Package javax.servlet

The javax.servlet package contains a number of classes and interfaces that describe and define the contracts between a servlet class and the runtime environment provided for an instance of such a class by a conforming servlet container.

See:
 Description

Interface Summary

AsyncContext	Class representing the execution context for an asynchronous operation that was initiated on a ServletRequest.
AsyncListener	Listener that will be notified in the event that an asynchronous operation initiated on a ServletRequest to which the listener had been added has completed, timed out, or resulted in an error.
Filter	A filter is an object that performs filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both.
FilterChain	A FilterChain is an object provided by the servlet container to the developer giving a view into the invocation chain of a filtered request for a resource.
FilterConfig	A filter configuration object used by a servlet container to pass information to a filter during initialization.
FilterRegistration	Interface through which a `Filter` may be further configured.
FilterRegistration.Dynamic	Interface through which a `Filter` registered via one of the `addFilter` methods on `ServletContext` may be further configured.
Registration	Interface through which a `Servlet` or `Filter` may be further configured.
Registration.Dynamic	Interface through which a `Servlet` or `Filter` registered via one of the `addServlet` or `addFilter` methods, respectively, on `ServletContext` may be further configured.
RequestDispatcher	Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server.
Servlet	Defines methods that all servlets must implement.
ServletConfig	A servlet configuration object used by a servlet container to pass information to a servlet during initialization.
ServletContainerInitializer	Interface which allows a library/runtime to be notified of a web application's startup phase and perform any required programmatic registration of servlets, filters, and listeners in response to it.

ServletContext	Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.
ServletContextAttributeListener	Interface for receiving notification events about ServletContext attribute changes.
ServletContextListener	Interface for receiving notification events about ServletContext lifecycle changes.
ServletRegistration	Interface through which a `Servlet` may be further configured.
ServletRegistration.Dynamic	Interface through which a `Servlet` registered via one of the `addServlet` methods on `ServletContext` may be further configured.
ServletRequest	Defines an object to provide client request information to a servlet.
ServletRequestAttributeListener	Interface for receiving notification events about ServletRequest attribute changes.
ServletRequestListener	Interface for receiving notification events about requests coming into and going out of scope of a web application.
ServletResponse	Defines an object to assist a servlet in sending a response to the client.
SessionCookieConfig	Class that may be used to configure various properties of cookies used for session tracking purposes.
SingleThreadModel	**Deprecated.** *As of Java Servlet API 2.4, with no direct replacement.*

Class Summary

AsyncEvent	Event that gets fired when the asynchronous operation initiated on a ServletRequest (via a call to `ServletRequest#startAsync` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`) has completed, timed out, or produced an error.
GenericServlet	Defines a generic, protocol-independent servlet.
HttpConstraintElement	Java Class representation of an `HttpConstraint` annotation value.
HttpMethodConstraintElement	Java Class represntation of an `HttpMethodConstraint` annotation value.
MultipartConfigElement	Java Class represntation of an `MultipartConfig` annotation value.
ServletContextAttributeEvent	Event class for notifications about changes to the attributes of the ServletContext of a web application.
ServletContextEvent	This is the event class for notifications about changes to the servlet context of a web application.
ServletInputStream	Provides an input stream for reading binary data from a client request, including an efficient `readLine` method for reading data one line at a time.
ServletOutputStream	Provides an output stream for sending binary data to the client.
ServletRequestAttributeEvent	This is the event class for notifications of changes to the attributes of the servlet request in an application.

ServletRequestEvent	Events of this kind indicate lifecycle events for a ServletRequest.
ServletRequestWrapper	Provides a convenient implementation of the ServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet.
ServletResponseWrapper	Provides a convenient implementation of the ServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet.
ServletSecurityElement	Java Class represntation of a `ServletSecurity` annotation value.

Enum Summary	
DispatcherType	Enumeration of filter dispatcher types.
SessionTrackingMode	Enumeration of session tracking modes.

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

Package javax.servlet Description

The javax.servlet package contains a number of classes and interfaces that describe and define the contracts between a servlet class and the runtime environment provided for an instance of such a class by a conforming servlet container.

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface AsyncContext

`public interface` **`AsyncContext`**

Class representing the execution context for an asynchronous operation that was initiated on a ServletRequest.

An AsyncContext is created and initialized by a call to `ServletRequest#startAsync()` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`. Repeated invocations of these methods will return the same AsyncContext instance, reinitialized as appropriate.

