.servlet.<u>ServletResponse</u>

 Returns a PrintWriter object that can send character text to the client.
- **getWriter()** Method in class javax.servlet.<u>ServletResponseWrapper</u>
 The default behavior of this method is to return getWriter() on the wrapped response object.

<u>HttpServlet</u> - class javax.servlet.http.<u>HttpServlet</u>.

Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site.

HttpServlet() - Constructor for class javax.servlet.http.<u>HttpServlet</u>
Does nothing, because this is an abstract class.

<u>HttpServletRequest</u> - interface javax.servlet.http.<u>HttpServletRequest</u>.

Extends the <u>ServletRequest</u> interface to provide request information for HTTP servlets.

HttpServletRequestWrapper - class

javax.servlet.http.HttpServletRequestWrapper.

Provides a convenient implementation of the HttpServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet.

<u>HttpServletRequestWrapper(HttpServletRequest)</u> - Constructor for class javax.servlet.http.<u>HttpServletRequestWrapper</u>

Constructs a request object wrapping the given request.

HttpServletResponse - interface javax.servlet.http.HttpServletResponse.

Extends the <u>ServletResponse</u> interface to provide HTTP-specific functionality in sending a response.

HttpServletResponseWrapper - class

 $javax. servlet. http. \underline{HttpServletResponseWrapper}.$

Provides a convenient implementation of the HttpServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet.

 $\underline{\textbf{HttpServletResponseWrapper(HttpServletResponse)}} \text{-} Constructor for class } javax.servlet.http. \underline{\textbf{HttpServletResponseWrapper}}$

Constructs a response adaptor wrapping the given response.

<u>HttpSession</u> - interface javax.servlet.http.<u>HttpSession</u>.

Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

HttpSessionActivationListener - interface

javax.servlet.http.<u>HttpSessionActivationListener</u>.

Objects that are bound to a session may listen to container events notifying them that sessions will be passivated and that session will be activated.

<u>HttpSessionAttributeListener</u> - interface

javax.servlet.http.<u>HttpSessionAttributeListener</u>.

This listener interface can be implemented in order to get notifications of changes to the attribute lists of sessions within this web application.

HttpSessionBindingEvent - class javax.servlet.http.HttpSessionBindingEvent.
Events of this type are either sent to an object that implements
HttpSessionBindingListener when it is bound or unbound from a session,
or to a HttpSessionAttributeListener that has been configured in the deployment descriptor when any attribute is bound, unbound or replaced in a session.

<u>HttpSessionBindingEvent(HttpSession, String)</u> - Constructor for class javax.servlet.http.<u>HttpSessionBindingEvent</u>

Constructs an event that notifies an object that it has been bound to or unbound from a session.

<u>HttpSessionBindingEvent(HttpSession, String, Object)</u> - Constructor for class javax.servlet.http.<u>HttpSessionBindingEvent</u>

Constructs an event that notifies an object that it has been bound to or unbound from a session.

HttpSessionBindingListener - interface

javax.servlet.http.<u>HttpSessionBindingListener</u>.

Causes an object to be notified when it is bound to or unbound from a session.

HttpSessionContext - interface javax.servlet.http.HttpSessionContext.

Deprecated. As of Java(tm) Servlet API 2.1 for security reasons, with no replacement. This interface will be removed in a future version of this API.

HttpSessionEvent - class javax.servlet.http.<u>HttpSessionEvent</u>.

This is the class representing event notifications for changes to sessions within a web application.

 $\underline{\textbf{HttpSessionEvent(HttpSession)}} \text{ - Constructor for class}$

javax.servlet.http.<u>HttpSessionEvent</u>

Construct a session event from the given source.

<u>HttpSessionListener</u> - interface javax.servlet.http.<u>HttpSessionListener</u>.

Implementations of this interface may are notified of changes to the list of active sessions in a web application.

<u>**HttpUtils**</u> - class javax.servlet.http.<u>HttpUtils</u>.

Deprecated. As of Java(tm) Servlet API 2.3. These methods were only useful with the default encoding and have been moved to the request interfaces.

HttpUtils() - Constructor for class javax.servlet.http.HttpUtils
 Deprecated. Constructs an empty HttpUtils object.

- include(ServletRequest, ServletResponse) Method in interface javax.servlet.RequestDispatcher
 - Includes the content of a resource (servlet, JSP page, HTML file) in the response.
- init() Method in class javax.servlet.GenericServlet
 - A convenience method which can be overridden so that there's no need to call super.init(config).
- <u>init(FilterConfig)</u> Method in interface javax.servlet.<u>Filter</u>
 Called by the web container to indicate to a filter that it is being placed into service.
- init(ServletConfig) Method in interface javax.servlet.Servlet
 Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
- <u>init(ServletConfig)</u> Method in class javax.servlet.<u>GenericServlet</u>
 Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
- invalidate() Method in interface javax.servlet.http.HttpSession
 Invalidates this session then unbinds any objects bound to it.
- **isCommitted()** Method in interface javax.servlet.<u>ServletResponse</u>
 Returns a boolean indicating if the response has been committed.
- **isCommitted()** Method in class javax.servlet.<u>ServletResponseWrapper</u>
 The default behavior of this method is to return isCommitted() on the wrapped response object.
- isNew() Method in interface javax.servlet.http.<u>HttpSession</u>
 Returns true if the client does not yet know about the session or if the client chooses not to join the session.
- **isPermanent()** Method in class javax.servlet.<u>UnavailableException</u>
 Returns a boolean indicating whether the servlet is permanently unavailable.
- isRequestedSessionIdFromCookie() Method in interface javax.servlet.http.<u>HttpServletRequest</u>
 - Checks whether the requested session ID came in as a cookie.
- isRequestedSessionIdFromCookie() Method in class
- javax.servlet.http.<u>HttpServletRequestWrapper</u>
 - The default behavior of this method is to return

isRequestedSessionIdFromCookie() on the wrapped request object.

isRequestedSessionIdFromUrl() - Method in interface

javax.servlet.http.<u>HttpServletRequest</u>

Deprecated. As of Version 2.1 of the Java Servlet API, use

HttpServletRequest.isRequestedSessionIdFromURL() instead.

isRequestedSessionIdFromUrl() - Method in class

javax.servlet.http.<u>HttpServletRequestWrapper</u>

The default behavior of this method is to return

isRequestedSessionIdFromUrl() on the wrapped request object.

isRequestedSessionIdFromURL() - Method in interface

javax.servlet.http.<u>HttpServletRequest</u>

Checks whether the requested session ID came in as part of the request URL.

isRequestedSessionIdFromURL() - Method in class

 $javax. servlet. http. \underline{HttpServletRequestWrapper}$

The default behavior of this method is to return

isRequestedSessionIdFromURL() on the wrapped request object.

isRequestedSessionIdValid() - Method in interface

javax.servlet.http.<u>HttpServletRequest</u>

Checks whether the requested session ID is still valid.

isRequestedSessionIdValid() - Method in class

 $javax. servlet. http. \underline{HttpServletRequestWrapper}$

The default behavior of this method is to return

isRequestedSessionIdValid() on the wrapped request object.

<u>isSecure()</u> - Method in class javax.servlet.<u>ServletRequestWrapper</u>

The default behavior of this method is to return is Secure() on the wrapped request object.

isSecure() - Method in interface javax.servlet.<u>ServletRequest</u>

Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.

isUserInRole(String) - Method in interface

javax.servlet.http.<u>HttpServletRequest</u>

Returns a boolean indicating whether the authenticated user is included in the specified logical "role".

isUserInRole(String) - Method in class

 $javax. servlet. http. \underline{HttpServletRequestWrapper}$

The default behavior of this method is to return is UserInRole(String role) on the wrapped request object.

J

<u>javax.servlet</u> - package javax.servlet

<u>javax.servlet.http</u> - package javax.servlet.http

\mathbf{L}

log(Exception, String) - Method in interface javax.servlet.<u>ServletContext</u> **Deprecated.** As of Java Servlet API 2.1, use <u>ServletContext.log(String message, Throwable throwable)</u> instead.

This method was originally defined to write an exception's stack trace and an explanatory error message to the servlet log file.

- log(String) Method in interface javax.servlet.ServletContext
 Writes the specified message to a servlet log file, usually an event log.
- log(String) Method in class javax.servlet.GenericServlet
 Writes the specified message to a servlet log file, prepended by the servlet's name.
- **log(String, Throwable)** Method in interface javax.servlet. ServletContext Writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file.
- **log(String, Throwable)** Method in class javax.servlet.<u>GenericServlet</u>
 Writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file, prepended by the servlet's name.

- parsePostData(int, ServletInputStream) Static method in class javax.servlet.http.HttpUtils
 - **Deprecated.** Parses data from an HTML form that the client sends to the server using the HTTP POST method and the *application/x-www-form-urlencoded* MIME type.
- **parseQueryString(String)** Static method in class javax.servlet.http.<u>HttpUtils</u> **Deprecated.** Parses a query string passed from the client to the server and builds a HashTable object with key-value pairs.
- print(boolean) Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes a boolean value to the client, with no carriage return-line feed
 (CRLF) character at the end.
- **print(char)** Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes a character to the client, with no carriage return-line feed (CRLF) at the end.
- print(double) Method in class javax.servlet.ServletOutputStream
 Writes a double value to the client, with no carriage return-line feed
 (CRLF) at the end.
- print(float) Method in class javax.servlet.ServletOutputStream
 Writes a float value to the client, with no carriage return-line feed (CRLF)
 at the end.
- **print(int)** Method in class javax.servlet.
 ServletOutputStream
 Writes an int to the client, with no carriage return-line feed (CRLF) at the end.
- print(long) Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes a long value to the client, with no carriage return-line feed (CRLF)
 at the end.
- print(String) Method in class javax.servlet.ServletOutputStream
 Writes a String to the client, without a carriage return-line feed (CRLF)
 character at the end.
- **println()** Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes a carriage return-line feed (CRLF) to the client.
- println(boolean) Method in class javax.servlet.ServletOutputStream
 Writes a boolean value to the client, followed by a carriage return-line feed
 (CRLF).
- println(char) Method in class javax.servlet.ServletOutputStream

- Writes a character to the client, followed by a carriage return-line feed (CRLF).
- println(double) Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes a double value to the client, followed by a carriage return-line feed
 (CRLF).
- println(float) Method in class javax.servlet.ServletOutputStream
 Writes a float value to the client, followed by a carriage return-line feed
 (CRLF).
- **println(int)** Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes an int to the client, followed by a carriage return-line feed (CRLF) character.
- **println(long)** Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes a long value to the client, followed by a carriage return-line feed (CRLF).
- println(String) Method in class javax.servlet.<u>ServletOutputStream</u>
 Writes a String to the client, followed by a carriage return-line feed
 (CRLF).
- putValue(String, Object) Method in interface javax.servlet.http.HttpSession
 Deprecated. As of Version 2.2, this method is replaced by
 HttpSession.setAttribute(java.lang.String, java.lang.Object)

- **readLine(byte[], int, int)** Method in class javax.servlet.<u>ServletInputStream</u>
 Reads the input stream, one line at a time.
- **removeAttribute(String)** Method in interface javax.servlet.<u>ServletContext</u>
 Removes the attribute with the given name from the servlet context.
- removeAttribute(String) Method in class
- $javax. servlet. \underline{ServletRequestWrapper}$
 - The default behavior of this method is to call removeAttribute(String name) on the wrapped request object.
- **removeAttribute(String)** Method in interface javax.servlet.<u>ServletRequest</u>
 Removes an attribute from this request.
- **removeAttribute(String)** Method in interface javax.servlet.http.<u>HttpSession</u> Removes the object bound with the specified name from this session.
- **removeValue(String)** Method in interface javax.servlet.http.<u>HttpSession</u> **Deprecated.** As of Version 2.2, this method is replaced by

 <u>HttpSession.removeAttribute(java.lang.String)</u>
- $\underline{RequestDispatcher} \text{ interface javax.servlet.} \underline{RequestDispatcher}.$
 - Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server.
- reset() Method in interface javax.servlet.<u>ServletResponse</u>Clears any data that exists in the buffer as well as the status code and headers.
- reset() Method in class javax.servlet.<u>ServletResponseWrapper</u>
 The default behavior of this method is to call reset() on the wrapped response object.
- resetBuffer() Method in interface javax.servlet.ServletResponse
 Clears the content of the underlying buffer in the response without clearing
 headers or status code.
- resetBuffer() Method in class javax.servlet.<u>ServletResponseWrapper</u>
 The default behavior of this method is to call resetBuffer() on the wrapped response object.

SC ACCEPTED - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (202) indicating that a request was accepted for processing, but was not completed.

SC BAD GATEWAY - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (502) indicating that the HTTP server received an invalid response from a server it consulted when acting as a proxy or gateway.

SC_BAD_REQUEST - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (400) indicating the request sent by the client was syntactically incorrect.

SC CONFLICT - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (409) indicating that the request could not be completed due to a conflict with the current state of the resource.

SC CONTINUE - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (100) indicating the client can continue.

SC_CREATED - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (201) indicating the request succeeded and created a new resource on the server.

SC_EXPECTATION_FAILED - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (417) indicating that the server could not meet the expectation given in the Expect request header.

SC_FORBIDDEN - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (403) indicating the server understood the request but refused to fulfill it.

SC GATEWAY TIMEOUT - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (504) indicating that the server did not receive a timely response from the upstream server while acting as a gateway or proxy.

SC_GONE - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u> Status code (410) indicating that the resource is no longer available at the server and no forwarding address is known.

SC_HTTP_VERSION_NOT_SUPPORTED - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (505) indicating that the server does not support or refuses to support the HTTP protocol version that was used in the request message.

SC_INTERNAL_SERVER_ERROR - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (500) indicating an error inside the HTTP server which prevented it from fulfilling the request.

SC LENGTH REQUIRED - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (411) indicating that the request cannot be handled without a defined *Content-Length*.

SC_METHOD_NOT_ALLOWED - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (405) indicating that the method specified in the *Request-Line* is not allowed for the resource identified by the *Request-URI*.

SC_MOVED_PERMANENTLY - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (301) indicating that the resource has permanently moved to a new location, and that future references should use a new URI with their requests.

SC MOVED TEMPORARILY - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (302) indicating that the resource has temporarily moved to another location, but that future references should still use the original URI to access the resource.

SC MULTIPLE CHOICES - Static variable in interface javax.servlet.http.HttpServletResponse

Status code (300) indicating that the requested resource corresponds to any one of a set of representations, each with its own specific location.

SC_NO_CONTENT - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (204) indicating that the request succeeded but that there was no new information to return.

SC_NON_AUTHORITATIVE_INFORMATION - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (203) indicating that the meta information presented by the client did not originate from the server.

SC_NOT_ACCEPTABLE - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (406) indicating that the resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headerssent in the request.

SC_NOT_FOUND - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (404) indicating that the requested resource is not available.

SC_NOT_IMPLEMENTED - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (501) indicating the HTTP server does not support the functionality needed to fulfill the request.

SC_NOT_MODIFIED - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (304) indicating that a conditional GET operation found that the resource was available and not modified.

SC_OK - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u> Status code (200) indicating the request succeeded normally.

SC PARTIAL CONTENT - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (206) indicating that the server has fulfilled the partial GET request for the resource.

SC_PAYMENT_REQUIRED - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (402) reserved for future use.

SC_PRECONDITION_FAILED - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (412) indicating that the precondition given in one or more of the request-header fields evaluated to false when it was tested on the server.

<u>SC_PROXY_AUTHENTICATION_REQUIRED</u> - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (407) indicating that the client *MUST* first authenticate itself with the proxy.

SC REQUEST ENTITY TOO LARGE - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (413) indicating that the server is refusing to process the request

because the request entity is larger than the server is willing or able to process.

SC REQUEST TIMEOUT - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (408) indicating that the client did not produce a requestwithin the time that the server was prepared to wait.

SC_REQUEST_URI_TOO_LONG - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (414) indicating that the server is refusing to service the request because the *Request-URI* is longer than the server is willing to interpret.

SC_REQUESTED_RANGE_NOT_SATISFIABLE - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (416) indicating that the server cannot serve the requested byte range.

SC RESET CONTENT - Static variable in interface

javax.servlet.http.HttpServletResponse

Status code (205) indicating that the agent *SHOULD* reset the document view which caused the request to be sent.

SC SEE OTHER - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (303) indicating that the response to the request can be found under a different URI.

SC_SERVICE_UNAVAILABLE - Static variable in interface

javax.servlet.http.HttpServletResponse

Status code (503) indicating that the HTTP server is temporarily overloaded, and unable to handle the request.

SC_SWITCHING_PROTOCOLS - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (101) indicating the server is switching protocols according to Upgrade header.

SC_TEMPORARY_REDIRECT - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (307) indicating that the requested resource resides temporarily under a different URI.

SC_UNAUTHORIZED - Static variable in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Status code (401) indicating that the request requires HTTP authentication.

SC UNSUPPORTED MEDIA TYPE - Static variable in interface javax.servlet.http.<u>HttpServletResponse</u>

Status code (415) indicating that the server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

SC USE PROXY - Static variable in interface

javax.servlet.http.<u>HttpServletResponse</u>

Status code (305) indicating that the requested resource *MUST* be accessed through the proxy given by the *Location* field.

sendError(int) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to call sendError(int sc) on the wrapped response object.

sendError(int) - Method in interface javax.servlet.http.<u>HttpServletResponse</u> Sends an error response to the client using the specified status code and clearing the buffer.

sendError(int, String) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to call sendError(int sc, String msg) on the wrapped response object.

sendError(int, String) - Method in interface

javax.servlet.http.<u>HttpServletResponse</u>

Sends an error response to the client using the specified status clearing the buffer.

sendRedirect(String) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to return sendRedirect(String location) on the wrapped response object.

sendRedirect(String) - Method in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Sends a temporary redirect response to the client using the specified redirect location URL.

service(HttpServletRequest, HttpServletResponse) - Method in class
javax.servlet.http.HttpServlet

Receives standard HTTP requests from the public service method and dispatches them to the do*XXX* methods defined in this class.

service(ServletRequest, ServletResponse) - Method in interface javax.servlet.<u>Servlet</u>

Called by the servlet container to allow the servlet to respond to a request. **service(ServletRequest, ServletResponse)** - Method in class javax.servlet.GenericServlet

Called by the servlet container to allow the servlet to respond to a request.

service(ServletRequest, ServletResponse) - Method in class javax.servlet.http.<u>HttpServlet</u>

Dispatches client requests to the protected service method.

Servlet - interface javax.servlet.**Servlet**.

Defines methods that all servlets must implement.

ServletConfig - interface javax.servlet.<u>ServletConfig</u>.

A servlet configuration object used by a servlet container used to pass information to a servlet during initialization.

<u>ServletContext</u> - interface javax.servlet.<u>ServletContext</u>.

Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.

ServletContextAttributeEvent - class

 $javax. servlet. \underline{ServletContextAttributeEvent}.$

This is the event class for notifications about changes to the attributes of the servlet context of a web application.

<u>ServletContextAttributeEvent(ServletContext, String, Object)</u> - Constructor for class javax.servlet.<u>ServletContextAttributeEvent</u>

Construct a ServletContextAttributeEvent from the given context for the given attribute name and attribute value.

ServletContextAttributeListener - interface

 $javax. servlet. \underline{ServletContextAttributeListener}.$

Implementations of this interface recieve notifications of changes to the attribute list on the servlet context of a web application.

<u>ServletContextEvent</u> - class javax.servlet.<u>ServletContextEvent</u>.

This is the event class for notifications about changes to the servlet context of a web application.

<u>ServletContextEvent(ServletContext)</u> - Constructor for class

 $javax. servlet. \underline{ServletContextEvent}$

Construct a ServletContextEvent from the given context.

<u>ServletContextListener</u> - interface javax.servlet.<u>ServletContextListener</u>.

Implementations of this interface recieve notifications about changes to the servlet context of the web application they are part of.

<u>ServletException</u> - exception javax.servlet.<u>ServletException</u>.

Defines a general exception a servlet can throw when it encounters difficulty.

<u>ServletException()</u> - Constructor for class javax.servlet.<u>ServletException</u>
Constructs a new servlet exception.

ServletException(String) - Constructor for class javax.servlet. ServletException Constructs a new servlet exception with the specified message.

<u>ServletException(String, Throwable)</u> - Constructor for class javax.servlet.<u>ServletException</u>

Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation, including a description message.

ServletException(Throwable) - Constructor for class

javax.servlet.ServletException

Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation.

<u>ServletInputStream</u> - class javax.servlet.<u>ServletInputStream</u>.

Provides an input stream for reading binary data from a client request, including an efficient readLine method for reading data one line at a time.

<u>ServletInputStream()</u> - Constructor for class javax.servlet.<u>ServletInputStream</u>
Does nothing, because this is an abstract class.

<u>ServletOutputStream</u> - class javax.servlet.<u>ServletOutputStream</u>.

Provides an output stream for sending binary data to the client.

ServletOutputStream() - Constructor for class

javax.servlet.ServletOutputStream

Does nothing, because this is an abstract class.

<u>ServletRequest</u> - interface javax.servlet.<u>ServletRequest</u>.

Defines an object to provide client request information to a servlet.

<u>ServletRequestWrapper</u> - class javax.servlet.<u>ServletRequestWrapper</u>.

Provides a convenient implementation of the ServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet.

<u>ServletRequestWrapper(ServletRequest)</u> - Constructor for class javax.servlet.<u>ServletRequestWrapper</u>

Creates a ServletRequest adaptor wrapping the given request object.

 $\underline{ServletResponse} \text{ - interface } javax.servlet. \underline{ServletResponse}.$

Defines an object to assist a servlet in sending a response to the client.

<u>ServletResponseWrapper</u> - class javax.servlet.<u>ServletResponseWrapper</u>.

Provides a convenient implementation of the ServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet.

<u>ServletResponseWrapper(ServletResponse)</u> - Constructor for class javax.servlet.ServletResponseWrapper

Creates a ServletResponse adaptor wrapping the given response object.

```
sessionCreated(HttpSessionEvent) - Method in interface
```

javax.servlet.http.<u>HttpSessionListener</u>

Notification that a session was created.

 $\underline{sessionDestroyed(HttpSessionEvent)} - Method in interface$

javax.servlet.http.<u>HttpSessionListener</u>

Notification that a session was invalidated.

sessionDidActivate(HttpSessionEvent) - Method in interface

 $javax. servlet. http. \underline{HttpSessionActivationListener}$

Notification that the session has just been activated.

sessionWillPassivate(HttpSessionEvent) - Method in interface

 $javax. servlet. http. \underline{HttpSessionActivationListener}$

Notification that the session is about to be passivated.

setAttribute(String, Object) - Method in interface javax.servlet.<u>ServletContext</u>
Binds an object to a given attribute name in this servlet context.

setAttribute(String, Object) - Method in class

javax.servlet.ServletRequestWrapper

The default behavior of this method is to return setAttribute(String name, Object o) on the wrapped request object.

setAttribute(String, Object) - Method in interface javax.servlet.<u>ServletRequest</u> Stores an attribute in this request.

setAttribute(String, Object) - Method in interface

javax.servlet.http.<u>HttpSession</u>

Binds an object to this session, using the name specified.

setBufferSize(int) - Method in interface javax.servlet.<u>ServletResponse</u>
Sets the preferred buffer size for the body of the response.

setBufferSize(int) - Method in class javax.servlet.ServletResponseWrapper

The default behavior of this method is to call setBufferSize(int size) on the wrapped response object.

setCharacterEncoding(String) - Method in class

 $javax. servlet. \underline{ServletRequestWrapper}$

The default behavior of this method is to set the character encoding on the wrapped request object.

 $\underline{\textbf{setCharacterEncoding}(\textbf{String})} \text{ - Method in interface}$

javax.servlet.<u>ServletRequest</u>

Overrides the name of the character encoding used in the body of this request.

setComment(String) - Method in class javax.servlet.http.Cookie

Specifies a comment that describes a cookie's purpose.

setContentLength(int) - Method in interface javax.servlet.<u>ServletResponse</u>

Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.

setContentLength(int) - Method in class javax.servlet.ServletResponseWrapper
The default behavior of this method is to call setContentLength(int len) on the wrapped response object.

setContentType(String) - Method in interface javax.servlet. <u>ServletResponse</u>
Sets the content type of the response being sent to the client.

setContentType(String) - Method in class

javax.servlet.ServletResponseWrapper

The default behavior of this method is to call setContentType(String type) on the wrapped response object.

setDateHeader(String, long) - Method in class

 $javax. servlet. http. \underline{HttpServletResponseWrapper}$

The default behavior of this method is to call setDateHeader(String name, long date) on the wrapped response object.

setDateHeader(String, long) - Method in interface

javax.servlet.http.HttpServletResponse

Sets a response header with the given name and date-value.

setDomain(String) - Method in class javax.servlet.http.Cookie

Specifies the domain within which this cookie should be presented.

setHeader(String, String) - Method in class

 $javax. servlet. http. \underline{HttpServletResponseWrapper}$

The default behavior of this method is to return setHeader(String name, String value) on the wrapped response object.

setHeader(String, String) - Method in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Sets a response header with the given name and value.

setIntHeader(String, int) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to call setIntHeader(String name, int value) on the wrapped response object.

setIntHeader(String, int) - Method in interface

 $javax. servlet. http. \underline{HttpServletResponse}$

Sets a response header with the given name and integer value.

<u>setLocale(Locale)</u> - Method in interface javax.servlet.<u>ServletResponse</u>

Sets the locale of the response, setting the headers (including the Content-Type's charset) as appropriate.

setLocale(Locale) - Method in class javax.servlet.ServletResponseWrapper

The default behavior of this method is to call setLocale(Locale loc) on the

wrapped response object.

setMaxAge(int) - Method in class javax.servlet.http.Cookie

Sets the maximum age of the cookie in seconds.

setMaxInactiveInterval(int) - Method in interface

javax.servlet.http.HttpSession

Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

setPath(String) - Method in class javax.servlet.http.Cookie

Specifies a path for the cookie to which the client should return the cookie.

setRequest(ServletRequest) - Method in class

 $javax. servlet. \underline{ServletRequestWrapper}$

Sets the request object being wrapped.

setResponse(ServletResponse) - Method in class

 $javax. servlet. \underline{ServletResponseWrapper}$

Sets the response being wrapped.

setSecure(boolean) - Method in class javax.servlet.http.Cookie

Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HTTPS or SSL.

setStatus(int) - Method in class javax.servlet.http.<u>HttpServletResponseWrapper</u>
The default behavior of this method is to call setStatus(int sc) on the wrapped response object.

setStatus(int) - Method in interface javax.servlet.http.<u>HttpServletResponse</u>
Sets the status code for this response.

setStatus(int, String) - Method in class

javax.servlet.http.<u>HttpServletResponseWrapper</u>

The default behavior of this method is to call setStatus(int sc, String sm) on the wrapped response object.

setStatus(int, String) - Method in interface

javax.servlet.http.<u>HttpServletResponse</u>

Deprecated. As of version 2.1, due to ambiguous meaning of the message parameter. To set a status code use setStatus(int), to send an error with a description use sendError(int, String). Sets the status code and message for this response.

setValue(String) - Method in class javax.servlet.http.Cookie

Assigns a new value to a cookie after the cookie is created.

setVersion(int) - Method in class javax.servlet.http.Cookie

Sets the version of the cookie protocol this cookie complies with.

