



LIVE SOFTWARE, INC.

# Java Servlet API

Version 2.1.1

LIVE SOFTWARE, INC.

# Java Servlet API

---

Version 2.1.1

© 1999 Live Software, Inc.  
20245 Stevens Creek Blvd, Suite 100  
Cupertino, CA 95014  
[info@livesoftware.com](mailto:info@livesoftware.com)  
<http://www.livesoftware.com>

<a href="#">Package javax.servlet</a> .....	4
<a href="#">javax.servlet Interface RequestDispatcher</a> .....	5
<a href="#">javax.servlet Interface Servlet</a> .....	6
<a href="#">javax.servlet Interface ServletConfig</a> .....	10
<a href="#">javax.servlet Interface ServletContext</a> .....	11
<a href="#">javax.servlet Interface ServletRequest</a> .....	18
<a href="#">javax.servlet Interface ServletResponse</a> .....	24
<a href="#">javax.servlet Interface SingleThreadModel</a> .....	27
<a href="#">javax.servlet Class GenericServlet</a> .....	27
<a href="#">javax.servlet Class ServletInputStream</a> .....	33
<a href="#">javax.servlet Class ServletOutputStream</a> .....	35
<a href="#">javax.servlet Class ServletException</a> .....	40
<a href="#">javax.servlet Class UnavailableException</a> .....	42
<a href="#">Package javax.servlet.http</a> .....	46
<a href="#">javax.servlet.http Interface HttpServletRequest</a> .....	46
<a href="#">javax.servlet.http Interface HttpServletResponse</a> .....	54
<a href="#">javax.servlet.http Interface HttpSession</a> .....	69
<a href="#">javax.servlet.http Interface HttpSessionBindingListener</a> .....	75
<a href="#">javax.servlet.http Interface HttpSessionContext</a> .....	76
<a href="#">javax.servlet.http Class Cookie</a> .....	77
<a href="#">javax.servlet.http Class HttpServlet</a> .....	83
<a href="#">javax.servlet.http Class HttpSessionBindingEvent</a> .....	91
<a href="#">javax.servlet.http Class HttpUtils</a> .....	93
<a href="#">A</a> .....	96
<a href="#">C</a> .....	96
<a href="#">D</a> .....	96
<a href="#">E</a> .....	97
<a href="#">F</a> .....	98
<a href="#">G</a> .....	98
<a href="#">H</a> .....	105
<a href="#">I</a> .....	106
<a href="#">J</a> .....	107
<a href="#">L</a> .....	107
<a href="#">P</a> .....	108
<a href="#">R</a> .....	109
<a href="#">S</a> .....	110
<a href="#">U</a> .....	117
<a href="#">V</a> .....	117



# Java Servlet API 2.1.1

## Interfaces

### Package javax.servlet

Interface Summary	
<a href="#"><u>RequestDispatcher</u></a>	Defines a request dispatcher object that receives request from the client and sends them to any resource (such as a servlet, CGI script, HTML file, or JSP file) available on the server.
<a href="#"><u>Servlet</u></a>	A Servlet is a small program that runs inside a web server.
<a href="#"><u>ServletConfig</u></a>	Defines an object that a servlet engine generates to pass configuration information to a servlet when such servlet is initialized.
<a href="#"><u>ServletContext</u></a>	A servlet engine generated object that gives servlets information about their environment.
<a href="#"><u>ServletRequest</u></a>	Defines a servlet engine generated object that enables a servlet to get information about a client request.
<a href="#"><u>ServletResponse</u></a>	Interface for sending MIME data from the servlet's service method to the client.
<a href="#"><u>SingleThreadModel</u></a>	Defines a "single" thread model for servlet execution.

Class Summary	
<a href="#"><u>GenericServlet</u></a>	The GenericServlet class implements the Servlet interface and, for convenience, the ServletConfig interface.
<a href="#"><u>ServletInputStream</u></a>	An input stream for reading servlet requests, it provides an efficient readLine method.
<a href="#"><u>ServletOutputStream</u></a>	An output stream for writing servlet responses.

Exception Summary	
<a href="#"><u>ServletException</u></a>	This exception is thrown to indicate a servlet problem.
<a href="#"><u>UnavailableException</u></a>	This exception indicates that a servlet is unavailable.

## Interface RequestDispatcher

public abstract interface **RequestDispatcher**

Defines a request dispatcher object that receives request from the client and sends them to any resource (such as a servlet, CGI script, HTML file, or JSP file) available on the server. The request dispatcher object is created by the servlet engine and serves as a wrapper around a server resource defined by a particular URL path.

The `RequestDispatcher` interface is defined primarily to wrap servlets, but a servlet engine can create request dispatcher objects to wrap any type of resource.

Request dispatcher objects are created by the servlet engine, not by the servlet developer.

### See Also:

[ServletContext.getRequestDispatcher\(java.lang.String\)](#)

## Method Summary

void	<a href="#">forward</a> ( <a href="#">ServletRequest</a> request, <a href="#">ServletResponse</a> response) Used for forwarding a request from this servlet to another resource on the server.
void	<a href="#">include</a> ( <a href="#">ServletRequest</a> request, <a href="#">ServletResponse</a> response) Used for including the content generated by another server resource in the body of a response.

## Method Detail

### forward

```
public void forward(ServletRequest request,
                  ServletResponse response)
    throws ServletException,
           java.io.IOException
```

Used for forwarding a request from this servlet to another resource on the server. This method is useful when one servlet does preliminary processing of a request and wants to let another object generate the response.

The `request` object passed to the target object will have its request URL path and other path parameters adjusted to reflect the target URL path of the target object.

You cannot use this method if a `ServletOutputStream` object or `PrintWriter` object has been obtained from the response. In that case, the method throws an `IllegalStateException`.

### Parameters:

`request` - the client's request on the servlet

response - the client's response from the servlet

**Throws:**

[ServletException](#) - if a servlet exception is thrown by the target servlet

java.io.IOException - if an I/O Exception occurs

IllegalStateException - if the ServletOutputStream or a writer had allready been obtained from the response object

---

**include**

```
public void include(ServletRequest request,  
                  ServletResponse response)  
    throws ServletException,  
          java.io.IOException
```

Used for including the content generated by another server resource in the body of a response. In essence, this method enables programmatic server side includes.

The request object passed to the target object will reflect the request URL path and path info of the calling request. The response object only has access to the calling servlet's ServletOutputStream object or PrintWriter object.

An included servlet cannot set headers. If the included servlet calls a method that may need to set headers (such as sessions might need to), the method is not guaranteed to work. As a servlet developer, you must ensure that any methods that might need direct access to headers are properly resolved. To ensure that a session works correctly, start the session outside of the included servlet, even if you use session tracking.

**Parameters:**

request - the client's request on the servlet

response - the client's response from the servlet

**Throws:**

[ServletException](#) - if a servlet exception is thrown by the target servlet

java.io.IOException - if the ServletOutputStream or a writer had already been obtained from the response object

---

javax.servlet

**Interface Servlet**

**All Known Implementing Classes:**

[GenericServlet](#)

---

---

public abstract interface **Servlet**

A Servlet is a small program that runs inside a web server. It receives and responds to requests from web clients.

All servlets implement this interface. Servlet writers typically do this by subclassing either `GenericServlet`, which implements the `Servlet` interface, or by subclassing `GenericServlet`'s descendent, `HttpServlet`.

The `Servlet` 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 by the network service in the following manner:

1. Servlet is created then **initialized**.
2. Zero or more **service** calls from clients are handled
3. Servlet is **destroyed** then garbage collected and finalized

In addition to the life-cycle methods, the `Servlet` interface provides for a method for the servlet to use to get any startup information, and a method that allows the servlet to return basic information about itself, such as its author, version and copyright.

**See Also:**

[GenericServlet](#), [HttpServlet](#)

---

<b>Method Summary</b>	
void	<a href="#"><b>destroy()</b></a> Called by the servlet engine when the servlet is removed from service.
<a href="#">ServletConfig</a>	<a href="#"><b>getServletConfig()</b></a> Returns a <code>ServletConfig</code> object, which contains any initialization parameters and startup configuration for this servlet.
java.lang. String	<a href="#"><b>getServletInfo()</b></a> Allows the servlet to provide information about itself to the host servlet runner such as author, version, and copyright.
void	<a href="#"><b>init(ServletConfig config)</b></a> Called by the web server when the Servlet is placed into service.
void	<a href="#"><b>service(ServletRequest req, ServletResponse res)</b></a> Called by the servlet engine to allow the servlet to respond to a request.

## Method Detail

### **init**

```
public void init(ServletConfig config)  
    throws ServletException
```

Called by the web server when the Servlet is placed into service. This method is

---

called exactly once by the host servlet engine after the Servlet object is instantiated and must successfully complete before any requests can be routed through the Servlet.

If a ServletException is thrown during the execution of this method, a servlet engine may not place the servlet into service. If the method does not return within a server defined time-out period, the servlet engine may assume that the servlet is nonfunctional and may not place it into service.

### Parameters:

config - object containing the servlet's startup- configuration and initialization parameters

### Throws:

[ServletException](#) - if a servlet exception has occurred

### See Also:

[UnavailableException](#), [getServletConfig\(\)](#)

---

## getServletConfig

```
public ServletConfig getServletConfig()
```

Returns a ServletConfig object, which contains any initialization parameters and startup configuration for this servlet. This is the ServletConfig object passed to the init method; the init method should have stored this object so that this method could return it.

The servlet writer is responsible for storing the ServletConfig object passed to the init method so it may be accessed via this method. For your convenience, the GenericServlet implementation of this interface already does this.

### See Also:

[init\(javax.servlet.ServletConfig\)](#)

---

## service

```
public void service(ServletRequest req,  
                   ServletResponse res)  
    throws ServletException,  
           java.io.IOException
```

Called by the servlet engine to allow the servlet to respond to a request. This method can only be called when the servlet has been properly initialized. The servlet engine may block pending requests to this servlet until initialization is complete. Similarly, when a servlet is removed from service (has its destroy method called), no more requests can be serviced by this instance of the servlet.

---

Note that servlets typically run inside of multi threaded servlet engines that can handle multiple requests simultaneously. It is the servlet writer's responsibility to synchronize access to any shared resources, such as network connections or the servlet's class and instance variables. Information on multi-threaded programming in Java can be found in [the Java tutorial on multi-threaded programming](#).

**Parameters:**

req - the client's request of the servlet

res - the servlet's response to the client

**Throws:**

[ServletException](#) - if a servlet exception has occurred

java.io.IOException - if an I/O exception has occurred

---

**getServletInfo**

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

Allows the servlet to provide information about itself to the host servlet runner such as author, version, and copyright. As this method may be called to display such information in an administrative tool, the string that this method returns should be plain text and not composed of markup of any kind (such as HTML, XML, etc).

**Returns:**

String containing servlet information

---

**destroy**

```
public void destroy()
```

Called by the servlet engine when the servlet is removed from service. The servlet engine may not call this method until all threads within in the servlet's service method have exited or an engine specified timeout period has passed. After this method is run, the service method may not be called by the servlet engine on this instance of the servlet.

This method gives the servlet an opportunity to clean up whatever resources are being held (e.g., memory, file handles, thread) and makes sure that any persistent state is synchronized with the servlet's current in-memory state.

---

javax.servlet

## Interface ServletConfig

All Known Implementing Classes:

[GenericServlet](#)

---

public abstract interface **ServletConfig**

Defines an object that a servlet engine generates to pass configuration information to a servlet when such servlet is initialized. The configuration information that this servlet will have access to is a set of name/value pairs that describe initialization parameters and the `ServletContext` object which describes the context within which the servlet will be running.

---

### Method Summary

java.lang.String	<a href="#">getInitParameter</a> (java.lang.String name) Returns a string containing the value of the named initialization parameter of the servlet, or null if the parameter does not exist.
java.util.Enumeration	<a href="#">getInitParameterNames</a> () Returns the names of the servlet's initialization parameters as an enumeration of strings, or an empty enumeration if there are no initialization parameters.
<a href="#">ServletContext</a>	<a href="#">getServletContext</a> () Returns the <code>ServletContext</code> for this servlet.

### Method Detail

#### getServletContext

```
public ServletContext getServletContext()
```

Returns the `ServletContext` for this servlet.

#### getInitParameter

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

Returns a string containing the value of the named initialization parameter of the servlet, or null if the parameter does not exist. Init parameters have a single string value; it is the responsibility of the servlet writer to interpret the string.

## Parameters:

name - the name of the parameter whose value is requested

---

## getInitParameterNames

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

Returns the names of the servlet's initialization parameters as an enumeration of strings, or an empty enumeration if there are no initialization parameters.

---

javax.servlet

## Interface ServletContext

---

```
public abstract interface ServletContext
```

A servlet engine generated object that gives servlets information about their environment. In a server that supports the concept of multiple hosts (and even virtual hosts), the context must be at least as unique as the host. Servlet engines may also provide context objects that are unique to a group of servlets and which is tied to a specific portion of the URL path namespace of the host. This grouping may be administratively assigned or defined by deployment information.

Servlets get the ServletContext object via the `getServletContext` method of `ServletConfig`. The `ServletConfig` object is provided to the servlet at initialization, and is accessible via the servlet's `getServletConfig` method.

### See Also:

[Servlet.getServletConfig\(\)](#), [ServletConfig.getServletContext\(\)](#)

---

Method Summary	
java.lang.Object	<a href="#">getAttribute</a> (java.lang.String name) Returns an object that is known to the context by a given name, or null if there is no such object associated with the name.
java.util.Enumeration	<a href="#">getAttributeNames</a> () Returns an enumeration of the attribute names present in this context.
ServletContext	<a href="#">getContext</a> (java.lang.String uripath) Returns a <code>ServletContext</code> object for a particular URL path.
int	<a href="#">getMajorVersion</a> ()

---

	Returns the major version of the servlet API that this servlet engine supports.
java.lang.String	<a href="#"><u>getMimeType</u></a> (java.lang.String file) Returns the mime type of the specified file, or null if not known.
int	<a href="#"><u>getMinorVersion</u></a> () Returns the minor version of the servlet API that this servlet engine supports.
java.lang.String	<a href="#"><u>getRealPath</u></a> (java.lang.String path) Applies alias rules to the specified virtual path in URL path format, that is, /dir/dir/file.ext.
<a href="#"><u>RequestDispatcher</u></a>	<a href="#"><u>getRequestDispatcher</u></a> (java.lang.String urlpath) Returns a <code>RequestDispatcher</code> object for the specified URL path if the context knows of an active source (such as a servlet, JSP page, CGI script, etc) of content for the particular path.
java.net.URL	<a href="#"><u>getResource</u></a> (java.lang.String path) Returns a URL object of a resource that is mapped to a corresponding URL path.
java.io.InputStream	<a href="#"><u>getResourceAsStream</u></a> (java.lang.String path) Returns an <code>InputStream</code> object allowing access to a resource that is mapped to a corresponding URL path.
java.lang.String	<a href="#"><u>getServerInfo</u></a> () Returns the name and version of the network service under which the servlet is running.
<a href="#"><u>Servlet</u></a>	<a href="#"><u>getServlet</u></a> (java.lang.String name) <b>Deprecated.</b> <i>This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.</i>
java.util.Enumeration	<a href="#"><u>getServletNames</u></a> () <b>Deprecated.</b> <i>This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.</i>
java.util.Enumeration	<a href="#"><u>getServlets</u></a> () <b>Deprecated.</b> <i>This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.</i>
void	<a href="#"><u>log</u></a> (java.lang.Exception exception, java.lang.String msg) <b>Deprecated.</b> <i>Use <code>log(String message, Throwable t)</code> instead</i>
void	<a href="#"><u>log</u></a> (java.lang.String msg) Logs the specified message to the context's log.
void	<a href="#"><u>log</u></a> (java.lang.String message, java.lang.Throwable throwable) Logs the specified message and a stack trace of the given <code>Throwable</code> object to the context's log.
void	<a href="#"><u>removeAttribute</u></a> (java.lang.String name) Removes the attribute from the context that is bound to a particular name.
void	<a href="#"><u>setAttribute</u></a> (java.lang.String name, java.lang.Object object) Binds an object to a given name in this context.

## Method Detail

## getContext

```
public ServletContext getContext(java.lang.String uripath)
```

Returns a `ServletContext` object for a particular URL path. This allows servlets to potentially gain access to the resources and to obtain `RequestDispatcher` objects from the target context.

In security conscious environments, the servlet engine may always return null for any given URL path.

### Parameters:

uripath -

---

## getMajorVersion

```
public int getMajorVersion()
```

Returns the major version of the servlet API that this servlet engine supports. All 2.1 compliant implementations must return the integer 2 from this method.

### Returns:

2

---

## getMimeType

```
public java.lang.String getMimeType(java.lang.String file)
```

Returns the mime type of the specified file, or null if not known. The MIME type is determined according to the configuration of the servlet engine.

### Parameters:

file - name of the file whose mime type is required

---

## getMinorVersion

```
public int getMinorVersion()
```

Returns the minor version of the servlet API that this servlet engine supports. All 2.1 compliant implementations must return the integer 1 from this method.

### Returns:

1

---

---

## **getResource**

```
public java.net.URL getResource(java.lang.String path)
                        throws java.net.MalformedURLException
```

Returns a URL object of a resource that is mapped to a corresponding URL path. The URL path must be of the form `/dir/dir/file.ext`. This method allows a servlet to access content to be served from the servlet engines document space in a system independent manner. Resources could be located on the local file system, a remote file system, a database, or a remote network site.