In the event that an asynchronous operation has timed out, the container must run through these steps:

1. Invoke, at their `onTimeout` method, all `AsyncListener` instances registered with the ServletRequest on which the asynchronous operation was initiated.
2. If none of the listeners called `complete()` or any of the `dispatch()` methods, perform an error dispatch with a status code equal to `HttpServletResponse.SC_INTERNAL_SERVER_ERROR`.
3. If no matching error page was found, or the error page did not call `complete()` or any of the `dispatch()` methods, call `complete()`.

Since:
Servlet 3.0

Field Summary

`static java.lang.String`	**`ASYNC_CONTEXT_PATH`** The name of the request attribute under which the original context path is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`
`static java.lang.String`	**`ASYNC_PATH_INFO`** The name of the request attribute under which the original path info is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`
`static java.lang.String`	**`ASYNC_QUERY_STRING`** The name of the request attribute under which the original query string is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`
`static java.lang.String`	**`ASYNC_REQUEST_URI`** The name of the request attribute under which the original request URI is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`
`static java.lang.String`	**`ASYNC_SERVLET_PATH`** The name of the request attribute under which the original servlet path is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`

Method Summary

`void`	**addListener**(`AsyncListener listener`) Registers the given `AsyncListener` with the most recent asynchronous cycle that was started by a call to one of the `ServletRequest#startAsync` methods.
`void`	**addListener**(`AsyncListener listener, ServletRequest servletRequest, ServletResponse servletResponse`) Registers the given `AsyncListener` with the most recent asynchronous cycle that was started by a call to one of the `ServletRequest#startAsync` methods.
`void`	**complete**() Completes the asynchronous operation that was started on the request that was used to initialze this AsyncContext, closing the response that was used to initialize this AsyncContext.
`<T extends AsyncListener> T`	**createListener**(`java.lang.Class<T> clazz`) Instantiates the given `AsyncListener` class.
`void`	**dispatch**() Dispatches the request and response objects of this AsyncContext to the servlet container.
`void`	**dispatch**(`ServletContext context, java.lang.String path`) Dispatches the request and response objects of this AsyncContext to the given `path` scoped to the given `context`.
`void`	**dispatch**(`java.lang.String path`) Dispatches the request and response objects of this AsyncContext to the given `path`.
`ServletRequest`	**getRequest**() Gets the request that was used to initialize this AsyncContext by calling `ServletRequest#startAsync()` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`.
`ServletResponse`	**getResponse**() Gets the response that was used to initialize this AsyncContext by calling `ServletRequest#startAsync()` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`.
`long`	**getTimeout**() Gets the timeout (in milliseconds) for this AsyncContext.
`boolean`	**hasOriginalRequestAndResponse**() Checks if this AsyncContext was initialized with the original or application-wrapped request and response objects.
`void`	**setTimeout**(`long timeout`) Sets the timeout (in milliseconds) for this AsyncContext.
`void`	**start**(`java.lang.Runnable run`) Causes the container to dispatch a thread, possibly from a managed thread pool, to run the specified `Runnable`.

Field Detail

ASYNC_REQUEST_URI

`static final java.lang.String` **`ASYNC_REQUEST_URI`**

The name of the request attribute under which the original request URI is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`

See Also:
Constant Field Values

ASYNC_CONTEXT_PATH

`static final java.lang.String` **`ASYNC_CONTEXT_PATH`**

The name of the request attribute under which the original context path is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`

See Also:
Constant Field Values

ASYNC_PATH_INFO

`static final java.lang.String` **`ASYNC_PATH_INFO`**

The name of the request attribute under which the original path info is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`

See Also:
Constant Field Values

ASYNC_SERVLET_PATH

`static final java.lang.String` **`ASYNC_SERVLET_PATH`**

The name of the request attribute under which the original servlet path is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`

See Also:
Constant Field Values

ASYNC_QUERY_STRING

`static final java.lang.String` **`ASYNC_QUERY_STRING`**

The name of the request attribute under which the original query string is made available to the target of a `dispatch(String)` or `dispatch(ServletContext,String)`

See Also:
Constant Field Values

Method Detail

getRequest

`ServletRequest getRequest()`

Gets the request that was used to initialize this AsyncContext by calling `ServletRequest#startAsync()` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`.

Returns:
the request that was used to initialize this AsyncContext

---

getResponse

`ServletResponse getResponse()`

Gets the response that was used to initialize this AsyncContext by calling `ServletRequest#startAsync()` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`.

Returns:
the response that was used to initialize this AsyncContext

---

hasOriginalRequestAndResponse

`boolean hasOriginalRequestAndResponse()`

Checks if this AsyncContext was initialized with the original or application-wrapped request and response objects.