<u>SingleThreadModel</u> - interface javax.servlet.<u>SingleThreadModel</u>.

Ensures that servlets handle only one request at a time.

<u>UnavailableException</u> - exception javax.servlet.<u>UnavailableException</u>.

Defines an exception that a servlet or filter throws to indicate that it is permanently or temporarily unavailable.

<u>UnavailableException(int, Servlet, String)</u> - Constructor for class javax.servlet.<u>UnavailableException</u>

Deprecated. As of Java Servlet API 2.2, use

<u>UnavailableException.UnavailableException(String, int)</u> instead.

<u>UnavailableException(Servlet, String)</u> - Constructor for class javax.servlet.<u>UnavailableException</u>

Deprecated. As of Java Servlet API 2.2, use

<u>UnavailableException.UnavailableException(String)</u> instead.

<u>UnavailableException(String)</u> - Constructor for class javax.servlet.<u>UnavailableException</u>

Constructs a new exception with a descriptive message indicating that the servlet is permanently unavailable.

<u>UnavailableException(String, int)</u> - Constructor for class javax.servlet.<u>UnavailableException</u>

Constructs a new exception with a descriptive message indicating that the servlet is temporarily unavailable and giving an estimate of how long it will be unavailable.

\mathbf{V}

<u>valueBound(HttpSessionBindingEvent)</u> - Method in interface

javax.servlet.http.HttpSessionBindingListener

Notifies the object that it is being bound to a session and identifies the session.

valueUnbound(HttpSessionBindingEvent) - Method in interface javax.servlet.http.HttpSessionBindingListener

Notifies the object that it is being unbound from a session and identifies the session.

ABCDEFGHIJLPRSUV

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Deprecated API

Deprecated Classes

javax.servlet.http.HttpUtils As of Java(tm) Servlet API 2.3. These methods were only useful with the default encoding and have been moved to the request interfaces.

Deprecated Interfaces

javax.servlet.http.HttpSessionContext

As of Java(tm) Servlet API 2.1 for security reasons, with no replacement. This interface will be removed in a future version of this API.

Deprecated Methods

<u>javax.servlet.http.HttpServletResponse.encodeRedirectUrl(String)</u>

As of version 2.1, use encodeRedirectURL(String url) instead

<u>javax.servlet.http.HttpServletResponse.encodeUrl(String)</u>

As of version 2.1, use encodeURL(String url) instead

javax.servlet.http.HttpSessionContext.getIds()

As of Java Servlet API 2.1 with no replacement. This method must return an empty Enumeration and will be removed in a future version of this API.

<u>javax.servlet.ServletRequest.getRealPath(String)</u>

As of Version 2.1 of the Java Servlet API, use

<u>javax.servlet.UnavailableException.getServlet()</u>

As of Java Servlet API 2.2, with no replacement. Returns the servlet that is reporting its unavailability.

<u>javax.servlet.ServletContext.getServlet(String)</u>

As of Java Servlet API 2.1, with no direct replacement.

This method was originally defined to retrieve a servlet from a ServletContext. In this version, this method always returns null and remains

only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

In lieu of this method, servlets can share information using the ServletContext class and can perform shared business logic by invoking methods on common non-servlet classes.

<u>javax.servlet.ServletContext.getServletNames()</u>

As of Java Servlet API 2.1, with no replacement.

This method was originally defined to return an Enumeration of all the servlet names known to this context. In this version, this method always returns an empty Enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

<u>javax.servlet.ServletContext.getServlets()</u>

As of Java Servlet API 2.0, with no replacement.

This method was originally defined to return an Enumeration of all the servlets known to this servlet context. In this version, this method always returns an empty enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

<u>javax.servlet.http.HttpSessionContext.getSession(String)</u>

As of Java Servlet API 2.1 with no replacement. This method must return null and will be removed in a future version of this API.

javax.servlet.http.HttpSession.getSessionContext()

As of Version 2.1, this method is deprecated and has no replacement. It will be removed in a future version of the Java Servlet API.

<u>javax.servlet.http.HttpSession.getValue(String)</u>

As of Version 2.2, this method is replaced by

<u> HttpSession.getAttribute(java.lang.String)</u>.

javax.servlet.http.HttpSession.getValueNames()

As of Version 2.2, this method is replaced by

<u> HttpSession.getAttributeNames()</u>

<u>javax.servlet.http.HttpServletRequest.isRequestedSessionIdFromUrl()</u>

As of Version 2.1 of the Java Servlet API, use

<u> HttpServletRequest.isRequestedSessionIdFromURL()</u> instead.

javax.servlet.ServletContext.log(Exception, String)

As of Java Servlet API 2.1, use <u>ServletContext.log(String message, Throwable throwable)</u> instead.

This method was originally defined to write an exception's stack trace and an explanatory error message to the servlet log file.

<u>javax.servlet.http.HttpSession.putValue(String, Object)</u>

As of Version 2.2, this method is replaced by

<u> HttpSession.setAttribute(java.lang.String, java.lang.Object)</u>

<u>javax.servlet.http.HttpSession.removeValue(String)</u>

As of Version 2.2, this method is replaced by

<u> HttpSession.removeAttribute(java.lang.String)</u>

<u>javax.servlet.http.HttpServletResponse.setStatus(int, String)</u>

As of version 2.1, due to ambiguous meaning of the message parameter. To set a status code use setStatus(int), to send an error with a description use sendError(int, String). Sets the status code and message for this response.

Deprecated Constructors

<u>javax.servlet.UnavailableException(int, Servlet, String)</u>

As of Java Servlet API 2.2, use

<u>UnavailableException.UnavailableException(String, int)</u> instead.

javax.servlet.UnavailableException(Servlet, String)

As of Java Servlet API 2.2, use

UnavailableException.UnavailableException(String) instead.

Overview Package Class Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Overview Package Class <u>Tree</u> <u>Deprecated Index Help</u>

PREV NEXT FRAMES NO FRAMES

Packages	
javax.servlet	
javax.servlet.http	

Overview Package Class <u>Tree</u> <u>Deprecated Index Help</u>

PREV NEXT FRAMES NO FRAMES

Overview Package Class Tree Deprecated Index Help

PREV <u>NEXT</u>

FRAMES NO FRAMES

Hierarchy For Package javax.servlet

Package Hierarchies: All Packages

Class Hierarchy

- class java.lang.Object
 - class java.util.EventObject (implements java.io.Serializable)
 - class javax.servlet.<u>ServletContextEvent</u>
 - class javax.servlet.<u>ServletContextAttributeEvent</u>
 - class javax.servlet.<u>GenericServlet</u> (implements java.io.Serializable, javax.servlet.<u>Servlet</u>, javax.servlet.<u>ServletConfig</u>)
 - class java.io.InputStream
 - class javax.servlet.<u>ServletInputStream</u>
 - class java.io.OutputStream
 - class javax.servlet.ServletOutputStream
 - class javax.servlet.<u>ServletRequestWrapper</u> (implements javax.servlet.<u>ServletRequest</u>)
 - class javax.servlet.<u>ServletResponseWrapper</u> (implements javax.servlet.<u>ServletResponse</u>)
 - class java.lang.Throwable (implements java.io.Serializable)
 - class java.lang.Exception
 - class javax.servlet.ServletException
 - class javax.servlet.<u>UnavailableException</u>

Interface Hierarchy

- interface java.util.EventListener
 - interface javax.servlet.<u>ServletContextAttributeListener</u>
 - interface javax.servlet.<u>ServletContextListener</u>
- interface javax.servlet.Filter
- interface javax.servlet.FilterChain
- interface javax.servlet.FilterConfig
- interface javax.servlet.<u>RequestDispatcher</u>
- interface javax.servlet.Servlet
- interface javax.servlet.<u>ServletConfig</u>
- interface javax.servlet.ServletContext
- interface javax.servlet.ServletRequest
- interface javax.servlet.<u>ServletResponse</u>
- interface javax.servlet.<u>SingleThreadModel</u>

Overview Package Class Tree Deprecated Index Help

PREV <u>NEXT</u> <u>FRAMES</u> <u>NO FRAMES</u>

PREV PACKAGE <u>NEXT PACKAGE</u>

FRAMES NO FRAMES

Package javax.servlet

Interface Summary	
<u>Filter</u>	A filter is an object than perform filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both. Filters perform filtering in the doFilter method.
<u>FilterChain</u>	A FilterChain is an object provided by the servlet container to the developer giving a view into the invocation chain of a filtered request for a resource.
<u>FilterConfig</u>	A filter configuration object used by a servlet container used to pass information to a filter during initialization.
<u>RequestDispatcher</u>	Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server.
<u>Servlet</u>	Defines methods that all servlets must implement.
<u>ServletConfig</u>	A servlet configuration object used by a servlet container used to pass information to a servlet during initialization.
<u>ServletContext</u>	Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.
${\color{red} Servlet Context Attribute Listener}$	Implementations of this interface recieve notifications of changes to the attribute list on the servlet context of a web application.
	Implementations of this interface recieve

<u>ServletContextListener</u>	notifications about changes to the servlet context of the web application they are part of.
<u>ServletRequest</u>	Defines an object to provide client request information to a servlet.
<u>ServletResponse</u>	Defines an object to assist a servlet in sending a response to the client.
<u>SingleThreadModel</u>	Ensures that servlets handle only one request at a time.

Class Summary		
<u>GenericServlet</u>	Defines a generic, protocol-independent servlet.	
<u>ServletContextAttributeEvent</u>	This is the event class for notifications about changes to the attributes of the servlet context of a web application.	
<u>ServletContextEvent</u>	This is the event class for notifications about changes to the servlet context of a web application.	
<u>ServletInputStream</u>	Provides an input stream for reading binary data from a client request, including an efficient readLine method for reading data one line at a time.	
<u>ServletOutputStream</u>	Provides an output stream for sending binary data to the client.	
<u>ServletRequestWrapper</u>	Provides a convenient implementation of the ServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet.	
<u>ServletResponseWrapper</u>	Provides a convenient implementation of the ServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet.	

Exception Summary		
"SPEVIELE VERNIMI	Defines a general exception a servlet can throw when it encounters difficulty.	
UnavailableException	Defines an exception that a servlet or filter throws to indicate that it is permanently or temporarily unavailable.	

PREV PACKAGE NEXT PACKAGE

FRAMES NO FRAMES

PREV CLASS NEXT CLASS

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface SingleThreadModel

public abstract interface SingleThreadModel

Ensures that servlets handle only one request at a time. This interface has no methods.

If a servlet implements this interface, you are *guaranteed* that no two threads will execute concurrently in the servlet's service method. The servlet container can make this guarantee by synchronizing access to a single instance of the servlet, or by maintaining a pool of servlet instances and dispatching each new request to a free servlet.

This interface does not prevent synchronization problems that result from servlets accessing shared resources such as static class variables or classes outside the scope of the servlet.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface Filter

public abstract interface Filter

A filter is an object than perform filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both.

Filters perform filtering in the doFilter method. Every Filter has access to a FilterConfig object from which it can obtain its initialization parameters, a reference to the ServletContext which it can use, for example, to load resources needed for filtering tasks.

Filters are configured in the deployment descriptor of a web application

Examples that have been identified for this design are

- 1) Authentication Filters
- 2) Logging and Auditing Filters
- 3) Image conversion Filters
- 4) Data compression Filters
- 5) Encryption Filters
- 6) Tokenizing Filters
- 7) Filters that trigger resource access events
- 8) XSL/T filters
- 9) Mime-type chain Filter

Since:

Servlet 2.3

Method Summary destroy() Called by the web container to indicate to a filter that it is being taken out of service. doFilter(ServletRequest request, ServletResponse response, FilterChain chain) The doFilter method of the Filter is called by the container each

time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.

init(FilterConfig filterConfig)

Called by the web container to indicate to a filter that it is being placed into service.

Method Detail

init

void

Called by the web container to indicate to a filter that it is being placed into service. The servlet container calls the init method exactly once after instantiating the filter. The init method must complete successfully before the filter is asked to do any filtering work.

The web container cannot place the filter into service if the init method either

- 1.Throws a ServletException
- 2.Does not return within a time period defined by the web container

doFilter

```
public void doFilter(<u>ServletRequest</u> request, <u>ServletResponse</u> response, <u>FilterChain</u> chain) throws java.io.IOException, <u>ServletException</u>
```

The doFilter method of the Filter is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain. The FilterChain passed in to this method allows the Filter to pass on the request and response to the next entity in the chain.

A typical implementation of this method would follow the following pattern:-

- 1. Examine the request
- 2. Optionally wrap the request object with a custom implementation to filter content or headers for input filtering
- 3. Optionally wrap the response object with a custom implementation to filter content or headers for output filtering
- 4. a) **Either** invoke the next entity in the chain using the FilterChain object (chain.doFilter()),
- 4. b) **or** not pass on the request/response pair to the next entity in the filter chain to block the request processing
- 5. Directly set headers on the response after invokation of the next entity in ther filter chain.

destroy

public void destroy()

Called by the web container to indicate to a filter that it is being taken out of service. This method is only called once all threads within the filter's doFilter method have exited or after a timeout period has passed. After the web container calls this method, it will not call the doFilter method again on this instance of the filter.

This method gives the filter an opportunity to clean up any resources that are being held (for example, memory, file handles, threads) and make sure that any persistent state is synchronized with the filter's current state in memory.

Overview Package Class Tree Deprecated Index Help

PREV CLASS <u>NEXT CLASS</u>
SUMMARY: INNER | FIELD | CONSTR | <u>METHOD</u>

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface FilterChain

public abstract interface FilterChain

A FilterChain is an object provided by the servlet container to the developer giving a view into the invocation chain of a filtered request for a resource. Filters use the FilterChain to invoke the next filter in the chain, or if the calling filter is the last filter in the chain, to invoke the rosource at the end of the chain.

Since:

Servlet 2.3

See Also:

Filter

Method Summary

void

doFilter(ServletRequest request, ServletResponse response)

Causes the next filter in the chain to be invoked, or if the calling filter is the last filter in the chain, causes the resource at the end of the chain to be invoked.

Method Detail

doFilter

```
public void doFilter(<u>ServletRequest</u> request, <u>ServletResponse</u> response) throws java.io.IOException, <u>ServletException</u>
```

Causes the next filter in the chain to be invoked, or if the calling filter is the last filter in the chain, causes the resource at the end of the chain to be invoked.

Parameters:

request - the request to pass along the chain.

response - the response to pass along the chain.

Since:

2.3

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface FilterConfig

public abstract interface FilterConfig

A filter configuration object used by a servlet container used to pass information to a filter during initialization.

Since:

Servlet 2.3

See Also:

Filter

Method Sumi	Method Summary		
java.lang.String	<pre>getFilterName() Returns the filter-name of this filter as defined in the deployment descriptor.</pre>		
java.lang.String	<pre>getInitParameter(java.lang.String name) Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.</pre>		
	<pre>getInitParameterNames() Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.</pre>		
<u>ServletContext</u>	<pre>getServletContext() Returns a reference to the ServletContext in which the caller is executing.</pre>		

Method Detail

getFilterName

```
public java.lang.String getFilterName()
```

Returns the filter-name of this filter as defined in the deployment descriptor.

getServletContext

```
public <u>ServletContext</u> getServletContext()
```

Returns a reference to the <u>ServletContext</u> in which the caller is executing. **Returns:**

a <u>ServletContext</u> object, used by the caller to interact with its servlet container

See Also:

ServletContext

getInitParameter

```
public java.lang.String getInitParameter(java.lang.String name)
```

Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

Parameters:

 ${\tt name}$ - a ${\tt String}$ specifying the name of the initialization parameter

Returns:

a String containing the value of the initialization parameter

getInitParameterNames

```
public java.util.Enumeration getInitParameterNames()
```

Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.

Returns:

an Enumeration of String objects containing the names of the servlet's initialization parameters

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

javax.servlet Class GenericServlet

Direct Known Subclasses:

HttpServlet

public abstract class **GenericServlet** extends java.lang.Object implements <u>Servlet</u>, <u>ServletConfig</u>, java.io.Serializable

Defines a generic, protocol-independent servlet. To write an HTTP servlet for use on the Web, extend httpServlet instead.

GenericServlet implements the Servlet and ServletConfig interfaces. GenericServlet may be directly extended by a servlet, although it's more common to extend a protocol-specific subclass such as HttpServlet.

GenericServlet makes writing servlets easier. It provides simple versions of the lifecycle methods init and destroy and of the methods in the ServletConfig interface. GenericServlet also implements the log method, declared in the ServletContext interface.

To write a generic servlet, you need only override the abstract service method.

See Also:

Serialized Form

Constructor Summary

GenericServlet()

Does nothing.

Method Summary

void	destroy() Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
java.lang.String	getInitParameter (java.lang.String name) Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.
java.util.Enumeration	getInitParameterNames() Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.
ServletConfig	<pre>getServletConfig() Returns this servlet's ServletConfig object.</pre>
<u>ServletContext</u>	<pre>getServletContext()</pre>
java.lang.String	getServletInfo() Returns information about the servlet, such as author, version, and copyright.
java.lang.String	getServletName() Returns the name of this servlet instance.
void	<pre>init() A convenience method which can be overridden so that there's no need to call super.init(config).</pre>
void	<pre>init(ServletConfig config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.</pre>
void	prepended by the servlet's name.
void	log(java.lang.String message, java.lang.Throwable t) Writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file, prepended by the servlet's name.

abstract void

service(ServletRequest req, ServletResponse res)
 Called by the servlet container to allow the servlet to
respond to a request.

Methods inherited from class java.lang.Object

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

Constructor Detail

GenericServlet

public GenericServlet()

Does nothing. All of the servlet initialization is done by one of the init methods.

Method Detail

destroy

public void destroy()

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service. See <u>Servlet.destroy()</u>.

Specified by:

destroy in interface Servlet

getInitParameter

public java.lang.String getInitParameter(java.lang.String name)

Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist. See ServletConfig.getInitParameter(java.lang.String).

This method is supplied for convenience. It gets the value of the named parameter from the servlet's ServletConfig object.

Specified by:

getInitParameter in interface ServletConfig

Parameters:

name - a String specifying the name of the initialization parameter

Returns:

String a String containing the value of the initalization parameter

getInitParameterNames

public java.util.Enumeration getInitParameterNames()

Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters. See

<u>ServletConfig.getInitParameterNames()</u>.

This method is supplied for convenience. It gets the parameter names from the servlet's ServletConfig object.

Specified by:

getInitParameterNames in interface ServletConfig

Returns:

Enumeration an enumeration of String objects containing the names of the servlet's initialization parameters

getServletConfig

public <u>ServletConfig</u> getServletConfig()

Returns this servlet's <u>ServletConfig</u> object.

Specified by:

getServletConfig in interface Servlet

Returns:

ServletConfig the ServletConfig object that initialized this servlet

getServletContext

```
public <u>ServletContext</u> getServletContext()
```

Returns a reference to the <u>ServletContext</u> in which this servlet is running. See <u>ServletConfig.getServletContext()</u>.

This method is supplied for convenience. It gets the context from the servlet's ServletConfig object.

Specified by:

getServletContext in interface ServletConfig

Returns:

ServletContext the ServletContext object passed to this servlet by the init method

getServletInfo

```
public java.lang.String getServletInfo()
```

Returns information about the servlet, such as author, version, and copyright. By default, this method returns an empty string. Override this method to have it return a meaningful value. See

Servlet.getServletInfo().

Specified by:

getServletInfo in interface Servlet

Returns:

String information about this servlet, by default an empty string

init

Called by the servlet container to indicate to a servlet that the servlet is being placed into service. See

Servlet.init(javax.servlet.ServletConfig).

This implementation stores the <u>ServletConfig</u> object it receives from the servlet container for later use. When overriding this form of the method, call super.init(config).

Specified by:

init in interface Servlet

Parameters:

config - the ServletConfig object that contains configutation information for this servlet

Throws:

<u>ServletException</u> - if an exception occurs that interrupts the servlet's normal operation

See Also:

UnavailableException

init

A convenience method which can be overridden so that there's no need to call super.init(config).

Instead of overriding <u>init(ServletConfig)</u>, simply override this method and it will be called by GenericServlet.init(ServletConfig config). The ServletConfig object can still be retrieved via <u>getServletConfig()</u>.

Throws:

<u>ServletException</u> - if an exception occurs that interrupts the servlet's normal operation

log

```
public void log(java.lang.String msg)
```

Writes the specified message to a servlet log file, prepended by the servlet's name. See <u>ServletContext.log(String)</u>.

Parameters:

msg - a String specifying the message to be written to the log file

log

Writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file, prepended by the servlet's name. See ServletContext.log(String, Throwable).

Parameters:

message - a String that describes the error or exception t - the java.lang.Throwable error or exception

service

```
public abstract void service(<u>ServletRequest</u> req, <u>ServletResponse</u> res) throws <u>ServletException</u>, java.io.IOException
```

Called by the servlet container to allow the servlet to respond to a request. See <u>Servlet.service(javax.servlet.ServletRequest,</u> javax.servlet.ServletResponse).

This method is declared abstract so subclasses, such as HttpServlet, must override it.

Specified by:

service in interface Servlet

Parameters:

req - the ServletRequest object that contains the client's request res - the ServletResponse object that will contain the servlet's response

Throws:

<u>ServletException</u> - if an exception occurs that interferes with the servlet's normal operation occurred

java.io.IOException - if an input or output exception occurs

getServletName

public java.lang.String getServletName()

Returns the name of this servlet instance. See <u>ServletConfig.getServletName()</u>.

Specified by:

getServletName in interface ServletConfig

Returns:

the name of this servlet instance

Overview Package Class Tree Deprecated Index Help

PREV CLASS <u>NEXT CLASS</u>
SUMMARY: INNER | FIELD | <u>CONSTR | METHOD</u>

PREV NEXT FRAMES NO FRAMES

Hierarchy For Package javax.servlet.http

Package Hierarchies:

All Packages

Class Hierarchy

- class java.lang.Object
 - class javax.servlet.http.<u>Cookie</u> (implements java.lang.Cloneable)
 - class java.util.EventObject (implements java.io.Serializable)
 - class javax.servlet.http.<u>HttpSessionEvent</u>
 - class javax.servlet.http.<u>HttpSessionBindingEvent</u>
 - class javax.servlet.<u>GenericServlet</u> (implements java.io.Serializable, javax.servlet.<u>Servlet</u>, javax.servlet.<u>ServletConfig</u>)
 - class javax.servlet.http.<u>HttpServlet</u> (implements java.io.Serializable)
 - class javax.servlet.http.<u>HttpUtils</u>
 - class javax.servlet.<u>ServletRequestWrapper</u> (implements javax.servlet.<u>ServletRequest</u>)
 - class javax.servlet.http.<u>HttpServletRequestWrapper</u> (implements javax.servlet.http.<u>HttpServletRequest</u>)
 - class javax.servlet.<u>ServletResponseWrapper</u> (implements javax.servlet.<u>ServletResponse</u>)
 - class javax.servlet.http.<u>HttpServletResponseWrapper</u> (implements javax.servlet.http.<u>HttpServletResponse</u>)

Interface Hierarchy

- interface java.util.EventListener
 - interface javax.servlet.http.<u>HttpSessionActivationListener</u>
 - interface javax.servlet.http.<u>HttpSessionAttributeListener</u>
 - interface javax.servlet.http.<u>HttpSessionBindingListener</u>
 - interface javax.servlet.http.<u>HttpSessionListener</u>
- interface javax.servlet.http.<u>HttpSession</u>
- interface javax.servlet.http.<u>HttpSessionContext</u>
- interface javax.servlet.<u>ServletRequest</u>
 - interface javax.servlet.http.<u>HttpServletRequest</u>
- interface javax.servlet.<u>ServletResponse</u>
 - interface javax.servlet.http.<u>HttpServletResponse</u>

Overview Package Class	Tree Deprecated Index Help
PREV NEXT	FRAMES NO FRAMES

PREV PACKAGE NEXT PACKAGE

FRAMES NO FRAMES

Package javax.servlet.http

Interface Summary			
<u>HttpServletRequest</u>	Extends the <u>ServletRequest</u> interface to provide request information for HTTP servlets.		
<u>HttpServletResponse</u>	Extends the <u>ServletResponse</u> interface to provide HTTP-specific functionality in sending a response.		
<u>HttpSession</u>	Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.		
HttpSessionActivationListener	bjects that are bound to a session may listen container events notifying them that sessions ill be passivated and that session will be ctivated.		
HttpSessionAttributeListener	This listener interface can be implemented in order to get notifications of changes to the attribute lists of sessions within this web application.		
HttpSessionBindingListener	Causes an object to be notified when it is bound to or unbound from a session.		
HttpSessionContext	Deprecated. As of Java(tm) Servlet API 2.1 for security reasons, with no replacement.		
<u>HttpSessionListener</u>	Implementations of this interface may are notified of changes to the list of active sessions in a web application.		

Class Summary	
<u>Cookie</u>	Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server.

<u>HttpServlet</u>	Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site.
<u>HttpServletRequestWrapper</u>	Provides a convenient implementation of the HttpServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet.
<u>HttpServletResponseWrapper</u>	Provides a convenient implementation of the HttpServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet.
<u>HttpSessionBindingEvent</u>	Events of this type are either sent to an object that implements HttpSessionBindingListener when it is bound or unbound from a session, or to a HttpSessionAttributeListener that has been configured in the deployment descriptor when any attribute is bound, unbound or replaced in a session.
<u>HttpSessionEvent</u>	This is the class representing event notifications for changes to sessions within a web application.
<u>HttpUtils</u>	Deprecated. As of Java(tm) Servlet API 2.3.