This method may return null if there is no resource mapped to the given URL path.

The servlet engine must implement whatever URL handlers and `URLConnection` objects are necessary to access the given content.

This method does not fill the same purpose as the `getResource` method of `java.lang.Class`. The method in `java.lang.Class` looks up resources based on class loader. This method allows servlet engines to make resources available to a servlet from any source without regards to class loaders, location, etc.

### **Parameters:**

path - Path of the content resource

### **Throws:**

`java.net.MalformedURLException` - if the resource path is not properly formed.

---

## **getResourceAsStream**

```
public java.io.InputStream getResourceAsStream(java.lang.String path)
```

Returns an `InputStream` object allowing access to a resource that is mapped to a corresponding URL path. The URL path must be of the form `/dir/dir/file.ext`.

Note that meta-information such as content length and content type that are available when using the `getResource` method of this class are lost when using this method.

This method may return null if there is no resource mapped to the given URL path.

The servlet engine must implement whatever URL handlers and `URLConnection` objects are necessary to access the given content.

This method does not fill the same purpose as the `getResourceAsStream` method of `java.lang.Class`. The method in `java.lang.Class` looks up resources based on class loader. This method allows servlet engines to make resources available to a servlet from any source without regards to class loaders, location, etc.

---

## Parameters:

name -

---

## getRequestDispatcher

```
public RequestDispatcher getRequestDispatcher(java.lang.String urlpath)
```

Returns a `RequestDispatcher` object for the specified URL path if the context knows of an active source (such as a servlet, JSP page, CGI script, etc) of content for the particular path. This format of the URL path must be of the form `/dir/dir/file.ext`. The servlet engine is responsible for implementing whatever functionality is required to wrap the target source with an implementation of the `RequestDispatcher` interface.

This method will return null if the context cannot provide a dispatcher for the path provided.

## Parameters:

urlpath - Path to use to look up the target server resource

## See Also:

[RequestDispatcher](#)

---

## getServlet

```
public Servlet getServlet(java.lang.String name)  
    throws ServletException
```

**Deprecated.** *This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.*

Originally defined to return a servlet from the context with the specified name. This method has been deprecated and only remains to preserve binary compatibility. This method will always return null.

---

## getServlets

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

**Deprecated.** *This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.*

Originally defined to return an `Enumeration` of `Servlet` objects containing all the servlets known to this context. This method has been deprecated and only remains to preserve binary compatibility. This method must always return an empty enumeration.

---

## getServletNames

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

**Deprecated.** *This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.*

Originally defined to return an Enumeration of String objects containing all the servlet names known to this context. This method has been deprecated and only remains to preserve binary compatibility. This method must always return an empty enumeration.

---

## log

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

Logs the specified message to the context's log. The name and type of the servlet log is servlet engine specific, but is normally an event log.

### Parameters:

msg - the message to be written

---

## log

```
public void log(java.lang.Exception exception,  
               java.lang.String msg)
```

**Deprecated.** *Use log(String message, Throwable t) instead*

Logs the specified message and a stack trace of the given exception to the context's log. The name and type of the servlet log is servlet engine specific, but is normally an event log.

### Parameters:

exception - the exception to be written

msg - the message to be written

---

## log

```
public void log(java.lang.String message,  
               java.lang.Throwable throwable)
```

Logs the specified message and a stack trace of the given Throwable object to the context's log. The name and type of the servlet log is servlet engine specific, but is normally an event log.

---

**Parameters:**

msg - the message to be written

throwable - the exception to be written

---

**getRealPath**

```
public java.lang.String getRealPath(java.lang.String path)
```

Applies alias rules to the specified virtual path in URL path format, that is, /dir/dir/file.ext. Returns a String representing the corresponding real path in the format that is appropriate for the operating system the servlet engine is running under (including the proper path separators).

This method returns null if the translation could not be performed for any reason.

**Parameters:**

path - the virtual path to be translated into a real path

---

**getServerInfo**

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

Returns the name and version of the network service under which the servlet is running. The form of this string must begin with <servername>/<versionnumber>. For example the Java Web Server could return a string of the form Java Web Server/1.1.3. Other optional information can be returned in parenthesis after the primary string. For example, Java Web Server/1.1.3 (JDK 1.1.6; Windows NT 4.0 x86) .

---

**getAttribute**

```
public java.lang.Object getAttribute(java.lang.String name)
```

Returns an object that is known to the context by a given name, or null if there is no such object associated with the name. This method allwos access to additional information about the servlet engine not already provided by other methods in this interface. Attribute names should follow the same convention as package names. Names matching java.\*, javax.\*, and sun.\* are reserved for definition by this specification or by the reference implementation.

**Parameters:**

name - the name of the attribute whose value is required

**Returns:**

the value of the attribute, or null if the attribute does not exist.

---

### **getAttributeNames**

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

Returns an enumeration of the attribute names present in this context.

---

### **setAttribute**

```
public void setAttribute(java.lang.String name,  
                           java.lang.Object object)
```

Binds an object to a given name in this context. If an object is already bound into the context with the given name, it will be replaced. Attribute names should follow the same convention as package names. Names matching `java.*`, `javax.*`, and `sun.*` are reserved for definition by this specification or by the reference implementation.

#### **Parameters:**

name - the name of the attribute to store

value - the value of the attribute

---

### **removeAttribute**

```
public void removeAttribute(java.lang.String name)
```

Removes the attribute from the context that is bound to a particular name.

#### **Parameters:**

name - the name of the attribute to remove from the context

---

javax.servlet

## **Interface ServletRequest**

#### **All Known Subinterfaces:**

[HttpServletRequest](#)

---

```
public abstract interface ServletRequest
```

Defines a servlet engine generated object that enables a servlet to get information about a client request.

---

Some of the data provided by the ServletRequest object includes parameter names and values, attributes, and an input stream. Subclasses of ServletRequest can provide additional protocol-specific data. For example, HTTP data is provided by the interface HttpServletRequest, which extends ServletRequest. This framework provides the servlet's only access to this data.

MIME bodies are either text or binary data. Use `getReader` to handle text, including the character encodings. The `getInputStream` call should be used to handle binary data.

Multipart MIME bodies are treated as binary data, since the headers are US-ASCII data.

**See Also:**

[HttpServletRequest](#)

<b>Method Summary</b>	
java.lang.Object	<a href="#">getAttribute</a> (java.lang.String name) Returns the value of the named attribute of this request.
java.util.Enumeration	<a href="#">getAttributeNames</a> () Returns an enumeration of attribute names contained in this request.
java.lang.String	<a href="#">getCharacterEncoding</a> () Returns the character set encoding for the input of this request.
int	<a href="#">getContentLength</a> () Returns the size of the request entity data, or -1 if not known.
java.lang.String	<a href="#">getContentType</a> () Returns the Internet Media (MIME) Type of the request entity data, or null if not known.
<a href="#">ServletInputStream</a>	<a href="#">getInputStream</a> () Returns an input stream for reading binary data in the request body.
java.lang.String	<a href="#">getParameter</a> (java.lang.String name) Returns a string containing the lone value of the specified parameter, or null if the parameter does not exist.
java.util.Enumeration	<a href="#">getParameterNames</a> () Returns the parameter names for this request as an enumeration of strings, or an empty enumeration if there are no parameters or the input stream is empty.
java.lang.String []	<a href="#">getParameterValues</a> (java.lang.String name) Returns the values of the specified parameter for the request as an array of strings, or null if the named parameter does not exist.
java.lang.String	<a href="#">getProtocol</a> () Returns the protocol and version of the request as a string of the form <protocol>/<major version>.<minor version>.
java.io.BufferedReader	<a href="#">getReader</a> () Returns a buffered reader for reading text in the request body.

	r	
java.lang.String		<a href="#"><u>getRealPath</u></a> (java.lang.String path) <b>Deprecated.</b> <i>This method has been deprecated in preference to the same method found in the ServletContext interface.</i>
java.lang.String		<a href="#"><u>getRemoteAddr</u></a> () Returns the IP address of the agent that sent the request.
java.lang.String		<a href="#"><u>getRemoteHost</u></a> () Returns the fully qualified host name of the agent that sent the request.
java.lang.String		<a href="#"><u>getScheme</u></a> () Returns the scheme of the URL used in this request, for example "http", "https", or "ftp".
java.lang.String		<a href="#"><u>getServerName</u></a> () Returns the host name of the server that received the request.
	int	<a href="#"><u>getServerPort</u></a> () Returns the port number on which this request was received.
	void	<a href="#"><u>setAttribute</u></a> (java.lang.String key, java.lang.Object o) This method stores an attribute in the request context; these attributes will be reset between requests.

## Method Detail

### getAttribute

```
public java.lang.Object getAttribute(java.lang.String name)
```

Returns the value of the named attribute of this request. This method may return null if the attribute does not exist. This method allows access to request information not already provided by other methods in this interface or data that was placed in the request object by other server components. Attribute names should follow the same convention as package names. Names matching java.\*, javax.\*, and sun.\* are reserved for definition by this specification or by the reference implementation.

#### Parameters:

name - the name of the attribute whose value is required

---

### getAttributeNames

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

Returns an enumeration of attribute names contained in this request.

---

### getCharacterEncoding

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

---

Returns the character set encoding for the input of this request. This method may return null if no character encoding is defined for this request body.

---

### **getContentLength**

```
public int getContentLength()
```

Returns the size of the request entity data, or -1 if not known. Same as the CGI variable CONTENT\_LENGTH.

---

### **getContentType**

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

Returns the Internet Media (MIME) Type of the request entity data, or null if not known. Same as the CGI variable CONTENT\_TYPE.

---

### **getInputStream**

```
public ServletInputStream getInputStream()  
throws java.io.IOException
```

Returns an input stream for reading binary data in the request body.

#### **Throws:**

IllegalStateException - if `getReader` has been called on this same request.

java.io.IOException - on other I/O related errors.

#### **See Also:**

`getReader`

---

### **getParameter**

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

Returns a string containing the lone value of the specified parameter, or null if the parameter does not exist. For example, in an HTTP servlet this method would return the value of the specified query string parameter. Servlet writers should use this method only when they are sure that there is only one value for the parameter. If the parameter has (or could have) multiple values, servlet writers should use `getParameterValues`. If a multiple valued parameter name is passed as an argument, the return value is implementation dependent.

#### **Parameters:**

---

name - the name of the parameter whose value is required.

**See Also:**

[getParameterValues\(java.lang.String\)](#)

---

## **getParameterNames**

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

Returns the parameter names for this request as an enumeration of strings, or an empty enumeration if there are no parameters or the input stream is empty. The input stream would be empty if all the data had been read from the stream returned by the method `getInputStream`.

---

## **getParameterValues**

```
public java.lang.String[] getParameterValues(java.lang.String name)
```

Returns the values of the specified parameter for the request as an array of strings, or null if the named parameter does not exist. For example, in an HTTP servlet this method would return the values of the specified query string or posted form as an array of strings.

**Parameters:**

name - the name of the parameter whose value is required.

**See Also:**

[getParameter\(java.lang.String\)](#)

---

## **getProtocol**

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

Returns the protocol and version of the request as a string of the form `<protocol>/<major version>.<minor version>`. Same as the CGI variable `SERVER_PROTOCOL`.

---

## **getScheme**

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

Returns the scheme of the URL used in this request, for example "http", "https", or "ftp". Different schemes have different rules for constructing URLs, as noted in RFC 1738. The URL used to create a request may be reconstructed using this scheme, the server name and port, and additional information such as URIs.

---

## **getServerName**

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

Returns the host name of the server that received the request. Same as the CGI variable SERVER\_NAME.

---

## **getServerPort**

```
public int getServerPort()
```

Returns the port number on which this request was received. Same as the CGI variable SERVER\_PORT.

---

## **getReader**

```
public java.io.BufferedReader getReader()  
                                throws java.io.IOException
```

Returns a buffered reader for reading text in the request body. This translates character set encodings as appropriate.

### **Throws:**

java.io.UnsupportedEncodingException - if the character set encoding is unsupported, so the text can't be correctly decoded.

IllegalStateException - if getInputStream has been called on this same request.

java.io.IOException - on other I/O related errors.

### **See Also:**

getInputStream

---

## **getRemoteAddr**

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

Returns the IP address of the agent that sent the request. Same as the CGI variable REMOTE\_ADDR.

---

## **getRemoteHost**

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

Returns the fully qualified host name of the agent that sent the request. Same as the CGI

---

variable REMOTE\_HOST.

---

## setAttribute

```
public void setAttribute(java.lang.String key,  
                          java.lang.Object o)
```

This method stores an attribute in the request context; these attributes will be reset between requests. Attribute names should follow the same convention as package names.

The package (and hence attribute) names beginning with java.\*, and javax.\* are reserved for use by Javasoft. Similarly, com.sun.\* is reserved for use by Sun Microsystems.

### Parameters:

key - a String specifying the name of the attribute

o - a context object stored with the key.

### Throws:

IllegalStateException - if the named attribute already has a value.

---

## getRealPath

```
public java.lang.String getRealPath(java.lang.String path)
```

**Deprecated.** *This method has been deprecated in preference to the same method found in the ServletContext interface.*

Applies alias rules to the specified virtual path and returns the corresponding real path, or null if the translation can not be performed for any reason. For example, an HTTP servlet would resolve the path using the virtual docroot, if virtual hosting is enabled, and with the default docroot otherwise. Calling this method with the string "/" as an argument returns the document root.

### Parameters:

path - the virtual path to be translated to a real path

---

javax.servlet

## Interface ServletResponse

### All Known Subinterfaces:

[HttpServletResponse](#)

---

## public abstract interface **ServletResponse**

Interface for sending MIME data from the servlet's service method to the client. Network service developers implement this interface; its methods are then used by servlets when the service method is run, to return data to clients. The `ServletResponse` object is passed as an argument to the service method.

To write MIME bodies which consist of binary data, use the output stream returned by `getOutputStream`. To write MIME bodies consisting of text data, use the writer returned by `getWriter`. If you need to mix binary and text data, for example because you're creating a multipart response, use the output stream to write the multipart headers, and use that to build your own text bodies.

If you don't explicitly set the character set in your MIME media type, with `setContentType`, one will be selected and the content type will be modified accordingly. If you will be using a writer, and want to call the `setContentType` method, you must do so before calling the `getWriter` method. If you will be using the output stream, and want to call `setContentType`, you must do so before using the output stream to write the MIME body. For more information about MIME, see the Internet RFCs such as [RFC 2045](#), the first in a series which defines MIME. Note that protocols such SMTP and HTTP define application-specific profiles of MIME, and that standards in this area are evolving.

---

<b>Method Summary</b>	
java.lang.String	<a href="#"><b>getCharacterEncoding</b></a> () Returns the character set encoding used for this MIME body.
<a href="#">ServletOutputStream</a>	<a href="#"><b>getOutputStream</b></a> () Returns an output stream for writing binary response data.
java.io.PrintWriter	<a href="#"><b>getWriter</b></a> () Returns a print writer for writing formatted text responses.
void	<a href="#"><b>setContentLength</b></a> (int len) Sets the content length for this response.
void	<a href="#"><b>setContentType</b></a> (java.lang.String type) Sets the content type for this response.

---

## **Method Detail**

### **getCharacterEncoding**

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

Returns the character set encoding used for this MIME body. The character encoding is either the one specified in the assigned content type, or one which the client understands. If no content type has yet been assigned, it is implicitly set to `text/plain`

See RFC 2047 for more information about character encoding and MIME.

---

## getOutputStream

```
public ServletOutputStream getOutputStream()  
    throws java.io.IOException
```

Returns an output stream for writing binary response data.

### Throws:

IllegalStateException - if getWriter has been called on this same request.

java.io.IOException - if an I/O exception has occurred

### See Also:

getWriter

---

## getWriter

```
public java.io.PrintWriter getWriter()  
    throws java.io.IOException
```

Returns a print writer for writing formatted text responses. The MIME type of the response will be modified, if necessary, to reflect the character encoding used, through the `charset=...` property. This means that the content type must be set before calling this method.

### Throws:

java.io.UnsupportedEncodingException - if no such encoding can be provided

IllegalStateException - if getOutputStream has been called on this same request.

java.io.IOException - on other errors.

### See Also:

getOutputStream, setContentType

---

## setContentLength

```
public void setContentLength(int len)
```

Sets the content length for this response.

### Parameters:

len - the content length

---

## setContentType

```
public void setContentType(java.lang.String type)
```

Sets the content type for this response. This type may later be implicitly modified by addition of properties such as the MIME `charset=<value>` if the service finds it necessary, and the appropriate media type property has not been set.

This response property may only be assigned one time. If a writer is to be used to write a text response, this method must be called before the method `getWriter`. If an output stream will be used to write a response, this method must be called before the output stream is used to write response data.

### Parameters:

`type` - the content's MIME type

### See Also:

`getOutputStream`, `getWriter`

---

javax.servlet

## Interface SingleThreadModel

---

```
public abstract interface SingleThreadModel
```

Defines a "single" thread model for servlet execution. This empty interface allows servlet implementers to specify how the system should handle concurrent calls to the same servlet.

If the target servlet is flagged with this interface, the servlet programmer is **guaranteed** that no two threads will execute concurrently the `service` method of that servlet. This guarantee is ensured by maintaining a pool of servlet instances for each such servlet, and dispatching each `service` call to a free servlet.

In essence, if the servlet implements this interface, the servlet will be thread safe. Note that this will not prevent synchronization problems associated with accessing shared resources (such as static class variables or classes outside the scope of the servlet).