This information may be used by filters invoked in the *outbound* direction, after a request was put into asynchronous mode, to determine whether any request and/or response wrappers that they added during their *inbound* invocation need to be preserved for the duration of the asynchronous operation, or may be released.

Returns:
true if this AsyncContext was initialized with the original request and response objects by calling `ServletRequest#startAsync()`, or if it was initialized by calling `ServletRequest#startAsync(ServletRequest, ServletResponse)`, and neither the ServletRequest nor ServletResponse arguments carried any application-provided wrappers; false otherwise

---

dispatch

`void dispatch()`

Dispatches the request and response objects of this AsyncContext to the servlet container.

If the asynchronous cycle was started with `ServletRequest#startAsync(ServletRequest,` ServletResponse), and the request passed is an instance of HttpServletRequest, then the dispatch is to the URI returned by HttpServletRequest.getRequestURI(). Otherwise, the dispatch is to the URI of the request when it was last dispatched by the container.

The following sequence illustrates how this will work:

```
// REQUEST dispatch to /url/A
AsyncContext ac = request.startAsync();
...
ac.dispatch(); // ASYNC dispatch to /url/A

// FORWARD dispatch to /url/B
getRequestDispatcher("/url/B").forward(request,response);
// Start async operation from within the target of the FORWARD
// dispatch
ac = request.startAsync();
...
ac.dispatch(); // ASYNC dispatch to /url/A

// FORWARD dispatch to /url/B
getRequestDispatcher("/url/B").forward(request,response);
// Start async operation from within the target of the FORWARD
// dispatch
ac = request.startAsync(request,response);
...
ac.dispatch(); // ASYNC dispatch to /url/B
```

This method returns immediately after passing the request and response objects to a container managed thread, on which the dispatch operation will be performed. If this method is called before the container-initiated dispatch that called startAsync has returned to the container, the dispatch operation will be delayed until after the container-initiated dispatch has returned to the container.

The dispatcher type of the request is set to DispatcherType.ASYNC. Unlike forward dispatches, the response buffer and headers will not be reset, and it is legal to dispatch even if the response has already been committed.

Control over the request and response is delegated to the dispatch target, and the response will be closed when the dispatch target has completed execution, unless ServletRequest#startAsync() or ServletRequest#startAsync(ServletRequest, ServletResponse) are called.

Any errors or exceptions that may occur during the execution of this method must be caught and handled by the container, as follows:

1. Invoke, at their onError method, all AsyncListener instances registered with the ServletRequest for which this AsyncContext was created, and make the caught Throwable available via AsyncEvent#getThrowable.
2. If none of the listeners called complete() or any of the dispatch() methods, perform an error dispatch with a status code equal to HttpServletResponse.SC_INTERNAL_SERVER_ERROR, and make the above Throwable available as the value of the RequestDispatcher.ERROR_EXCEPTION request attribute.
3. If no matching error page was found, or the error page did not call complete() or any of the dispatch() methods, call complete().

There can be at most one asynchronous dispatch operation per asynchronous cycle, which is started by a call to one of the ServletRequest#startAsync methods. Any attempt to perform an additional asynchronous dispatch operation within the same asynchronous cycle will result in an

IllegalStateException. If startAsync is subsequently called on the dispatched request, then any of the dispatch or `complete()` methods may be called.

Throws:

`IllegalStateException` - if one of the dispatch methods has been called and the startAsync method has not been called during the resulting dispatch, or if `complete()` was called

See Also:

`ServletRequest#getDispatcherType`

---

dispatch

```
void dispatch(java.lang.String path)
```

Dispatches the request and response objects of this AsyncContext to the given `path`.

The `path` parameter is interpreted in the same way as in `ServletRequest#getRequestDispatcher(String)`, within the scope of the `ServletContext` from which this AsyncContext was initialized.

All path related query methods of the request must reflect the dispatch target, while the original request URI, context path, path info, servlet path, and query string may be recovered from the `ASYNC_REQUEST_URI`, `ASYNC_CONTEXT_PATH`, `ASYNC_PATH_INFO`, `ASYNC_SERVLET_PATH`, and `ASYNC_QUERY_STRING` attributes of the request. These attributes will always reflect the original path elements, even under repeated dispatches.

There can be at most one asynchronous dispatch operation per asynchronous cycle, which is started by a call to one of the `ServletRequest#startAsync` methods. Any attempt to perform an additional asynchronous dispatch operation within the same asynchronous cycle will result in an IllegalStateException. If startAsync is subsequently called on the dispatched request, then any of the dispatch or `complete()` methods may be called.