Overview Package Class Tree Deprecated Index Help	Overview	Package	Class	Tree	Deprecated	Index	Help
---	-----------------	---------	-------	------	-------------------	-------	------

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

javax.servlet.http Class Cookie

public class **Cookie** extends java.lang.Object implements java.lang.Cloneable

Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server. A cookie's value can uniquely identify a client, so cookies are commonly used for session management.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number. Some Web browsers have bugs in how they handle the optional attributes, so use them sparingly to improve the interoperability of your servlets.

The servlet sends cookies to the browser by using the httpServletResponse.addCookie(javax.servlet.http.Cookie) method, which adds fields to HTTP response headers to send cookies to the browser, one at a time. The browser is expected to support 20 cookies for each Web server, 300 cookies total, and may limit cookie size to 4 KB each.

The browser returns cookies to the servlet by adding fields to HTTP request headers. Cookies can be retrieved from a request by using the <u>HttpServletRequest.getCookies</u>() method. Several cookies might have the same name but different path attributes.

Cookies affect the caching of the Web pages that use them. HTTP 1.0 does not cache pages that use cookies created with this class. This class does not support the cache control defined with HTTP 1.1.

This class supports both the Version 0 (by Netscape) and Version 1 (by RFC 2109) cookie specifications. By default, cookies are created using Version 0 to ensure the best interoperability.

Constructor Summary

Cookie(java.lang.String name, java.lang.String value)
Constructs a cookie with a specified name and value.

Method Su	ımmary
java.lang.Object	clone() Overrides the standard java.lang.Object.clone method to return a copy of this cookie.
java.lang.String	getComment() Returns the comment describing the purpose of this cookie, or null if the cookie has no comment.
java.lang.String	getDomain() Returns the domain name set for this cookie.
int	getMaxAge() Returns the maximum age of the cookie, specified in seconds, By default, -1 indicating the cookie will persist until browser shutdown.
java.lang.String	getName() Returns the name of the cookie.
java.lang.String	getPath() Returns the path on the server to which the browser returns this cookie.
boolean	<pre>getSecure() Returns true if the browser is sending cookies only over a secure protocol, or false if the browser can send cookies using any protocol.</pre>
java.lang.String	getValue() Returns the value of the cookie.
int	getVersion() Returns the version of the protocol this cookie complies with.
void	<pre>setComment(java.lang.String purpose) Specifies a comment that describes a cookie's purpose.</pre>

void	setDomain(java.lang.String pattern) Specifies the domain within which this cookie should be presented.	
void	setMaxAge(int expiry) Sets the maximum age of the cookie in seconds.	
void	setPath(java.lang.String uri) Specifies a path for the cookie to which the client should return the cookie.	
void	setSecure (boolean flag) Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HTTPS or SSL.	
void	setValue(java.lang.String newValue) Assigns a new value to a cookie after the cookie is created.	
void	setVersion(int v) Sets the version of the cookie protocol this cookie complies with.	

Methods inherited from class java.lang.Object

equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Cookie

Constructs a cookie with a specified name and value.

The name must conform to RFC 2109. That means it can contain only ASCII alphanumeric characters and cannot contain commas, semicolons, or white space or begin with a \$ character. The cookie's name cannot be changed after creation.

The value can be anything the server chooses to send. Its value is probably of interest only to the server. The cookie's value can be changed after creation with the setValue method.

By default, cookies are created according to the Netscape cookie specification. The version can be changed with the setVersion method.

Parameters:

name - a String specifying the name of the cookie value - a String specifying the value of the cookie

Throws:

java.lang.IllegalArgumentException - if the cookie name contains illegal characters (for example, a comma, space, or semicolon) or it is one of the tokens reserved for use by the cookie protocol

See Also:

setValue(java.lang.String), setVersion(int)

Method Detail

setComment

public void setComment(java.lang.String purpose)

Specifies a comment that describes a cookie's purpose. The comment is useful if the browser presents the cookie to the user. Comments are not supported by Netscape Version 0 cookies.

Parameters:

purpose - a String specifying the comment to display to the user

See Also:

getComment()

getComment

```
public java.lang.String getComment()
```

Returns the comment describing the purpose of this cookie, or null if the cookie has no comment.

Returns:

a String containing the comment, or null if none

See Also:

setComment(java.lang.String)

setDomain

```
public void setDomain(java.lang.String pattern)
```

Specifies the domain within which this cookie should be presented.

The form of the domain name is specified by RFC 2109. A domain name begins with a dot (.foo.com) and means that the cookie is visible to servers in a specified Domain Name System (DNS) zone (for example, www.foo.com, but not a.b.foo.com). By default, cookies are only returned to the server that sent them.

Parameters:

pattern - a String containing the domain name within which this cookie is visible; form is according to RFC 2109

See Also:

getDomain()

getDomain

```
public java.lang.String getDomain()
```

Returns the domain name set for this cookie. The form of the domain name is set by RFC 2109.

Returns:

a String containing the domain name

See Also:

setDomain(java.lang.String)

setMaxAge

```
public void setMaxAge(int expiry)
```

Sets the maximum age of the cookie in seconds.

A positive value indicates that the cookie will expire after that many seconds have passed. Note that the value is the *maximum* age when the cookie will expire, not the cookie's current age.

A negative value means that the cookie is not stored persistently and will be deleted when the Web browser exits. A zero value causes the cookie to be deleted.

Parameters:

expiry - an integer specifying the maximum age of the cookie in seconds; if negative, means the cookie is not stored; if zero, deletes the cookie

See Also:

getMaxAge()

getMaxAge

```
public int getMaxAge()
```

Returns the maximum age of the cookie, specified in seconds, By default, -1 indicating the cookie will persist until browser shutdown.

Returns:

an integer specifying the maximum age of the cookie in seconds; if negative, means the cookie persists until browser shutdown

See Also:

setMaxAge(int)

setPath

```
public void setPath(java.lang.String uri)
```

Specifies a path for the cookie to which the client should return the cookie.

The cookie is visible to all the pages in the directory you specify, and all the pages in that directory's subdirectories. A cookie's path must include the servlet that set the cookie, for example, /catalog, which makes the cookie

visible to all directories on the server under /catalog.

Consult RFC 2109 (available on the Internet) for more information on setting path names for cookies.

Parameters:

uri - a String specifying a path

See Also:

getPath()

getPath

```
public java.lang.String getPath()
```

Returns the path on the server to which the browser returns this cookie. The cookie is visible to all subpaths on the server.

Returns:

a String specifying a path that contains a servlet name, for example, /catalog

See Also:

setPath(java.lang.String)

setSecure

```
public void setSecure(boolean flag)
```

Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HTTPS or SSL.

The default value is false.

Parameters:

flag - if true, sends the cookie from the browser to the server using only when using a secure protocol; if false, sent on any protocol

See Also:

getSecure()

getSecure

```
public boolean getSecure()
```

Returns true if the browser is sending cookies only over a secure protocol, or false if the browser can send cookies using any protocol.

Returns:

true if the browser uses a secure protocol; otherwise, true

See Also:

setSecure(boolean)

getName

```
public java.lang.String getName()
```

Returns the name of the cookie. The name cannot be changed after creation.

Returns:

a String specifying the cookie's name

setValue

```
public void setValue(java.lang.String newValue)
```

Assigns a new value to a cookie after the cookie is created. If you use a binary value, you may want to use BASE64 encoding.

With Version 0 cookies, values should not contain white space, brackets, parentheses, equals signs, commas, double quotes, slashes, question marks, at signs, colons, and semicolons. Empty values may not behave the same way on all browsers.

Parameters:

newValue - a String specifying the new value

See Also:

getValue(), Cookie

getValue

```
public java.lang.String getValue()
```

Returns the value of the cookie.

Returns:

a String containing the cookie's present value

See Also:

setValue(java.lang.String), Cookie

getVersion

```
public int getVersion()
```

Returns the version of the protocol this cookie complies with. Version 1 complies with RFC 2109, and version 0 complies with the original cookie specification drafted by Netscape. Cookies provided by a browser use and identify the browser's cookie version.

Returns:

0 if the cookie complies with the original Netscape specification; 1 if the cookie complies with RFC 2109

See Also:

setVersion(int)

setVersion

```
public void setVersion(int v)
```

Sets the version of the cookie protocol this cookie complies with. Version 0 complies with the original Netscape cookie specification. Version 1 complies with RFC 2109.

Since RFC 2109 is still somewhat new, consider version 1 as experimental; do not use it yet on production sites.

Parameters:

v - 0 if the cookie should comply with the original Netscape specification; 1 if the cookie should comply with RFC 2109

See Also:

getVersion()

clone

public java.lang.Object clone()

Overrides the standard java.lang.Object.clone method to return a copy of this cookie.

Overrides:

clone in class java.lang.Object

Overview Package Class Tree Deprecated Index Help

PREV CLASS <u>NEXT CLASS</u>
SUMMARY: INNER | FIELD | <u>CONSTR | METHOD</u>

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

javax.servlet.http Class HttpServlet

public abstract class **HttpServlet** extends <u>GenericServlet</u> implements java.io.Serializable

Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. A subclass of HttpServlet must override at least one method, usually one of these:

- doGet, if the servlet supports HTTP GET requests
- doPost, for HTTP POST requests
- doPut, for HTTP PUT requests
- doDelete, for HTTP DELETE requests
- init and destroy, to manage resources that are held for the life of the servlet
- getServletInfo, which the servlet uses to provide information about itself

There's almost no reason to override the service method. service handles standard HTTP requests by dispatching them to the handler methods for each HTTP request type (the do*XXX* methods listed above).

Likewise, there's almost no reason to override the doOptions and doTrace methods.

Servlets typically run on multithreaded servers, so be aware that a servlet must handle concurrent requests and be careful to synchronize access to shared resources. Shared resources include in-memory data such as instance or class variables and external objects such as files, database connections, and network connections. See the <u>Java Tutorial on Multithreaded Programming</u> for more information on handling multiple threads in a Java program.

See Also:

Serialized Form

Constructor Summary

HttpServlet()
Does nothing, because this is an abstract class.

Method Summary		
protected void	<pre>doDelete(HttpServletRequest req, HttpServletResponse resp)</pre>	
protected void	<pre>doGet(HttpServletRequest req, HttpServletResponse resp)</pre>	
protected void	doHead(HttpServletRequest req, HttpServletResponse resp) Receives an HTTP HEAD request from the protected service method and handles the request.	
protected void	<pre>doOptions(HttpServletRequest req, HttpServletResponse resp)</pre>	
protected void	<pre>doPost(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a POST request.</pre>	
protected void	doPut(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a PUT request.	
protected void	doTrace(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a TRACE request.	
protected long	getLastModified(HttpServletRequest req) Returns the time the HttpServletRequest object was last modified, in milliseconds since midnight January 1, 1970 GMT.	
protected	service(HttpServletRequest req, HttpServletResponse resp) Receives standard HTTP requests from the public service	

	method and dispatches them to the $doXXX$ methods defined in this class.
void	<pre>service(ServletRequest req, ServletResponse res) Dispatches client requests to the protected service method.</pre>

Methods inherited from class javax.servlet.<u>GenericServlet</u>

destroy, getInitParameter, getInitParameterNames,
getServletConfig, getServletContext, getServletInfo,
getServletName, init, init, log, log

Methods inherited from class java.lang.Object

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

Constructor Detail

HttpServlet

public HttpServlet()

Does nothing, because this is an abstract class.

Method Detail

doGet

```
protected void doGet(<a href="httpServletRequest">httpServletRequest</a> req,
<a href="httpServletResponse">httpServletResponse</a> resp)
<a href="httpServletException">throws</a> ServletException,
<a href="mailto:java.io.10Exception">java.io.10Exception</a>
```

Called by the server (via the service method) to allow a servlet to handle a GET request.

Overriding this method to support a GET request also automatically supports an HTTP HEAD request. A HEAD request is a GET request that

returns no body in the response, only the request header fields.

When overriding this method, read the request data, write the response headers, get the response's writer or output stream object, and finally, write the response data. It's best to include content type and encoding. When using a PrintWriter object to return the response, set the content type before accessing the PrintWriter object.

The servlet container must write the headers before committing the response, because in HTTP the headers must be sent before the response body.

Where possible, set the Content-Length header (with the ServletResponse.setContentLength(int) method), to allow the servlet container to use a persistent connection to return its response to the client, improving performance. The content length is automatically set if the entire response fits inside the response buffer.

The GET method should be safe, that is, without any side effects for which users are held responsible. For example, most form queries have no side effects. If a client request is intended to change stored data, the request should use some other HTTP method.

The GET method should also be idempotent, meaning that it can be safely repeated. Sometimes making a method safe also makes it idempotent. For example, repeating queries is both safe and idempotent, but buying a product online or modifying data is neither safe nor idempotent.

If the request is incorrectly formatted, doGet returns an HTTP "Bad Request" message.

Parameters:

req - an <u>HttpServletRequest</u> object that contains the request the client has made of the servlet

resp - an <u>HttpServletResponse</u> object that contains the response the servlet sends to the client

Throws:

java.io.IOException - if an input or output error is detected when the servlet handles the GET request

<u>ServletException</u> - if the request for the GET could not be handled

ServletResponse.setContentType(java.lang.String)

getLastModified

protected long getLastModified(HttpServletRequest req)

Returns the time the HttpServletRequest object was last modified, in milliseconds since midnight January 1, 1970 GMT. If the time is unknown, this method returns a negative number (the default).

Servlets that support HTTP GET requests and can quickly determine their last modification time should override this method. This makes browser and proxy caches work more effectively, reducing the load on server and network resources.

Parameters:

req - the HttpServletRequest object that is sent to the servlet **Returns:**

a long integer specifying the time the HttpServletRequest object was last modified, in milliseconds since midnight, January 1, 1970 GMT, or -1 if the time is not known

doHead

```
protected void doHead(<a href="httpServletRequest"><u>HttpServletRequest</u></a> req,
<a href="httpServletResponse"><u>HttpServletResponse</u></a> resp)
<a href="httpServletResponse">throws</a> ServletException,
<a href="java.io.IOException">java.io.IOException</a>
```

Receives an HTTP HEAD request from the protected service method and handles the request. The client sends a HEAD request when it wants to see only the headers of a response, such as Content-Type or Content-Length. The HTTP HEAD method counts the output bytes in the response to set the Content-Length header accurately.

If you override this method, you can avoid computing the response body and just set the response headers directly to improve performance. Make sure that the doHead method you write is both safe and idempotent (that is, protects itself from being called multiple times for one HTTP HEAD request).

If the HTTP HEAD request is incorrectly formatted, doHead returns an HTTP "Bad Request" message.

Parameters:

req - the request object that is passed to the servletresp - the response object that the servlet uses to return the headers to the clien

Throws:

java.io.IOException - if an input or output error occurs

ServletException - if the request for the HEAD could not be handled

doPost

```
protected void doPost(<a href="httpServletRequest"><u>HttpServletRequest</u></a> req,
<a href="httpServletResponse"><u>HttpServletResponse</u></a> resp)
<a href="httpServletResponse">throws</a> ServletException,
<a href="mailto:java.io.IOException">java.io.IOException</a>
```

Called by the server (via the service method) to allow a servlet to handle a POST request. The HTTP POST method allows the client to send data of unlimited length to the Web server a single time and is useful when posting information such as credit card numbers.

When overriding this method, read the request data, write the response headers, get the response's writer or output stream object, and finally, write the response data. It's best to include content type and encoding. When using a PrintWriter object to return the response, set the content type before accessing the PrintWriter object.

The servlet container must write the headers before committing the response, because in HTTP the headers must be sent before the response body.

Where possible, set the Content-Length header (with the ServletResponse.setContentLength(int) method), to allow the servlet

container to use a persistent connection to return its response to the client, improving performance. The content length is automatically set if the entire response fits inside the response buffer.

When using HTTP 1.1 chunked encoding (which means that the response has a Transfer-Encoding header), do not set the Content-Length header.

This method does not need to be either safe or idempotent. Operations requested through POST can have side effects for which the user can be held accountable, for example, updating stored data or buying items online.

If the HTTP POST request is incorrectly formatted, doPost returns an HTTP "Bad Request" message.

Parameters:

req - an <u>HttpServletRequest</u> object that contains the request the client has made of the servlet

resp - an <u>HttpServletResponse</u> object that contains the response the servlet sends to the client

Throws:

java.io.IOException - if an input or output error is detected when the servlet handles the request

<u>ServletException</u> - if the request for the POST could not be handled

See Also:

ServletOutputStream,
ServletResponse.setContentType(java.lang.String)

doPut

```
protected void doPut(<a href="httpServletRequest">httpServletRequest</a> req,
<a href="httpServletResponse">httpServletResponse</a> resp)
<a href="httpServletException">throws</a> ServletException,
<a href="mailto:java.io.10Exception">java.io.10Exception</a>
```

Called by the server (via the service method) to allow a servlet to handle a PUT request. The PUT operation allows a client to place a file on the server and is similar to sending a file by FTP.

When overriding this method, leave intact any content headers sent with the

request (including Content-Length, Content-Type, Content-Transfer-Encoding, Content-Encoding, Content-Base, Content-Language, Content-Location, Content-MD5, and Content-Range). If your method cannot handle a content header, it must issue an error message (HTTP 501 - Not Implemented) and discard the request. For more information on HTTP 1.1, see RFC 2068.

This method does not need to be either safe or idempotent. Operations that doPut performs can have side effects for which the user can be held accountable. When using this method, it may be useful to save a copy of the affected URL in temporary storage.

If the HTTP PUT request is incorrectly formatted, doPut returns an HTTP "Bad Request" message.

Parameters:

req - the HttpServletRequest object that contains the request the client made of the servlet

resp - the httpServletResponse object that contains the response the servlet returns to the client

Throws:

java.io.IOException - if an input or output error occurs while the servlet is handling the PUT request

<u>ServletException</u> - if the request for the PUT cannot be handled

doDelete

```
protected void doDelete(<a href="httpServletRequest">HttpServletRequest</a> req,
<a href="httpServletResponse">HttpServletResponse</a> resp)
<a href="httpServletException">throws</a> ServletException,
<a href="mailto:java.io.10Exception">java.io.10Exception</a>
```

Called by the server (via the service method) to allow a servlet to handle a DELETE request. The DELETE operation allows a client to remove a document or Web page from the server.

This method does not need to be either safe or idempotent. Operations requested through DELETE can have side effects for which users can be held accountable. When using this method, it may be useful to save a copy

of the affected URL in temporary storage.

If the HTTP DELETE request is incorrectly formatted, doDelete returns an HTTP "Bad Request" message.

Parameters:

req - the <u>HttpServletRequest</u> object that contains the request the client made of the servlet

resp - the <u>HttpServletResponse</u> object that contains the response the servlet returns to the client

Throws:

java.io.IOException - if an input or output error occurs while the servlet is handling the DELETE request

<u>ServletException</u> - if the request for the DELETE cannot be handled

doOptions

```
protected void doOptions(<a href="httpServletRequest">HttpServletRequest</a> req,
<a href="httpServletResponse">HttpServletResponse</a> resp)
<a href="httpServletException">throws</a> ServletException,
<a href="mailto:java.io.IOException">java.io.IOException</a>
```

Called by the server (via the service method) to allow a servlet to handle a OPTIONS request. The OPTIONS request determines which HTTP methods the server supports and returns an appropriate header. For example, if a servlet overrides doGet, this method returns the following header:

```
Allow: GET, HEAD, TRACE, OPTIONS
```

There's no need to override this method unless the servlet implements new HTTP methods, beyond those implemented by HTTP 1.1.

Parameters:

req - the HttpServletRequest object that contains the request the client made of the servlet

resp - the <u>HttpServletResponse</u> object that contains the response the servlet returns to the client

Throws:

java.io.IOException - if an input or output error occurs while the servlet is handling the OPTIONS request ServletException - if the request for the OPTIONS cannot be handled

doTrace

```
protected void doTrace(<a href="httpServletRequest"><u>HttpServletRequest</u></a> req,
<a href="httpServletResponse"><u>HttpServletResponse</u></a> resp)
<a href="httpServletResponse">throws</a> ServletException,
<a href="mailto:java.io.10Exception">java.io.10Exception</a>
```

Called by the server (via the service method) to allow a servlet to handle a TRACE request. A TRACE returns the headers sent with the TRACE request to the client, so that they can be used in debugging. There's no need to override this method.

Parameters:

req - the <u>HttpServletRequest</u> object that contains the request the client made of the servlet

resp - the httpServletResponse object that contains the response the servlet returns to the client

Throws:

java.io.IOException - if an input or output error occurs while the servlet is handling the TRACE request

<u>ServletException</u> - if the request for the TRACE cannot be handled

service

```
protected void service(<u>HttpServletRequest</u> req, <u>HttpServletResponse</u> resp) throws <u>ServletException</u>, java.io.IOException
```

Receives standard HTTP requests from the public service method and dispatches them to the do*XXX* methods defined in this class. This method is an HTTP-specific version of the

<u>Servlet.service(javax.servlet.ServletRequest,</u>

<u>javax.servlet.ServletResponse</u>) method. There's no need to override this method.

Parameters:

req - the <u>HttpServletRequest</u> object that contains the request the client made of the servlet

resp - the httpServletResponse object that contains the response the servlet returns to the client

Throws:

java.io.IOException - if an input or output error occurs while the servlet is handling the TRACE request

<u>ServletException</u> - if the request for the TRACE cannot be handled

See Also:

<u>Servlet.service(javax.servlet.ServletRequest, javax.servlet.ServletResponse)</u>

service

Dispatches client requests to the protected service method. There's no need to override this method.

Parameters:

req - the <u>HttpServletRequest</u> object that contains the request the client made of the servlet

resp - the httpServletResponse object that contains the response the servlet returns to the client

Throws:

java.io.IOException - if an input or output error occurs while the servlet is handling the TRACE request

<u>ServletException</u> - if the request for the TRACE cannot be handled

Overrides:

service in class GenericServlet

See Also:

<u>Servlet.service(javax.servlet.ServletRequest,</u> javax.servlet.ServletResponse)

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpServletRequest

All Known Implementing Classes:

<u>HttpServletRequestWrapper</u>

public abstract interface **HttpServletRequest** extends <u>ServletRequest</u>

Extends the <u>ServletRequest</u> interface to provide request information for HTTP servlets.

The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).

Field Summary		
static java.lang.String	BASIC AUTH String identifier for Basic authentication.	
static java.lang.String	CLIENT CERT AUTH String identifier for Basic authentication.	
static java.lang.String	DIGEST AUTH String identifier for Basic authentication.	
static java.lang.String	FORM AUTH String identifier for Basic authentication.	

Method Summary				
java.lang.String	getAuthType() Returns the name of the authentication scheme used to protect the servlet.			
java.lang.String	getContextPath() Returns the portion of the request URI that indicates the context of the request.			
	<pre>getCookies()</pre>			

<u>Cookie</u> []	Returns an array containing all of the Cookie objects the client sent with this request.
long	<pre>getDateHeader(java.lang.String name) Returns the value of the specified request header as a long value that represents a Date object.</pre>
java.lang.String	<pre>getHeader(java.lang.String name) Returns the value of the specified request header as a String.</pre>
java.util.Enumeration	getHeaderNames() Returns an enumeration of all the header names this request contains.
java.util.Enumeration	<pre>getHeaders(java.lang.String name) Returns all the values of the specified request header as an Enumeration of String objects.</pre>
int	<pre>getIntHeader(java.lang.String name) Returns the value of the specified request header as an int.</pre>
java.lang.String	<pre>getMethod() Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.</pre>
java.lang.String	getPathInfo() Returns any extra path information associated with the URL the client sent when it made this request.
	getPathTranslated() Returns any extra path information after the servlet name but before the query string, and translates it to a real path.
java.lang.String	<pre>getQueryString() Returns the query string that is contained in the request URL after the path.</pre>
java.lang.String	<pre>getRemoteUser() Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated.</pre>
java.lang.String	getRequestedSessionId() Returns the session ID specified by the client.

java.lang.String	getRequestURI() Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.
java.lang.StringBuffer	getRequestURL() Reconstructs the URL the client used to make the request.
java.lang.String	<pre>getServletPath() Returns the part of this request's URL that calls the servlet.</pre>
HttpSession	getSession() Returns the current session associated with this request, or if the request does not have a session, creates one.
HttpSession	getSession (boolean create) Returns the current HttpSession associated with this request or, if if there is no current session and create is true, returns a new session.
java.security.Principal	getUserPrincipal() Returns a java.security.Principal object containing the name of the current authenticated user.
boolean	isRequestedSessionIdFromCookie() Checks whether the requested session ID came in as a cookie.
boolean	isRequestedSessionIdFromUrl() Deprecated. As of Version 2.1 of the Java Servlet API, use isRequestedSessionIdFromURL() instead.
boolean	isRequestedSessionIdFromURL() Checks whether the requested session ID came in as part of the request URL.
boolean	isRequestedSessionIdValid() Checks whether the requested session ID is still valid.
boolean	isUserInRole(java.lang.String role) Returns a boolean indicating whether the

authenticated user is included in the specified logical "role".

Methods inherited from interface javax.servlet.ServletRequest

getAttribute, getAttributeNames, getCharacterEncoding,
getContentLength, getContentType, getInputStream, getLocale,
getLocales, getParameter, getParameterMap, getParameterNames,
getParameterValues, getProtocol, getReader, getRealPath,
getRemoteAddr, getRemoteHost, getRequestDispatcher, getScheme,
getServerName, getServerPort, isSecure, removeAttribute,
setAttribute, setCharacterEncoding

Field Detail

BASIC_AUTH

public static final java.lang.String BASIC_AUTH

String identifier for Basic authentication. Value "BASIC"

FORM_AUTH

public static final java.lang.String ${f FORM_AUTH}$

String identifier for Basic authentication. Value "FORM"

CLIENT_CERT_AUTH

public static final java.lang.String CLIENT_CERT_AUTH

String identifier for Basic authentication. Value "CLIENT_CERT"

DIGEST_AUTH

public static final java.lang.String DIGEST_AUTH

Method Detail

getAuthType

```
public java.lang.String getAuthType()
```

Returns the name of the authentication scheme used to protect the servlet. All servlet containers support basic, form and client certificate authentication, and may additionally support digest authentication. If the servlet is not authenticated null is returned.

Same as the value of the CGI variable AUTH TYPE.