---

javax.servlet

## Class GenericServlet

```
java.lang.Object
|
+-- javax.servlet.GenericServlet
```

### Direct Known Subclasses:

[HttpServlet](#)

---

public abstract class **GenericServlet**

extends java.lang.Object

implements [Servlet](#), [ServletConfig](#), java.io.Serializable

The GenericServlet class implements the Servlet interface and, for convenience, the ServletConfig interface. Servlet developers typically subclass GenericServlet, or its descendent HttpServlet, unless the servlet needs another class as a parent. (If a servlet does need to subclass another class, the servlet must implement the Servlet interface directly. This would be necessary when, for example, RMI or CORBA objects act as servlets.)

The GenericServlet class was created to make writing servlets easier. It provides simple versions of the life-cycle methods init and destroy, and of the methods in the ServletConfig interface. It also provides a log method, from the ServletContext interface. The servlet writer must override only the service method, which is abstract. Though not required, the servlet implementer should also override the getServletInfo method, and will want to specialize the init and destroy methods if expensive servlet-wide resources are to be managed.

**See Also:**

[Serialized Form](#)

---

## Constructor Summary

[GenericServlet](#)()

The default constructor does no work.

## Method Summary

void	<a href="#">destroy</a> () Destroys the servlet, cleaning up whatever resources are being held, and logs the destruction in the servlet log file.
java.lang.String	<a href="#">getInitParameter</a> (java.lang.String name) Returns a string containing the value of the named initialization parameter, or null if the requested parameter does not exist.
java.util.Enumeration	<a href="#">getInitParameterNames</a> () Returns the names of the initialization parameters for this servlet as an enumeration of Strings, or an empty enumeration if there are no initialization parameters.
<a href="#">ServletConfig</a>	<a href="#">getServletConfig</a> () Returns a servletConfig object containing any startup configuration information for this servlet.
<a href="#">ServletContext</a>	<a href="#">getServletContext</a> () Returns a ServletContext object, which contains information about the network service in

<a href="#">ServletContext</a>	which the servlet is running.
java.lang.String	<a href="#">getServletInfo()</a> Returns a string that contains information about the servlet, such as its author, version, and copyright.
void	<a href="#">init()</a> This method is provided as a convenience so that servlet writers do not have to worry about storing the ServletConfig object.
void	<a href="#">init(ServletConfig config)</a> Initializes the servlet and logs the initialization.
void	<a href="#">log(java.lang.String msg)</a> Writes the class name of the servlet and the given message to the servlet log file.
void	<a href="#">log(java.lang.String message, java.lang.Throwable t)</a> Logs the message with the root cause
abstract void	<a href="#">service(ServletRequest req, ServletResponse res)</a> Carries out a single request from the client.

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

## Constructor Detail

### GenericServlet

```
public GenericServlet()
```

The default constructor does no work.

## Method Detail

### destroy

```
public void destroy()
```

Destroys the servlet, cleaning up whatever resources are being held, and logs the destruction in the servlet log file. This method is called, once, automatically, by the network service each time it removes the servlet. After destroy is run, it cannot be called again until the network service reloads the servlet.

When the network service removes a servlet, it calls destroy after all service calls have been completed, or a service-specific number of seconds have passed, whichever comes first. In the case of long-running operations, there could be other threads running service requests when destroy is called. The servlet writer is responsible for making sure that any threads still in the service method complete.

**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 requested parameter does not exist. Init parameters have a single string value; it is the responsibility of the servlet writer to interpret the string.

This is a convenience method; it gets the parameter's value from the ServletConfig object. (The ServletConfig object was passed into and stored by the init method.)

**Specified by:**

[getInitParameter](#) in interface [ServletConfig](#)

**Parameters:**

name - the name of the parameter whose value is requested

---

**getInitParameterNames**

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

Returns the names of the initialization parameters for this servlet as an enumeration of Strings, or an empty enumeration if there are no initialization parameters.

This method is supplied for convenience. It gets the parameter names from the ServletConfig object. (The ServletConfig object was passed into and stored by the init method.)

**Specified by:**

[getInitParameterNames](#) in interface [ServletConfig](#)

---

**getServletConfig**

```
public ServletConfig getServletConfig()
```

Returns a servletConfig object containing any startup configuration information for this servlet.

**Specified by:**

[getServletConfig](#) in interface [Servlet](#)

---

---

## getServletContext

```
public ServletContext getServletContext()
```

Returns a ServletContext object, which contains information about the network service in which the servlet is running. This is a convenience method; it gets the ServletContext object from the ServletConfig object. (The ServletConfig object was passed into and stored by the init method.)

### Specified by:

[getServletContext](#) in interface [ServletConfig](#)

---

## getServletInfo

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

Returns a string that contains information about the servlet, such as its author, version, and copyright. This method must be overridden in order to return this information. If it is not overridden, an empty string is returned.

### Specified by:

[getServletInfo](#) in interface [Servlet](#)

---

## init

```
public void init(ServletConfig config)  
    throws ServletException
```

Initializes the servlet and logs the initialization. The init method is called once, automatically, by the network service each time it loads the servlet. It is guaranteed to finish before any service requests are accepted. On fatal initialization errors, an UnavailableException should be thrown. Do not call the method System.exit.

The init method stores the ServletConfig object. Servlet writers who specialize this method should call either super.init, or store the ServletConfig object themselves. If an implementor decides to store the ServletConfig object in a different location, then the getServletConfig method must also be overridden.

### Specified by:

[init](#) in interface [Servlet](#)

### Parameters:

config - servlet configuration information

---

**Throws:**

[ServletException](#) - if a servlet exception has occurred

**See Also:**

[UnavailableException](#)

---

**init**

```
public void init()  
    throws ServletException
```

This method is provided as a convenience so that servlet writers do not have to worry about storing the ServletConfig object. When extending this class, simply override this method and it will be called by GenericServlet.init(ServletConfig config);

---

**log**

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

Writes the class name of the servlet and the given message to the servlet log file. The name of the servlet log file is server specific; it is normally an event log.

If a servlet will have multiple instances (for example, if the network service runs the servlet for multiple virtual hosts), the servlet writer should override this method. The specialized method should log an instance identifier and possibly a thread identifier, along with the requested message. The default message prefix, the class name of the servlet, does not allow the log entries of the instances to be distinguished from one another.

**Parameters:**

msg - the message string to be logged

---

**log**

```
public void log(java.lang.String message,  
               java.lang.Throwable t)
```

Logs the message with the root cause

---

**service**

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

---

Carries out a single request from the client. The request object contains parameters provided by the client, and an input stream, which can also bring data to the servlet. To return information to the client, write to the output stream of the response object.

Service requests handled after servlet initialization has completed. Any requests for service that are received during initialization block until it is complete.

Note that servlets typically run inside multi-threaded network services, which can handle multiple service requests simultaneously. It is the servlet writer's responsibility to synchronize access to any shared resources, such as database or network connections. The simplest way to do this is to synchronize the entire service call. This can have a major performance impact, however, and should be avoided whenever possible in favor of techniques that are less coarse. For more information on synchronization, see the [the Java tutorial on multithreaded programming](#).

**Specified by:**

[service](#) in interface [Servlet](#)

**Parameters:**

req - the servlet request

res - the servlet response

**Throws:**

[ServletException](#) - if a servlet exception has occurred

java.io.IOException - if an I/O exception has occurred

---

javax.servlet

## Class ServletInputStream

```
java.lang.Object
|
+--java.io.InputStream
|
+--javax.servlet.ServletInputStream
```

---

public abstract class **ServletInputStream**

extends java.io.InputStream

An input stream for reading servlet requests, it provides an efficient readLine method. This is an abstract class, to be implemented by a network services writer. For some application protocols, such as the HTTP POST and PUT methods, servlet writers use the input stream to get data from clients. They access the input stream via the ServletRequest's getInputStream method, available from within the servlet's service

method. Subclasses of `ServletInputStream` must provide an implementation of the `read()` method.

**See Also:**

`InputStream.read()`

---

<b>Constructor Summary</b>	
protected	<a href="#">ServletInputStream()</a> The default constructor does no work.

<b>Method Summary</b>	
int	<a href="#">readLine</a> (byte[] b, int off, int len) Starting at the specified offset, reads into the given array of bytes until all requested bytes have been read or a '\n' is encountered, in which case the '\n' is read into the array as well.

<b>Methods inherited from class java.io.InputStream</b>
available, close, mark, markSupported, read, read, read, reset, skip

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

<b>Constructor Detail</b>
---------------------------

**ServletInputStream**

protected `ServletInputStream()`

The default constructor does no work.

<b>Method Detail</b>
----------------------

**readLine**

```
public int readLine(byte[] b,  
                   int off,  
                   int len)  
    throws java.io.IOException
```

Starting at the specified offset, reads into the given array of bytes until all requested bytes have been read or a '\n' is encountered, in which case the '\n' is read into the array as well.

**Parameters:**

b - the buffer into which the data is read

off - the start offset of the data

len - the maximum number of bytes to read

**Returns:**

the actual number of bytes read, or -1 if the end of the stream is reached

**Throws:**

java.io.IOException - if an I/O error has occurred

---

javax.servlet

## Class ServletOutputStream

```
java.lang.Object
|
+-- java.io.OutputStream
    |
    +-- javax.servlet.ServletOutputStream
```

---

public abstract class **ServletOutputStream**

extends java.io.OutputStream

An output stream for writing servlet responses. This is an abstract class, to be implemented by a network services implementor. Servlet writers use the output stream to return data to clients. They access it via the ServletResponse's `getOutputStream` method, available from within the servlet's service method. Subclasses of `ServletOutputStream` must provide an implementation of the `write(int)` method.

**See Also:**

OutputStream.write(int)

---

### Constructor Summary

protected	<a href="#"><u>ServletOutputStream()</u></a> The default constructor does no work.
-----------	---

---

### Method Summary

void	<a href="#"><u>print</u></a> (boolean b) Prints the boolean provided.
void	<a href="#"><u>print</u></a> (char c) Prints the character provided.

void	<a href="#"><u>print</u></a> (double d) Prints the double provided.
void	<a href="#"><u>print</u></a> (float f) Prints the float provided.
void	<a href="#"><u>print</u></a> (int i) Prints the integer provided.
void	<a href="#"><u>print</u></a> (long l) Prints the long provided.
void	<a href="#"><u>print</u></a> (java.lang.String s) Prints the string provided.
void	<a href="#"><u>println</u></a> () Prints a CRLF.
void	<a href="#"><u>println</u></a> (boolean b) Prints the boolean provided, followed by a CRLF.
void	<a href="#"><u>println</u></a> (char c) Prints the character provided, followed by a CRLF.
void	<a href="#"><u>println</u></a> (double d) Prints the double provided, followed by a CRLF.
void	<a href="#"><u>println</u></a> (float f) Prints the float provided, followed by a CRLF.
void	<a href="#"><u>println</u></a> (int i) Prints the integer provided, followed by a CRLF.
void	<a href="#"><u>println</u></a> (long l) Prints the long provided, followed by a CRLF.
void	<a href="#"><u>println</u></a> (java.lang.String s) Prints the string provided, followed by a CRLF.

<b>Methods inherited from class java.io.OutputStream</b>
close, flush, write, write, write

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

<b>Constructor Detail</b>
---------------------------

### ServletOutputStream

protected `ServletOutputStream()`

The default constructor does no work.

<b>Method Detail</b>
----------------------

## **print**

```
public void print(java.lang.String s)
    throws java.io.IOException
```

Prints the string provided.

### **Throws:**

java.io.IOException - if an I/O error has occurred

---

## **print**

```
public void print(boolean b)
    throws java.io.IOException
```

Prints the boolean provided.

### **Throws:**

java.io.IOException - if an I/O error has occurred.

---

## **print**

```
public void print(char c)
    throws java.io.IOException
```

Prints the character provided.

### **Throws:**

java.io.IOException - if an I/O error has occurred

---

## **print**

```
public void print(int i)
    throws java.io.IOException
```

Prints the integer provided.

### **Throws:**

java.io.IOException - if an I/O error has occurred

---

## **print**

```
public void print(long l)
    throws java.io.IOException
```

Prints the long provided.

---

**Throws:**

java.io.IOException - if an I/O error has occurred

---

**print**

```
public void print(float f)
    throws java.io.IOException
```

Prints the float provided.

**Throws:**

java.io.IOException - if an I/O error has occurred

---

**print**

```
public void print(double d)
    throws java.io.IOException
```

Prints the double provided.

**Throws:**

java.io.IOException - if an I/O error has occurred

---

**println**

```
public void println()
    throws java.io.IOException
```

Prints a CRLF.

**Throws:**

java.io.IOException - if an I/O error has occurred

---

**println**

```
public void println(java.lang.String s)
    throws java.io.IOException
```

Prints the string provided, followed by a CRLF.

**Throws:**

java.io.IOException - if an I/O error has occurred

---

## **println**

```
public void println(boolean b)
    throws java.io.IOException
```

Prints the boolean provided, followed by a CRLF.

### **Throws:**

java.io.IOException - if an I/O error has occurred.

---

## **println**

```
public void println(char c)
    throws java.io.IOException
```

Prints the character provided, followed by a CRLF.

### **Throws:**

java.io.IOException - if an I/O error has occurred

---

## **println**

```
public void println(int i)
    throws java.io.IOException
```

Prints the integer provided, followed by a CRLF.

### **Throws:**

java.io.IOException - if an I/O error has occurred

---

## **println**

```
public void println(long l)
    throws java.io.IOException
```

Prints the long provided, followed by a CRLF.

### **Throws:**

java.io.IOException - if an I/O error has occurred

---

## **println**

```
public void println(float f)
    throws java.io.IOException
```

---

Prints the float provided, followed by a CRLF.

**Throws:**

java.io.IOException - if an I/O error has occurred

---

**println**

```
public void println(double d)
    throws java.io.IOException
```

Prints the double provided, followed by a CRLF.

**Throws:**

java.io.IOException - if an I/O error has occurred

---

javax.servlet

## Class ServletException

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- javax.servlet.ServletException
```

**Direct Known Subclasses:**

[UnavailableException](#)

---

public class **ServletException**

extends java.lang.Exception

This exception is thrown to indicate a servlet problem.

**See Also:**

[Serialized Form](#)

---

<b>Constructor Summary</b>	
<a href="#">ServletException</a> ()	
Constructs a new ServletException.	
<a href="#">ServletException</a> (java.lang.String message)	
Constructs a new ServletException with the specified message.	

---

<a href="#">ServletException</a> (java.lang.String message, java.lang.Throwable rootCause)	
Constructs a new ServletException with the specified message and root cause.	
<a href="#">ServletException</a> (java.lang.Throwable rootCause)	
Constructs a new ServletException with the specified message and root cause.	

<b>Method Summary</b>	
java.lang.Throwable	<a href="#">getRootCause</a> ()
	Returns the root cause of this exception.

<b>Methods inherited from class java.lang.Throwable</b>
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

<b>Methods inherited from class java.lang.Object</b>
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

## Constructor Detail

### ServletException

```
public ServletException()
```

Constructs a new ServletException.

---

### ServletException

```
public ServletException(java.lang.String message)
```

Constructs a new ServletException with the specified message.

#### Parameters:

message - Message of exception

---

### ServletException

```
public ServletException(java.lang.String message,
                       java.lang.Throwable rootCause)
```

Constructs a new ServletException with the specified message and root cause.

#### Parameters:

message - Message of exception

rootCause - Exception that caused this exception to be raised

---

---

## ServletException

public **ServletException**(java.lang.Throwable rootCause)

Constructs a new ServletException with the specified message and root cause.

### Parameters:

rootCause - Exception that caused this exception to be raised

## Method Detail

### getRootCause

public java.lang.Throwable **getRootCause**()

Returns the root cause of this exception.

### Returns:

Throwable

---

javax.servlet

## Class UnavailableException

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- javax.servlet.ServletException
            |
            +-- javax.servlet.UnavailableException
```

---

public class **UnavailableException**

extends [ServletException](#)

This exception indicates that a servlet is unavailable. Servlets may report this exception at any time, and the network service running the servlet should behave appropriately. There are two types of unavailability, and sophisticated services will to deal with these differently:

- Permanent unavailability. The servlet will not be able to handle client requests until some administrative action is taken to correct a servlet problem. For example, the servlet might be misconfigured, or the state of the servlet may be corrupted. Well written servlets
-

will log both the error and the corrective action which an administrator must perform to let the servlet become available.

- Temporary unavailability. The servlet can not handle requests at this moment due to a 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. The problem may be self correcting, such as those due to excessive load, or corrective action may need to be taken by an administrator.

Network services may safely treat both types of exceptions as "permanent", but good treatment of temporary unavailability leads to more robust network services. Specifically, requests to the servlet might be blocked (or otherwise deferred) for a servlet-suggested amount of time, rather than being rejected until the service itself restarts.

**See Also:**

[Serialized Form](#)

<b>Constructor Summary</b>	
<a href="#">UnavailableException</a> (int seconds, <a href="#">Servlet</a> servlet, java.lang.String msg)	
Constructs a new exception with the specified descriptive message, indicating that the servlet is temporarily unavailable and giving an estimate of how long it will be unavailable.	
<a href="#">UnavailableException</a> ( <a href="#">Servlet</a> servlet, java.lang.String msg)	
Constructs a new exception with the specified descriptive message, indicating that the servlet is permanently unavailable.	