Returns:

one of the static members BASIC_AUTH, FORM_AUTH, CLIENT_CERT_AUTH, DIGEST_AUTH (suitable for == comparison) indicating the authentication scheme, or null if the request was not authenticated.

getCookies

```
public Cookie[] getCookies()
```

Returns an array containing all of the Cookie objects the client sent with this request. This method returns null if no cookies were sent.

Returns:

an array of all the Cookies included with this request, or null if the request has no cookies

getDateHeader

```
public long getDateHeader(java.lang.String name)
```

Returns the value of the specified request header as a long value that represents a Date object. Use this method with headers that contain dates,

such as If-Modified-Since.

The date is returned as the number of milliseconds since January 1, 1970 GMT. The header name is case insensitive.

If the request did not have a header of the specified name, this method returns -1. If the header can't be converted to a date, the method throws an IllegalArgumentException.

Parameters:

name - a String specifying the name of the header

Returns:

a long value representing the date specified in the header expressed as the number of milliseconds since January 1, 1970 GMT, or -1 if the named header was not included with the regest

Throws:

java.lang.IllegalArgumentException - If the header value can't be converted to a date

getHeader

public java.lang.String getHeader(java.lang.String name)

Returns the value of the specified request header as a String. If the request did not include a header of the specified name, this method returns null. The header name is case insensitive. You can use this method with any request header.

Parameters:

name - a String specifying the header name

Returns:

a String containing the value of the requested header, or null if the request does not have a header of that name

getHeaders

public java.util.Enumeration getHeaders(java.lang.String name)

Returns all the values of the specified request header as an Enumeration of String objects.

Some headers, such as Accept-Language can be sent by clients as several headers each with a different value rather than sending the header as a comma separated list.

If the request did not include any headers of the specified name, this method returns an empty Enumeration. The header name is case insensitive. You can use this method with any request header.

Parameters:

name - a String specifying the header name

Returns:

an Enumeration containing the values of the requested header. If the request does not have any headers of that name return an empty enumeration. If the container does not allow access to header information, return null

getHeaderNames

public java.util.Enumeration getHeaderNames()

Returns an enumeration of all the header names this request contains. If the request has no headers, this method returns an empty enumeration.

Some servlet containers do not allow do not allow servlets to access headers using this method, in which case this method returns null

Returns:

an enumeration of all the header names sent with this request; if the request has no headers, an empty enumeration; if the servlet container does not allow servlets to use this method, null

getIntHeader

public int getIntHeader(java.lang.String name)

Returns the value of the specified request header as an int. If the request does not have a header of the specified name, this method returns -1. If the header cannot be converted to an integer, this method throws a NumberFormatException.

The header name is case insensitive.

Parameters:

name - a String specifying the name of a request header

Returns:

an integer expressing the value of the request header or -1 if the request doesn't have a header of this name

Throws:

java.lang.NumberFormatException - If the header value can't be converted to an int

getMethod

```
public java.lang.String getMethod()
```

Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT. Same as the value of the CGI variable REQUEST METHOD.

Returns:

a String specifying the name of the method with which this request was made

getPathInfo

```
public java.lang.String getPathInfo()
```

Returns any extra path information associated with the URL the client sent when it made this request. The extra path information follows the servlet path but precedes the query string. This method returns null if there was no extra path information.

Same as the value of the CGI variable PATH_INFO.

Returns:

a String, decoded by the web container, specifying extra path information that comes after the servlet path but before the query string in the request URL; or null if the URL does not have any extra path information

getPathTranslated

public java.lang.String getPathTranslated()

Returns any extra path information after the servlet name but before the query string, and translates it to a real path. Same as the value of the CGI variable PATH_TRANSLATED.

If the URL does not have any extra path information, this method returns null. The web container does not decode thins string.

Returns:

a String specifying the real path, or null if the URL does not have any extra path information

getContextPath

public java.lang.String getContextPath()

Returns the portion of the request URI that indicates the context of the request. The context path always comes first in a request URI. The path starts with a "/" character but does not end with a "/" character. For servlets in the default (root) context, this method returns "". The container does not decode this string.

Returns:

a String specifying the portion of the request URI that indicates the context of the request

getQueryString

```
public java.lang.String getQueryString()
```

Returns the query string that is contained in the request URL after the path. This method returns null if the URL does not have a query string. Same as the value of the CGI variable QUERY_STRING.

Returns:

a String containing the query string or null if the URL contains no query string. The value is not decoded by the container.

getRemoteUser

```
public java.lang.String getRemoteUser()
```

Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated. Whether the user name is sent with each subsequent request depends on the browser and type of authentication. Same as the value of the CGI variable REMOTE_USER.

Returns:

a String specifying the login of the user making this request, or null

isUserInRole

```
public boolean isUserInRole(java.lang.String role)
```

Returns a boolean indicating whether the authenticated user is included in the specified logical "role". Roles and role membership can be defined using deployment descriptors. If the user has not been authenticated, the method returns false.

Parameters:

role - a String specifying the name of the role

Returns:

a boolean indicating whether the user making this request belongs to a given role; false if the user has not been authenticated

getUserPrincipal

```
public java.security.Principal getUserPrincipal()
```

Returns a java.security.Principal object containing the name of the current authenticated user. If the user has not been authenticated, the method returns null.

Returns:

a java.security.Principal containing the name of the user making this request; null if the user has not been authenticated

getRequestedSessionId

```
public java.lang.String getRequestedSessionId()
```

Returns the session ID specified by the client. This may not be the same as the ID of the actual session in use. For example, if the request specified an old (expired) session ID and the server has started a new session, this method gets a new session with a new ID. If the request did not specify a session ID, this method returns null.

Returns:

a String specifying the session ID, or null if the request did not specify a session ID

See Also:

isRequestedSessionIdValid()

getRequestURI

```
public java.lang.String getRequestURI()
```

Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request. The web container does not decode this String. For example:

First line of HTTP request Returned Value

POST /some/path.html HTTP/1.1 /some/path.html

GET http://foo.bar/a.html HTTP/1.0 /a.html HEAD /xyz?a=b HTTP/1.1 /xyz

To reconstruct an URL with a scheme and host, use HttpUtils.getRequestURL(javax.servlet.http.HttpServletRequesture)

Returns:

a String containing the part of the URL from the protocol name up to the query string

See Also:

HttpUtils.getRequestURL(javax.servlet.http.HttpServletRegreesture)

getRequestURL

```
public java.lang.StringBuffer getRequestURL()
```

Reconstructs the URL the client used to make the request. The returned URL contains a protocol, server name, port number, and server path, but it does not include query string parameters.

Because this method returns a StringBuffer, not a string, you can modify the URL easily, for example, to append query parameters.

This method is useful for creating redirect messages and for reporting errors.

Returns:

a StringBuffer object containing the reconstructed URL

getServletPath

```
public java.lang.String getServletPath()
```

Returns the part of this request's URL that calls the servlet. This includes either the servlet name or a path to the servlet, but does not include any extra path information or a query string. Same as the value of the CGI variable SCRIPT_NAME.

Returns:

a String containing the name or path of the servlet being called, as specified in the request URL, decoded.

getSession

```
public HttpSession getSession(boolean create)
```

Returns the current HttpSession associated with this request or, if if there is no current session and create is true, returns a new session.

If create is false and the request has no valid HttpSession, this method returns null.

To make sure the session is properly maintained, you must call this method before the response is committed. If the container is using cookies to maintain session integrity and is asked to create a new session when the response is committed, an IllegalStateException is thrown.

Parameters:

true - to create a new session for this request if necessary; false to return null if there's no current session

Returns:

the HttpSession associated with this request or null if create is false and the request has no valid session

See Also:

getSession()

getSession

```
public HttpSession getSession()
```

Returns the current session associated with this request, or if the request does not have a session, creates one.

Returns:

the HttpSession associated with this request

See Also:

getSession(boolean)

isRequestedSessionIdValid

```
public boolean isRequestedSessionIdValid()
```

Checks whether the requested session ID is still valid.

Returns:

true if this request has an id for a valid session in the current

session context; false otherwise

See Also:

getRequestedSessionId(), getSession(boolean),
HttpSessionContext

is Requested Session Id From Cookie

public boolean isRequestedSessionIdFromCookie()

Checks whether the requested session ID came in as a cookie.

Returns:

true if the session ID came in as a cookie; otherwise, false

See Also:

getSession(boolean)

is Requested Session Id From URL

public boolean isRequestedSessionIdFromURL()

Checks whether the requested session ID came in as part of the request URL.

Returns:

true if the session ID came in as part of a URL; otherwise, false **See Also:**

getSession(boolean)

is Requested Session Id From Url

public boolean isRequestedSessionIdFromUrl()

Deprecated. As of Version 2.1 of the Java Servlet API, use isRequestedSessionIdFromURL() instead.

Overview Package Class Tree Deprecated Index Help

PREV CLASS <u>NEXT CLASS</u>
SUMMARY: INNER | <u>FIELD</u> | CONSTR | <u>METHOD</u>

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

javax.servlet.http Class HttpServletRequestWrapper

public class **HttpServletRequestWrapper**

extends <u>ServletRequestWrapper</u> implements <u>HttpServletRequest</u>

Provides a convenient implementation of the HttpServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet. This class implements the Wrapper or Decorator pattern. Methods default to calling through to the wrapped request object.

Since:

v 2.3

See Also:

<u>HttpServletRequest</u>

Constructor Summary

HttpServletRequestWrapper(HttpServletRequest request)

Constructs a request object wrapping the given request.

Method Summary	
java.lang.String	<pre>getAuthType() The default behavior of this method is to return getAuthType() on the wrapped request object.</pre>
java.lang.String	<pre>getContextPath() The default behavior of this method is to return getContextPath() on the wrapped request object.</pre>
	<u>getCookies</u> ()

<u>Cookie</u> []	The default behavior of this method is to return getCookies() on the wrapped request object.
long	<pre>getDateHeader(java.lang.String name) The default behavior of this method is to return getDateHeader(String name) on the wrapped request object.</pre>
java.lang.String	<pre>getHeader(java.lang.String name) The default behavior of this method is to return getHeader(String name) on the wrapped request object.</pre>
java.util.Enumeration	getHeaderNames() The default behavior of this method is to return getHeaderNames() on the wrapped request object.
java.util.Enumeration	<pre>getHeaders(java.lang.String name) The default behavior of this method is to return getHeaders(String name) on the wrapped request object.</pre>
int	<pre>getIntHeader(java.lang.String name) The default behavior of this method is to return getIntHeader(String name) on the wrapped request object.</pre>
java.lang.String	<pre>getMethod() The default behavior of this method is to return getMethod() on the wrapped request object.</pre>
java.lang.String	<pre>getPathInfo() The default behavior of this method is to return getPathInfo() on the wrapped request object.</pre>
java.lang.String	<pre>getPathTranslated() The default behavior of this method is to return getPathTranslated() on the wrapped request object.</pre>
java.lang.String	<pre>getQueryString() The default behavior of this method is to return getQueryString() on the wrapped request object.</pre>
java.lang.String	<pre>getRemoteUser() The default behavior of this method is to return getRemoteUser() on the wrapped request object.</pre>
java.lang.String	<pre>getRequestedSessionId()</pre>

	The default behavior of this method is to return getRequestedSessionId() on the wrapped request object.
java.lang.String	<pre>getRequestURI() The default behavior of this method is to return getRequestURI() on the wrapped request object.</pre>
java.lang.StringBuffer	getRequestURL() The default behavior of this method is to return getRequestURL() on the wrapped request object.
java.lang.String	<pre>getServletPath() The default behavior of this method is to return getServletPath() on the wrapped request object.</pre>
HttpSession	getSession() The default behavior of this method is to return getSession() on the wrapped request object.
HttpSession	getSession(boolean create) The default behavior of this method is to return getSession(boolean create) on the wrapped request object.
java.security.Principal	<pre>getUserPrincipal() The default behavior of this method is to return getUserPrincipal() on the wrapped request object.</pre>
boolean	isRequestedSessionIdFromCookie() The default behavior of this method is to return isRequestedSessionIdFromCookie() on the wrapped request object.
boolean	isRequestedSessionIdFromUrl() The default behavior of this method is to return isRequestedSessionIdFromUrl() on the wrapped request object.
boolean	isRequestedSessionIdFromURL() The default behavior of this method is to return isRequestedSessionIdFromURL() on the wrapped request object.
boolean	isRequestedSessionIdValid() The default behavior of this method is to return isRequestedSessionIdValid() on the wrapped request

	object.
boolean	isUserInRole(java.lang.String role) The default behavior of this method is to return isUserInRole(String role) on the wrapped request object.

Methods inherited from class javax.servlet.ServletRequestWrapper

getAttribute, getAttributeNames, getCharacterEncoding,
getContentLength, getContentType, getInputStream, getLocale,
getLocales, getParameter, getParameterMap, getParameterNames,
getParameterValues, getProtocol, getReader, getRealPath,
getRemoteAddr, getRemoteHost, getRequest, getRequestDispatcher,
getScheme, getServerName, getServerPort, isSecure,
removeAttribute, setAttribute, setCharacterEncoding, setRequest

Methods inherited from class java.lang.Object

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

Constructor Detail

HttpServletRequestWrapper

public HttpServletRequestWrapper(HttpServletRequest request)

Constructs a request object wrapping the given request.

Throws:

java.lang.IllegalArgumentException - if the request is null

Method Detail

getAuthType

public java.lang.String getAuthType()

The default behavior of this method is to return getAuthType() on the wrapped request object.

Specified by:

getAuthType in interface HttpServletRequest

getCookies

```
public Cookie[] getCookies()
```

The default behavior of this method is to return getCookies() on the wrapped request object.

Specified by:

getCookies in interface HttpServletRequest

getDateHeader

```
public long getDateHeader(java.lang.String name)
```

The default behavior of this method is to return getDateHeader(String name) on the wrapped request object.

Specified by:

getDateHeader in interface <u>HttpServletRequest</u>

getHeader

```
public java.lang.String getHeader(java.lang.String name)
```

The default behavior of this method is to return getHeader(String name) on the wrapped request object.

Specified by:

getHeader in interface HttpServletRequest

getHeaders

```
public java.util.Enumeration getHeaders(java.lang.String name)
```

The default behavior of this method is to return getHeaders(String name) on

the wrapped request object.

Specified by:

getHeaders in interface HttpServletRequest

getHeaderNames

```
public java.util.Enumeration getHeaderNames()
```

The default behavior of this method is to return getHeaderNames() on the wrapped request object.

Specified by:

getHeaderNames in interface HttpServletRequest

getIntHeader

```
public int getIntHeader(java.lang.String name)
```

The default behavior of this method is to return getIntHeader(String name) on the wrapped request object.

Specified by:

getIntHeader in interface HttpServletRequest

getMethod

```
public java.lang.String getMethod()
```

The default behavior of this method is to return getMethod() on the wrapped request object.

Specified by:

getMethod in interface HttpServletRequest

getPathInfo

```
public java.lang.String getPathInfo()
```

The default behavior of this method is to return getPathInfo() on the wrapped request object.

Specified by:

getPathInfo in interface HttpServletRequest

getPathTranslated

```
public java.lang.String getPathTranslated()
```

The default behavior of this method is to return getPathTranslated() on the wrapped request object.

Specified by:

getPathTranslated in interface HttpServletRequest

getContextPath

```
public java.lang.String getContextPath()
```

The default behavior of this method is to return getContextPath() on the wrapped request object.

Specified by:

getContextPath in interface HttpServletRequest

getQueryString

```
public java.lang.String getQueryString()
```

The default behavior of this method is to return getQueryString() on the wrapped request object.

Specified by:

getQueryString in interface HttpServletRequest

getRemoteUser

```
public java.lang.String getRemoteUser()
```

The default behavior of this method is to return getRemoteUser() on the wrapped request object.

Specified by:

getRemoteUser in interface HttpServletRequest

isUserInRole

```
public boolean isUserInRole(java.lang.String role)
```

The default behavior of this method is to return is UserInRole(String role) on the wrapped request object.

Specified by:

isUserInRole in interface HttpServletRequest

getUserPrincipal

```
public java.security.Principal getUserPrincipal()
```

The default behavior of this method is to return getUserPrincipal() on the wrapped request object.

Specified by:

getUserPrincipal in interface HttpServletRequest

getRequestedSessionId

```
public java.lang.String getRequestedSessionId()
```

The default behavior of this method is to return getRequestedSessionId() on the wrapped request object.

Specified by:

getRequestedSessionId in interface HttpServletRequest

getRequestURI

```
public java.lang.String getRequestURI()
```

The default behavior of this method is to return getRequestURI() on the wrapped request object.

Specified by:

getRequestURI in interface HttpServletRequest

getRequestURL

```
public java.lang.StringBuffer getRequestURL()
```

The default behavior of this method is to return getRequestURL() on the wrapped request object.

Specified by:

getRequestURL in interface HttpServletRequest

getServletPath

```
public java.lang.String getServletPath()
```

The default behavior of this method is to return getServletPath() on the wrapped request object.

Specified by:

getServletPath in interface HttpServletRequest

getSession

```
public HttpSession getSession(boolean create)
```

The default behavior of this method is to return getSession(boolean create) on the wrapped request object.

Specified by:

getSession in interface HttpServletRequest

getSession

```
public HttpSession getSession()
```

The default behavior of this method is to return getSession() on the wrapped request object.

Specified by:

getSession in interface HttpServletRequest

isRequestedSessionIdValid

public boolean isRequestedSessionIdValid()

The default behavior of this method is to return isRequestedSessionIdValid() on the wrapped request object. **Specified by:**

isRequestedSessionIdValid in interface HttpServletRequest

isRequestedSessionIdFromCookie

public boolean isRequestedSessionIdFromCookie()

The default behavior of this method is to return is Requested Session IdFrom Cookie() on the wrapped request object. **Specified by:**

isRequestedSessionIdFromCookie in interface HttpServletRequest

isRequestedSessionIdFromURL

public boolean isRequestedSessionIdFromURL()

The default behavior of this method is to return is Requested Session IdFrom URL() on the wrapped request object.

Specified by:

isRequestedSessionIdFromURL in interface HttpServletRequest

isRequestedSessionIdFromUrl

public boolean isRequestedSessionIdFromUrl()

The default behavior of this method is to return isRequestedSessionIdFromUrl() on the wrapped request object. **Specified by:**

isRequestedSessionIdFromUrl in interface HttpServletRequest

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES DETAIL: FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpServletResponse

All Known Implementing Classes:

HttpServletResponseWrapper

public abstract interface **HttpServletResponse** extends <u>ServletResponse</u>

Extends the <u>ServletResponse</u> interface to provide HTTP-specific functionality in sending a response. For example, it has methods to access HTTP headers and cookies.

The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).

See Also:

<u>ServletResponse</u>

Field S	Field Summary	
static int	SC ACCEPTED Status code (202) indicating that a request was accepted for processing, but was not completed.	
static int	Status code (502) indicating that the HTTP server received an invalid response from a server it consulted when acting as a proxy or gateway.	
static int	SC BAD REQUEST Status code (400) indicating the request sent by the client was syntactically incorrect.	
static int	completed due to a conflict with the current state of the resource.	
static int	Status code (100) indicating the client can continue.	

static int	Status code (201) indicating the request succeeded and created a new resource on the server.
static int	SC EXPECTATION FAILED Status code (417) indicating that the server could not meet the expectation given in the Expect request header.
static int	SC FORBIDDEN Status code (403) indicating the server understood the request but refused to fulfill it.
static int	SC GATEWAY TIMEOUT Status code (504) indicating that the server did not receive a timely response from the upstream server while acting as a gateway or proxy.
static int	Status code (410) indicating that the resource is no longer available at the server and no forwarding address is known.
static int	Status code (505) indicating that the server does not support or refuses to support the HTTP protocol version that was used in the request message.
static int	SC INTERNAL SERVER ERROR Status code (500) indicating an error inside the HTTP server which prevented it from fulfilling the request.
static int	SC LENGTH REQUIRED Status code (411) indicating that the request cannot be handled without a defined Content-Length.
static int	SC METHOD NOT ALLOWED Status code (405) indicating that the method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
static int	Status code (301) indicating that the resource has permanently moved to a new location, and that future references should use a new URI with their requests.
	Status code (302) indicating that the resource has temporarily

static int	moved to another location, but that future references should still use the original URI to access the resource.
static int	Status code (300) indicating that the requested resource corresponds to any one of a set of representations, each with its own specific location.
static int	Status code (204) indicating that the request succeeded but that there was no new information to return.
static int	SC NON AUTHORITATIVE INFORMATION Status code (203) indicating that the meta information presented by the client did not originate from the server.
static int	Status code (406) indicating that the resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headerssent in the request.
static int	SC NOT FOUND Status code (404) indicating that the requested resource is not available.
static int	SC NOT IMPLEMENTED Status code (501) indicating the HTTP server does not support the functionality needed to fulfill the request.
static int	Status code (304) indicating that a conditional GET operation found that the resource was available and not modified.
static int	Status code (200) indicating the request succeeded normally.
static int	SC PARTIAL CONTENT Status code (206) indicating that the server has fulfilled the partial GET request for the resource.
static int	SC PAYMENT REQUIRED Status code (402) reserved for future use.
	SC PRECONDITION FAILED Status code (412) indicating that the precondition given in one

static int	or more of the request-header fields evaluated to false when it was tested on the server.
static int	SC PROXY AUTHENTICATION REQUIRED Status code (407) indicating that the client MUST first authenticate itself with the proxy.
static int	Status code (413) indicating that the server is refusing to process the request because the request entity is larger than the server is willing or able to process.
static int	sc request TIMEOUT Status code (408) indicating that the client did not produce a requestwithin the time that the server was prepared to wait.
static int	Status code (414) indicating that the server is refusing to service the request because the <i>Request-URI</i> is longer than the server is willing to interpret.
static int	SC REQUESTED RANGE NOT SATISFIABLE Status code (416) indicating that the server cannot serve the requested byte range.
static int	SC RESET CONTENT Status code (205) indicating that the agent SHOULD reset the document view which caused the request to be sent.
static int	SC SEE OTHER Status code (303) indicating that the response to the request can be found under a different URI.
static int	SC SERVICE UNAVAILABLE Status code (503) indicating that the HTTP server is temporarily overloaded, and unable to handle the request.
static int	SC SWITCHING PROTOCOLS Status code (101) indicating the server is switching protocols according to Upgrade header.
static int	SC TEMPORARY REDIRECT Status code (307) indicating that the requested resource resides temporarily under a different URI.
	SC UNAUTHORIZED

static int	Status code (401) indicating that the request requires HTTP authentication.
static int	Status code (415) indicating that the server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.
static int	Status code (305) indicating that the requested resource <i>MUST</i> be accessed through the proxy given by the <i>Location</i> field.

Method Summary		
void	addCookie (Cookie cookie) Adds the specified cookie to the response.	
void	addDateHeader(java.lang.String name, long date) Adds a response header with the given name and date-value.	
void	<pre>addHeader(java.lang.String name, java.lang.String value) Adds a response header with the given name and value.</pre>	
void	<pre>addIntHeader(java.lang.String name, int value) Adds a response header with the given name and integer value.</pre>	
boolean	containsHeader(java.lang.String name) Returns a boolean indicating whether the named response header has already been set.	
java.lang.String	encodeRedirectUrl(java.lang.String url) Deprecated. As of version 2.1, use encodeRedirectURL(String url) instead	
java.lang.String	encodeRedirectURL(java.lang.String url) Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged.	
java.lang.String	encodeUrl(java.lang.String url) Deprecated. As of version 2.1, use encodeURL(String	

	url) instead
java.lang.String	encodeURL(java.lang.String url) Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged.
void	sendError(int sc) Sends an error response to the client using the specified status code and clearing the buffer.
void	sendError(int sc, java.lang.String msg) Sends an error response to the client using the specified status clearing the buffer.
void	sendRedirect(java.lang.String location) Sends a temporary redirect response to the client using the specified redirect location URL.
void	<pre>setDateHeader(java.lang.String name, long date) Sets a response header with the given name and date- value.</pre>
void	<pre>setHeader(java.lang.String name, java.lang.String value) Sets a response header with the given name and value.</pre>
void	<pre>setIntHeader(java.lang.String name, int value) Sets a response header with the given name and integer value.</pre>
void	setStatus(int sc) Sets the status code for this response.
void	setStatus(int sc, java.lang.String sm) Deprecated. As of version 2.1, due to ambiguous meaning of the message parameter. To set a status code use setStatus(int), to send an error with a description use sendError(int, String). Sets the status code and message for this response.

Methods inherited from interface javax.servlet.ServletResponse

flushBuffer, getBufferSize, getCharacterEncoding, getLocale, getOutputStream, getWriter, isCommitted, reset, resetBuffer, setBufferSize, setContentLength, setContentType, setLocale

Field Detail

SC_CONTINUE

public static final int SC_CONTINUE

Status code (100) indicating the client can continue.

SC_SWITCHING_PROTOCOLS

public static final int SC_SWITCHING_PROTOCOLS

Status code (101) indicating the server is switching protocols according to Upgrade header.

SC_OK

public static final int SC_OK

Status code (200) indicating the request succeeded normally.

SC_CREATED

public static final int SC_CREATED

Status code (201) indicating the request succeeded and created a new resource on the server.

SC_ACCEPTED

public static final int SC ACCEPTED

Status code (202) indicating that a request was accepted for processing, but was not completed.

SC_NON_AUTHORITATIVE_INFORMATION

public static final int SC_NON_AUTHORITATIVE_INFORMATION

Status code (203) indicating that the meta information presented by the client did not originate from the server.

SC_NO_CONTENT

public static final int SC_NO_CONTENT

Status code (204) indicating that the request succeeded but that there was no new information to return.

SC_RESET_CONTENT

public static final int SC_RESET_CONTENT

Status code (205) indicating that the agent *SHOULD* reset the document view which caused the request to be sent.

SC_PARTIAL_CONTENT

public static final int SC PARTIAL CONTENT

Status code (206) indicating that the server has fulfilled the partial GET request for the resource.

SC_MULTIPLE_CHOICES

public static final int SC_MULTIPLE_CHOICES

Status code (300) indicating that the requested resource corresponds to any one of a set of representations, each with its own specific location.

SC_MOVED_PERMANENTLY

public static final int SC_MOVED_PERMANENTLY

Status code (301) indicating that the resource has permanently moved to a new location, and that future references should use a new URI with their requests.

SC_MOVED_TEMPORARILY

public static final int SC_MOVED_TEMPORARILY

Status code (302) indicating that the resource has temporarily moved to another location, but that future references should still use the original URI to access the resource.

SC_SEE_OTHER

public static final int SC_SEE_OTHER

Status code (303) indicating that the response to the request can be found under a different URI.

SC_NOT_MODIFIED

public static final int SC NOT MODIFIED

Status code (304) indicating that a conditional GET operation found that the resource was available and not modified.

SC_USE_PROXY

public static final int SC_USE_PROXY

Status code (305) indicating that the requested resource *MUST* be accessed through the proxy given by the *Location* field.

SC_TEMPORARY_REDIRECT

public static final int SC_TEMPORARY_REDIRECT

Status code (307) indicating that the requested resource resides temporarily under a different URI. The temporary URI *SHOULD* be given by the *Location* field in the response.

SC_BAD_REQUEST

public static final int SC_BAD_REQUEST

Status code (400) indicating the request sent by the client was syntactically incorrect.

SC_UNAUTHORIZED

public static final int SC_UNAUTHORIZED

Status code (401) indicating that the request requires HTTP authentication.

SC_PAYMENT_REQUIRED

public static final int SC_PAYMENT_REQUIRED

Status code (402) reserved for future use.

SC_FORBIDDEN

public static final int SC_FORBIDDEN

Status code (403) indicating the server understood the request but refused to fulfill it.

SC_NOT_FOUND

public static final int SC_NOT_FOUND

Status code (404) indicating that the requested resource is not available.

SC_METHOD_NOT_ALLOWED

public static final int SC_METHOD_NOT_ALLOWED

Status code (405) indicating that the method specified in the *Request-Line* is not allowed for the resource identified by the *Request-URI*.

SC_NOT_ACCEPTABLE

public static final int SC_NOT_ACCEPTABLE

Status code (406) indicating that the resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headerssent in the request.

SC_PROXY_AUTHENTICATION_REQUIRED

public static final int SC_PROXY_AUTHENTICATION_REQUIRED

Status code (407) indicating that the client *MUST* first authenticate itself with the proxy.

SC_REQUEST_TIMEOUT

public static final int SC_REQUEST_TIMEOUT

Status code (408) indicating that the client did not produce a requestwithin the time that the server was prepared to wait.

SC_CONFLICT

public static final int SC_CONFLICT

Status code (409) indicating that the request could not be completed due to a conflict with the current state of the resource.

SC_GONE

public static final int SC_GONE

Status code (410) indicating that the resource is no longer available at the server and no forwarding address is known. This condition *SHOULD* be considered permanent.

SC_LENGTH_REQUIRED

public static final int SC_LENGTH_REQUIRED

Status code (411) indicating that the request cannot be handled without a defined *Content-Length*.

SC_PRECONDITION_FAILED

public static final int SC_PRECONDITION_FAILED

Status code (412) indicating that the precondition given in one or more of the request-header fields evaluated to false when it was tested on the server.

SC_REQUEST_ENTITY_TOO_LARGE

public static final int SC_REQUEST_ENTITY_TOO_LARGE

Status code (413) indicating that the server is refusing to process the request because the request entity is larger than the server is willing or able to

SC_REQUEST_URI_TOO_LONG

public static final int SC_REQUEST_URI_TOO_LONG

Status code (414) indicating that the server is refusing to service the request because the *Request-URI* is longer than the server is willing to interpret.

SC_UNSUPPORTED_MEDIA_TYPE

public static final int SC_UNSUPPORTED_MEDIA_TYPE

Status code (415) indicating that the server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

SC_REQUESTED_RANGE_NOT_SATISFIABLE

public static final int SC_REQUESTED_RANGE_NOT_SATISFIABLE

Status code (416) indicating that the server cannot serve the requested byte range.

SC_EXPECTATION_FAILED

public static final int SC_EXPECTATION_FAILED

Status code (417) indicating that the server could not meet the expectation given in the Expect request header.

SC_INTERNAL_SERVER_ERROR

public static final int SC_INTERNAL_SERVER_ERROR

Status code (500) indicating an error inside the HTTP server which prevented it from fulfilling the request.

SC_NOT_IMPLEMENTED

public static final int SC_NOT_IMPLEMENTED

Status code (501) indicating the HTTP server does not support the functionality needed to fulfill the request.

SC_BAD_GATEWAY

public static final int SC_BAD_GATEWAY

Status code (502) indicating that the HTTP server received an invalid response from a server it consulted when acting as a proxy or gateway.

SC_SERVICE_UNAVAILABLE

public static final int SC_SERVICE_UNAVAILABLE

Status code (503) indicating that the HTTP server is temporarily overloaded, and unable to handle the request.

SC_GATEWAY_TIMEOUT

public static final int SC_GATEWAY_TIMEOUT

Status code (504) indicating that the server did not receive a timely response from the upstream server while acting as a gateway or proxy.

SC_HTTP_VERSION_NOT_SUPPORTED

public static final int SC_HTTP_VERSION_NOT_SUPPORTED

Status code (505) indicating that the server does not support or refuses to support the HTTP protocol version that was used in the request message.

Method Detail

addCookie

public void addCookie(Cookie cookie)

Adds the specified cookie to the response. This method can be called multiple times to set more than one cookie.

Parameters:

cookie - the Cookie to return to the client

containsHeader

public boolean containsHeader(java.lang.String name)

Returns a boolean indicating whether the named response header has already been set.

Parameters:

name - the header name

Returns:

true if the named response header has already been set; false otherwise

encodeURL

public java.lang.String encodeURL(java.lang.String url)

Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged. The implementation of this method includes the logic to determine whether the session ID needs to be encoded in the URL. For example, if the browser supports cookies, or session tracking is turned off, URL encoding is unnecessary.

For robust session tracking, all URLs emitted by a servlet should be run

through this method. Otherwise, URL rewriting cannot be used with browsers which do not support cookies.

Parameters:

url - the url to be encoded.

Returns:

the encoded URL if encoding is needed; the unchanged URL otherwise.

encodeRedirectURL

public java.lang.String encodeRedirectURL(java.lang.String url)

Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged. The implementation of this method includes the logic to determine whether the session ID needs to be encoded in the URL. Because the rules for making this determination can differ from those used to decide whether to encode a normal link, this method is seperate from the encodeURL method.

All URLs sent to the HttpServletResponse.sendRedirect method should be run through this method. Otherwise, URL rewriting cannot be used with browsers which do not support cookies.

Parameters:

url - the url to be encoded.

Returns:

the encoded URL if encoding is needed; the unchanged URL otherwise.

See Also:

sendRedirect(java.lang.String), encodeUrl(java.lang.String)

encodeUrl

public java.lang.String encodeUrl(java.lang.String url)

Deprecated. As of version 2.1, use encodeURL(String url) instead

Parameters:

url - the url to be encoded.

Returns:

the encoded URL if encoding is needed; the unchanged URL otherwise.

encodeRedirectUrl

```
public java.lang.String encodeRedirectUrl(java.lang.String url)
```

Deprecated. As of version 2.1, use encodeRedirectURL(String url) instead **Parameters:**

url - the url to be encoded.

Returns:

the encoded URL if encoding is needed; the unchanged URL otherwise.

sendError

Sends an error response to the client using the specified status clearing the buffer. The server defaults to creating the response to look like an HTML-formatted server error page containing the specified message, setting the content type to "text/html", leaving cookies and other headers unmodified. If an error-page declaration has been made for the web application corresponding to the status code passed in, it will be served back in preference to the suggested msg parameter.

If the response has already been committed, this method throws an IllegalStateException. After using this method, the response should be considered to be committed and should not be written to.

Parameters:

```
sc - the error status codemsg - the descriptive message
```

Throws:

java.io.IOException - If an input or output exception occurs IllegalStateException - If the response was committed

sendError

Sends an error response to the client using the specified status code and clearing the buffer.

If the response has already been committed, this method throws an IllegalStateException. After using this method, the response should be considered to be committed and should not be written to.

Parameters:

sc - the error status code

Throws:

java.io.IOException - If an input or output exception occurs IllegalStateException - If the response was committed before this method call

sendRedirect

Sends a temporary redirect response to the client using the specified redirect location URL. This method can accept relative URLs; the servlet container must convert the relative URL to an absolute URL before sending the response to the client. If the location is relative without a leading '/' the container interprets it as relative to the current request URI. If the location is relative with a leading '/' the container interprets it as relative to the servlet container root.

If the response has already been committed, this method throws an IllegalStateException. After using this method, the response should be

considered to be committed and should not be written to.

Parameters:

location - the redirect location URL

Throws:

java.io.IOException - If an input or output exception occurs IllegalStateException - If the response was committed

setDateHeader

Sets a response header with the given name and date-value. The date is specified in terms of milliseconds since the epoch. If the header had already been set, the new value overwrites the previous one. The containsHeader method can be used to test for the presence of a header before setting its value.

Parameters:

name - the name of the header to set value - the assigned date value

See Also:

containsHeader(java.lang.String),
addDateHeader(java.lang.String, long)

addDateHeader

Adds a response header with the given name and date-value. The date is specified in terms of milliseconds since the epoch. This method allows response headers to have multiple values.

Parameters:

name - the name of the header to set value - the additional date value

See Also:

setDateHeader(java.lang.String, long)

setHeader

Sets a response header with the given name and value. If the header had already been set, the new value overwrites the previous one. The containsHeader method can be used to test for the presence of a header before setting its value.

Parameters:

name - the name of the header value - the header value

See Also:

```
containsHeader(java.lang.String),
addHeader(java.lang.String, java.lang.String)
```

addHeader

Adds a response header with the given name and value. This method allows response headers to have multiple values.

Parameters:

name - the name of the header value - the additional header value

See Also:

setHeader(java.lang.String, java.lang.String)

setIntHeader

Sets a response header with the given name and integer value. If the header had already been set, the new value overwrites the previous one. The containsHeader method can be used to test for the presence of a header

before setting its value.

Parameters:

name - the name of the header value - the assigned integer value

See Also:

containsHeader(java.lang.String),
addIntHeader(java.lang.String, int)

addIntHeader

Adds a response header with the given name and integer value. This method allows response headers to have multiple values.

Parameters:

name - the name of the header value - the assigned integer value

See Also:

setIntHeader(java.lang.String, int)

setStatus

```
public void setStatus(int sc)
```

Sets the status code for this response. This method is used to set the return status code when there is no error (for example, for the status codes SC_OK or SC_MOVED_TEMPORARILY). If there is an error, and the caller wishes to invoke an defined in the web application, the sendError method should be used instead.

The container clears the buffer and sets the Location header, preserving cookies and other headers.

Parameters:

sc - the status code

See Also:

sendError(int, java.lang.String)

setStatus

Deprecated. As of version 2.1, due to ambiguous meaning of the message parameter. To set a status code use setStatus(int), to send an error with a description use sendError(int, String). Sets the status code and message for this response.

Parameters:

sc - the status codesm - the status message

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

javax.servlet.http Class HttpServletResponseWrapper

$public\ class\ \textbf{HttpServletResponseWrapper}$

extends <u>ServletResponseWrapper</u> implements <u>HttpServletResponse</u>

Provides a convenient implementation of the HttpServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet. This class implements the Wrapper or Decorator pattern. Methods default to calling through to the wrapped response object.

Since:

v 2.3

See Also:

HttpServletResponse

Constructor Summary

HttpServletResponseWrapper(HttpServletResponse response)Constructs a response adaptor wrapping the given response.

Method Summary		
void	addCookie(Cookie cookie) The default behavior of this method is to call addCookie(Cookie cookie) on the wrapped response object.	
void	addDateHeader(java.lang.String name, long date) The default behavior of this method is to call addDateHeader(String name, long date) on the wrapped response object.	
	addHeader(java.lang.String name,	

void	java.lang.String value) The default behavior of this method is to return addHeader(String name, String value) on the wrapped response object.
void	addIntHeader(java.lang.String name, int value) The default behavior of this method is to call addIntHeader(String name, int value) on the wrapped response object.
boolean	<pre>containsHeader(java.lang.String name) The default behavior of this method is to call containsHeader(String name) on the wrapped response object.</pre>
java.lang.String	<pre>encodeRedirectUrl(java.lang.String url) The default behavior of this method is to return encodeRedirectUrl(String url) on the wrapped response object.</pre>
java.lang.String	<pre>encodeRedirectURL(java.lang.String url) The default behavior of this method is to return encodeRedirectURL(String url) on the wrapped response object.</pre>
java.lang.String	encodeUrl(java.lang.String url) The default behavior of this method is to call encodeUrl(String url) on the wrapped response object.
java.lang.String	encodeURL(java.lang.String url) The default behavior of this method is to call encodeURL(String url) on the wrapped response object.
void	sendError(int sc) The default behavior of this method is to call sendError(int sc) on the wrapped response object.
void	<pre>sendError(int sc, java.lang.String msg) The default behavior of this method is to call sendError(int sc, String msg) on the wrapped response object.</pre>
void	sendRedirect(java.lang.String location) The default behavior of this method is to return sendRedirect(String location) on the wrapped response object.
void	setDateHeader(java.lang.String name, long date) The default behavior of this method is to call setDateHeader(String name, long date) on the wrapped

	response object.
void	<pre>setHeader(java.lang.String name, java.lang.String value) The default behavior of this method is to return setHeader(String name, String value) on the wrapped response object.</pre>
void	<pre>setIntHeader(java.lang.String name, int value) The default behavior of this method is to call setIntHeader(String name, int value) on the wrapped response object.</pre>
void	setStatus(int sc) The default behavior of this method is to call setStatus(int sc) on the wrapped response object.
void	setStatus(int sc, java.lang.String sm) The default behavior of this method is to call setStatus(int sc, String sm) on the wrapped response object.

Methods inherited from class javax.servlet.ServletResponseWrapper

flushBuffer, getBufferSize, getCharacterEncoding, getLocale, getOutputStream, getResponse, getWriter, isCommitted, reset, resetBuffer, setBufferSize, setContentLength, setContentType, setLocale, setResponse

Methods inherited from class java.lang.Object

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

Constructor Detail

Http Servlet Response Wrapper

public HttpServletResponseWrapper(HttpServletResponse response)

Constructs a response adaptor wrapping the given response.

Throws:

java.lang.IllegalArgumentException - if the response is null

Method Detail

addCookie

public void addCookie(Cookie cookie)

The default behavior of this method is to call addCookie(Cookie cookie) on the wrapped response object.

Specified by:

addCookie in interface HttpServletResponse

containsHeader

public boolean containsHeader(java.lang.String name)

The default behavior of this method is to call containsHeader(String name) on the wrapped response object.

Specified by:

containsHeader in interface HttpServletResponse

encodeURL

public java.lang.String encodeURL(java.lang.String url)

The default behavior of this method is to call encodeURL(String url) on the wrapped response object.

Specified by:

encodeURL in interface HttpServletResponse

encodeRedirectURL

 $public java.lang. String \ \textbf{encodeRedirectURL} (java.lang. String \ url)$

The default behavior of this method is to return encodeRedirectURL(String url) on the wrapped response object.

Specified by:

encodeUrl

```
public java.lang.String encodeUrl(java.lang.String url)
```

The default behavior of this method is to call encodeUrl(String url) on the wrapped response object.

Specified by:

encodeUrl in interface HttpServletResponse

encodeRedirectUrl

```
public java.lang.String encodeRedirectUrl(java.lang.String url)
```

The default behavior of this method is to return encodeRedirectUrl(String url) on the wrapped response object.

Specified by:

encodeRedirectUrl in interface HttpServletResponse

sendError

The default behavior of this method is to call sendError(int sc, String msg) on the wrapped response object.

Specified by:

sendError in interface HttpServletResponse

sendError

The default behavior of this method is to call sendError(int sc) on the wrapped response object.

Specified by:

sendError in interface HttpServletResponse

sendRedirect

The default behavior of this method is to return sendRedirect(String location) on the wrapped response object.

Specified by:

sendRedirect in interface HttpServletResponse

setDateHeader

The default behavior of this method is to call setDateHeader(String name, long date) on the wrapped response object.

Specified by:

<u>setDateHeader</u> in interface <u>HttpServletResponse</u>

addDateHeader

The default behavior of this method is to call addDateHeader(String name, long date) on the wrapped response object.

Specified by:

addDateHeader in interface HttpServletResponse

setHeader

The default behavior of this method is to return setHeader(String name, String value) on the wrapped response object.

Specified by:

setHeader in interface HttpServletResponse

addHeader

The default behavior of this method is to return addHeader(String name, String value) on the wrapped response object.

Specified by:

addHeader in interface HttpServletResponse

setIntHeader

The default behavior of this method is to call setIntHeader(String name, int value) on the wrapped response object.

Specified by:

<u>setIntHeader</u> in interface <u>HttpServletResponse</u>

addIntHeader

The default behavior of this method is to call addIntHeader(String name, int value) on the wrapped response object.

Specified by:

addIntHeader in interface HttpServletResponse

setStatus

```
public void setStatus(int sc)
```

The default behavior of this method is to call setStatus(int sc) on the wrapped response object.

Specified by:

setStatus in interface HttpServletResponse

setStatus

The default behavior of this method is to call setStatus(int sc, String sm) on the wrapped response object.

Specified by:

setStatus in interface HttpServletResponse

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpSession

public abstract interface HttpSession

Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user. A session usually corresponds to one user, who may visit a site many times. The server can maintain a session in many ways such as using cookies or rewriting URLs.

This interface allows servlets to

- View and manipulate information about a session, such as the session identifier, creation time, and last accessed time
- Bind objects to sessions, allowing user information to persist across multiple user connections

When an application stores an object in or removes an object from a session, the session checks whether the object implements httpSessionBindingListener. If it does, the servlet notifies the object that it has been bound to or unbound from the session. Notifications are sent after the binding methods complete. For session that are invalidated or expire, notifications are sent after the session has been invalidated or expired.

When container migrates a session between VMs in a distributed container setting, all session atributes implementing the httpSessionActivationListener interface are notified.

A servlet should be able to handle cases in which the client does not choose to join a session, such as when cookies are intentionally turned off. Until the client joins the session, is New returns true. If the client chooses not to join the session, getSession will return a different session on each request, and is New will always return true.

Session information is scoped only to the current web application (ServletContext), so information stored in one context will not be directly visible in another.

See Also:

HttpSessionBindingListener, HttpSessionContext

Method Summary		
java.lang.Object	<pre>getAttribute(java.lang.String name) Returns the object bound with the specified name in this session, or null if no object is bound under the name.</pre>	
java.util.Enumeration	<pre>getAttributeNames() Returns an Enumeration of String objects containing the names of all the objects bound to this session.</pre>	
long	getCreationTime() Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.	
java.lang.String	<pre>getId() Returns a string containing the unique identifier assigned to this session.</pre>	
	getLastAccessedTime() Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container recieved the request.	
int	getMaxInactiveInterval() Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.	
<u>ServletContext</u>	getServletContext() Returns the ServletContext to which this session belongs.	
	<pre>getSessionContext()</pre>	

<u>HttpSessionContext</u>	Deprecated. As of Version 2.1, this method is deprecated and has no replacement. It will be removed in a future version of the Java Servlet API.
java.lang.Object	<pre>getValue(java.lang.String name) Deprecated. As of Version 2.2, this method is replaced by getAttribute(java.lang.String).</pre>
java.lang.String[]	<pre>getValueNames() Deprecated. As of Version 2.2, this method is replaced by getAttributeNames()</pre>
void	invalidate() Invalidates this session then unbinds any objects bound to it.
boolean	isNew() Returns true if the client does not yet know about the session or if the client chooses not to join the session.
void	<pre>putValue(java.lang.String name, java.lang.Object value) Deprecated. As of Version 2.2, this method is replaced by setAttribute(java.lang.String, java.lang.Object)</pre>
void	removeAttribute(java.lang.String name) Removes the object bound with the specified name from this session.
void	<pre>removeValue(java.lang.String name) Deprecated. As of Version 2.2, this method is replaced by removeAttribute(java.lang.String)</pre>
void	setAttribute(java.lang.String name, java.lang.Object value) Binds an object to this session, using the name specified.
void	setMaxInactiveInterval(int interval) Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

Method Detail

getCreationTime

```
public long getCreationTime()
```

Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.

Returns:

a long specifying when this session was created, expressed in milliseconds since $1/1/1970~\mathrm{GMT}$

Throws:

IllegalStateException - if this method is called on an invalidated session

getId

```
public java.lang.String getId()
```

Returns a string containing the unique identifier assigned to this session. The identifier is assigned by the servlet container and is implementation dependent.

Returns:

a string specifying the identifier assigned to this session

getLastAccessedTime

```
public long getLastAccessedTime()
```

Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container recieved the request.

Actions that your application takes, such as getting or setting a value associated with the session, do not affect the access time.

Returns:

a long representing the last time the client sent a request associated with this session, expressed in milliseconds since 1/1/1970 GMT

getServletContext

```
public <u>ServletContext</u> getServletContext()
```

Returns the ServletContext to which this session belongs.

Returns:

The ServletContext object for the web application

Since:

2.3

setMaxInactiveInterval

```
public void setMaxInactiveInterval(int interval)
```

Specifies the time, in seconds, between client requests before the servlet container will invalidate this session. A negative time indicates the session should never timeout.

Parameters:

interval - An integer specifying the number of seconds

getMaxInactiveInterval

```
public int getMaxInactiveInterval()
```

Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses. After this interval, the servlet container will invalidate the session. The maximum time interval can be set with the setMaxInactiveInterval method. A negative time indicates the session should never timeout.

Returns:

an integer specifying the number of seconds this session remains open between client requests

See Also:

getSessionContext

```
public HttpSessionContext getSessionContext()
```

Deprecated. As of Version 2.1, this method is deprecated and has no replacement. It will be removed in a future version of the Java Servlet API.

getAttribute

```
public java.lang.Object getAttribute(java.lang.String name)
```

Returns the object bound with the specified name in this session, or null if no object is bound under the name.

Parameters:

name - a string specifying the name of the object

Returns:

the object with the specified name

Throws:

IllegalStateException - if this method is called on an invalidated session

getValue

```
public java.lang.Object getValue(java.lang.String name)
```

Deprecated. As of Version 2.2, this method is replaced by getAttribute(java.lang.String).

Parameters:

name - a string specifying the name of the object

Returns:

the object with the specified name

Throws:

IllegalStateException - if this method is called on an invalidated session

getAttributeNames

```
public java.util.Enumeration getAttributeNames()
```

Returns an Enumeration of String objects containing the names of all the objects bound to this session.

Returns:

an Enumeration of String objects specifying the names of all the objects bound to this session

Throws:

IllegalStateException - if this method is called on an invalidated session

getValueNames

```
public java.lang.String[] getValueNames()
```

Deprecated. As of Version 2.2, this method is replaced by getAttributeNames()

Returns:

an array of String objects specifying the names of all the objects bound to this session

Throws:

IllegalStateException - if this method is called on an invalidated session

setAttribute

Binds an object to this session, using the name specified. If an object of the same name is already bound to the session, the object is replaced.

After this method executes, and if the new object implements
HttpSessionBindingListener, the container calls
HttpSessionBindingListener.valueBound. The container then notifies

any HttpSessionAttributeListeners in the web application.

If an object was already bound to this session of this name that implements HttpSessionBindingListener, its HttpSessionBindingListener.valueUnbound method is called.

If the value passed in is null, this has the same effect as calling removeAttribute().

Parameters:

name - the name to which the object is bound; cannot be null value - the object to be bound

Throws:

IllegalStateException - if this method is called on an invalidated session

putValue

Deprecated. As of Version 2.2, this method is replaced by setAttribute(java.lang.String, java.lang.Object)

Parameters:

name - the name to which the object is bound; cannot be null value - the object to be bound; cannot be null

Throws:

IllegalStateException - if this method is called on an invalidated session

removeAttribute

```
public void removeAttribute(java.lang.String name)
```

Removes the object bound with the specified name from this session. If the session does not have an object bound with the specified name, this method does nothing.

After this method executes, and if the object implements
HttpSessionBindingListener, the container calls
HttpSessionBindingListener.valueUnbound. The container then notifies
any HttpSessionAttributeListeners in the web application.

Parameters:

name - the name of the object to remove from this session

Throws:

IllegalStateException - if this method is called on an invalidated session

removeValue

public void removeValue(java.lang.String name)

Deprecated. As of Version 2.2, this method is replaced by removeAttribute(java.lang.String)

Parameters:

name - the name of the object to remove from this session

Throws:

IllegalStateException - if this method is called on an invalidated session

invalidate

```
public void invalidate()
```

Invalidates this session then unbinds any objects bound to it.

Throws:

IllegalStateException - if this method is called on an already invalidated session

isNew

```
public boolean isNew()
```

Returns true if the client does not yet know about the session or if the

client chooses not to join the session. For example, if the server used only cookie-based sessions, and the client had disabled the use of cookies, then a session would be new on each request.

Returns:

true if the server has created a session, but the client has not yet joined

Throws:

IllegalStateException - if this method is called on an already invalidated session

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpSessionActivationListener

public abstract interface **HttpSessionActivationListener** extends java.util.EventListener

Objects that are bound to a session may listen to container events notifying them that sessions will be passivated and that session will be activated. A container that migrates session between VMs or persists sessions is required to notify all attributes bound to sessions implementing HttpSessionActivationListener.

Since:

2.3

Method Summary	
void	sessionDidActivate(HttpSessionEvent se) Notification that the session has just been activated.
void	sessionWillPassivate(HttpSessionEvent se) Notification that the session is about to be passivated.

Method Detail

sessionWillPassivate

public void sessionWillPassivate(HttpSessionEvent se)

Notification that the session is about to be passivated.

sessionDidActivate

public void sessionDidActivate(HttpSessionEvent se)

Notification that the session has just been activated.

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpSessionAttributeListener

public abstract interface **HttpSessionAttributeListener** extends java.util.EventListener

This listener interface can be implemented in order to get notifications of changes to the attribute lists of sessions within this web application.

Since:

v2.3

Method Summary	
void	<pre>attributeAdded(HttpSessionBindingEvent se) Notification that an attribute has been added to a session.</pre>
void	<pre>attributeRemoved(HttpSessionBindingEvent se) Notification that an attribute has been removed from a session.</pre>
void	<pre>attributeReplaced(HttpSessionBindingEvent se) Notification that an attribute has been replaced in a session.</pre>

Method Detail

attributeAdded

public void attributeAdded(HttpSessionBindingEvent se)

Notification that an attribute has been added to a session. Called after the attribute is added.

attributeRemoved

public void attributeRemoved(HttpSessionBindingEvent se)

Notification that an attribute has been removed from a session. Called after the attribute is removed.

attributeReplaced

public void attributeReplaced(HttpSessionBindingEvent se)

Notification that an attribute has been replaced in a session. Called after the attribute is replaced.