<b>Method Summary</b>	
<a href="#">Servlet</a>	<a href="#">getServlet</a> () Returns the servlet that is reporting its unavailability.
int	<a href="#">getUnavailableSeconds</a> () Returns the amount of time the servlet expects to be temporarily unavailable.
boolean	<a href="#">isPermanent</a> () Returns true if the servlet is "permanently" unavailable, indicating that the service administrator must take some corrective action to make the servlet be usable.

<b>Methods inherited from class javax.servlet.<a href="#">ServletException</a></b>
<a href="#">getRootCause</a>

<b>Methods inherited from class java.lang.Throwable</b>
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

<b>Methods inherited from class java.lang.Object</b>
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

## Constructor Detail

### UnavailableException

```
public UnavailableException(Servlet servlet,  
                           java.lang.String msg)
```

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

#### Parameters:

servlet - the servlet which is unavailable

msg - the descriptive message

---

### UnavailableException

```
public UnavailableException(int seconds,  
                           Servlet servlet,  
                           java.lang.String msg)
```

Constructs a new exception with the specified descriptive message, indicating that the servlet is temporarily unavailable and giving an estimate of how long it will be unavailable. In some cases, no estimate can be made; this is indicated by a non-positive time. For example, the servlet might know a server it needs is "down", but not be able to report how long it will take to restore it to an adequate level of functionality.

#### Parameters:

seconds - number of seconds that the servlet is anticipated to be unavailable. If negative or zero, no estimate is available.

servlet - the servlet which is unavailable

msg - the descriptive message

---

## Method Detail

### isPermanent

```
public boolean isPermanent()
```

Returns true if the servlet is "permanently" unavailable, indicating that the service administrator must take some corrective action to make the servlet be usable.

---

## **getServlet**

public [Servlet](#) getServlet()

Returns the servlet that is reporting its unavailability.

---

## **getUnavailableSeconds**

public int getUnavailableSeconds()

Returns the amount of time the servlet expects to be temporarily unavailable. If the servlet is permanently unavailable, or no estimate was provided, returns a negative number. No effort is made to correct for the time elapsed since the exception was first reported.

---

## Package javax.servlet.http

Interface Summary	
<a href="#"><u>HttpServletRequest</u></a>	An HTTP servlet request.
<a href="#"><u>HttpServletResponse</u></a>	An HTTP servlet response.
<a href="#"><u>HttpSession</u></a>	The HttpSession interface is implemented by services to provide an association between an HTTP client and HTTP server.
<a href="#"><u>HttpSessionBindingListener</u></a>	Objects implement this interface so that they can be notified when they are being bound or unbound from a HttpSession.
<a href="#"><u>HttpSessionContext</u></a>	<b>Deprecated.</b> <i>The HttpSessionContext class has been deprecated for security reasons.</i>

Class Summary	
<a href="#"><u>Cookie</u></a>	This class represents a "Cookie", as used for session management with HTTP and HTTPS protocols.
<a href="#"><u>HttpServlet</u></a>	An abstract class that simplifies writing HTTP servlets.
<a href="#"><u>HttpSessionBindingEvent</u></a>	This event is communicated to a HttpSessionBindingListener whenever the listener is bound to or unbound from a HttpSession value.
<a href="#"><u>HttpUtils</u></a>	A collection of static utility methods useful to HTTP servlets.

javax.servlet.http

## Interface HttpServletRequest

public abstract interface **HttpServletRequest**

extends [ServletRequest](#)

An HTTP servlet request. This interface gets data from the client to the servlet for use in the `HttpServletRequest.service` method. It allows the HTTP-protocol specified header information to be accessed from the `service` method. This interface is implemented by network-service developers for use within servlets.

Method Summary	
<small>java.lang.String</small>	<a href="#"><u>getAuthType()</u></a> Gets the authentication scheme of this request.
<small>Cookie[]</small>	<a href="#"><u>getCookies()</u></a> Gets the array of cookies found in this request.

long	<a href="#"><u>getDateHeader</u></a> (java.lang.String name) Gets the value of the requested date header field of this request.
java.lang.String	<a href="#"><u>getHeader</u></a> (java.lang.String name) Gets the value of the requested header field of this request.
java.util.Enumeration	<a href="#"><u>getHeaderNames</u></a> () Gets the header names for this request.
int	<a href="#"><u>getIntHeader</u></a> (java.lang.String name) Gets the value of the specified integer header field of this request.
java.lang.String	<a href="#"><u>getMethod</u></a> () Gets the HTTP method (for example, GET, POST, PUT) with which this request was made.
java.lang.String	<a href="#"><u>getPathInfo</u></a> () Gets any optional extra path information following the servlet path of this request's URI, but immediately preceding its query string.
java.lang.String	<a href="#"><u>getPathTranslated</u></a> () Gets any optional extra path information following the servlet path of this request's URI, but immediately preceding its query string, and translates it to a real path.
java.lang.String	<a href="#"><u>getQueryString</u></a> () Gets any query string that is part of the HTTP request URI.
java.lang.String	<a href="#"><u>getRemoteUser</u></a> () Gets the name of the user making this request.
java.lang.String	<a href="#"><u>getRequestSessionId</u></a> () Gets the session id specified with this request.
java.lang.String	<a href="#"><u>getRequestURI</u></a> () Gets, from the first line of the HTTP request, the part of this request's URI that is to the left of any query string.
java.lang.String	<a href="#"><u>getServletPath</u></a> () Gets the part of this request's URI that refers to the servlet being invoked.
<a href="#"><u>HttpSession</u></a>	<a href="#"><u>getSession</u></a> () Gets the current valid session associated with this request, if create is false or, if necessary, creates a new session for the request.
<a href="#"><u>HttpSession</u></a>	<a href="#"><u>getSession</u></a> (boolean create) Gets the current valid session associated with this request, if create is false or, if necessary, creates a new session for the request, if create is true.
boolean	<a href="#"><u>isRequestedSessionIdFromCookie</u></a> () Checks whether the session id specified by this request came in as a cookie.
boolean	<a href="#"><u>isRequestedSessionIdFromUrl</u></a> () <b>Deprecated.</b> <i>use <a href="#"><u>isRequestSessionIdFromURL</u></a>() instead</i>
boolean	<a href="#"><u>isRequestedSessionIdFromURL</u></a> () Checks whether the session id specified by this request came in as part of the URL.
boolean	<a href="#"><u>isRequestedSessionIdValid</u></a> () Checks whether this request is associated with a session that is valid in the current session

context.

### Methods inherited from interface [javax.servlet.ServletRequest](#)

[getAttribute](#), [getAttributeNames](#), [getCharacterEncoding](#),  
[getContentTypeLength](#), [getContentType](#), [getInputStream](#), [getParameter](#),  
[getParameterNames](#), [getParameterValues](#), [getProtocol](#), [getReader](#),  
[getRealPath](#), [getRemoteAddr](#), [getRemoteHost](#), [getScheme](#), [getServerName](#),  
[getServerPort](#), [setAttribute](#)

## Method Detail

### **getAuthType**

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

Gets the authentication scheme of this request. Same as the CGI variable AUTH\_TYPE.

**Returns:**

this request's authentication scheme, or null if none.

---

### **getCookies**

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

Gets the array of cookies found in this request.

**Returns:**

the array of cookies found in this request

---

### **getDateHeader**

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

Gets the value of the requested date header field of this request. If the header can't be converted to a date, the method throws an `IllegalArgumentException`. The case of the header field name is ignored.

**Parameters:**

name - the String containing the name of the requested header field

**Returns:**

the value the requested date header field, or -1 if not found.

---

## **getHeader**

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

Gets the value of the requested header field of this request. The case of the header field name is ignored.

### **Parameters:**

name - the String containing the name of the requested header field

### **Returns:**

the value of the requested header field, or null if not known.

---

## **getHeaderNames**

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

Gets the header names for this request.

### **Returns:**

an enumeration of strings representing the header names for this request. Some server implementations do not allow headers to be accessed in this way, in which case this method will return null.

---

## **getIntHeader**

```
public int getIntHeader(java.lang.String name)
```

Gets the value of the specified integer header field of this request. The case of the header field name is ignored. If the header can't be converted to an integer, the method throws a `NumberFormatException`.

### **Parameters:**

name - the String containing the name of the requested header field

### **Returns:**

the value of the requested header field, or -1 if not found.

---

## **getMethod**

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

Gets the HTTP method (for example, GET, POST, PUT) with which this request was

---

made. Same as the CGI variable REQUEST\_METHOD.

**Returns:**

the HTTP method with which this request was made

---

**getPathInfo**

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

Gets any optional extra path information following the servlet path of this request's URI, but immediately preceding its query string. Same as the CGI variable PATH\_INFO.

**Returns:**

the optional path information following the servlet path, but before the query string, in this request's URI; null if this request's URI contains no extra path information

---

**getPathTranslated**

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

Gets any optional extra path information following the servlet path of this request's URI, but immediately preceding its query string, and translates it to a real path. Same as the CGI variable PATH\_TRANSLATED.

**Returns:**

extra path information translated to a real path or null if no extra path information is in the request's URI

---

**getQueryString**

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

Gets any query string that is part of the HTTP request URI. Same as the CGI variable QUERY\_STRING.

**Returns:**

query string that is part of this request's URI, or null if it contains no query string

---

**getRemoteUser**

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

Gets the name of the user making this request. The user name is set with HTTP

---

authentication. Whether the user name will continue to be sent with each subsequent communication is browser-dependent. Same as the CGI variable REMOTE\_USER.

**Returns:**

the name of the user making this request, or null if not known.

---

### **getRequestedSessionId**

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

Gets the session id specified with this request. This may differ from the actual session id. For example, if the request specified an id for an invalid session, then this will get a new session with a new id.

**Returns:**

the session id specified by this request, or null if the request did not specify a session id

**See Also:**

[isRequestedSessionIdValid\(\)](#)

---

### **getRequestURI**

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

Gets, from the first line of the HTTP request, the part of this request's URI that is to the left of any query string. For example,

<b>First line of HTTP request</b>	<b>Return from <code>getRequestURI</code></b>
POST /some/path.html HTTP/1.1	/some/path.html
GET http://foo.bar/a.html HTTP/1.0	http://foo.bar/a.html
HEAD /xyz?a=b HTTP/1.1	/xyz

To reconstruct a URL with a URL scheme and host, use the method `javax.servlet.http.HttpUtils.getRequestURL`, which returns a `StringBuffer`.

**Returns:**

this request's URI

**See Also:**

[HttpUtils.getRequestURL\(javax.servlet.http.HttpServletRequest\)](#)

---

### **getServletPath**

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

---

Gets the part of this request's URI that refers to the servlet being invoked. Analogous to the CGI variable SCRIPT\_NAME.

**Returns:**

the servlet being invoked, as contained in this request's URI

---

## getSession

```
public HttpSession getSession(boolean create)
```

Gets the current valid session associated with this request, if create is false or, if necessary, creates a new session for the request, if create is true.

**Note:** to ensure the session is properly maintained, the servlet developer must call this method (at least once) before any output is written to the response.

Additionally, application-writers need to be aware that newly created sessions (that is, sessions for which `HttpSession.isNew` returns true) do not have any application-specific state.

**Returns:**

the session associated with this request or null if create was false and no valid session is associated with this request.

---

## getSession

```
public HttpSession getSession()
```

Gets the current valid session associated with this request, if create is false or, if necessary, creates a new session for the request.

---

## isRequestedSessionIdValid

```
public boolean isRequestedSessionIdValid()
```

Checks whether this request is associated with a session that is valid in the current session context. If it is not valid, the requested session will never be returned from the `getSession` method.

**Returns:**

true if this request is associated with a session that is valid in the current session context.

**See Also:**

[getRequestedSessionId\(\)](#), [HttpSessionContext](#), [getSession\(boolean\)](#)

---

## isRequestedSessionIdFromCookie

```
public boolean isRequestedSessionIdFromCookie()
```

Checks whether the session id specified by this request came in as a cookie. (The requested session may not be one returned by the `getSession` method.)

### Returns:

true if the session id specified by this request came in as a cookie; false otherwise

### See Also:

[getSession\(boolean\)](#)

---

## isRequestedSessionIdFromURL

```
public boolean isRequestedSessionIdFromURL()
```

Checks whether the session id specified by this request came in as part of the URL. (The requested session may not be the one returned by the `getSession` method.)

### Returns:

true if the session id specified by the request for this session came in as part of the URL;  
false otherwise

### See Also:

[getSession\(boolean\)](#)

---

## isRequestedSessionIdFromUrl

```
public boolean isRequestedSessionIdFromUrl()
```

**Deprecated.** *use `isRequestSessionIdFromURL()` instead*

Checks whether the session id specified by this request came in as part of the URL. (The requested session may not be the one returned by the `getSession` method.)

### Returns:

true if the session id specified by the request for this session came in as part of the URL;  
false otherwise

### See Also:

[getSession\(boolean\)](#)

---

## Interface `HttpServletResponse`

public abstract interface `HttpServletResponse`

extends [ServletResponse](#)

An HTTP servlet response. This interface allows a servlet's `service` method to manipulate HTTP-protocol specified header information and return data to its client. It is implemented by network service developers for use within servlets.

### Field Summary

static int	<a href="#">SC_ACCEPTED</a> Status code (202) indicating that a request was accepted for processing, but was not completed.
static int	<a href="#">SC_BAD_GATEWAY</a> 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	<a href="#">SC_BAD_REQUEST</a> Status code (400) indicating the request sent by the client was syntactically incorrect.
static int	<a href="#">SC_CONFLICT</a> Status code (409) indicating that the request could not be completed due to a conflict with the current state of the resource.
static int	<a href="#">SC_CONTINUE</a> Status code (100) indicating the client can continue.
static int	<a href="#">SC_CREATED</a> Status code (201) indicating the request succeeded and created a new resource on the server.
static int	<a href="#">SC_FORBIDDEN</a> Status code (403) indicating the server understood the request but refused to fulfill it.
static int	<a href="#">SC_GATEWAY_TIMEOUT</a> 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	<a href="#">SC_GONE</a> Status code (410) indicating that the resource is no longer available at the server and no forwarding address is known.
static int	<a href="#">SC_HTTP_VERSION_NOT_SUPPORTED</a> 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	<a href="#">SC_INTERNAL_SERVER_ERROR</a> Status code (500) indicating an error inside the HTTP server which prevented it from fulfilling the request.

static int	<a href="#"><u>SC_LENGTH_REQUIRED</u></a> Status code (411) indicating that the request cannot be handled without a defined Content-Length.
static int	<a href="#"><u>SC_METHOD_NOT_ALLOWED</u></a> 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	<a href="#"><u>SC_MOVED_PERMANENTLY</u></a> 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.
static int	<a href="#"><u>SC_MOVED_TEMPORARILY</u></a> 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.
static int	<a href="#"><u>SC_MULTIPLE_CHOICES</u></a> 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	<a href="#"><u>SC_NO_CONTENT</u></a> Status code (204) indicating that the request succeeded but that there was no new information to return.
static int	<a href="#"><u>SC_NON_AUTHORITATIVE_INFORMATION</u></a> Status code (203) indicating that the meta information presented by the client did not originate from the server.
static int	<a href="#"><u>SC_NOT_ACCEPTABLE</u></a> 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 headers sent in the request.
static int	<a href="#"><u>SC_NOT_FOUND</u></a> Status code (404) indicating that the requested resource is not available.
static int	<a href="#"><u>SC_NOT_IMPLEMENTED</u></a> Status code (501) indicating the HTTP server does not support the functionality needed to fulfill the request.
static int	<a href="#"><u>SC_NOT_MODIFIED</u></a> Status code (304) indicating that a conditional GET operation found that the resource was available and not modified.
static int	<a href="#"><u>SC_OK</u></a> Status code (200) indicating the request succeeded normally.
static int	<a href="#"><u>SC_PARTIAL_CONTENT</u></a> Status code (206) indicating that the server has fulfilled the partial GET request for the resource.
static int	<a href="#"><u>SC_PAYMENT_REQUIRED</u></a> Status code (402) reserved for future use.
static int	<a href="#"><u>SC_PRECONDITION_FAILED</u></a> 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.
static int	<a href="#"><u>SC_PROXY_AUTHENTICATION_REQUIRED</u></a>

	Status code (407) indicating that the client <b>MUST</b> first authenticate itself with the proxy.
static int	<a href="#"><u>SC_REQUEST_ENTITY_TOO_LARGE</u></a> 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	<a href="#"><u>SC_REQUEST_TIMEOUT</u></a> Status code (408) indicating that the client did not produce a request within the time that the server was prepared to wait.
static int	<a href="#"><u>SC_REQUEST_URI_TOO_LONG</u></a> 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.
static int	<a href="#"><u>SC_RESET_CONTENT</u></a> Status code (205) indicating that the agent <b>SHOULD</b> reset the document view which caused the request to be sent.
static int	<a href="#"><u>SC_SEE_OTHER</u></a> Status code (303) indicating that the response to the request can be found under a different URI.
static int	<a href="#"><u>SC_SERVICE_UNAVAILABLE</u></a> Status code (503) indicating that the HTTP server is temporarily overloaded, and unable to handle the request.
static int	<a href="#"><u>SC_SWITCHING_PROTOCOLS</u></a> Status code (101) indicating the server is switching protocols according to Upgrade header.
static int	<a href="#"><u>SC_UNAUTHORIZED</u></a> Status code (401) indicating that the request requires HTTP authentication.
static int	<a href="#"><u>SC_UNSUPPORTED_MEDIA_TYPE</u></a> 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	<a href="#"><u>SC_USE_PROXY</u></a> Status code (305) indicating that the requested resource <b>MUST</b> be accessed through the proxy given by the Location field.