Overview Package Class Tree Deprecated Index Help

SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Class HttpSessionBindingEvent

public class **HttpSessionBindingEvent** extends <u>HttpSessionEvent</u>

Events of this type are either sent to an object that implements <u>HttpSessionBindingListener</u> when it is bound or unbound from a session, or to a <u>HttpSessionAttributeListener</u> that has been configured in the deployment descriptor when any attribute is bound, unbound or replaced in a session.

The session binds the object by a call to HttpSession.setAttribute and unbinds the object by a call to HttpSession.removeAttribute.

See Also:

HttpSession, HttpSessionBindingListener,
HttpSessionAttributeListener, Serialized Form

Fields inherited from class java.util.EventObject

source

Constructor Summary

```
HttpSessionBindingEvent(HttpSession session,
java.lang.String name)
```

Constructs an event that notifies an object that it has been bound to or unbound from a session.

```
HttpSessionBindingEvent(HttpSession session,
java.lang.String name, java.lang.Object value)
```

Constructs an event that notifies an object that it has been bound to or unbound from a session.

Method Summary	
java.lang.String	getName() Returns the name with which the attribute is bound to or unbound from the session.
HttpSession	getSession() Return the session that changed.
java.lang.Object	getValue() Returns the value of the attribute that has been added, removed or replaced.

Methods inherited from class java.util.EventObject	
getSource, toString	

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

Constructor Detail

HttpSessionBindingEvent

Constructs an event that notifies an object that it has been bound to or unbound from a session. To receive the event, the object must implement httpSessionBindingListener.

Parameters:

session - the session to which the object is bound or unbound name - the name with which the object is bound or unbound

See Also:

getName(), getSession()

HttpSessionBindingEvent

Constructs an event that notifies an object that it has been bound to or unbound from a session. To receive the event, the object must implement httpSessionBindingListener.

Parameters:

session - the session to which the object is bound or unbound name - the name with which the object is bound or unbound

See Also:

getName(), getSession()

Method Detail

getSession

```
public HttpSession getSession()
```

Return the session that changed.

Overrides:

getSession in class HttpSessionEvent

getName

```
public java.lang.String getName()
```

Returns the name with which the attribute is bound to or unbound from the session.

Returns:

a string specifying the name with which the object is bound to or unbound from the session

getValue

public java.lang.Object getValue()

Returns the value of the attribute that has been added, removed or replaced. If the attribute was added (or bound), this is the value of the attribute. If the attribute was removed (or unbound), this is the value of the removed attribute. If the attribute was replaced, this is the old value of the attribute. **Since:**

2.3

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpSessionBindingListener

public abstract interface **HttpSessionBindingListener** extends java.util.EventListener

Causes an object to be notified when it is bound to or unbound from a session. The object is notified by an httpSessionBindingEvent object. This may be as a result of a servlet programmer explicitly unbinding an attribute from a session, due to a session being invalidated, or due to a session timing out.

See Also:

HttpSession, HttpSessionBindingEvent

Me	Method Summary	
void	valueBound (HttpSessionBindingEvent event) Notifies the object that it is being bound to a session and identifies the session.	
void	<pre>valueUnbound(HttpSessionBindingEvent event) Notifies the object that it is being unbound from a session and identifies the session.</pre>	

Method Detail

valueBound

public void valueBound(HttpSessionBindingEvent event)

Notifies the object that it is being bound to a session and identifies the session.

Parameters:

event - the event that identifies the session

See Also:

valueUnbound(javax.servlet.http.HttpSessionBindingEvent)

valueUnbound

public void valueUnbound(<u>HttpSessionBindingEvent</u> event)

Notifies the object that it is being unbound from a session and identifies the session.

Parameters:

event - the event that identifies the session

See Also:

valueBound(javax.servlet.http.HttpSessionBindingEvent)

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpSessionContext

Deprecated. As of Java(tm) Servlet API 2.1 for security reasons, with no replacement. This interface will be removed in a future version of this API.

public abstract interface **HttpSessionContext**

See Also:

HttpSession, HttpSessionBindingEvent, HttpSessionBindingListener

Method Summary	
java.util.Enumeration	getIds() Deprecated. As of Java Servlet API 2.1 with no replacement. This method must return an empty Enumeration and will be removed in a future version of this API.
HttpSession	<pre>getSession(java.lang.String sessionId) Deprecated. As of Java Servlet API 2.1 with no replacement. This method must return null and will be removed in a future version of this API.</pre>

Method Detail

getSession

public HttpSession getSession(java.lang.String sessionId)

Deprecated. As of Java Servlet API 2.1 with no replacement. This method must return null and will be removed in a future version of this API.

getIds

public java.util.Enumeration getIds()

Deprecated. As of Java Servlet API 2.1 with no replacement. This method must return an empty Enumeration and will be removed in a future version of this API.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Class HttpSessionEvent

Direct Known Subclasses:

HttpSessionBindingEvent

public class **HttpSessionEvent** extends java.util.EventObject

This is the class representing event notifications for changes to sessions within a web application.

Since:

v2.3

See Also:

Serialized Form

Fields inherited from class java.util.EventObject

source

Constructor Summary

HttpSessionEvent(HttpSession source)

Construct a session event from the given source.

Method Summary

<u>HttpSession</u>

getSession()

Return the session that changed.

Methods inherited from class java.util.EventObject

getSource, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HttpSessionEvent

public HttpSessionEvent(HttpSession source)

Construct a session event from the given source.

Method Detail

getSession

public HttpSession getSession()

Return the session that changed.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Interface HttpSessionListener

public abstract interface **HttpSessionListener** extends java.util.EventListener

Implementations of this interface may are notified of changes to the list of active sessions in a web application. To recieve notification events, the implementation class must be configured in the deployment descriptor for the web application.

Since:

v2.3

See Also:

HttpSessionEvent

Method Summary		
void	sessionCreated(HttpSessionEvent se) Notification that a session was created.	
void	sessionDestroyed(HttpSessionEvent se) Notification that a session was invalidated.	

Method Detail

sessionCreated

public void sessionCreated(HttpSessionEvent se)

Notification that a session was created.

Parameters:

se - the notification event

sessionDestroyed

public void sessionDestroyed(HttpSessionEvent se)

Notification that a session was invalidated.

Parameters:

se - the notification event

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet.http Class HttpUtils

Deprecated. As of Java(tm) Servlet API 2.3. These methods were only useful with the default encoding and have been moved to the request interfaces.

public class **HttpUtils** extends java.lang.Object

Constructor Summary

HttpUtils()

Deprecated. Constructs an empty HttpUtils object.

Method Summary			
static java.lang.StringBuffer	<pre>getRequestURL(HttpServletRequest req)</pre>		
static java.util.Hashtable	<pre>parsePostData(int len, ServletInputStream in) Deprecated. Parses data from an HTML form that the client sends to the server using the HTTP POST method and the application/x-www-form- urlencoded MIME type.</pre>		
	<pre>parseQueryString(java.lang.String s) Deprecated. Parses a query string passed from the client to the server and builds a HashTable object with key-value pairs.</pre>		

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll,

Constructor Detail

HttpUtils

public HttpUtils()

Deprecated.

Constructs an empty HttpUtils object.

Method Detail

parseQueryString

public static java.util.Hashtable parseQueryString(java.lang.String

Deprecated.

Parses a query string passed from the client to the server and builds a HashTable object with key-value pairs. The query string should be in the form of a string packaged by the GET or POST method, that is, it should have key-value pairs in the form *key=value*, with each pair separated from the next by a & character.

A key can appear more than once in the query string with different values. However, the key appears only once in the hashtable, with its value being an array of strings containing the multiple values sent by the query string.

The keys and values in the hashtable are stored in their decoded form, so any + characters are converted to spaces, and characters sent in hexadecimal notation (like %xx) are converted to ASCII characters.

Parameters:

s - a string containing the query to be parsed

Returns:

a HashTable object built from the parsed key-value pairs

Throws:

parsePostData

Deprecated.

Parses data from an HTML form that the client sends to the server using the HTTP POST method and the *application/x-www-form-urlencoded* MIME type.

The data sent by the POST method contains key-value pairs. A key can appear more than once in the POST data with different values. However, the key appears only once in the hashtable, with its value being an array of strings containing the multiple values sent by the POST method.

The keys and values in the hashtable are stored in their decoded form, so any + characters are converted to spaces, and characters sent in hexadecimal notation (like %xx) are converted to ASCII characters.

Parameters:

len - an integer specifying the length, in characters, of the
ServletInputStream object that is also passed to this method
in - the ServletInputStream object that contains the data sent from
the client

Returns:

a HashTable object built from the parsed key-value pairs

Throws:

java.lang.IllegalArgumentException - if the data sent by the POST method is invalid

getRequestURL

public static java.lang.StringBuffer getRequestURL(HttpServletReques

Deprecated.

Reconstructs the URL the client used to make the request, using

information in the HttpServletRequest object. The returned URL contains a protocol, server name, port number, and server path, but it does not include query string parameters.

Because this method returns a StringBuffer, not a string, you can modify the URL easily, for example, to append query parameters.

This method is useful for creating redirect messages and for reporting errors.

Parameters:

req - a HttpServletRequest object containing the client's request
Returns:

a StringBuffer object containing the reconstructed URL

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface RequestDispatcher

public abstract interface RequestDispatcher

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server. The servlet container creates the RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or given by a particular name.

This interface is intended to wrap servlets, but a servlet container can create RequestDispatcher objects to wrap any type of resource.

See Also:

ServletContext.getRequestDispatcher(java.lang.String), ServletContext.getNamedDispatcher(java.lang.String), ServletRequest.getRequestDispatcher(java.lang.String)

Method Summary

forward (ServletRequest request, ServletResponse response)

Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

include(ServletRequest request, ServletResponse response)

Includes the content of a resource (servlet, JSP page, HTML file) in the response.

Method Detail

forward

```
java.io.IOException
```

Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server. This method allows one servlet to do preliminary processing of a request and another resource to generate the response.

For a RequestDispatcher obtained via getRequestDispatcher(), the ServletRequest object has its path elements and parameters adjusted to match the path of the target resource.

forward should be called before the response has been committed to the client (before response body output has been flushed). If the response already has been committed, this method throws an IllegalStateException. Uncommitted output in the response buffer is automatically cleared before the forward.

The request and response parameters must be either the same objects as were passed to the calling servlet's service method or be subclasses of the ServletRequestWrapper or ServletResponseWrapper classes that wrap them.

Parameters:

request - a <u>ServletRequest</u> object that represents the request the client makes of the servlet

response - a <u>ServletResponse</u> object that represents the response the servlet returns to the client

Throws:

<u>ServletException</u> - if the target resource throws this exception java.io.IOException - if the target resource throws this exception IllegalStateException - if the response was already committed

include

Includes the content of a resource (servlet, JSP page, HTML file) in the

response. In essence, this method enables programmatic server-side includes.

The <u>ServletResponse</u> object has its path elements and parameters remain unchanged from the caller's. The included servlet cannot change the response status code or set headers; any attempt to make a change is ignored.

The request and response parameters must be either the same objects as were passed to the calling servlet's service method or be subclasses of the ServletResponseWrapper classes that wrap them.

Parameters:

request - a <u>ServletRequest</u> object that contains the client's request response - a <u>ServletResponse</u> object that contains the servlet's response

Throws:

<u>ServletException</u> - if the included resource throws this exception java.io.IOException - if the included resource throws this exception

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface Servlet

All Known Implementing Classes:

GenericServlet

public abstract interface Servlet

Defines methods that all servlets must implement.

A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.

To implement this interface, you can write a generic servlet that extends javax.servlet.GenericServlet or an HTTP servlet that extends javax.servlet.http.HttpServlet.

This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods and are called in the following sequence:

- 1. The servlet is constructed, then initialized with the init method.
- 2. Any calls from clients to the service method are handled.
- 3. The servlet is taken out of service, then destroyed with the destroy method, then garbage collected and finalized.

In addition to the life-cycle methods, this interface provides the getServletConfig method, which the servlet can use to get any startup information, and the getServletInfo method, which allows the servlet to return basic information about itself, such as author, version, and copyright.

See Also:

GenericServlet, HttpServlet

Method Summary

void	destroy() Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
ServletConfig	getServletConfig() Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.
java.lang.String	getServletInfo() Returns information about the servlet, such as author, version, and copyright.
void	<pre>init(ServletConfig config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.</pre>
void	service(ServletRequest req, ServletResponse res) Called by the servlet container to allow the servlet to respond to a request.

Method Detail

init

Called by the servlet container to indicate to a servlet that the servlet is being placed into service.

The servlet container calls the init method exactly once after instantiating the servlet. The init method must complete successfully before the servlet can receive any requests.

The servlet container cannot place the servlet into service if the init method

- Throws a ServletException
- 2. Does not return within a time period defined by the Web server

Parameters:

config - a ServletConfig object containing the servlet's configuration and initialization parameters

Throws:

<u>ServletException</u> - if an exception has occurred that interferes with the servlet's normal operation

See Also:

UnavailableException, getServletConfig()

getServletConfig

```
public ServletConfig getServletConfig()
```

Returns a <u>ServletConfig</u> object, which contains initialization and startup parameters for this servlet. The ServletConfig object returned is the one passed to the init method.

Implementations of this interface are responsible for storing the ServletConfig object so that this method can return it. The GenericServlet class, which implements this interface, already does this.

Returns:

the ServletConfig object that initializes this servlet

See Also:

init(javax.servlet.ServletConfig)

service

Called by the servlet container to allow the servlet to respond to a request.

This method is only called after the servlet's init() method has completed successfully.

The status code of the response always should be set for a servlet that

throws or sends an error.

Servlets typically run inside multithreaded servlet containers that can handle multiple requests concurrently. Developers must be aware to synchronize access to any shared resources such as files, network connections, and as well as the servlet's class and instance variables. More information on multithreaded programming in Java is available in the Java tutorial on multi-threaded programming.

Parameters:

req - the ServletRequest object that contains the client's request res - the ServletResponse object that contains the servlet's response

Throws:

<u>ServletException</u> - if an exception occurs that interferes with the servlet's normal operation java.io.IOException - if an input or output exception occurs

getServletInfo

```
public java.lang.String getServletInfo()
```

Returns information about the servlet, such as author, version, and copyright.

The string that this method returns should be plain text and not markup of any kind (such as HTML, XML, etc.).

Returns:

a String containing servlet information

destroy

```
public void destroy()
```

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service. This method is only called once all threads within the servlet's service method have exited or after a timeout period has passed. After the servlet container calls this method, it will not call the

service method again on this servlet.

This method gives the servlet an opportunity to clean up any resources that are being held (for example, memory, file handles, threads) and make sure that any persistent state is synchronized with the servlet's current state in memory.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface ServletConfig

All Known Implementing Classes:

GenericServlet

public abstract interface ServletConfig

A servlet configuration object used by a servlet container used to pass information to a servlet during initialization.

Method Summary			
java.lang.String	<pre>getInitParameter(java.lang.String name) Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.</pre>		
	getInitParameterNames() Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.		
<u>ServletContext</u>	<pre>getServletContext()</pre>		
java.lang.String	getServletName() Returns the name of this servlet instance.		

Method Detail

getServletName

public java.lang.String getServletName()

Returns the name of this servlet instance. The name may be provided via server administration, assigned in the web application deployment descriptor, or for an unregistered (and thus unnamed) servlet instance it will be the servlet's class name.

Returns:

the name of the servlet instance

getServletContext

```
public ServletContext getServletContext()
```

Returns a reference to the <u>ServletContext</u> in which the caller is executing. **Returns:**

a <u>ServletContext</u> object, used by the caller to interact with its servlet container

See Also:

ServletContext

getInitParameter

```
public java.lang.String getInitParameter(java.lang.String name)
```

Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

Parameters:

name - a String specifying the name of the initialization parameter

Returns:

a String containing the value of the initialization parameter

getInitParameterNames

```
public java.util.Enumeration getInitParameterNames()
```

Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.

Returns:

an Enumeration of String objects containing the names of the servlet's initialization parameters

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface ServletContext

public abstract interface ServletContext

Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.

There is one context per "web application" per Java Virtual Machine. (A "web application" is a collection of servlets and content installed under a specific subset of the server's URL namespace such as /catalog and possibly installed via a .war file.)

In the case of a web application marked "distributed" in its deployment descriptor, there will be one context instance for each virtual machine. In this situation, the context cannot be used as a location to share global information (because the information won't be truly global). Use an external resource like a database instead.

The ServletContext object is contained within the <u>ServletConfig</u> object, which the Web server provides the servlet when the servlet is initialized.

See Also:

Servlet.getServletConfig(), ServletConfig.getServletContext()

Method Summary				
java.lang.Object	<pre>getAttribute(java.lang.String name) Returns the servlet container attribute with the given name, or null if there is no attribute by that name.</pre>			
java.util.Enumeration	getAttributeNames() Returns an Enumeration containing the attribute names available within this servlet context.			
<u>ServletContext</u>	<pre>getContext(java.lang.String uripath) Returns a ServletContext object that corresponds</pre>			

	to a specified URL on the server.
java.lang.String	getInitParameter (java.lang.String name) Returns a String containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.
java.util.Enumeration	getInitParameterNames() Returns the names of the context's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the context has no initialization parameters.
int	<pre>getMajorVersion() Returns the major version of the Java Servlet API that this servlet container supports.</pre>
java.lang.String	<pre>getMimeType(java.lang.String file) Returns the MIME type of the specified file, or null if the MIME type is not known.</pre>
int	getMinorVersion() Returns the minor version of the Servlet API that this servlet container supports.
RequestDispatcher	<pre>getNamedDispatcher(java.lang.String name) Returns a RequestDispatcher object that acts as a wrapper for the named servlet.</pre>
java.lang.String	<pre>getRealPath(java.lang.String path) Returns a String containing the real path for a given virtual path.</pre>
RequestDispatcher	<pre>getRequestDispatcher(java.lang.String path) Returns a RequestDispatcher object that acts as a wrapper for the resource located at the given path.</pre>
java.net.URL	<pre>getResource(java.lang.String path) Returns a URL to the resource that is mapped to a specified path.</pre>
java.io.InputStream	<pre>getResourceAsStream(java.lang.String path) Returns the resource located at the named path as an InputStream object.</pre>
	getResourcePaths(java.lang.String path) Returns a directory-like listing of all the paths to

java.util.Set	resources within the web application whose longest subpath matches the supplied path argument.
java.lang.String	getServerInfo() Returns the name and version of the servlet container on which the servlet is running.
	<pre>getServlet(java.lang.String name) Deprecated. As of Java Servlet API 2.1, with no direct replacement.</pre>
<u>Servlet</u>	This method was originally defined to retrieve a servlet from a ServletContext. In this version, this method always returns null and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.
	In lieu of this method, servlets can share information using the ServletContext class and can perform shared business logic by invoking methods on common non-servlet classes.
java.lang.String	getServletContextName() Returns the name of this web application correponding to this ServletContext as specified in the deployment descriptor for this web application by the display-name element.
	getServletNames() Deprecated. As of Java Servlet API 2.1, with no replacement.
java.util.Enumeration	This method was originally defined to return an Enumeration of all the servlet names known to this context. In this version, this method always returns an empty Enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.
	getServlets() Deprecated. As of Java Servlet API 2.0, with no replacement.

java.util.Enumeration	This method was originally defined to return an Enumeration of all the servlets known to this servlet context. In this version, this method always returns an empty enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.
void	log(java.lang.Exception exception, java.lang.String msg) Deprecated. As of Java Servlet API 2.1, use log(String message, Throwable throwable) instead. This method was originally defined to write an exception's stack trace and an explanatory error message to the servlet log file.
void	<pre>log(java.lang.String msg) Writes the specified message to a servlet log file, usually an event log.</pre>
void	<pre>log(java.lang.String message, java.lang.Throwable throwable) Writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file.</pre>
void	removeAttribute(java.lang.String name) Removes the attribute with the given name from the servlet context.
void	<pre>setAttribute(java.lang.String name, java.lang.Object object) Binds an object to a given attribute name in this servlet context.</pre>

Method Detail

getContext

public ServletContext getContext(java.lang.String uripath)

Returns a ServletContext object that corresponds to a specified URL on the server.

This method allows servlets to gain access to the context for various parts of the server, and as needed obtain RequestDispatcher objects from the context. The given path must be begin with "/", is interpreted relative to the server's document root and is matched against the context roots of other web applications hosted on this container.

In a security conscious environment, the servlet container may return null for a given URL.

Parameters:

uripath - a String specifying the context path of another web application in the container.

Returns:

the ServletContext object that corresponds to the named URL, or null if either none exists or the container wishes to restrict this access.

See Also:

RequestDispatcher

getMajorVersion

```
public int getMajorVersion()
```

Returns the major version of the Java Servlet API that this servlet container supports. All implementations that comply with Version 2.3 must have this method return the integer 2.

Returns:

2

getMinorVersion

```
public int getMinorVersion()
```

Returns the minor version of the Servlet API that this servlet container supports. All implementations that comply with Version 2.3 must have this method return the integer 3.

3

getMimeType

```
public java.lang.String getMimeType(java.lang.String file)
```

Returns the MIME type of the specified file, or null if the MIME type is not known. The MIME type is determined by the configuration of the servlet container, and may be specified in a web application deployment descriptor. Common MIME types are "text/html" and "image/gif".

Parameters:

file - a String specifying the name of a file

Returns:

a String specifying the file's MIME type

getResourcePaths

```
public java.util.Set getResourcePaths(java.lang.String path)
```

Returns a directory-like listing of all the paths to resources within the web application whose longest sub-path matches the supplied path argument. Paths indicating subdirectory paths end with a '/'. The returned paths are all relative to the root of the web application and have a leading '/'. For example, for a web application containing

```
/welcome.html
/catalog/index.html
/catalog/products.html
/catalog/offers/books.html
/catalog/offers/music.html
/customer/login.jsp
/WEB-INF/web.xml
/WEB-INF/classes/com.acme.OrderServlet.class,

getResourcePaths("/") returns {"/welcome.html", "/catalog/", "/customer/", "/WEB-INF/"}
```

getResourcePaths("/catalog/") returns {"/catalog/index.html", "/catalog/products.html", "/catalog/offers/"}.

Parameters:

the - partial path used to match the resources, which must start with a / **Returns:**

a Set containing the directory listing, or null if there are no resources in the web application whose path begins with the supplied path.

Since:

Servlet 2.3

getResource

Returns a URL to the resource that is mapped to a specified path. The path must begin with a "/" and is interpreted as relative to the current context root.

This method allows the servlet container to make a resource available to servlets from any source. Resources can be located on a local or remote file system, in a database, or in a .war file.

The servlet container must implement the URL handlers and URLConnection objects that are necessary to access the resource.

This method returns null if no resource is mapped to the pathname.

Some containers may allow writing to the URL returned by this method using the methods of the URL class.

The resource content is returned directly, so be aware that requesting a .jsp page returns the JSP source code. Use a RequestDispatcher instead to include results of an execution.

This method has a different purpose than java.lang.Class.getResource, which looks up resources based on a class loader. This method does not use class loaders.

Parameters:

path - a String specifying the path to the resource

Returns:

the resource located at the named path, or null if there is no resource at that path

Throws:

java.net.MalformedURLException - if the pathname is not given in the correct form

getResourceAsStream

public java.io.InputStream getResourceAsStream(java.lang.String path

Returns the resource located at the named path as an InputStream object.

The data in the InputStream can be of any type or length. The path must be specified according to the rules given in getResource. This method returns null if no resource exists at the specified path.

Meta-information such as content length and content type that is available via getResource method is lost when using this method.

The servlet container must implement the URL handlers and URLConnection objects necessary to access the resource.

This method is different from java.lang.Class.getResourceAsStream, which uses a class loader. This method allows servlet containers to make a resource available to a servlet from any location, without using a class loader.

Parameters:

name - a String specifying the path to the resource

Returns:

the InputStream returned to the servlet, or null if no resource exists at the specified path

getRequestDispatcher

public RequestDispatcher getRequestDispatcher(java.lang.String path)

Returns a <u>RequestDispatcher</u> object that acts as a wrapper for the resource located at the given path. A RequestDispatcher object can be used to forward a request to the resource or to include the resource in a response. The resource can be dynamic or static.

The pathname must begin with a "/" and is interpreted as relative to the current context root. Use getContext to obtain a RequestDispatcher for resources in foreign contexts. This method returns null if the ServletContext cannot return a RequestDispatcher.

Parameters:

path - a String specifying the pathname to the resource

Returns:

a RequestDispatcher object that acts as a wrapper for the resource at the specified path

See Also:

RequestDispatcher, getContext(java.lang.String)

getNamedDispatcher

public RequestDispatcher getNamedDispatcher(java.lang.String name)

Returns a <u>RequestDispatcher</u> object that acts as a wrapper for the named servlet.

Servlets (and JSP pages also) may be given names via server administration or via a web application deployment descriptor. A servlet instance can determine its name using ServletConfig.getServletName().

This method returns null if the ServletContext cannot return a RequestDispatcher for any reason.

Parameters:

name - a String specifying the name of a servlet to wrap

Returns:

a RequestDispatcher object that acts as a wrapper for the named servlet

See Also:

RequestDispatcher, getContext(java.lang.String),
ServletConfig.getServletName()

getServlet

Deprecated. As of Java Servlet API 2.1, with no direct replacement.

This method was originally defined to retrieve a servlet from a ServletContext. In this version, this method always returns null and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

In lieu of this method, servlets can share information using the ServletContext class and can perform shared business logic by invoking methods on common non-servlet classes.

getServlets

```
public java.util.Enumeration getServlets()
```

Deprecated. As of Java Servlet API 2.0, with no replacement.

This method was originally defined to return an Enumeration of all the servlets known to this servlet context. In this version, this method always returns an empty enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

getServletNames

```
public java.util.Enumeration getServletNames()
```

Deprecated. As of Java Servlet API 2.1, with no replacement.

This method was originally defined to return an Enumeration of all the servlet names known to this context. In this version, this method always returns an empty Enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.

log

```
public void log(java.lang.String msg)
```

Writes the specified message to a servlet log file, usually an event log. The name and type of the servlet log file is specific to the servlet container.