<b>Method Summary</b>	
void	<a href="#"><u>addCookie</u></a> ( <a href="#"><u>Cookie</u></a> cookie) Adds the specified cookie to the response.
boolean	<a href="#"><u>containsHeader</u></a> (java.lang.String name) Checks whether the response message header has a field with the specified name.
java.lang. String	<a href="#"><u>encodeRedirectUrl</u></a> (java.lang.String url) <b>Deprecated.</b> Use <i>encodeRedirectURL(String url)</i>
java.lang. String	<a href="#"><u>encodeRedirectURL</u></a> (java.lang.String url) Encodes the specified URL for use in the <code>sendRedirect</code> method or, if encoding is not needed, returns the URL unchanged.
java.lang. String	<a href="#"><u>encodeUrl</u></a> (java.lang.String url)

	<b>Deprecated.</b> Use <code>encodeURL(String url)</code>
java.lang. String	<a href="#"><u>encodeURL</u></a> (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	<a href="#"><u>sendError</u></a> (int sc) Sends an error response to the client using the specified status code and a default message.
void	<a href="#"><u>sendError</u></a> (int sc, java.lang.String msg) Sends an error response to the client using the specified status code and descriptive message.
void	<a href="#"><u>sendRedirect</u></a> (java.lang.String location) Sends a temporary redirect response to the client using the specified redirect location URL.
void	<a href="#"><u>setDateHeader</u></a> (java.lang.String name, long date) Adds a field to the response header with the given name and date-valued field.
void	<a href="#"><u>setHeader</u></a> (java.lang.String name, java.lang.String value) Adds a field to the response header with the given name and value.
void	<a href="#"><u>setIntHeader</u></a> (java.lang.String name, int value) Adds a field to the response header with the given name and integer value.
void	<a href="#"><u>setStatus</u></a> (int sc) Sets the status code for this response.
void	<a href="#"><u>setStatus</u></a> (int sc, java.lang.String sm) <b>Deprecated.</b> <i>ambiguous meaning. To send an error with a description page, use <code>sendError(int sc, String msg)</code>;</i>

**Methods inherited from interface [javax.servlet.ServletResponse](#)**  
[getCharacterEncoding](#), [getOutputStream](#), [getWriter](#), [setContentLength](#), [setContentType](#)

## 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\_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 headers sent 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 request within 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 process.

---

## **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\_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. It 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)
```

Checks whether the response message header has a field with the specified name.

#### Parameters:

name - the header field name

#### Returns:

true if the response message header has a field with the specified name; 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 should include 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.

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 should include the logic to determine whether the session ID needs to be encoded in the URL. Because the rules for making this determination differ from those used to decide whether to encode a normal link, this method is separate 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.

After this method is called, the response should be considered to be committed and should not be written to.

### **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.** *Use `encodeURL(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 should include 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.

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)
```

**Deprecated.** *Use `encodeRedirectURL(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 should include the logic to determine whether the session ID needs to be encoded in the URL. Because the rules for making this determination differ from those used to decide whether to encode a normal link, this method is separate 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.

---

## sendError

```
public void sendError(int sc,  
                      java.lang.String msg)  
    throws java.io.IOException
```

Sends an error response to the client using the specified status code and descriptive message. If setStatus has previously been called, it is reset to the error status code. The message is sent as the body of an HTML page, which is returned to the user to describe the problem. The page is sent with a default HTML header; the message is enclosed in simple body tags (<body></body>).

After using this method, the response should be considered to be committed and should not be written to.

### Parameters:

sc - the status code

msg - the detail message

### Throws:

java.io.IOException - If an I/O error has occurred.

---

## sendError

```
public void sendError(int sc)  
    throws java.io.IOException
```

Sends an error response to the client using the specified status code and a default message.

### Parameters:

sc - the status code

### Throws:

java.io.IOException - If an I/O error has occurred.

---

## sendRedirect

```
public void sendRedirect(java.lang.String location)  
    throws java.io.IOException
```

Sends a temporary redirect response to the client using the specified redirect location URL. The URL must be absolute (for example, https://hostname/path/file.html). Relative URLs are not permitted here.

### Parameters:

---

location - the redirect location URL

**Throws:**

java.io.IOException - If an I/O error has occurred.

---

**setDateHeader**

```
public void setDateHeader(java.lang.String name,  
                           long date)
```

Adds a field to the response header with the given name and date-valued field. The date is specified in terms of milliseconds since the epoch. If the date field 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 field

value - the header field's date value

**See Also:**

[containsHeader\(java.lang.String\)](#)

---

**setHeader**

```
public void setHeader(java.lang.String name,  
                       java.lang.String value)
```

Adds a field to the response header with the given name and value. If the field 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 field

value - the header field's value

**See Also:**

[containsHeader\(java.lang.String\)](#)

---

**setIntHeader**

```
public void setIntHeader(java.lang.String name,  
                          int value)
```

---

Adds a field to the response header with the given name and integer value. If the field 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 field

`value` - the header field's integer value

**See Also:**

[containsHeader\(java.lang.String\)](#)

---

**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, the `sendError` method should be used instead.

**Parameters:**

`sc` - the status code

**See Also:**

[sendError\(int, java.lang.String\)](#)

---

**setStatus**

```
public void setStatus(int sc,  
                    java.lang.String sm)
```

**Deprecated.** *ambiguous meaning. To send an error with a description page, use `sendError(int sc, String msg)`;*

Sets the status code and message for this response. If the field had already been set, the new value overwrites the previous one. The message is sent as the body of an HTML page, which is returned to the user to describe the problem. The page is sent with a default HTML header; the message is enclosed in simple body tags (`<body></body>`).

**Parameters:**

`sc` - the status code

`sm` - the status message

---

## Interface HttpSession

---

public abstract interface **HttpSession**

The HttpSession interface is implemented by services to provide an association between an HTTP client and HTTP server. This association, or session, persists over multiple connections and/or requests during a given time period. Sessions are used to maintain state and user identity across multiple page requests.

A session can be maintained either by using cookies or by URL rewriting.

HttpSession defines methods which store these types of data:

- Standard session properties, such as an identifier for the session, and the context for the session.
- Application layer data, accessed using this interface and stored using a dictionary-like interface.

The following code snippet illustrates getting and setting the the session data value.

```
//Get the session object - "request" represents the HTTP servlet request
HttpSession session = request.getSession(true);

//Get the session data value - an Integer object is read from
//the session, incremented, then written back to the session.
//sessiontest.counter identifies values in the session
Integer ival = (Integer) session.getValue("sessiontest.counter");
if (ival==null)
    ival = new Integer(1);
else
    ival = new Integer(ival.intValue() + 1);
session.putValue("sessiontest.counter", ival);
```

When an application layer stores or removes data from the session, the session layer checks whether the object implements HttpSessionBindingListener. If it does, then the object is notified that it has been bound or unbound from the session.

An implementation of HttpSession represents the server's view of the session. The server considers a session to be new until it has been joined by the client. Until the client joins the session, the isNew method returns true. A value of true can indicate one of these three cases:

- the client does not yet know about the session
- the session has not yet begun
- the client chooses not to join the session. This case will occur if the client supports only cookies and chooses to reject any cookies sent by the server. If the server supports URL rewriting, this case will not commonly occur.

It is the responsibility of developers to design their applications to account for situations where a client has not joined a session. For example, in the following code snippet isNew is called to determine whether a session is new. If it is, the server will require the client to start a session by directing the client to a welcome page welcomeURL where a user might be required to enter some information and send it to the server before gaining access to subsequent pages.

```
//Get the session object - "request" represents the HTTP servlet request
```

---

```

HttpSession session = request.getSession(true);

//insist that the client starts a session
//before access to data is allowed
//"response" represents the HTTP servlet response
if (session.isNew()) {
    response.sendRedirect (welcomeURL);
}

```

**See Also:**

[HttpSessionBindingListener](#), [HttpSessionContext](#)

<b>Method Summary</b>	
long	<a href="#">getCreationTime()</a> Returns the time at which this session representation was created, in milliseconds since midnight, January 1, 1970 UTC.
java.lang.String	<a href="#">getId()</a> Returns the identifier assigned to this session.
long	<a href="#">getLastAccessedTime()</a> Returns the last time the client sent a request carrying the identifier assigned to the session.
int	<a href="#">getMaxInactiveInterval()</a>
<a href="#">HttpSessionContext</a>	<a href="#">getSessionContext()</a> <b>Deprecated.</b>
java.lang.Object	<a href="#">getValue()</a> (java.lang.String name) Returns the object bound to the given name in the session's application layer data.
java.lang.String[]	<a href="#">getValueNames()</a> Returns an array of the names of all the application layer data objects bound into the session.
void	<a href="#">invalidate()</a> Causes this representation of the session to be invalidated and removed from its context.
boolean	<a href="#">isNew()</a> A session is considered to be "new" if it has been created by the server, but the client has not yet acknowledged joining the session.
void	<a href="#">putValue()</a> (java.lang.String name, java.lang.Object value) Binds the specified object into the session's application layer data with the given name.
void	<a href="#">removeValue()</a> (java.lang.String name) Removes the object bound to the given name in the session's application layer data.
void	<a href="#">setMaxInactiveInterval()</a> (int interval)

	Sets the maximum interval between requests that this session will be kept by the host server.
--	---

## Method Detail

### **getCreationTime**

```
public long getCreationTime()
```

Returns the time at which this session representation was created, in milliseconds since midnight, January 1, 1970 UTC.

**Returns:**

the time when the session was created

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

### **getId**

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

Returns the identifier assigned to this session. An HttpSession's identifier is a unique string that is created and maintained by HttpSessionContext.

**Returns:**

the identifier assigned to this session

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

### **getLastAccessedTime**

```
public long getLastAccessedTime()
```

Returns the last time the client sent a request carrying the identifier assigned to the session. Time is expressed as milliseconds since midnight, January 1, 1970 UTC. Application level operations, such as getting or setting a value associated with the session, does not affect the access time.

This information is particularly useful in session management policies. For example,

---

- a session manager could leave all sessions which have not been used in a long time in a given context.
- the sessions can be sorted according to age to optimize some task.

**Returns:**

the last time the client sent a request carrying the identifier assigned to the session

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

### **getMaxInactiveInterval**

```
public int getMaxInactiveInterval()
```

---

### **getSessionContext**

```
public HttpSessionContext getSessionContext()
```

**Deprecated.**

Returns the context in which this session is bound.

**Returns:**

the name of the context in which this session is bound

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

### **getValue**

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

Returns the object bound to the given name in the session's application layer data. Returns null if there is no such binding.

**Parameters:**

name - the name of the binding to find

**Returns:**

the value bound to that name, or null if the binding does not exist.

---

**Throws:**

IllegalStateException - if an attempt is made to access HttpSession's session data after it has been invalidated

---

**getValueNames**

```
public java.lang.String[] getValueNames()
```

Returns an array of the names of all the application layer data objects bound into the session. For example, if you want to delete all of the data objects bound into the session, use this method to obtain their names.

**Returns:**

an array containing the names of all of the application layer data objects bound into the session

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

**invalidate**

```
public void invalidate()
```

Causes this representation of the session to be invalidated and removed from its context.

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

**isNew**

```
public boolean isNew()
```

A session is considered to be "new" if it has been created by the server, but the client has not yet acknowledged joining the session. For example, if the server supported only cookie-based sessions and the client had completely disabled the use of cookies, then calls to `HttpServletRequest.getSession()` would always return "new" sessions.

**Returns:**

true if the session has been created by the server but the client has not yet acknowledged joining the session; false otherwise

---

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

**putValue**

```
public void putValue(java.lang.String name,  
                    java.lang.Object value)
```

Binds the specified object into the session's application layer data with the given name. Any existing binding with the same name is replaced. New (or existing) values that implement the HttpSessionBindingListener interface will call its valueBound() method.

**Parameters:**

name - the name to which the data object will be bound. This parameter cannot be null.

value - the data object to be bound. This parameter cannot be null.

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

**removeValue**

```
public void removeValue(java.lang.String name)
```

Removes the object bound to the given name in the session's application layer data. Does nothing if there is no object bound to the given name. The value that implements the HttpSessionBindingListener interface will call its valueUnbound() method.

**Parameters:**

name - the name of the object to remove

**Throws:**

IllegalStateException - if an attempt is made to access session data after the session has been invalidated

---

**setMaxInactiveInterval**

```
public void setMaxInactiveInterval(int interval)
```

Sets the maximum interval between requests that this session will be kept by the host server.

---

---

javax.servlet.http

## Interface HttpSessionBindingListener

---

public abstract interface **HttpSessionBindingListener**

extends java.util.EventListener

Objects implement this interface so that they can be notified when they are being bound or unbound from a HttpSession. When a binding occurs (using HttpSession.putValue) HttpSessionBindingEvent communicates the event and identifies the session into which the object is bound.

Similarly, when an unbinding occurs (using HttpSession.removeValue) HttpSessionBindingEvent communicates the event and identifies the session from which the object is unbound.

### See Also:

[HttpSession](#), [HttpSessionBindingEvent](#)

---

### Method Summary

void	<a href="#">valueBound</a> ( <a href="#">HttpSessionBindingEvent</a> event) Notifies the listener that it is being bound into a session.
void	<a href="#">valueUnbound</a> ( <a href="#">HttpSessionBindingEvent</a> event) Notifies the listener that it is being unbound from a session.

### Method Detail

#### valueBound

public void **valueBound**([HttpSessionBindingEvent](#) event)

Notifies the listener that it is being bound into a session.

#### Parameters:

event - the event identifying the session into which the listener is being bound.

---

#### valueUnbound

public void **valueUnbound**([HttpSessionBindingEvent](#) event)

Notifies the listener that it is being unbound from a session.

---

## Parameters:

event - the event identifying the session from which the listener is being unbound.

---

javax.servlet.http

## Interface HttpSessionContext

---

**Deprecated.** *The HttpSessionContext class has been deprecated for security reasons. It will be removed in a future version of the Servlet API.*

public abstract interface **HttpSessionContext**

A HttpSessionContext is a grouping of HttpSession objects associated with a single entity. This interface gives servlets access to methods for listing the IDs and for retrieving a session based on its ID.

Servlets get the HttpSessionContext object by calling the getSessionContext() method of HttpSession.

### See Also:

[HttpSession](#)

---

## Method Summary

java.util.Enumeration	<a href="#">getIds()</a> <b>Deprecated.</b> <i>This method has been deprecated for security reasons. It will be removed in a future version of the Servlet API.</i>
<a href="#">HttpSession</a>	<a href="#">getSession()</a> (java.lang.String sessionId) <b>Deprecated.</b> <i>This method has been deprecated for security reasons. It will be removed in a future version of the Servlet API.</i>

## Method Detail

### getSession

public [HttpSession](#) getSession(java.lang.String sessionId)

**Deprecated.** *This method has been deprecated for security reasons. It will be removed in a future version of the Servlet API.*

This method is deprecated and retained only for binary compatibility. It must always return null.

---

## getIds

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

**Deprecated.** *This method has been deprecated for security reasons. It will be removed in a future version of the Servlet API.*

This method is deprecated and retained only for binary compatibility. It must always return an empty enumeration.

---

javax.servlet.http

## Class Cookie

```
java.lang.Object
```

```
|  
+-- javax.servlet.http.Cookie
```

---

```
public class Cookie
```

```
extends java.lang.Object
```

```
implements java.lang.Cloneable
```

This class represents a "Cookie", as used for session management with HTTP and HTTPS protocols. Cookies are used to get user agents (web browsers etc) to hold small amounts of state associated with a user's web browsing. Common applications for cookies include storing user preferences, automating low security user signon facilities, and helping collect data used for "shopping cart" style applications.

---

Cookies are named, and have a single value. They may have optional attributes, including a comment presented to the user, path and domain qualifiers for which hosts see the cookie, a maximum age, and a version. Current web browsers often have bugs in how they treat those attributes, so interoperability can be improved by not relying on them heavily.

Cookies are assigned by servers, using fields added to HTTP response headers. In this API, cookies are saved one at a time into such HTTP response headers, using the `javax.servlet.http.HttpServletResponse.addCookie` method. User agents are expected to support twenty cookies per host, of at least four kilobytes each; use of large numbers of cookies is discouraged.

Cookies are passed back to those servers using fields added to HTTP request headers. In this API, HTTP request fields are retrieved using the cookie module's `javax.servlet.http.HttpServletRequest.getCookies` method. This returns all of the cookies found in the request. Several cookies with the same name can be returned; they have different path attributes, but those attributes will not be visible when using "old format" cookies.

Cookies affect the caching of the web pages used to set their values. At this time, none of the sophisticated HTTP/1.1 cache control models are supported by this class. Standard HTTP/1.0 caches will not cache pages which contain cookies created by this class.

---

Cookies are being standardized by the IETF. This class supports the original Cookie specification (from Netscape Communications Corp.) as well as the updated RFC 2109 specification. By default, cookies are stored using the original specification. This promotes maximal interoperability; an updated RFC will provide better interoperability by defining a new HTTP header field for setting cookies.

## Constructor Summary

<a href="#">Cookie</a> (java.lang.String name, java.lang.String value)	
Defines a cookie with an initial name/value pair.	

## Method Summary

java.lang. Object	<a href="#">clone</a> () Returns a copy of this object.
java.lang. String	<a href="#">getComment</a> () Returns the comment describing the purpose of this cookie, or null if no such comment has been defined.
java.lang. String	<a href="#">getDomain</a> () Returns the domain of this cookie.
int	<a href="#">getMaxAge</a> () Returns the maximum specified age of the cookie.
java.lang. String	<a href="#">getName</a> () Returns the name of the cookie.
java.lang. String	<a href="#">getPath</a> () Returns the prefix of all URLs for which this cookie is targeted.
boolean	<a href="#">getSecure</a> () Returns the value of the 'secure' flag.
java.lang. String	<a href="#">getValue</a> () Returns the value of the cookie.
int	<a href="#">getVersion</a> () Returns the version of the cookie.
void	<a href="#">setComment</a> (java.lang.String purpose) If a user agent (web browser) presents this cookie to a user, the cookie's purpose will be described using this comment.
void	<a href="#">setDomain</a> (java.lang.String pattern) This cookie should be presented only to hosts satisfying this domain name pattern.
void	<a href="#">setMaxAge</a> (int expiry) Sets the maximum age of the cookie.
void	<a href="#">setPath</a> (java.lang.String uri) This cookie should be presented only with requests beginning with this URL.
void	<a href="#">setSecure</a> (boolean flag) Indicates to the user agent that the cookie should only be sent using a secure protocol

	(https).
void	<a href="#">setValue</a> (java.lang.String newValue) Sets the value of the cookie.
void	<a href="#">setVersion</a> (int v) Sets the version of the cookie protocol used when this cookie saves itself.

<b>Methods inherited from class java.lang.Object</b>
equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### Cookie

```
public Cookie(java.lang.String name,
              java.lang.String value)
```

Defines a cookie with an initial name/value pair. Names must not contain whitespace, comma, or semicolons and should only contain ASCII alphanumeric characters.

Names starting with a "\$" character are reserved by RFC 2109.

#### Parameters:

name - name of the cookie

value - value of the cookie

#### Throws:

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

## Method Detail

### setComment

```
public void setComment(java.lang.String purpose)
```

If a user agent (web browser) presents this cookie to a user, the cookie's purpose will be described using this comment. This is not supported by version zero cookies.

#### See Also:

getComment

## **getComment**

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

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

### **See Also:**

setComment

---

## **setDomain**

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

This cookie should be presented only to hosts satisfying this domain name pattern. Read RFC 2109 for specific details of the syntax. Briefly, a domain name begins with a dot (".foo.com") and means that hosts in that DNS zone ("www.foo.com", but not "a.b.foo.com") should see the cookie. By default, cookies are only returned to the host which saved them.

### **See Also:**

getDomain

---

## **getDomain**

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

Returns the domain of this cookie.

### **See Also:**

setDomain

---

## **setMaxAge**

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

Sets the maximum age of the cookie. The cookie will expire after that many seconds have passed. Negative values indicate the default behaviour: the cookie is not stored persistently, and will be deleted when the user agent (web browser) exits. A zero value causes the cookie to be deleted.

### **See Also:**

getMaxAge

---

## **getMaxAge**

```
public int getMaxAge()
```

Returns the maximum specified age of the cookie. If none was specified, a negative value is returned, indicating the default behaviour described with `setMaxAge`.

### **See Also:**

`setMaxAge`

---

## **setPath**

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

This cookie should be presented only with requests beginning with this URL. Read RFC 2109 for a specification of the default behaviour. Basically, URLs in the same "directory" as the one which set the cookie, and in subdirectories, can all see the cookie unless a different path is set.

### **See Also:**

`getPath`

---

## **getPath**

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

Returns the prefix of all URLs for which this cookie is targeted.

### **See Also:**

`setPath`

---

## **setSecure**

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

Indicates to the user agent that the cookie should only be sent using a secure protocol (https). This should only be set when the cookie's originating server used a secure protocol to set the cookie's value.

### **See Also:**

`getSecure`

---

## **getSecure**

```
public boolean getSecure()
```

---

Returns the value of the 'secure' flag.

**See Also:**

setSecure

---

**getName**

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

Returns the name of the cookie. This name may not be changed after the cookie is created.

---

**setValue**

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

Sets the value of the cookie. BASE64 encoding is suggested for use with binary values.

With version zero cookies, you need to be careful about the kinds of values you use. Values with various special characters (whitespace, brackets and parentheses, the equals sign, comma, double quote, slashes, question marks, the "at" sign, colon, and semicolon) should be avoided. Empty values may not behave the same way on all browsers.

**See Also:**

getValue

---

**getValue**

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

Returns the value of the cookie.

**See Also:**

setValue

---

**getVersion**

```
public int getVersion()
```

Returns the version of the cookie. Version 1 complies with RFC 2109, version 0 indicates the original version, as specified by Netscape. Newly constructed cookies use version 0 by default, to maximize interoperability. Cookies provided by a user agent will identify the cookie version used by the browser.

**See Also:**

---

setVersion

---

## setVersion

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

Sets the version of the cookie protocol used when this cookie saves itself. Since the IETF standards are still being finalized, consider version 1 as experimental; do not use it (yet) on production sites.

### See Also:

[getVersion](#)

---

## clone

```
public java.lang.Object clone()
```

Returns a copy of this object.

### Overrides:

clone in class [java.lang.Object](#)

---

[javax.servlet.http](#)

## Class HttpServlet

```
java.lang.Object
|
+-- javax.servlet.GenericServlet
    |
    +-- javax.servlet.http.HttpServlet
```

---

```
public abstract class HttpServlet
```

extends [GenericServlet](#)

implements [java.io.Serializable](#)

An abstract class that simplifies writing HTTP servlets. It extends the `GenericServlet` base class and provides an framework for handling the HTTP protocol. Because it is an abstract class, servlet writers must subclass it and override at least one method. The methods normally overridden are:

- `doGet`, if HTTP GET requests are supported. Overriding the `doGet` method automatically also provides support for the HEAD and conditional GET operations. Where practical, the `getLastModified` method should also be overridden, to facilitate caching the HTTP response data. This improves performance by enabling smarter conditional GET support.
  - `doPost`, if HTTP POST requests are supported.
-

- `doPut`, if HTTP PUT requests are supported.
- `doDelete`, if HTTP DELETE requests are supported.
- The lifecycle methods `init` and `destroy`, if the servlet writer needs to manage resources that are held for the lifetime of the servlet. Servlets that do not manage resources do not need to specialize these methods.
- `getServletInfo`, to provide descriptive information through a service's administrative interfaces.

Notice that the `service` method is not typically overridden. The `service` method, as provided, supports standard HTTP requests by dispatching them to appropriate methods, such as the methods listed above that have the prefix "do". In addition, the `service` method also supports the HTTP 1.1 protocol's TRACE and OPTIONS methods by dispatching to the `doTrace` and `doOptions` methods. The `doTrace` and `doOptions` methods are not typically overridden.

Servlets typically run inside multi-threaded servers; servlets must be written to handle multiple service requests simultaneously. It is the servlet writer's responsibility to synchronize access to any shared resources. Such resources include in-memory data such as instance or class variables of the servlet, as well as external components such as files, database and network connections. Information on multithreaded programming in Java can be found in the [Java Tutorial on Multithreaded Programming](#).

**See Also:**

[Serialized Form](#)

<b>Constructor Summary</b>	
<a href="#">HttpServlet()</a>	The default constructor does nothing.

<b>Method Summary</b>	
protected void	<a href="#">doDelete</a> ( <a href="#">HttpServletRequest</a> req, <a href="#">HttpServletResponse</a> resp) Performs the HTTP DELETE operation; the default implementation reports an HTTP BAD_REQUEST error.
protected void	<a href="#">doGet</a> ( <a href="#">HttpServletRequest</a> req, <a href="#">HttpServletResponse</a> resp) Performs the HTTP GET operation; the default implementation reports an HTTP BAD_REQUEST error.
protected void	<a href="#">doOptions</a> ( <a href="#">HttpServletRequest</a> req, <a href="#">HttpServletResponse</a> resp) Performs the HTTP OPTIONS operation; the default implementation of this method automatically determines what HTTP Options are supported.
protected void	<a href="#">doPost</a> ( <a href="#">HttpServletRequest</a> req, <a href="#">HttpServletResponse</a> resp) Performs the HTTP POST operation; the default implementation reports an HTTP BAD_REQUEST error.
protected void	<a href="#">doPut</a> ( <a href="#">HttpServletRequest</a> req, <a href="#">HttpServletResponse</a> resp) Performs the HTTP PUT operation; the default implementation reports an HTTP

	BAD_REQUEST error.
protected void	<a href="#">doTrace</a> ( <a href="#">HttpServletRequest</a> req, <a href="#">HttpServletResponse</a> resp) Performs the HTTP TRACE operation; the default implementation of this method causes a response with a message containing all of the headers sent in the trace request.
protected long	<a href="#">getLastModified</a> ( <a href="#">HttpServletRequest</a> req) Gets the time the requested entity was last modified; the default implementation returns a negative number, indicating that the modification time is unknown and hence should not be used for conditional GET operations or for other cache control operations as this implementation will always return the contents.
protected void	<a href="#">service</a> ( <a href="#">HttpServletRequest</a> req, <a href="#">HttpServletResponse</a> resp) This is an HTTP-specific version of the <code>Servlet.service</code> method, which accepts HTTP specific parameters.
void	<a href="#">service</a> ( <a href="#">ServletRequest</a> req, <a href="#">ServletResponse</a> res) Implements the high level <code>Servlet.service</code> method by delegating to the HTTP-specific service method.

**Methods inherited from class [javax.servlet.GenericServlet](#)**  
[destroy](#), [getInitParameter](#), [getInitParameterNames](#), [getServletConfig](#),  
[getServletContext](#), [getServletInfo](#), [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()
```

The default constructor does nothing.

## Method Detail

### doGet

```
protected void doGet(HttpServletRequest req,  
                    HttpServletResponse resp)  
    throws ServletException,  
           java.io.IOException
```

Performs the HTTP GET operation; the default implementation reports an HTTP BAD\_REQUEST error. Overriding this method to support the GET operation also automatically supports the HEAD operation. (HEAD is a GET that returns no body in the response; it just returns the request HEADER fields.)

Servlet writers who override this method should read any data from the request, set entity

headers in the response, access the writer or output stream, and, finally, write any response data. The headers that are set should include content type, and encoding. If a writer is to be used to write response data, the content type must be set before the writer is accessed. In general, the servlet implementor must write the headers before the response data because the headers can be flushed at any time after the data starts to be written.

Setting content length allows the servlet to take advantage of HTTP "connection keep alive". If content length can not be set in advance, the performance penalties associated with not using keep alives will sometimes be avoided if the response entity fits in an internal buffer.

Entity data written for a HEAD request is ignored. Servlet writers can, as a simple performance optimization, omit writing response data for HEAD methods. If no response data is to be written, then the content length field must be set explicitly.

The GET operation is expected to be safe: without any side effects for which users might be held responsible. For example, most form queries have no side effects. Requests intended to change stored data should use some other HTTP method. (There have been cases of significant security breaches reported because web-based applications used GET inappropriately.)

The GET operation is also expected to be idempotent: it can safely be repeated. This is not quite the same as being safe, but in some common examples the requirements have the same result. For example, repeating queries is both safe and idempotent (unless payment is required!), but buying something or modifying data is neither safe nor idempotent.

#### **Parameters:**

req - `HttpServletRequest` that encapsulates the request to the servlet

resp - `HttpServletResponse` that encapsulates the response from the servlet

#### **Throws:**

`java.io.IOException` - if detected when handling the request

[ServletException](#) - if the request could not be handled

#### **See Also:**

[ServletResponse.setContentType\(java.lang.String\)](#)

---

### **getLastModified**

protected long `getLastModified`([HttpServletRequest](#) req)

Gets the time the requested entity was last modified; the default implementation returns a negative number, indicating that the modification time is unknown and hence should not

be used for conditional GET operations or for other cache control operations as this implementation will always return the contents.

Implementations supporting the GET request should override this method to provide an accurate object modification time. This makes browser and proxy caches work more effectively, reducing the load on server and network resources.

**Parameters:**

req - `HttpServletRequest` that encapsulates the request to the servlet

**Returns:**

the time the requested entity was last modified, as the difference, measured in milliseconds, between that time and midnight, January 1, 1970 UTC. Negative numbers indicate this time is unknown.

---

**doPost**

```
protected void doPost(HttpServletRequest req,  
                     HttpServletResponse resp)  
    throws ServletException,  
           java.io.IOException
```

Performs the HTTP POST operation; the default implementation reports an HTTP BAD\_REQUEST error. Servlet writers who override this method should read any data from the request (for example, form parameters), set entity headers in the response, access the writer or output stream and, finally, write any response data using the servlet output stream. The headers that are set should include content type, and encoding. If a writer is to be used to write response data, the content type must be set before the writer is accessed. In general, the servlet implementor must write the headers before the response data because the headers can be flushed at any time after the data starts to be written.

If HTTP/1.1 chunked encoding is used (that is, if the transfer-encoding header is present), then the content-length header should not be set. For HTTP/1.1 communications that do not use chunked encoding and HTTP 1.0 communications, setting content length allows the servlet to take advantage of HTTP "connection keep alive". For just such communications, if content length can not be set, the performance penalties associated with not using keep alives will sometimes be avoided if the response entity fits in an internal buffer.

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. Specific examples including updating stored data or buying things online.

**Parameters:**

req - `HttpServletRequest` that encapsulates the request to the servlet

resp - HttpServletResponse that encapsulates the response from the servlet

**Throws:**

java.io.IOException - if detected when handling the request

[ServletException](#) - if the request could not be handled

**See Also:**

[ServletResponse.setContentType\(java.lang.String\)](#)

---

**doPut**

```
protected void doPut(HttpServletRequest req,  
                    HttpServletResponse resp)  
    throws ServletException,  
           java.io.IOException
```

Performs the HTTP PUT operation; the default implementation reports an HTTP BAD\_REQUEST error. The PUT operation is analogous to sending a file via FTP.

Servlet writers who override this method must respect any Content-\* headers sent with the request. (These headers include content-length, content-type, content-transfer-encoding, content-encoding, content-base, content-language, content-location, content-MD5, and content-range.) If the subclass cannot honor a content header, then it must issue an error response (501) and discard the request. For more information, see the [HTTP 1.1 RFC](#).

This method does not need to be either "safe" or "idempotent". Operations requested through PUT can have side effects for which the user can be held accountable. Although not required, servlet writers who override this method may wish to save a copy of the affected URI in temporary storage.

**Parameters:**

req - HttpServletRequest that encapsulates the request to the servlet

resp - HttpServletResponse that encapsulates the response from the servlet

**Throws:**

java.io.IOException - if detected when handling the request

[ServletException](#) - if the request could not be handled

---

**doDelete**

```
protected void doDelete(HttpServletRequest req,  
                       HttpServletResponse resp)
```

---

throws [ServletException](#),  
java.io.IOException

Performs the HTTP DELETE operation; the default implementation reports an HTTP BAD\_REQUEST error. The DELETE operation allows a client to request a URI to be removed 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 may be held accountable. Although not required, servlet writers who subclass this method may wish to save a copy of the affected URI in temporary storage.

**Parameters:**

req - HttpServletRequest that encapsulates the request to the servlet

resp - HttpServletResponse that encapsulates the response from the servlet

**Throws:**

java.io.IOException - if detected when handling the request

[ServletException](#) - if the request could not be handled

---

## doOptions

```
protected void doOptions(HttpServletRequest req,  
                        HttpServletResponse resp)  
    throws ServletException,  
           java.io.IOException
```

Performs the HTTP OPTIONS operation; the default implementation of this method automatically determines what HTTP Options are supported. For example, if a servlet writer subclasses HttpServlet and overrides the doGet method, then this method will return the following header:

Allow: GET,HEAD,TRACE,OPTIONS

This method does not need to be overridden unless the servlet implements new methods, beyond those supported by the HTTP/1.1 protocol.

**Parameters:**

req - HttpServletRequest that encapsulates the request to the servlet

resp - HttpServletResponse that encapsulates the response from the servlet

**Throws:**

java.io.IOException - if detected when handling the request

[ServletException](#) - if the request could not be handled

---

---

## doTrace

```
protected void doTrace(HttpServletRequest req,  
                      HttpServletResponse resp)  
    throws ServletException,  
           java.io.IOException
```

Performs the HTTP TRACE operation; the default implementation of this method causes a response with a message containing all of the headers sent in the trace request. This method is not typically overridden.

### Parameters:

req - [HttpServletRequest](#) that encapsulates the request to the servlet

resp - [HttpServletResponse](#) that encapsulates the response from the servlet

### Throws:

java.io.IOException - if detected when handling the request

[ServletException](#) - if the request could not be handled

---

## service

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

This is an HTTP-specific version of the `Servlet.service` method, which accepts HTTP specific parameters. This method is rarely overridden. Standard HTTP requests are supported by dispatching to Java methods specialized to implement them.

### Parameters:

req - [HttpServletRequest](#) that encapsulates the request to the servlet

resp - [HttpServletResponse](#) that encapsulates the response from the servlet

### Throws:

java.io.IOException - if detected when handling the request

[ServletException](#) - if the request could not be handled

### See Also:

[Servlet.service\(javax.servlet.ServletRequest, javax.servlet.ServletResponse\)](#)

---

## service