Parameters:

msg - a String specifying the message to be written to the log file

log

Deprecated. As of Java Servlet API 2.1, use <u>log(String message, Throwable throwable)</u> instead.

This method was originally defined to write an exception's stack trace and an explanatory error message to the servlet log file.

log

Writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file. The name and type of the servlet log file is specific to the servlet container, usually an event log.

Parameters:

message - a String that describes the error or exception throwable - the Throwable error or exception

getRealPath

public java.lang.String getRealPath(java.lang.String path)

Returns a String containing the real path for a given virtual path. For example, the path "/index.html" returns the absolute file path on the server's filesystem would be served by a request for

"http://host/contextPath/index.html", where contextPath is the context path of this ServletContext..

The real path returned will be in a form appropriate to the computer and operating system on which the servlet container is running, including the proper path separators. This method returns null if the servlet container cannot translate the virtual path to a real path for any reason (such as when the content is being made available from a .war archive).

Parameters:

path - a String specifying a virtual path

Returns:

a String specifying the real path, or null if the translation cannot be performed

getServerInfo

public java.lang.String getServerInfo()

Returns the name and version of the servlet container on which the servlet is running.

The form of the returned string is *servername/versionnumber*. For example, the JavaServer Web Development Kit may return the string JavaServer Web Dev Kit/1.0.

The servlet container may return other optional information after the primary string in parentheses, for example, JavaServer Web Dev Kit/1.0 (JDK 1.1.6; Windows NT 4.0 x86).

Returns:

a String containing at least the servlet container name and version number

getInitParameter

public java.lang.String getInitParameter(java.lang.String name)

Returns a String containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.

This method can make available configuration information useful to an entire "web application". For example, it can provide a webmaster's email address or the name of a system that holds critical data.

Parameters:

name - a String containing the name of the parameter whose value is requested

Returns:

a String containing at least the servlet container name and version number

See Also:

ServletConfig.getInitParameter(java.lang.String)

getInitParameterNames

public java.util.Enumeration getInitParameterNames()

Returns the names of the context's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the context has no initialization parameters.

Returns:

an Enumeration of String objects containing the names of the context's initialization parameters

See Also:

ServletConfig.getInitParameter(java.lang.String)

getAttribute

```
public java.lang.Object getAttribute(java.lang.String name)
```

Returns the servlet container attribute with the given name, or null if there is no attribute by that name. An attribute allows a servlet container to give the servlet additional information not already provided by this interface. See your server documentation for information about its attributes. A list of supported attributes can be retrieved using getAttributeNames.

The attribute is returned as a java.lang.Object or some subclass. Attribute names should follow the same convention as package names. The Java Servlet API specification reserves names matching java.*, javax.*, and sun.*.

Parameters:

name - a String specifying the name of the attribute

Returns:

an Object containing the value of the attribute, or null if no attribute exists matching the given name

See Also:

getAttributeNames()

getAttributeNames

```
public java.util.Enumeration getAttributeNames()
```

Returns an Enumeration containing the attribute names available within this servlet context. Use the getAttribute(java.lang.String) method with an attribute name to get the value of an attribute.

Returns:

an Enumeration of attribute names

See Also:

getAttribute(java.lang.String)

setAttribute

public void setAttribute(java.lang.String name,

```
java.lang.Object object)
```

Binds an object to a given attribute name in this servlet context. If the name specified is already used for an attribute, this method will replace the attribute with the new to the new attribute.

If listeners are configured on the ServletContext the container notifies them accordingly.

If a null value is passed, the effect is the same as calling removeAttribute().

Attribute names should follow the same convention as package names. The Java Servlet API specification reserves names matching java.*, javax.*, and sun.*.

Parameters:

name - a String specifying the name of the attribute
object - an Object representing the attribute to be bound

removeAttribute

public void removeAttribute(java.lang.String name)

Removes the attribute with the given name from the servlet context. After removal, subsequent calls to getAttribute(java.lang.String) to retrieve the attribute's value will return null.

If listeners are configured on the ServletContext the container notifies them accordingly.

Parameters:

name - a String specifying the name of the attribute to be removed

getServletContextName

public java.lang.String getServletContextName()

Returns the name of this web application correponding to this ServletContext as specified in the deployment descriptor for this web application by the display-name element.

Returns:

The name of the web application or null if no name has been declared in the deployment descriptor.

Since:

Servlet 2.3

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Class ServletContextAttributeEvent

public class ServletContextAttributeEvent extends ServletContextEvent

This is the event class for notifications about changes to the attributes of the servlet context of a web application.

Since:

v2.3

See Also:

<u>ServletContextAttributeListener</u>, <u>Serialized Form</u>

Fields inherited from class java.util.EventObject

source

Constructor Summary

ServletContextAttributeEvent(ServletContext source,

java.lang.String name, java.lang.Object value)

Construct a ServletContextAttributeEvent from the given context for the given attribute name and attribute value.

Method Summary		
java.lang.String	<pre>getName() Return the name of the attribute that changed on the ServletContext.</pre>	
	getValue()	

java.lang.Object

Returns the value of the attribute that has been added removed or replaced.

Methods inherited from class javax.servlet.<u>ServletContextEvent</u>

getServletContext

Methods inherited from class java.util.EventObject

getSource, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ServletContextAttributeEvent

Construct a ServletContextAttributeEvent from the given context for the given attribute name and attribute value.

Method Detail

getName

```
public java.lang.String getName()
```

Return the name of the attribute that changed on the ServletContext.

getValue

```
public java.lang.Object getValue()
```

Returns the value of the attribute that has been added removed or replaced. If the attribute was added, this is the value of the attribute. If the attribute was removed, this is the value of the removed attribute. If the attribute was replaced, this is the old value of the attribute.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface ServletContextAttributeListener

public abstract interface **ServletContextAttributeListener** extends java.util.EventListener

Implementations of this interface recieve notifications of changes to the attribute list on the servlet context of a web application. To recieve notification events, the implementation class must be configured in the deployment descriptor for the web application.

Since:

v 2.3

See Also:

<u>ServletContextAttributeEvent</u>

Method Summary		
void	<pre>attributeAdded(ServletContextAttributeEvent scab) Notification that a new attribute was added to the servlet context.</pre>	
void	attributeRemoved(ServletContextAttributeEvent scab) Notification that an existing attribute has been remved from the servlet context.	
void	attributeReplaced(ServletContextAttributeEvent scab) Notification that an attribute on the servlet context has been replaced.	

Method Detail

attributeAdded

public void attributeAdded(ServletContextAttributeEvent scab)

Notification that a new attribute was added to the servlet context. Called

after the attribute is added.

attributeRemoved

public void attributeRemoved(ServletContextAttributeEvent scab)

Notification that an existing attribute has been remved from the servlet context. Called after the attribute is removed.

attributeReplaced

public void attributeReplaced(ServletContextAttributeEvent scab)

Notification that an attribute on the servlet context has been replaced. Called after the attribute is replaced.

FRAMES NO FRAMES

DETAIL: FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS

SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Class ServletContextEvent

```
java.lang.Object
  +--java.util.EventObject
        +--javax.servlet.ServletContextEvent
```

Direct Known Subclasses:

ServletContextAttributeEvent

public class ServletContextEvent extends java.util.EventObject

This is the event class for notifications about changes to the servlet context of a web application.

Since:

v2.3

See Also:

ServletContextListener, Serialized Form

Fields inherited from class java.util.EventObject

source

Constructor Summary

ServletContextEvent(ServletContext source)

Construct a ServletContextEvent from the given context.

Method Summary

ServletContext getServletContext()

Return the ServletContext that changed.

Methods inherited from class java.util.EventObject

getSource, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ServletContextEvent

public ServletContextEvent(ServletContext source)

Construct a ServletContextEvent from the given context.

Parameters:

source - - the ServletContext that is sending the event.

Method Detail

getServletContext

public ServletContext getServletContext()

Return the ServletContext that changed.

Returns:

the ServletContext that sent the event.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface ServletContextListener

public abstract interface **ServletContextListener** extends java.util.EventListener

Implementations of this interface recieve notifications about changes to the servlet context of the web application they are part of. To recieve notification events, the implementation class must be configured in the deployment descriptor for the web application.

Since:

v2.3

See Also:

<u>ServletContextEvent</u>

Method Summary		
void	<pre>contextDestroyed(ServletContextEvent sce) Notification that the servlet context is about to be shut down.</pre>	
void	<pre>contextInitialized(ServletContextEvent sce) Notification that the web application is ready to process requests.</pre>	

Method Detail

contextInitialized

public void contextInitialized(ServletContextEvent sce)

Notification that the web application is ready to process requests.

contextDestroyed

public void contextDestroyed(ServletContextEvent sce)

Notification that the servlet context is about to be shut down.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS

SUMMARY: INNER | FIELD | CONSTR | $\underline{\text{METHOD}}$

FRAMES NO FRAMES

DETAIL: FIELD | CONSTR | $\underline{\mathsf{METHOD}}$

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

javax.servlet Class ServletException

Direct Known Subclasses:

UnavailableException

```
public class ServletException extends java.lang.Exception
```

Defines a general exception a servlet can throw when it encounters difficulty.

See Also:

Serialized Form

Constructor Summary

<u>ServletException()</u>

Constructs a new servlet exception.

ServletException(java.lang.String message)

Constructs a new servlet exception with the specified message.

```
<u>ServletException</u>(java.lang.String message,
```

java.lang.Throwable rootCause)

Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation, including a description message.

```
<u>ServletException</u>(java.lang.Throwable rootCause)
```

Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation.

Method Summary

java.lang.Throwable

getRootCause()

Returns the exception that caused this servlet exception.

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

Constructor Detail

ServletException

public ServletException()

Constructs a new servlet exception.

ServletException

public ServletException(java.lang.String message)

Constructs a new servlet exception with the specified message. The message can be written to the server log and/or displayed for the user.

Parameters:

message - a String specifying the text of the exception message

ServletException

Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation, including a description message.

Parameters:

message - a String containing the text of the exception message rootCause - the Throwable exception that interfered with the servlet's normal operation, making this servlet exception necessary

ServletException

public ServletException(java.lang.Throwable rootCause)

Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation. The exception's message is based on the localized message of the underlying exception.

This method calls the getLocalizedMessage method on the Throwable exception to get a localized exception message. When subclassing ServletException, this method can be overridden to create an exception message designed for a specific locale.

Parameters:

rootCause - the Throwable exception that interfered with the servlet's normal operation, making the servlet exception necessary

Method Detail

getRootCause

public java.lang.Throwable getRootCause()

Returns the exception that caused this servlet exception.

Returns:

the Throwable that caused this servlet exception

Overview Package Class Tree Deprecated Index Help

PREV CLASS <u>NEXT CLASS</u>

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Class ServletInputStream

public abstract class **ServletInputStream** extends java.io.InputStream

Provides an input stream for reading binary data from a client request, including an efficient readLine method for reading data one line at a time. With some protocols, such as HTTP POST and PUT, a ServletInputStream object can be used to read data sent from the client.

A ServletInputStream object is normally retrieved via the <u>ServletRequest.getInputStream()</u> method.

This is an abstract class that a servlet container implements. Subclasses of this class must implement the java.io.InputStream.read() method.

See Also:

ServletRequest

Constructor Summary

protected

ServletInputStream()

Does nothing, because this is an abstract class.

Method Summary

int

readLine(byte[] b, int off, int len)

Reads the input stream, one line at a time.

Methods inherited from class java.io.InputStream

available, close, mark, markSupported, read, read, read, reset,

skip

Methods inherited from class java.lang.Object

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

Constructor Detail

ServletInputStream

```
protected ServletInputStream()
```

Does nothing, because this is an abstract class.

Method Detail

readLine

Reads the input stream, one line at a time. Starting at an offset, reads bytes into an array, until it reads a certain number of bytes or reaches a newline character, which it reads into the array as well.

This method returns -1 if it reaches the end of the input stream before reading the maximum number of bytes.

Parameters:

b - an array of bytes into which data is readoff - an integer specifying the character at which this method begins reading

len - an integer specifying the maximum number of bytes to read

Returns:

an integer specifying the actual number of bytes read, or -1 if the end

of the stream is reached Throws:

java.io.IOException - if an input or output exception has occurred

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS

SUMMARY: INNER | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Class ServletOutputStream

public abstract class **ServletOutputStream** extends java.io.OutputStream

Provides an output stream for sending binary data to the client. A ServletOutputStream object is normally retrieved via the ServletResponse.getOutputStream() method.

This is an abstract class that the servlet container implements. Subclasses of this class must implement the java.io.OutputStream.write(int) method.

See Also:

<u>ServletResponse</u>

Constructor Summary protected ServletOutputStream() Does nothing, because this is an abstract class.

```
Method Summary

print(boolean b)
Writes a boolean value to the client, with no carriage return-line feed (CRLF) character at the end.

print(char c)
Writes a character to the client, with no carriage return-line feed (CRLF) at the end.

print(double d)
Writes a double value to the client, with no carriage return-line feed (CRLF) at the end.
```

void	<pre>print(float f) Writes a float value to the client, with no carriage return-line feed (CRLF) at the end.</pre>
void	<pre>print(int i) Writes an int to the client, with no carriage return-line feed (CRLF) at the end.</pre>
void	print(long 1) Writes a long value to the client, with no carriage return-line feed (CRLF) at the end.
void	<pre>print(java.lang.String s) Writes a String to the client, without a carriage return-line feed (CRLF) character at the end.</pre>
void	println() Writes a carriage return-line feed (CRLF) to the client.
void	println(boolean b) Writes a boolean value to the client, followed by a carriage returnline feed (CRLF).
void	println(char c) Writes a character to the client, followed by a carriage return-line feed (CRLF).
void	println(double d) Writes a double value to the client, followed by a carriage returnline feed (CRLF).
void	<pre>println(float f) Writes a float value to the client, followed by a carriage return-line feed (CRLF).</pre>
void	println(int i) Writes an int to the client, followed by a carriage return-line feed (CRLF) character.
void	println(long 1) Writes a long value to the client, followed by a carriage return-line feed (CRLF).
void	<pre>println(java.lang.String s) Writes a String to the client, followed by a carriage return-line feed (CRLF).</pre>

Methods inherited from class java.io.OutputStream

close, flush, write, write, write

Methods inherited from class java.lang.Object

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

Constructor Detail

ServletOutputStream

protected ServletOutputStream()

Does nothing, because this is an abstract class.

Method Detail

print

Writes a String to the client, without a carriage return-line feed (CRLF) character at the end.

Parameters:

s - the String

Throws:

java.io.IOException - if an input or output exception occurred

print

Writes a boolean value to the client, with no carriage return-line feed (CRLF) character at the end.

Parameters:

b - the boolean value to send to the client

Throws:

java.io.IOException - if an input or output exception occurred

print

Writes a character to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:

c - the character to send to the client

Throws:

java.io.IOException - if an input or output exception occurred

print

Writes an int to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:

i - the int to send to the client

Throws:

java.io.IOException - if an input or output exception occurred

print

Writes a long value to the client, with no carriage return-line feed (CRLF)

at the end.

Parameters:

1 - the long value to send to the client

Throws:

java.io.IOException - if an input or output exception occurred

print

Writes a float value to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:

f - the float value to send to the client

Throws:

java.io.IOException - if an input or output exception occurred

print

Writes a double value to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:

d - the double value to send to the client

Throws:

java.io.IOException - if an input or output exception occurred

println

Writes a carriage return-line feed (CRLF) to the client.

Throws:

println

Writes a String to the client, followed by a carriage return-line feed (CRLF).

Parameters:

s - the String to write to the client

Throws:

java.io.IOException - if an input or output exception occurred

println

Writes a boolean value to the client, followed by a carriage return-line feed (CRLF).

Parameters:

b - the boolean value to write to the client

Throws:

java.io.IOException - if an input or output exception occurred

println

Writes a character to the client, followed by a carriage return-line feed (CRLF).

Parameters:

c - the character to write to the client

Throws:

java.io.IOException - if an input or output exception occurred

println

Writes an int to the client, followed by a carriage return-line feed (CRLF) character.

Parameters:

i - the int to write to the client

Throws:

java.io.IOException - if an input or output exception occurred

println

Writes a long value to the client, followed by a carriage return-line feed (CRLF).

Parameters:

1 - the long value to write to the client

Throws:

java.io.IOException - if an input or output exception occurred

println

Writes a float value to the client, followed by a carriage return-line feed (CRLF).

Parameters:

f - the float value to write to the client

Throws:

java.io.IOException - if an input or output exception occurred

println

Writes a double value to the client, followed by a carriage return-line feed (CRLF).

Parameters:

d - the double value to write to the client

Throws:

java.io.IOException - if an input or output exception occurred

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface ServletRequest

All Known Subinterfaces:

HttpServletRequest

All Known Implementing Classes:

<u>ServletRequestWrapper</u>

public abstract interface **ServletRequest**

Defines an object to provide client request information to a servlet. The servlet container creates a ServletRequest object and passes it as an argument to the servlet's service method.

A ServletRequest object provides data including parameter name and values, attributes, and an input stream. Interfaces that extend ServletRequest can provide additional protocol-specific data (for example, HTTP data is provided by HttpServletRequest.

See Also:

HttpServletRequest

Method Summary	
java.lang.Object	getAttribute(java.lang.String name)Returns the value of the named attribute as anObject, or null if no attribute of the given name exists.
java.util.Enumeration	getAttributeNames() Returns an Enumeration containing the names of the attributes available to this request.
java.lang.String	getCharacterEncoding() Returns the name of the character encoding used in the body of this request.
int	<pre>getContentLength() Returns the length, in bytes, of the request body and</pre>

	made available by the input stream, or -1 if the length is not known.
java.lang.String	<pre>getContentType() Returns the MIME type of the body of the request, or null if the type is not known.</pre>
<u>ServletInputStream</u>	<pre>getInputStream() Retrieves the body of the request as binary data using a ServletInputStream.</pre>
java.util.Locale	getLocale() Returns the preferred Locale that the client will accept content in, based on the Accept-Language header.
java.util.Enumeration	getLocales() Returns an Enumeration of Locale objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header.
java.lang.String	<pre>getParameter(java.lang.String name) Returns the value of a request parameter as a String, or null if the parameter does not exist.</pre>
java.util.Map	getParameterMap() Returns a java.util.Map of the parameters of this request.
java.util.Enumeration	getParameterNames() Returns an Enumeration of String objects containing the names of the parameters contained in this request.
java.lang.String[]	getParameterValues(java.lang.String name) Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist.
java.lang.String	getProtocol() Returns the name and version of the protocol the request uses in the form protocol/majorVersion.minorVersion, for example, HTTP/1.1.
	<u>getReader</u> ()

java.io.BufferedReader	Retrieves the body of the request as character data using a BufferedReader.
java.lang.String	<pre>getRealPath(java.lang.String path)</pre>
java.lang.String	<pre>getRemoteAddr() Returns the Internet Protocol (IP) address of the client that sent the request.</pre>
java.lang.String	<pre>getRemoteHost() Returns the fully qualified name of the client that sent the request.</pre>
RequestDispatcher	<pre>getRequestDispatcher(java.lang.String path) Returns a RequestDispatcher object that acts as a wrapper for the resource located at the given path.</pre>
java.lang.String	getScheme() Returns the name of the scheme used to make this request, for example, https, or ftp.
java.lang.String	getServerName() Returns the host name of the server that received the request.
int	getServerPort() Returns the port number on which this request was received.
boolean	isSecure() Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.
void	<pre>removeAttribute(java.lang.String name) Removes an attribute from this request.</pre>
void	<pre>setAttribute(java.lang.String name, java.lang.Object o) Stores an attribute in this request.</pre>
void	setCharacterEncoding(java.lang.String env) Overrides the name of the character encoding used in the body of this request.

Method Detail

getAttribute

public java.lang.Object getAttribute(java.lang.String name)

Returns the value of the named attribute as an Object, or null if no attribute of the given name exists.

Attributes can be set two ways. The servlet container may set attributes to make available custom information about a request. For example, for requests made using HTTPS, the attribute

javax.servlet.request.X509Certificate can be used to retrieve information on the certificate of the client. Attributes can also be set programatically using setAttribute(java.lang.String).

<u>java.lang.Object</u>). This allows information to be embedded into a request before a RequestDispatcher call.

Attribute names should follow the same conventions as package names. This specification reserves names matching java.*, javax.*, and sun.*.

Parameters:

name - a String specifying the name of the attribute

Returns:

an Object containing the value of the attribute, or null if the attribute does not exist

getAttributeNames

public java.util.Enumeration getAttributeNames()

Returns an Enumeration containing the names of the attributes available to this request. This method returns an empty Enumeration if the request has no attributes available to it.

Returns:

an Enumeration of strings containing the names of the request's

getCharacterEncoding

```
public java.lang.String getCharacterEncoding()
```

Returns the name of the character encoding used in the body of this request. This method returns null if the request does not specify a character encoding

Returns:

a String containing the name of the character encoding, or null if the request does not specify a character encoding

setCharacterEncoding

Overrides the name of the character encoding used in the body of this request. This method must be called prior to reading request parameters or reading input using getReader().

Parameters:

a - String containing the name of the character encoding.

Throws:

java.io.UnsupportedEncodingException - if this is not a valid encoding

getContentLength

```
public int getContentLength()
```

Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known. For HTTP servlets, same as the value of the CGI variable CONTENT_LENGTH.

Returns:

an integer containing the length of the request body or -1 if the length is not known

getContentType

```
public java.lang.String getContentType()
```

Returns the MIME type of the body of the request, or null if the type is not known. For HTTP servlets, same as the value of the CGI variable CONTENT_TYPE.

Returns:

a String containing the name of the MIME type of the request, or null if the type is not known

getInputStream

Retrieves the body of the request as binary data using a ServletInputStream. Either this method or getReader() may be called to read the body, not both.

Returns:

a <u>ServletInputStream</u> object containing the body of the request

Throws:

IllegalStateException - if the getReader() method has already been called for this request

java.io.IOException - if an input or output exception occurred

getParameter

```
public java.lang.String getParameter(java.lang.String name)
```

Returns the value of a request parameter as a String, or null if the parameter does not exist. Request parameters are extra information sent with the request. For HTTP servlets, parameters are contained in the query string or posted form data.

You should only use this method when you are sure the parameter has only

one value. If the parameter might have more than one value, use getParameterValues(java.lang.String).

If you use this method with a multivalued parameter, the value returned is equal to the first value in the array returned by getParameterValues.

If the parameter data was sent in the request body, such as occurs with an HTTP POST request, then reading the body directly via getInputStream()
or getReader() can interfere with the execution of this method.

Parameters:

name - a String specifying the name of the parameter

Returns:

a String representing the single value of the parameter

See Also:

getParameterValues(java.lang.String)

getParameterNames

public java.util.Enumeration getParameterNames()

Returns an Enumeration of String objects containing the names of the parameters contained in this request. If the request has no parameters, the method returns an empty Enumeration.

Returns:

an Enumeration of String objects, each String containing the name of a request parameter; or an empty Enumeration if the request has no parameters

getParameterValues

public java.lang.String[] getParameterValues(java.lang.String name)

Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist.

If the parameter has a single value, the array has a length of 1.

Parameters:

name - a String containing the name of the parameter whose value is requested

Returns:

an array of String objects containing the parameter's values

See Also:

getParameter(java.lang.String)

getParameterMap

```
public java.util.Map getParameterMap()
```

Returns a java.util.Map of the parameters of this request. Request parameters are extra information sent with the request. For HTTP servlets, parameters are contained in the query string or posted form data.

Returns:

an immutable java.util.Map containing parameter names as keys and parameter values as map values. The keys in the parameter map are of type String. The values in the parameter map are of type String array.

getProtocol

```
public java.lang.String getProtocol()
```

Returns the name and version of the protocol the request uses in the form *protocol/majorVersion.minorVersion*, for example, HTTP/1.1. For HTTP servlets, the value returned is the same as the value of the CGI variable SERVER PROTOCOL.

Returns:

a String containing the protocol name and version number

getScheme

```
public java.lang.String getScheme()
```

Returns the name of the scheme used to make this request, for example,

http, https, or ftp. Different schemes have different rules for constructing URLs, as noted in RFC 1738.

Returns:

a String containing the name of the scheme used to make this request

getServerName

```
public java.lang.String getServerName()
```

Returns the host name of the server that received the request. For HTTP servlets, same as the value of the CGI variable SERVER_NAME.

Returns:

a String containing the name of the server to which the request was sent

getServerPort

```
public int getServerPort()
```

Returns the port number on which this request was received. For HTTP servlets, same as the value of the CGI variable SERVER_PORT.

Returns:

an integer specifying the port number

getReader

Retrieves the body of the request as character data using a BufferedReader. The reader translates the character data according to the character encoding used on the body. Either this method or getInputStream() may be called to read the body, not both.

Returns:

a BufferedReader containing the body of the request

Throws:

java.io.UnsupportedEncodingException - if the character set encoding used is not supported and the text cannot be decoded IllegalStateException - if getInputStream() method has been called on this request

java.io.IOException - if an input or output exception occurred

See Also:

getInputStream()

getRemoteAddr

```
public java.lang.String getRemoteAddr()
```

Returns the Internet Protocol (IP) address of the client that sent the request. For HTTP servlets, same as the value of the CGI variable REMOTE_ADDR. **Returns:**

a String containing the IP address of the client that sent the request

getRemoteHost

```
public java.lang.String getRemoteHost()
```

Returns the fully qualified name of the client that sent the request. If the engine cannot or chooses not to resolve the hostname (to improve performance), this method returns the dotted-string form of the IP address. For HTTP servlets, same as the value of the CGI variable REMOTE_HOST.

Returns:

a String containing the fully qualified name of the client

setAttribute

Stores an attribute in this request. Attributes are reset between requests. This method is most often used in conjunction with RequestDispatcher.

Attribute names should follow the same conventions as package names.

Names beginning with java.*, javax.*, and com.sun.*, are reserved for use by Sun Microsystems.

If the value passed in is null, the effect is the same as calling removeAttribute(java.lang.String).

Parameters:

name - a String specifying the name of the attributeo - the Object to be stored

removeAttribute

```
public void removeAttribute(java.lang.String name)
```

Removes an attribute from this request. This method is not generally needed as attributes only persist as long as the request is being handled.

Attribute names should follow the same conventions as package names. Names beginning with java.*, javax.*, and com.sun.*, are reserved for use by Sun Microsystems.