```
public void service(ServletRequest req,  
                   ServletResponse res)  
    throws ServletException,  
           java.io.IOException
```

Implements the high level `Servlet.service` method by delegating to the HTTP-specific service method. This method is not normally overridden.

### Parameters:

req - `ServletRequest` that encapsulates the request to the servlet

res - `ServletResponse` that encapsulates the response from the servlet

### Throws:

`java.io.IOException` - if an I/O exception has occurred

[ServletException](#) - if a servlet exception has occurred

### Overrides:

[service](#) in class [GenericServlet](#)

### See Also:

[Servlet.service\(javax.servlet.ServletRequest, javax.servlet.ServletResponse\)](#)

---

`javax.servlet.http`

## Class `HttpSessionBindingEvent`

```
java.lang.Object  
|  
+-- java.util.EventObject  
    |  
    +-- javax.servlet.http.HttpSessionBindingEvent
```

---

```
public class HttpSessionBindingEvent
```

extends `java.util.EventObject`

This event is communicated to a `HttpSessionBindingListener` whenever the listener is bound to or unbound from a `HttpSession` value.

The event's source is the `HttpSession`: binding occurs with a call to `HttpSession.putValue`; unbinding occurs with a call to `HttpSession.removeValue`.

### See Also:

[HttpSession](#), [HttpSessionBindingListener](#), [Serialized Form](#)

---

## Fields inherited from class `java.util.EventObject`

source

## Constructor Summary

[HttpSessionBindingEvent](#)([HttpSession](#) session, java.lang.String name)

Constructs a new HttpSessionBindingEvent

## Method Summary

java.lang.String

[getName](#)()

Returns the name to which the object is being bound or the name from which the object is being unbound.

[HttpSession](#)

[getSession](#)()

Returns the session into which the listener is being bound or from which the listener is being unbound.

## 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

### HttpSessionBindingEvent

```
public HttpSessionBindingEvent(HttpSession session,  
                               java.lang.String name)
```

Constructs a new HttpSessionBindingEvent

#### Parameters:

session - the session acting as the source of the event

name - the name to which the object is being bound or the name from which the object is being unbound

## Method Detail

### getName

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

Returns the name to which the object is being bound or the name from which the object is being unbound.

---

## getSession

public [HttpSession](#) getSession()

Returns the session into which the listener is being bound or from which the listener is being unbound.

---

javax.servlet.http

## Class HttpUtils

java.lang.Object

|  
+-- **javax.servlet.http.HttpUtils**

---

public class **HttpUtils**

extends java.lang.Object

A collection of static utility methods useful to HTTP servlets.

### Version:

1.15

---

## Constructor Summary

[HttpUtils](#)()

Creates an empty HttpUtils object.

## Method Summary

static java.lang. StringBuf fer	<a href="#">getRequestURL</a> ( <a href="#">HttpServletRequest</a> req) Reconstructs the URL used by the client used to make the request.
static java.util. Hashtabl e	<a href="#">parsePostData</a> (int len, <a href="#">ServletInputStream</a> in) Parses FORM data that is posted to the server using the HTTP POST method and the application/x-www-form-urlencoded mime type.
static java.util. Hashtabl e	<a href="#">parseQueryString</a> (java.lang.String s) Parses a query string and builds a hashtable of key-value pairs, where the values are arrays of strings.

## Methods inherited from class java.lang.Object

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

---

## Constructor Detail

### HttpUtils

```
public HttpUtils()
```

Creates an empty HttpUtils object.

## Method Detail

### parseQueryString

```
public static java.util.Hashtable parseQueryString(java.lang.String s)
```

Parses a query string and builds a hashtable of key-value pairs, where the values are arrays of strings. The query string should have the form of a string packaged by the GET or POST method. (For example, it should have its key-value pairs delimited by ampersands (&) and its keys separated from its values by equal signs (=).)

A key can appear one or more times in the query string. Each time a key appears, its corresponding value is inserted into its string array in the hash table. (So keys that appear once in the query string have, in the hash table, a string array of length one as their value, keys that appear twice have a string array of length two, etc.)

When the keys and values are moved into the hashtable, any plus signs (+) are returned to spaces and characters sent in hexadecimal notation (%xx) are converted back to characters.

#### Parameters:

s - query string to be parsed

#### Returns:

a hashtable built from the parsed key-value pairs; the .hashtable's values are arrays of strings

#### Throws:

java.lang.IllegalArgumentException - if the query string is invalid.

---

### parsePostData

```
public static java.util.Hashtable parsePostData(int len,  
ServletInputStream in)
```

Parses FORM data that is posted to the server using the HTTP POST method and the application/x-www-form-urlencoded mime type.

---

**Parameters:**

len - the length of the data in the input stream.

in - the input stream

**Returns:**

a hashtable of the parsed key, value pairs. Keys with multiple values have their values stored as an array of strings

**Throws:**

java.lang.IllegalArgumentException - if the POST data is invalid.

---

**getRequestURL**