Parameters:

name - a String specifying the name of the attribute to remove

getLocale

```
public java.util.Locale getLocale()
```

Returns the preferred Locale that the client will accept content in, based on the Accept-Language header. If the client request doesn't provide an Accept-Language header, this method returns the default locale for the server.

Returns:

the preferred Locale for the client

getLocales

```
public java.util.Enumeration getLocales()
```

Returns an Enumeration of Locale objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header. If the client request doesn't provide an Accept-Language header, this method returns an Enumeration containing one Locale, the default locale for the server.

Returns:

an Enumeration of preferred Locale objects for the client

isSecure

```
public boolean isSecure()
```

Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.

Returns:

a boolean indicating if the request was made using a secure channel

getRequestDispatcher

```
public RequestDispatcher getRequestDispatcher(java.lang.String path)
```

Returns a <u>RequestDispatcher</u> object that acts as a wrapper for the resource located at the given path. A RequestDispatcher object can be used to forward a request to the resource or to include the resource in a response. The resource can be dynamic or static.

The pathname specified may be relative, although it cannot extend outside the current servlet context. If the path begins with a "/" it is interpreted as relative to the current context root. This method returns null if the servlet container cannot return a RequestDispatcher.

The difference between this method and ServletContext.getRequestDispatcher(java.lang.String) is that this method can take a relative path.

Parameters:

path - a String specifying the pathname to the resource

Returns:

a RequestDispatcher object that acts as a wrapper for the resource at the specified path

See Also:

RequestDispatcher,
ServletContext.getRequestDispatcher(java.lang.String)

getRealPath

public java.lang.String getRealPath(java.lang.String path)

Deprecated. As of Version 2.1 of the Java Servlet API, use <u>ServletContext.getRealPath(java.lang.String)</u> instead.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Class ServletRequestWrapper

Direct Known Subclasses:

<u>HttpServletRequestWrapper</u>

public class **ServletRequestWrapper** extends java.lang.Object implements <u>ServletRequest</u>

Provides a convenient implementation of the ServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet. This class implements the Wrapper or Decorator pattern. Methods default to calling through to the wrapped request object.

Since:

v 2.3

See Also:

<u>ServletRequest</u>

Constructor Summary

<u>ServletRequestWrapper(ServletRequest</u> request)

Creates a ServletRequest adaptor wrapping the given request object.

Method Summary	
java.lang.Object	<pre>getAttribute(java.lang.String name) The default behavior of this method is to call getAttribute(String name) on the wrapped request object.</pre>
java.util.Enumeration	getAttributeNames() The default behavior of this method is to return getAttributeNames() on the wrapped request object.

E	<u></u>
java.lang.String	<pre>getCharacterEncoding() The default behavior of this method is to return getCharacterEncoding() on the wrapped request object.</pre>
int	getContentLength() The default behavior of this method is to return getContentLength() on the wrapped request object.
java.lang.String	<pre>getContentType() The default behavior of this method is to return getContentType() on the wrapped request object.</pre>
<u>ServletInputStream</u>	getInputStream() The default behavior of this method is to return getInputStream() on the wrapped request object.
java.util.Locale	getLocale() The default behavior of this method is to return getLocale() on the wrapped request object.
java.util.Enumeration	getLocales() The default behavior of this method is to return getLocales() on the wrapped request object.
java.lang.String	<pre>getParameter(java.lang.String name) The default behavior of this method is to return getParameter(String name) on the wrapped request object.</pre>
java.util.Map	getParameterMap() The default behavior of this method is to return getParameterMap() on the wrapped request object.
java.util.Enumeration	getParameterNames() The default behavior of this method is to return getParameterNames() on the wrapped request object.
java.lang.String[]	<pre>getParameterValues(java.lang.String name) The default behavior of this method is to return getParameterValues(String name) on the wrapped request object.</pre>
java.lang.String	<pre>getProtocol() The default behavior of this method is to return getProtocol() on the wrapped request object.</pre>
	<u>getReader</u> ()

java.io.BufferedReader	The default behavior of this method is to return getReader() on the wrapped request object.
java.lang.String	<pre>getRealPath(java.lang.String path) The default behavior of this method is to return getRealPath(String path) on the wrapped request object.</pre>
java.lang.String	<pre>getRemoteAddr() The default behavior of this method is to return getRemoteAddr() on the wrapped request object.</pre>
java.lang.String	<pre>getRemoteHost() The default behavior of this method is to return getRemoteHost() on the wrapped request object.</pre>
<u>ServletRequest</u>	getRequest() Return the wrapped request object.
RequestDispatcher	<pre>getRequestDispatcher(java.lang.String path) The default behavior of this method is to return getRequestDispatcher(String path) on the wrapped request object.</pre>
java.lang.String	<pre>getScheme() The default behavior of this method is to return getScheme() on the wrapped request object.</pre>
java.lang.String	getServerName() The default behavior of this method is to return getServerName() on the wrapped request object.
int	getServerPort() The default behavior of this method is to return getServerPort() on the wrapped request object.
boolean	isSecure() The default behavior of this method is to return isSecure() on the wrapped request object.
void	removeAttribute(java.lang.String name) The default behavior of this method is to call removeAttribute(String name) on the wrapped request object.
void	<pre>setAttribute(java.lang.String name, java.lang.Object o) The default behavior of this method is to return</pre>

	setAttribute(String name, Object o) on the wrapped request object.
void	setCharacterEncoding(java.lang.String enc) The default behavior of this method is to set the character encoding on the wrapped request object.
void	setRequest (ServletRequest request) Sets the request object being wrapped.

Methods inherited from class java.lang.Object

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

Constructor Detail

ServletRequestWrapper

public ServletRequestWrapper(ServletRequest request)

 $Creates\ a\ Servlet Request\ adaptor\ wrapping\ the\ given\ request\ object.$

Throws:

java.lang.IllegalArgumentException - if the request is null

Method Detail

getRequest

public ServletRequest getRequest()

Return the wrapped request object.

setRequest

public void setRequest(ServletRequest request)

Sets the request object being wrapped.

Throws:

java.lang.IllegalArgumentException - if the request is null.

getAttribute

```
public java.lang.Object getAttribute(java.lang.String name)
```

The default behavior of this method is to call getAttribute(String name) on the wrapped request object.

Specified by:

getAttribute in interface ServletRequest

getAttributeNames

```
public java.util.Enumeration getAttributeNames()
```

The default behavior of this method is to return getAttributeNames() on the wrapped request object.

Specified by:

getAttributeNames in interface ServletRequest

getCharacterEncoding

```
public java.lang.String getCharacterEncoding()
```

The default behavior of this method is to return getCharacterEncoding() on the wrapped request object.

Specified by:

getCharacterEncoding in interface ServletRequest

setCharacterEncoding

The default behavior of this method is to set the character encoding on the

wrapped request object.

Specified by:

setCharacterEncoding in interface ServletRequest

getContentLength

```
public int getContentLength()
```

The default behavior of this method is to return getContentLength() on the wrapped request object.

Specified by:

getContentLength in interface ServletRequest

getContentType

```
public java.lang.String getContentType()
```

The default behavior of this method is to return getContentType() on the wrapped request object.

Specified by:

getContentType in interface ServletRequest

getInputStream

The default behavior of this method is to return getInputStream() on the wrapped request object.

Specified by:

getInputStream in interface ServletRequest

getParameter

```
public java.lang.String getParameter(java.lang.String name)
```

The default behavior of this method is to return getParameter(String name) on the wrapped request object.

Specified by:

getParameter in interface ServletRequest

getParameterMap

```
public java.util.Map getParameterMap()
```

The default behavior of this method is to return getParameterMap() on the wrapped request object.

Specified by:

getParameterMap in interface ServletRequest

getParameterNames

```
public java.util.Enumeration getParameterNames()
```

The default behavior of this method is to return getParameterNames() on the wrapped request object.

Specified by:

getParameterNames in interface ServletRequest

getParameterValues

```
public java.lang.String[] getParameterValues(java.lang.String name)
```

The default behavior of this method is to return getParameterValues(String name) on the wrapped request object.

Specified by:

getParameterValues in interface ServletRequest

getProtocol

```
public java.lang.String getProtocol()
```

The default behavior of this method is to return getProtocol() on the wrapped request object.

Specified by:

getProtocol in interface ServletRequest

getScheme

```
public java.lang.String getScheme()
```

The default behavior of this method is to return getScheme() on the wrapped request object.

Specified by:

getScheme in interface ServletRequest

getServerName

```
public java.lang.String getServerName()
```

The default behavior of this method is to return getServerName() on the wrapped request object.

Specified by:

getServerName in interface ServletRequest

getServerPort

```
public int getServerPort()
```

The default behavior of this method is to return getServerPort() on the wrapped request object.

Specified by:

getServerPort in interface ServletRequest

getReader

```
public java.io.BufferedReader getReader()
```

```
throws java.io.IOException
```

The default behavior of this method is to return getReader() on the wrapped request object.

Specified by:

getReader in interface ServletRequest

getRemoteAddr

```
public java.lang.String getRemoteAddr()
```

The default behavior of this method is to return getRemoteAddr() on the wrapped request object.

Specified by:

getRemoteAddr in interface ServletRequest

getRemoteHost

```
public java.lang.String getRemoteHost()
```

The default behavior of this method is to return getRemoteHost() on the wrapped request object.

Specified by:

getRemoteHost in interface ServletRequest

setAttribute

The default behavior of this method is to return setAttribute(String name, Object o) on the wrapped request object.

Specified by:

setAttribute in interface ServletRequest

removeAttribute

```
public void removeAttribute(java.lang.String name)
```

The default behavior of this method is to call removeAttribute(String name) on the wrapped request object.

Specified by:

removeAttribute in interface ServletRequest

getLocale

```
public java.util.Locale getLocale()
```

The default behavior of this method is to return getLocale() on the wrapped request object.

Specified by:

getLocale in interface ServletRequest

getLocales

```
public java.util.Enumeration getLocales()
```

The default behavior of this method is to return getLocales() on the wrapped request object.

Specified by:

getLocales in interface ServletRequest

isSecure

```
public boolean isSecure()
```

The default behavior of this method is to return is Secure() on the wrapped request object.

Specified by:

isSecure in interface ServletRequest

getRequestDispatcher

public RequestDispatcher getRequestDispatcher(java.lang.String path)

The default behavior of this method is to return getRequestDispatcher(String path) on the wrapped request object. **Specified by:**

getRequestDispatcher in interface ServletRequest

getRealPath

public java.lang.String getRealPath(java.lang.String path)

The default behavior of this method is to return getRealPath(String path) on the wrapped request object.

Specified by:

getRealPath in interface ServletRequest

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Interface ServletResponse

All Known Subinterfaces:

HttpServletResponse

All Known Implementing Classes:

<u>ServletResponseWrapper</u>

public abstract interface **ServletResponse**

Defines an object to assist a servlet in sending a response to the client. The servlet container creates a ServletResponse object and passes it as an argument to the servlet's service method.

To send binary data in a MIME body response, use the <u>ServletOutputStream</u> returned by <u>getOutputStream()</u>. To send character data, use the <u>PrintWriter</u> object returned by <u>getWriter()</u>. To mix binary and text data, for example, to create a multipart response, use a <u>ServletOutputStream</u> and manage the character sections manually.

The charset for the MIME body response can be specified with setContentType(java.lang.String). For example, "text/html;
charset=Shift_JIS". The charset can alternately be set using
setLocale(java.util.Locale). If no charset is specified, ISO-8859-1 will be
used. The setContentType or setLocale method must be called before
qetWriter for the charset to affect the construction of the writer.

See the Internet RFCs such as RFC 2045 for more information on MIME. Protocols such as SMTP and HTTP define profiles of MIME, and those standards are still evolving.

See Also:

ServletOutputStream

Method Summary

void	flushBuffer() Forces any content in the buffer to be written to the client.
int	getBufferSize() Returns the actual buffer size used for the response.
java.lang.String	getCharacterEncoding() Returns the name of the charset used for the MIME body sent in this response.
java.util.Locale	getLocale() Returns the locale assigned to the response.
<u>ServletOutputStream</u>	getOutputStream() Returns a <u>ServletOutputStream</u> suitable for writing binary data in the response.
java.io.PrintWriter	<pre>getWriter() Returns a PrintWriter object that can send character text to the client.</pre>
boolean	isCommitted() Returns a boolean indicating if the response has been committed.
void	reset() Clears any data that exists in the buffer as well as the status code and headers.
void	resetBuffer() Clears the content of the underlying buffer in the response without clearing headers or status code.
void	setBufferSize(int size) Sets the preferred buffer size for the body of the response.
void	setContentLength(int len) Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
void	<pre>setContentType(java.lang.String type) Sets the content type of the response being sent to the client.</pre>

void setLocale (java.util.Locale loc)

Sets the locale of the response, setting the headers (including the Content-Type's charset) as appropriate.

Method Detail

getCharacterEncoding

public java.lang.String getCharacterEncoding()

Returns the name of the charset used for the MIME body sent in this response.

If no charset has been assigned, it is implicitly set to ISO-8859-1 (Latin-1).

See RFC 2047 (http://ds.internic.net/rfc/rfc2045.txt) for more information about character encoding and MIME.

Returns:

a String specifying the name of the charset, for example, ISO-8859-1

getOutputStream

Returns a <u>ServletOutputStream</u> suitable for writing binary data in the response. The servlet container does not encode the binary data.

Calling flush() on the ServletOutputStream commits the response. Either this method or getWriter() may be called to write the body, not both.

Returns:

a <u>ServletOutputStream</u> for writing binary data

Throws:

IllegalStateException - if the getWriter method has been called on this response

java.io.IOException - if an input or output exception occurred **See Also:**

getWriter()

getWriter

Returns a PrintWriter object that can send character text to the client. The character encoding used is the one specified in the charset= property of the setContentType(java.lang.String) method, which must be called before calling this method for the charset to take effect.

If necessary, the MIME type of the response is modified to reflect the character encoding used.

Calling flush() on the PrintWriter commits the response.

Either this method or getOutputStream() may be called to write the body, not both.

Returns:

a PrintWriter object that can return character data to the client

Throws:

java.io.UnsupportedEncodingException - if the charset specified in setContentType cannot be used

IllegalStateException - if the getOutputStream method has already been called for this response object

java.io.IOException - if an input or output exception occurred

See Also:

getOutputStream(), setContentType(java.lang.String)

setContentLength

```
public void setContentLength(int len)
```

Sets the length of the content body in the response In HTTP servlets, this

method sets the HTTP Content-Length header.

Parameters:

len - an integer specifying the length of the content being returned to the client; sets the Content-Length header

setContentType

```
public void setContentType(java.lang.String type)
```

Sets the content type of the response being sent to the client. The content type may include the type of character encoding used, for example, text/html; charset=ISO-8859-4.

If obtaining a PrintWriter, this method should be called first.

Parameters:

type - a String specifying the MIME type of the content **See Also:**

getOutputStream(), getWriter()

setBufferSize

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

Sets the preferred buffer size for the body of the response. The servlet container will use a buffer at least as large as the size requested. The actual buffer size used can be found using getBufferSize.

A larger buffer allows more content to be written before anything is actually sent, thus providing the servlet with more time to set appropriate status codes and headers. A smaller buffer decreases server memory load and allows the client to start receiving data more quickly.

This method must be called before any response body content is written; if content has been written, this method throws an IllegalStateException.

Parameters:

size - the preferred buffer size

Throws:

IllegalStateException - if this method is called after content has been written

See Also:

getBufferSize(), flushBuffer(), isCommitted(), reset()

getBufferSize

```
public int getBufferSize()
```

Returns the actual buffer size used for the response. If no buffering is used, this method returns 0.

Returns:

the actual buffer size used

See Also:

setBufferSize(int), flushBuffer(), isCommitted(), reset()

flushBuffer

Forces any content in the buffer to be written to the client. A call to this method automatically commits the response, meaning the status code and headers will be written.

See Also:

setBufferSize(int), getBufferSize(), isCommitted(), reset()

resetBuffer

```
public void resetBuffer()
```

Clears the content of the underlying buffer in the response without clearing headers or status code. If the response has been committed, this method throws an IllegalStateException.

Since:

See Also:

setBufferSize(int), getBufferSize(), isCommitted(), reset()

isCommitted

```
public boolean isCommitted()
```

Returns a boolean indicating if the response has been committed. A committed response has already had its status code and headers written.

Returns:

a boolean indicating if the response has been committed **See Also:**

setBufferSize(int), getBufferSize(), flushBuffer(), reset()

reset

```
public void reset()
```

Clears any data that exists in the buffer as well as the status code and headers. If the response has been committed, this method throws an IllegalStateException.

Throws:

IllegalStateException - if the response has already been committed **See Also:**

setBufferSize(int), getBufferSize(), flushBuffer(),
isCommitted()

setLocale

```
public void setLocale(java.util.Locale loc)
```

Sets the locale of the response, setting the headers (including the Content-Type's charset) as appropriate. This method should be called before a call to getWriter(). By default, the response locale is the default locale for the server.

Parameters:

loc - the locale of the response

See Also:

getLocale()

getLocale

public java.util.Locale getLocale()

Returns the locale assigned to the response.

See Also:

setLocale(java.util.Locale)

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

javax.servlet Class ServletResponseWrapper

Direct Known Subclasses:

HttpServletResponseWrapper

public class **ServletResponseWrapper** extends java.lang.Object implements <u>ServletResponse</u>

Provides a convenient implementation of the ServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet. This class implements the Wrapper or Decorator pattern. Methods default to calling through to the wrapped response object.

Since:

v 2.3

See Also:

<u>ServletResponse</u>

Constructor Summary

<u>ServletResponseWrapper(ServletResponse</u> response)

Creates a ServletResponse adaptor wrapping the given response object.

Method Summary		
void	flushBuffer() The default behavior of this method is to call flushBuffer() on the wrapped response object.	
int	getBufferSize() The default behavior of this method is to return getBufferSize() on the wrapped response object.	

java.lang.String	<pre>getCharacterEncoding() The default behavior of this method is to return getCharacterEncoding() on the wrapped response object.</pre>
java.util.Locale	getLocale() The default behavior of this method is to return getLocale() on the wrapped response object.
ServletOutputStream	<pre>getOutputStream() The default behavior of this method is to return getOutputStream() on the wrapped response object.</pre>
<u>ServletResponse</u>	getResponse() Return the wrapped ServletResponse object.
java.io.PrintWriter	<pre>getWriter() The default behavior of this method is to return getWriter() on the wrapped response object.</pre>
boolean	isCommitted() The default behavior of this method is to return isCommitted() on the wrapped response object.
void	reset() The default behavior of this method is to call reset() on the wrapped response object.
void	resetBuffer() The default behavior of this method is to call resetBuffer() on the wrapped response object.
void	setBufferSize(int size) The default behavior of this method is to call setBufferSize(int size) on the wrapped response object.
void	<pre>setContentLength(int len) The default behavior of this method is to call setContentLength(int len) on the wrapped response object.</pre>
void	<pre>setContentType(java.lang.String type) The default behavior of this method is to call setContentType(String type) on the wrapped response object.</pre>
void	setLocale(java.util.Locale loc) The default behavior of this method is to call setLocale(Locale loc) on the wrapped response object.

setResponse (ServletResponse response)
Sets the response being wrapped.

Methods inherited from class java.lang.Object

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

Constructor Detail

ServletResponseWrapper

public ServletResponseWrapper(ServletResponse response)

Creates a ServletResponse adaptor wrapping the given response object.

Throws:

java.lang.IllegalArgumentException - if the response is null.

Method Detail

getResponse

public ServletResponse getResponse()

Return the wrapped ServletResponse object.

set Response

public void setResponse(ServletResponse response)

Sets the response being wrapped.

Throws:

java.lang.IllegalArgumentException - if the response is null.

getCharacterEncoding

```
public java.lang.String getCharacterEncoding()
```

The default behavior of this method is to return getCharacterEncoding() on the wrapped response object.

Specified by:

getCharacterEncoding in interface ServletResponse

getOutputStream

The default behavior of this method is to return getOutputStream() on the wrapped response object.

Specified by:

getOutputStream in interface ServletResponse

getWriter

The default behavior of this method is to return getWriter() on the wrapped response object.

Specified by:

getWriter in interface ServletResponse

setContentLength

```
public void setContentLength(int len)
```

The default behavior of this method is to call setContentLength(int len) on the wrapped response object.

Specified by:

setContentLength in interface ServletResponse

setContentType

```
public void setContentType(java.lang.String type)
```

The default behavior of this method is to call setContentType(String type) on the wrapped response object.

Specified by:

setContentType in interface ServletResponse

setBufferSize

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

The default behavior of this method is to call setBufferSize(int size) on the wrapped response object.

Specified by:

setBufferSize in interface ServletResponse

getBufferSize

```
public int getBufferSize()
```

The default behavior of this method is to return getBufferSize() on the wrapped response object.

Specified by:

getBufferSize in interface ServletResponse

flushBuffer

The default behavior of this method is to call flushBuffer() on the wrapped response object.

Specified by:

flushBuffer in interface ServletResponse

isCommitted

```
public boolean isCommitted()
```

The default behavior of this method is to return isCommitted() on the wrapped response object.

Specified by:

isCommitted in interface ServletResponse

reset

```
public void reset()
```

The default behavior of this method is to call reset() on the wrapped response object.

Specified by:

reset in interface ServletResponse

resetBuffer

```
public void resetBuffer()
```

The default behavior of this method is to call resetBuffer() on the wrapped response object.

Specified by:

resetBuffer in interface ServletResponse

setLocale

```
public void setLocale(java.util.Locale loc)
```

The default behavior of this method is to call setLocale(Locale loc) on the wrapped response object.

Specified by:

<u>setLocale</u> in interface <u>ServletResponse</u>

getLocale

public java.util.Locale getLocale()

The default behavior of this method is to return getLocale() on the wrapped response object.

Specified by:

getLocale in interface ServletResponse

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

Overview Package Class Tree Deprecated Index Help

PREV CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD

javax.servlet Class UnavailableException

public class **UnavailableException** extends <u>ServletException</u>

Defines an exception that a servlet or filter throws to indicate that it is permanently or temporarily unavailable.

When a servlet or filter is permanently unavailable, something is wrong with the it, and it cannot handle requests until some action is taken. For example, a servlet might be configured incorrectly, or a filter's state may be corrupted. The component should log both the error and the corrective action that is needed.

A servlet or filter is temporarily unavailable if it cannot handle requests momentarily due to some system-wide problem. For example, a third-tier server might not be accessible, or there may be insufficient memory or disk storage to handle requests. A system administrator may need to take corrective action.

Servlet containers can safely treat both types of unavailable exceptions in the same way. However, treating temporary unavailability effectively makes the servlet container more robust. Specifically, the servlet container might block requests to the servlet or filter for a period of time suggested by the exception, rather than rejecting them until the servlet container restarts.

See Also:

Serialized Form

Constructor Summary

UnavailableException(int seconds, Servlet servlet,

java.lang.String msg)

Deprecated. As of Java Servlet API 2.2, use

<u>UnavailableException(String, int)</u> instead.

UnavailableException(Servlet servlet, java.lang.String msg)

Deprecated. As of Java Servlet API 2.2, use

<u>UnavailableException(String)</u> instead.

UnavailableException(java.lang.String msg)

Constructs a new exception with a descriptive message indicating that the servlet is permanently unavailable.

UnavailableException(java.lang.String msg, int seconds)

Constructs a new exception with a descriptive message indicating that the servlet is temporarily unavailable and giving an estimate of how long it will be unavailable.

Method Summary		
Servlet	getServlet() Deprecated. As of Java Servlet API 2.2, with no replacement. Returns the servlet that is reporting its unavailability.	
int	getUnavailableSeconds() Returns the number of seconds the servlet expects to be temporarily unavailable.	
boolean	isPermanent() Returns a boolean indicating whether the servlet is permanently unavailable.	

Methods inherited from class javax.servlet.ServletException

getRootCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait

Constructor Detail

UnavailableException

Deprecated. As of Java Servlet API 2.2, use <u>UnavailableException(String)</u> instead.

Parameters:

servlet - the Servlet instance that is unavailable
msg - a String specifying the descriptive message

UnavailableException

Deprecated. As of Java Servlet API 2.2, use <u>UnavailableException(String, int)</u> instead.

Parameters:

seconds - an integer specifying the number of seconds the servlet expects to be unavailable; if zero or negative, indicates that the servlet can't make an estimate servlet - the Servlet that is unavailable

msg - a String specifying the descriptive message, which can be written to a log file or displayed for the user.

UnavailableException

public UnavailableException(java.lang.String msg)

Constructs a new exception with a descriptive message indicating that the servlet is permanently unavailable.

Parameters:

msg - a String specifying the descriptive message

UnavailableException

Constructs a new exception with a descriptive message indicating that the servlet is temporarily unavailable and giving an estimate of how long it will be unavailable.

In some cases, the servlet cannot make an estimate. For example, the servlet might know that a server it needs is not running, but not be able to report how long it will take to be restored to functionality. This can be indicated with a negative or zero value for the seconds argument.

Parameters:

msg - a String specifying the descriptive message, which can be written to a log file or displayed for the user. seconds - an integer specifying the number of seconds the servlet expects to be unavailable; if zero or negative, indicates that the servlet can't make an estimate

Method Detail

isPermanent

```
public boolean isPermanent()
```

Returns a boolean indicating whether the servlet is permanently unavailable. If so, something is wrong with the servlet, and the system administrator must take some corrective action.

Returns:

true if the servlet is permanently unavailable; false if the servlet is available or temporarily unavailable

getServlet

```
public Servlet getServlet()
```

Deprecated. As of Java Servlet API 2.2, with no replacement. Returns the servlet that is reporting its unavailability.

Returns:

the Servlet object that is throwing the UnavailableException

getUnavailableSeconds

```
public int getUnavailableSeconds()
```

Returns the number of seconds the servlet expects to be temporarily unavailable.

If this method returns a negative number, the servlet is permanently unavailable or cannot provide an estimate of how long it will be unavailable. No effort is made to correct for the time elapsed since the exception was first reported.

Returns:

an integer specifying the number of seconds the servlet will be temporarily unavailable, or a negative number if the servlet is permanently unavailable or cannot make an estimate

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS
SUMMARY: INNER | FIELD | CONSTR | METHOD

FRAMES NO FRAMES
DETAIL: FIELD | CONSTR | METHOD