```
public static java.lang.StringBuffer getRequestURL(HttpServletRequest req)
```

Reconstructs the URL used by the client used to make the request. This accounts for differences such as addressing scheme (http, https) and default ports, but does not attempt to include query parameters. Since it returns a StringBuffer, not a String, the URL can be modified efficiently (for example, by appending query parameters).

This method is useful for creating redirect messages and for reporting errors.

---

---

## A

[\*\*addCookie\(Cookie\)\*\*](#) - Method in interface javax.servlet.http.[HttpServletResponse](#)

Adds the specified cookie to the response.

---

## C

[\*\*clone\(\)\*\*](#) - Method in class javax.servlet.http.[Cookie](#)

Returns a copy of this object.

[\*\*containsHeader\(String\)\*\*](#) - Method in interface javax.servlet.http.[HttpServletResponse](#)

Checks whether the response message header has a field with the specified name.

[Cookie](#) - class javax.servlet.http.[Cookie](#).

This class represents a "Cookie", as used for session management with HTTP and HTTPS protocols.

[\*\*Cookie\(String, String\)\*\*](#) - Constructor for class javax.servlet.http.[Cookie](#)

Defines a cookie with an initial name/value pair.

---

## D

[\*\*destroy\(\)\*\*](#) - Method in interface javax.servlet.[Servlet](#)

Called by the servlet engine when the servlet is removed from service.

[\*\*destroy\(\)\*\*](#) - Method in class javax.servlet.[GenericServlet](#)

Destroys the servlet, cleaning up whatever resources are being held, and logs the destruction in the servlet log file.

[\*\*doDelete\(HttpServletRequest, HttpServletResponse\)\*\*](#) - Method in class javax.servlet.http.[HttpServlet](#)

Performs the HTTP DELETE operation; the default implementation reports an HTTP BAD\_REQUEST error.

[\*\*doGet\(HttpServletRequest, HttpServletResponse\)\*\*](#) - Method in class javax.servlet.http.[HttpServlet](#)

---

Performs the HTTP GET operation; the default implementation reports an HTTP BAD\_REQUEST error.

[doOptions\(HttpServletRequest, HttpServletResponse\)](#) - Method in class `javax.servlet.http.HttpServlet`

Performs the HTTP OPTIONS operation; the default implementation of this method automatically determines what HTTP Options are supported.

[doPost\(HttpServletRequest, HttpServletResponse\)](#) - Method in class `javax.servlet.http.HttpServlet`

Performs the HTTP POST operation; the default implementation reports an HTTP BAD\_REQUEST error.

[doPut\(HttpServletRequest, HttpServletResponse\)](#) - Method in class `javax.servlet.http.HttpServlet`

Performs the HTTP PUT operation; the default implementation reports an HTTP BAD\_REQUEST error.

[doTrace\(HttpServletRequest, HttpServletResponse\)](#) - Method in class `javax.servlet.http.HttpServlet`

Performs the HTTP TRACE operation; the default implementation of this method causes a response with a message containing all of the headers sent in the trace request.

---

## E

[encodeRedirectUrl\(String\)](#) - Method in interface `javax.servlet.http.HttpServletResponse`

**Deprecated.** Use `encodeRedirectURL(String url)`

[encodeRedirectURL\(String\)](#) - Method in interface `javax.servlet.http.HttpServletResponse`

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

[encodeUrl\(String\)](#) - Method in interface `javax.servlet.http.HttpServletResponse`

**Deprecated.** Use `encodeURL(String url)`

[encodeURL\(String\)](#) - Method in interface `javax.servlet.http.HttpServletResponse`

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

---

## F

[forward\(ServletRequest, ServletResponse\)](#) - Method in interface [javax.servlet.RequestDispatcher](#)

Used for forwarding a request from this servlet to another resource on the server.

---

## G

[GenericServlet](#) - class [javax.servlet.GenericServlet](#).

The GenericServlet class implements the Servlet interface and, for convenience, the ServletConfig interface.

[GenericServlet\(\)](#) - Constructor for class [javax.servlet.GenericServlet](#)

The default constructor does no work.

[getAttribute\(String\)](#) - Method in interface [javax.servlet.ServletContext](#)

Returns an object that is known to the context by a given name, or null if there is no such object associated with the name.

[getAttribute\(String\)](#) - Method in interface [javax.servlet.ServletRequest](#)

Returns the value of the named attribute of this request.

[getAttributeNames\(\)](#) - Method in interface [javax.servlet.ServletContext](#)

Returns an enumeration of the attribute names present in this context.

[getAttributeNames\(\)](#) - Method in interface [javax.servlet.ServletRequest](#)

Returns an enumeration of attribute names contained in this request.

[getAuthType\(\)](#) - Method in interface [javax.servlet.http.HttpServletRequest](#)

Gets the authentication scheme of this request.

[getCharacterEncoding\(\)](#) - Method in interface [javax.servlet.ServletResponse](#)

Returns the character set encoding used for this MIME body.

[getCharacterEncoding\(\)](#) - Method in interface [javax.servlet.ServletRequest](#)

Returns the character set encoding for the input of this request.

[getComment\(\)](#) - Method in class [javax.servlet.http.Cookie](#)

---

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

[getContentLength\(\)](#) - Method in interface javax.servlet.[ServletRequest](#)

Returns the size of the request entity data, or -1 if not known.

[getContentType\(\)](#) - Method in interface javax.servlet.[ServletRequest](#)

Returns the Internet Media (MIME) Type of the request entity data, or null if not known.

[getContext\(String\)](#) - Method in interface javax.servlet.[ServletContext](#)

Returns a `ServletContext` object for a particular URL path.

[getCookies\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the array of cookies found in this request.

[getCreationTime\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Returns the time at which this session representation was created, in milliseconds since midnight, January 1, 1970 UTC.

[getDateHeader\(String\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the value of the requested date header field of this request.

[getDomain\(\)](#) - Method in class javax.servlet.http.[Cookie](#)

Returns the domain of this cookie.

[getHeader\(String\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the value of the requested header field of this request.

[getHeaderNames\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the header names for this request.

[getId\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Returns the identifier assigned to this session.

[getIds\(\)](#) - Method in interface javax.servlet.http.[HttpSessionContext](#)

**Deprecated.** *This method has been deprecated for security reasons. It will be removed in a future version of the Servlet API.*

[getInitParameter\(String\)](#) - Method in interface javax.servlet.[ServletConfig](#)

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

[getInitParameter\(String\)](#) - Method in class javax.servlet.[GenericServlet](#)

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

[getInitParameterNames\(\)](#) - Method in interface javax.servlet.[ServletConfig](#)

Returns the names of the servlet's initialization parameters as an enumeration of strings, or an empty enumeration if there are no initialization parameters.

[getInitParameterNames\(\)](#) - Method in class javax.servlet.[GenericServlet](#)

Returns the names of the initialization parameters for this servlet as an enumeration of Strings, or an empty enumeration if there are no initialization parameters.

[getInputStream\(\)](#) - Method in interface javax.servlet.[ServletRequest](#)

Returns an input stream for reading binary data in the request body.

[getIntHeader\(String\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the value of the specified integer header field of this request.

[getLastAccessedTime\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Returns the last time the client sent a request carrying the identifier assigned to the session.

[getLastModified\(HttpServletRequest\)](#) - Method in class javax.servlet.http.[HttpServlet](#)

Gets the time the requested entity was last modified; the default implementation returns a negative number, indicating that the modification time is unknown and hence should not be used for conditional GET operations or for other cache control operations as this implementation will always return the contents.

[getMajorVersion\(\)](#) - Method in interface javax.servlet.[ServletContext](#)

Returns the major version of the servlet API that this servlet engine supports.

[getMaxAge\(\)](#) - Method in class javax.servlet.http.[Cookie](#)

Returns the maximum specified age of the cookie.

[getMaxInactiveInterval\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

[getMethod\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the HTTP method (for example, GET, POST, PUT) with which this request was made.

[getMimeType\(String\)](#) - Method in interface javax.servlet.[ServletContext](#)

Returns the mime type of the specified file, or null if not known.

[getMinorVersion\(\)](#) - Method in interface javax.servlet.[ServletContext](#)

Returns the minor version of the servlet API that this servlet engine supports.

[getName\(\)](#) - Method in class javax.servlet.http.[HttpSessionBindingEvent](#)

Returns the name to which the object is being bound or the name from which the object is being unbound.

[getName\(\)](#) - Method in class javax.servlet.http.[Cookie](#)

Returns the name of the cookie.

[getOutputStream\(\)](#) - Method in interface javax.servlet.[ServletResponse](#)

Returns an output stream for writing binary response data.

[getParameter\(String\)](#) - Method in interface javax.servlet.[ServletRequest](#)

Returns a string containing the lone value of the specified parameter, or null if the parameter does not exist.

[getParameterNames\(\)](#) - Method in interface javax.servlet.[ServletRequest](#)

Returns the parameter names for this request as an enumeration of strings, or an empty enumeration if there are no parameters or the input stream is empty.

[getParameterValues\(String\)](#) - Method in interface javax.servlet.[ServletRequest](#)

Returns the values of the specified parameter for the request as an array of strings, or null if the named parameter does not exist.

[getPath\(\)](#) - Method in class javax.servlet.http.[Cookie](#)

Returns the prefix of all URLs for which this cookie is targeted.

[getPathInfo\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets any optional extra path information following the servlet path of this request's URI, but immediately preceding its query string.

[getPathTranslated\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets any optional extra path information following the servlet path of this request's URI, but immediately preceding its query string, and translates it to a real path.

[getProtocol\(\)](#) - Method in interface javax.servlet.[ServletRequest](#)

Returns the protocol and version of the request as a string of the form <protocol>/<major version>.<minor version>.

[getQueryString\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets any query string that is part of the HTTP request URI.

[getReader\(\)](#) - Method in interface [javax.servlet.ServletRequest](#)

Returns a buffered reader for reading text in the request body.

[getRealPath\(String\)](#) - Method in interface [javax.servlet.ServletContext](#)

Applies alias rules to the specified virtual path in URL path format, that is, /dir/dir/file.ext.

[getRealPath\(String\)](#) - Method in interface [javax.servlet.ServletRequest](#)

**Deprecated.** *This method has been deprecated in preference to the same method found in the ServletContext interface.*

[getRemoteAddr\(\)](#) - Method in interface [javax.servlet.ServletRequest](#)

Returns the IP address of the agent that sent the request.

[getRemoteHost\(\)](#) - Method in interface [javax.servlet.ServletRequest](#)

Returns the fully qualified host name of the agent that sent the request.

[getRemoteUser\(\)](#) - Method in interface [javax.servlet.http.HttpServletRequest](#)

Gets the name of the user making this request.

[getRequestDispatcher\(String\)](#) - Method in interface [javax.servlet.ServletContext](#)

Returns a `RequestDispatcher` object for the specified URL path if the context knows of an active source (such as a servlet, JSP page, CGI script, etc) of content for the particular path.

[getRequesteSessionId\(\)](#) - Method in interface [javax.servlet.http.HttpServletRequest](#)

Gets the session id specified with this request.

[getRequestURI\(\)](#) - Method in interface [javax.servlet.http.HttpServletRequest](#)

Gets, from the first line of the HTTP request, the part of this request's URI that is to the left of any query string.

[getRequestURL\(HttpServletRequest\)](#) - Static method in class [javax.servlet.http.HttpUtils](#)

Reconstructs the URL used by the client used to make the request.

[getResource\(String\)](#) - Method in interface [javax.servlet.ServletContext](#)

Returns a URL object of a resource that is mapped to a corresponding URL path.

[getResourceAsStream\(String\)](#) - Method in interface [javax.servlet.ServletContext](#)

Returns an `InputStream` object allowing access to a resource that is mapped to a corresponding URL path.

[`getRootCause\(\)`](#) - Method in class `javax.servlet.ServletException`

Returns the root cause of this exception.

[`getScheme\(\)`](#) - Method in interface `javax.servlet.ServletRequest`

Returns the scheme of the URL used in this request, for example "http", "https", or "ftp".

[`getSecure\(\)`](#) - Method in class `javax.servlet.http.Cookie`

Returns the value of the 'secure' flag.

[`getServerInfo\(\)`](#) - Method in interface `javax.servlet.ServletContext`

Returns the name and version of the network service under which the servlet is running.

[`getServerName\(\)`](#) - Method in interface `javax.servlet.ServletRequest`

Returns the host name of the server that received the request.

[`getServerPort\(\)`](#) - Method in interface `javax.servlet.ServletRequest`

Returns the port number on which this request was received.

[`getServlet\(\)`](#) - Method in class `javax.servlet.UnavailableException`

Returns the servlet that is reporting its unavailability.

[`getServlet\(String\)`](#) - Method in interface `javax.servlet.ServletContext`

**Deprecated.** *This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.*

[`getServletConfig\(\)`](#) - Method in interface `javax.servlet.Servlet`

Returns a `ServletConfig` object, which contains any initialization parameters and startup configuration for this servlet.

[`getServletConfig\(\)`](#) - Method in class `javax.servlet.GenericServlet`

Returns a `ServletConfig` object containing any startup configuration information for this servlet.

[`getServletContext\(\)`](#) - Method in interface `javax.servlet.ServletConfig`

Returns the `ServletContext` for this servlet.

[`getServletContext\(\)`](#) - Method in class `javax.servlet.GenericServlet`

Returns a `ServletContext` object, which contains information about the network service in

which the servlet is running.

[getServletInfo\(\)](#) - Method in interface javax.servlet.[Servlet](#)

Allows the servlet to provide information about itself to the host servlet runner such as author, version, and copyright.

[getServletInfo\(\)](#) - Method in class javax.servlet.[GenericServlet](#)

Returns a string that contains information about the servlet, such as its author, version, and copyright.

[getServletNames\(\)](#) - Method in interface javax.servlet.[ServletContext](#)

**Deprecated.** *This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.*

[getServletPath\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the part of this request's URI that refers to the servlet being invoked.

[getServlets\(\)](#) - Method in interface javax.servlet.[ServletContext](#)

**Deprecated.** *This method has been deprecated for servlet lifecycle reasons. This method will be permanently removed in a future version of the Servlet API.*

[getSession\(\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the current valid session associated with this request, if create is false or, if necessary, creates a new session for the request.

[getSession\(\)](#) - Method in class javax.servlet.http.[HttpSessionBindingEvent](#)

Returns the session into which the listener is being bound or from which the listener is being unbound.

[getSession\(boolean\)](#) - Method in interface javax.servlet.http.[HttpServletRequest](#)

Gets the current valid session associated with this request, if create is false or, if necessary, creates a new session for the request, if create is true.

[getSession\(String\)](#) - Method in interface javax.servlet.http.[HttpSessionContext](#)

**Deprecated.** *This method has been deprecated for security reasons. It will be removed in a future version of the Servlet API.*

[getSessionContext\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

**Deprecated.**

[getUnavailableSeconds\(\)](#) - Method in class javax.servlet.[UnavailableException](#)

Returns the amount of time the servlet expects to be temporarily unavailable.

[getValue\(\)](#) - Method in class javax.servlet.http.[Cookie](#)

Returns the value of the cookie.

[getValue\(String\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Returns the object bound to the given name in the session's application layer data.

[getValueNames\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Returns an array of the names of all the application layer data objects bound into the session.

[getVersion\(\)](#) - Method in class javax.servlet.http.[Cookie](#)

Returns the version of the cookie.

[getWriter\(\)](#) - Method in interface javax.servlet.[ServletResponse](#)

Returns a print writer for writing formatted text responses.

---

## H

[HttpServlet](#) - class javax.servlet.http.[HttpServlet](#).

An abstract class that simplifies writing HTTP servlets.

[HttpServlet\(\)](#) - Constructor for class javax.servlet.http.[HttpServlet](#)

The default constructor does nothing.

[HttpServletRequest](#) - interface javax.servlet.http.[HttpServletRequest](#).

An HTTP servlet request.

[HttpServletResponse](#) - interface javax.servlet.http.[HttpServletResponse](#).

An HTTP servlet response.

[HttpSession](#) - interface javax.servlet.http.[HttpSession](#).

The HttpSession interface is implemented by services to provide an association between an HTTP client and HTTP server.

[HttpSessionBindingEvent](#) - class javax.servlet.http.[HttpSessionBindingEvent](#).

This event is communicated to a HttpSessionBindingListener whenever the listener is bound to or unbound from a HttpSession value.

[HttpSessionBindingEvent\(HttpSession, String\)](#) - Constructor for class

---

javax.servlet.http.[HttpSessionBindingEvent](#)

Constructs a new HttpSessionBindingEvent

[HttpSessionBindingListener](#) - interface javax.servlet.http.[HttpSessionBindingListener](#).

Objects implement this interface so that they can be notified when they are being bound or unbound from a HttpSession.

[HttpSessionContext](#) - interface javax.servlet.http.[HttpSessionContext](#).

**Deprecated.** *The HttpSessionContext class has been deprecated for security reasons. It will be removed in a future version of the Servlet API.*

[HttpUtils](#) - class javax.servlet.http.[HttpUtils](#).

A collection of static utility methods useful to HTTP servlets.

[HttpUtils\(\)](#) - Constructor for class javax.servlet.http.[HttpUtils](#)

Creates an empty HttpUtils object.

---

## I

[include\(ServletRequest, ServletResponse\)](#) - Method in interface javax.servlet.[RequestDispatcher](#)

Used for including the content generated by another server resource in the body of a response.

[init\(\)](#) - Method in class javax.servlet.[GenericServlet](#)

This method is provided as a convenience so that servlet writers do not have to worry about storing the ServletConfig object.

[init\(ServletConfig\)](#) - Method in interface javax.servlet.[Servlet](#)

Called by the web server when the Servlet is placed into service.

[init\(ServletConfig\)](#) - Method in class javax.servlet.[GenericServlet](#)

Initializes the servlet and logs the initialization.

[invalidate\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Causes this representation of the session to be invalidated and removed from its context.

[isNew\(\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

A session is considered to be "new" if it has been created by the server, but the client has

---

not yet acknowledged joining the session.

**[isPermanent\(\)](#)** - Method in class javax.servlet.[UnavailableException](#)

Returns true if the servlet is "permanently" unavailable, indicating that the service administrator must take some corrective action to make the servlet be usable.

**[isRequestedSessionIdFromCookie\(\)](#)** - Method in interface javax.servlet.http.[HttpServletRequest](#)

Checks whether the session id specified by this request came in as a cookie.

**[isRequestedSessionIdFromUrl\(\)](#)** - Method in interface javax.servlet.http.[HttpServletRequest](#)

**Deprecated.** *use `isRequestSessionIdFromURL()` instead*

**[isRequestedSessionIdFromURL\(\)](#)** - Method in interface javax.servlet.http.[HttpServletRequest](#)

Checks whether the session id specified by this request came in as part of the URL.

**[isRequestedSessionIdValid\(\)](#)** - Method in interface javax.servlet.http.[HttpServletRequest](#)

Checks whether this request is associated with a session that is valid in the current session context.

---

## J

[javax.servlet](#) - package javax.servlet

[javax.servlet.http](#) - package javax.servlet.http

---

## L

**[log\(Exception, String\)](#)** - Method in interface javax.servlet.[ServletContext](#)

**Deprecated.** *Use `log(String message, Throwable t)` instead*

**[log\(String\)](#)** - Method in interface javax.servlet.[ServletContext](#)

Logs the specified message to the context's log.

**[log\(String\)](#)** - Method in class javax.servlet.[GenericServlet](#)

Writes the class name of the servlet and the given message to the servlet log file.

---

[\*\*log\(String, Throwable\)\*\*](#) - Method in interface javax.servlet.[ServletContext](#)

Logs the specified message and a stack trace of the given `Throwable` object to the context's log.

[\*\*log\(String, Throwable\)\*\*](#) - Method in class javax.servlet.[GenericServlet](#)

Logs the message with the root cause

---

## **P**

[\*\*parsePostData\(int, ServletInputStream\)\*\*](#) - Static method in class javax.servlet.http.[HttpUtils](#)

Parses FORM data that is posted 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.[HttpUtils](#)

Parses a query string and builds a hashtable of key-value pairs, where the values are arrays of strings.

[\*\*print\(boolean\)\*\*](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the boolean provided.

[\*\*print\(char\)\*\*](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the character provided.

[\*\*print\(double\)\*\*](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the double provided.

[\*\*print\(float\)\*\*](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the float provided.

[\*\*print\(int\)\*\*](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the integer provided.

[\*\*print\(long\)\*\*](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the long provided.

[\*\*print\(String\)\*\*](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the string provided.

---

[println\(\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints a CRLF.

[println\(boolean\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the boolean provided, followed by a CRLF.

[println\(char\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the character provided, followed by a CRLF.

[println\(double\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the double provided, followed by a CRLF.

[println\(float\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the float provided, followed by a CRLF.

[println\(int\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the integer provided, followed by a CRLF.

[println\(long\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the long provided, followed by a CRLF.

[println\(String\)](#) - Method in class javax.servlet.[ServletOutputStream](#)

Prints the string provided, followed by a CRLF.

[putValue\(String, Object\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Binds the specified object into the session's application layer data with the given name.

---

## R

[readLine\(byte\[\], int, int\)](#) - Method in class javax.servlet.[ServletInputStream](#)

Starting at the specified offset, reads into the given array of bytes until all requested bytes have been read or a '\n' is encountered, in which case the '\n' is read into the array as well.

[removeAttribute\(String\)](#) - Method in interface javax.servlet.[ServletContext](#)

Removes the attribute from the context that is bound to a particular name.

[removeValue\(String\)](#) - Method in interface javax.servlet.http.[HttpSession](#)

Removes the object bound to the given name in the session's application layer data.

---

**[RequestDispatcher](#)** - interface javax.servlet.[RequestDispatcher](#).

Defines a request dispatcher object that receives request from the client and sends them to any resource (such as a servlet, CGI script, HTML file, or JSP file) available on the server.

---

## S

**[SC ACCEPTED](#)** - Static variable in interface javax.servlet.http.[HttpServletResponse](#)

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.[HttpServletResponse](#)

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.[HttpServletResponse](#)

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

**[SC CONFLICT](#)** - Static variable in interface javax.servlet.http.[HttpServletResponse](#)

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.[HttpServletResponse](#)

Status code (100) indicating the client can continue.

**[SC CREATED](#)** - Static variable in interface javax.servlet.http.[HttpServletResponse](#)

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

**[SC FORBIDDEN](#)** - Static variable in interface javax.servlet.http.[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.[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.[HttpServletResponse](#)

---

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.HttpServletResponse`

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.HttpServletResponse`

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.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.HttpServletResponse`

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.HttpServletResponse`

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.HttpServletResponse`

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.HttpServletResponse`

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.[HttpServletResponse](#)

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.[HttpServletResponse](#)

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 headers sent in the request.

**SC NOT FOUND** - Static variable in interface javax.servlet.http.[HttpServletResponse](#)

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

**SC NOT IMPLEMENTED** - Static variable in interface

javax.servlet.http.[HttpServletResponse](#)

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.[HttpServletResponse](#)

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.[HttpServletResponse](#)

Status code (200) indicating the request succeeded normally.

**SC PARTIAL CONTENT** - Static variable in interface

javax.servlet.http.[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.[HttpServletResponse](#)

Status code (402) reserved for future use.

**SC PRECONDITION FAILED** - Static variable in interface

javax.servlet.http.[HttpServletResponse](#)

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 PROXY AUTHENTICATION REQUIRED** - Static variable in interface

javax.servlet.http.[HttpServletResponse](#)

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.[HttpServletResponse](#)

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.[HttpServletResponse](#)

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

**[SC REQUEST URI TOO LONG](#)** - Static variable in interface  
javax.servlet.http.[HttpServletResponse](#)

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 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.[HttpServletResponse](#)

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.[HttpServletResponse](#)

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

**[SC UNAUTHORIZED](#)** - Static variable in interface  
javax.servlet.http.[HttpServletResponse](#)

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

**[SC UNSUPPORTED MEDIA TYPE](#)** - Static variable in interface  
javax.servlet.http.[HttpServletResponse](#)

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.HttpServletResponse`

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

**[sendError\(int\)](#)** - Method in interface `javax.servlet.http.HttpServletResponse`

Sends an error response to the client using the specified status code and a default message.

**[sendError\(int, String\)](#)** - Method in interface `javax.servlet.http.HttpServletResponse`

Sends an error response to the client using the specified status code and descriptive message.

**[sendRedirect\(String\)](#)** - Method in interface `javax.servlet.http.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`

This is an HTTP-specific version of the `Servlet.service` method, which accepts HTTP specific parameters.

**[service\(ServletRequest, ServletResponse\)](#)** - Method in interface `javax.servlet.Servlet`

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

**[service\(ServletRequest, ServletResponse\)](#)** - Method in class `javax.servlet.GenericServlet`

Carries out a single request from the client.

**[service\(ServletRequest, ServletResponse\)](#)** - Method in class `javax.servlet.http.HttpServlet`

Implements the high level `Servlet.service` method by delegating to the HTTP-specific service method.

**[Servlet](#)** - interface `javax.servlet.Servlet`.

A Servlet is a small program that runs inside a web server.

**[ServletConfig](#)** - interface `javax.servlet.ServletConfig`.

Defines an object that a servlet engine generates to pass configuration information to a

Servlet when such Servlet is initialized.

[ServletContext](#) - interface javax.servlet.[ServletContext](#).

A Servlet engine generated object that gives Servlets information about their environment.

[ServletException](#) - exception javax.servlet.[ServletException](#).

This exception is thrown to indicate a Servlet problem.

[ServletException\(\)](#) - Constructor for class javax.servlet.[ServletException](#)

Constructs a new ServletException.

[ServletException\(String\)](#) - Constructor for class javax.servlet.[ServletException](#)

Constructs a new ServletException with the specified message.

[ServletException\(String, Throwable\)](#) - Constructor for class javax.servlet.[ServletException](#)

Constructs a new ServletException with the specified message and root cause.

[ServletException\(Throwable\)](#) - Constructor for class javax.servlet.[ServletException](#)

Constructs a new ServletException with the specified message and root cause.

[ServletInputStream](#) - class javax.servlet.[ServletInputStream](#).

An input stream for reading Servlet requests, it provides an efficient readLine method.

[ServletInputStream\(\)](#) - Constructor for class javax.servlet.[ServletInputStream](#)

The default constructor does no work.

[ServletOutputStream](#) - class javax.servlet.[ServletOutputStream](#).

An output stream for writing Servlet responses.

[ServletOutputStream\(\)](#) - Constructor for class javax.servlet.[ServletOutputStream](#)

The default constructor does no work.

[ServletRequest](#) - interface javax.servlet.[ServletRequest](#).

Defines a Servlet engine generated object that enables a Servlet to get information about a client request.

[ServletResponse](#) - interface javax.servlet.[ServletResponse](#).

Interface for sending MIME data from the Servlet's service method to the client.

[setAttribute\(String, Object\)](#) - Method in interface javax.servlet.[ServletContext](#)

Binds an object to a given name in this context.

**[setAttribute\(String, Object\)](#)** - Method in interface javax.servlet.[ServletRequest](#)

This method stores an attribute in the request context; these attributes will be reset between requests.

**[setComment\(String\)](#)** - Method in class javax.servlet.http.[Cookie](#)

If a user agent (web browser) presents this cookie to a user, the cookie's purpose will be described using this comment.

**[setContentLength\(int\)](#)** - Method in interface javax.servlet.[ServletResponse](#)

Sets the content length for this response.

**[setContentType\(String\)](#)** - Method in interface javax.servlet.[ServletResponse](#)

Sets the content type for this response.

**[setDateHeader\(String, long\)](#)** - Method in interface javax.servlet.http.[HttpServletResponse](#)

Adds a field to the response header with the given name and date-valued field.

**[setDomain\(String\)](#)** - Method in class javax.servlet.http.[Cookie](#)

This cookie should be presented only to hosts satisfying this domain name pattern.

**[setHeader\(String, String\)](#)** - Method in interface javax.servlet.http.[HttpServletResponse](#)

Adds a field to the response header with the given name and value.

**[setIntHeader\(String, int\)](#)** - Method in interface javax.servlet.http.[HttpServletResponse](#)

Adds a field to the response header with the given name and integer value.

**[setMaxAge\(int\)](#)** - Method in class javax.servlet.http.[Cookie](#)

Sets the maximum age of the cookie.

**[setMaxInactiveInterval\(int\)](#)** - Method in interface javax.servlet.http.[HttpSession](#)

Sets the maximum interval between requests that this session will be kept by the host server.

**[setPath\(String\)](#)** - Method in class javax.servlet.http.[Cookie](#)

This cookie should be presented only with requests beginning with this URL.

**[setSecure\(boolean\)](#)** - Method in class javax.servlet.http.[Cookie](#)

Indicates to the user agent that the cookie should only be sent using a secure protocol

(https).

[setStatus\(int\)](#) - Method in interface javax.servlet.http.[HttpServletResponse](#)

Sets the status code for this response.

[setStatus\(int, String\)](#) - Method in interface javax.servlet.http.[HttpServletResponse](#)

**Deprecated.** *ambiguous meaning. To send an error with a description page, use `sendError(int sc, String msg)`;*

[setValue\(String\)](#) - Method in class javax.servlet.http.[Cookie](#)

Sets the value of the cookie.

[setVersion\(int\)](#) - Method in class javax.servlet.http.[Cookie](#)

Sets the version of the cookie protocol used when this cookie saves itself.

[SingleThreadModel](#) - interface javax.servlet.[SingleThreadModel](#).

Defines a "single" thread model for servlet execution.

---

## U

[UnavailableException](#) - exception javax.servlet.[UnavailableException](#).

This exception indicates that a servlet is unavailable.

[UnavailableException\(int, Servlet, String\)](#) - Constructor for class javax.servlet.[UnavailableException](#)

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

[UnavailableException\(Servlet, String\)](#) - Constructor for class javax.servlet.[UnavailableException](#)

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

---

## V

[valueBound\(HttpSessionBindingEvent\)](#) - Method in interface javax.servlet.http.[HttpSessionBindingListener](#)

---

Notifies the listener that it is being bound into a session.

**[valueUnbound\(HttpSessionBindingEvent\)](#)** - Method in interface  
javax.servlet.http.[HttpSessionBindingListener](#)

Notifies the listener that it is being unbound from a session.

---