See `dispatch()` for additional details, including error handling.

Parameters:

`path` - the path of the dispatch target, scoped to the ServletContext from which this AsyncContext was initialized

Throws:

`IllegalStateException` - if one of the dispatch methods has been called and the startAsync method has not been called during the resulting dispatch, or if `complete()` was called

See Also:

`ServletRequest#getDispatcherType`

---

dispatch

```
void dispatch(ServletContext context,
              java.lang.String path)
```

Dispatches the request and response objects of this AsyncContext to the given `path` scoped to the given `context`.

The `path` parameter is interpreted in the same way as in

`ServletRequest#getRequestDispatcher(String)`, except that it is scoped to the given `context`.

All path related query methods of the request must reflect the dispatch target, while the original request URI, context path, path info, servlet path, and query string may be recovered from the `ASYNC_REQUEST_URI`, `ASYNC_CONTEXT_PATH`, `ASYNC_PATH_INFO`, `ASYNC_SERVLET_PATH`, and `ASYNC_QUERY_STRING` attributes of the request. These attributes will always reflect the original path elements, even under repeated dispatches.

There can be at most one asynchronous dispatch operation per asynchronous cycle, which is started by a call to one of the `ServletRequest#startAsync` methods. Any attempt to perform an additional asynchronous dispatch operation within the same asynchronous cycle will result in an IllegalStateException. If startAsync is subsequently called on the dispatched request, then any of the dispatch or `complete()` methods may be called.

See `dispatch()` for additional details, including error handling.

Parameters:

`context` - the ServletContext of the dispatch target
`path` - the path of the dispatch target, scoped to the given ServletContext

Throws:

`IllegalStateException` - if one of the dispatch methods has been called and the startAsync method has not been called during the resulting dispatch, or if `complete()` was called

See Also:

`ServletRequest#getDispatcherType`

---

complete

`void` **`complete`**`()`

Completes the asynchronous operation that was started on the request that was used to initialze this AsyncContext, closing the response that was used to initialize this AsyncContext.

Any listeners of type `AsyncListener` that were registered with the ServletRequest for which this AsyncContext was created will be invoked at their `onComplete` method.

It is legal to call this method any time after a call to `ServletRequest#startAsync()` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`, and before a call to one of the `dispatch` methods of this class. If this method is called before the container-initiated dispatch that called `startAsync` has returned to the container, then the call will not take effect (and any invocations of `AsyncListener#onComplete(AsyncEvent)` will be delayed) until after the container-initiated dispatch has returned to the container.

---

start

`void` **`start`**`(java.lang.Runnable run)`

Causes the container to dispatch a thread, possibly from a managed thread pool, to run the specified `Runnable`. The container may propagate appropriate contextual information to the `Runnable`.

Parameters:

`run` - the asynchronous handler

addListener

```
void addListener(AsyncListener listener)
```

Registers the given `AsyncListener` with the most recent asynchronous cycle that was started by a call to one of the `ServletRequest#startAsync` methods.

The given AsyncListener will receive an `AsyncEvent` when the asynchronous cycle completes successfully, times out, or results in an error.

AsyncListener instances will be notified in the order in which they were added.

Parameters:

`listener` - the AsyncListener to be registered

Throws:

`IllegalStateException` - if this method is called after the container-initiated dispatch, during which one of the `ServletRequest#startAsync` methods was called, has returned to the container

addListener

```
void addListener(AsyncListener listener,
                 ServletRequest servletRequest,
                 ServletResponse servletResponse)
```

Registers the given `AsyncListener` with the most recent asynchronous cycle that was started by a call to one of the `ServletRequest#startAsync` methods.

The given AsyncListener will receive an `AsyncEvent` when the asynchronous cycle completes successfully, times out, or results in an error.

AsyncListener instances will be notified in the order in which they were added.

The given ServletRequest and ServletResponse objects will be made available to the given AsyncListener via the `getSuppliedRequest` and `getSuppliedResponse` methods, respectively, of the `AsyncEvent` delivered to it. These objects should not be read from or written to, respectively, at the time the AsyncEvent is delivered, because additional wrapping may have occurred since the given AsyncListener was registered, but may be used in order to release any resources associated with them.

Parameters:

`listener` - the AsyncListener to be registered
`servletRequest` - the ServletRequest that will be included in the AsyncEvent
`servletResponse` - the ServletResponse that will be included in the AsyncEvent

Throws:

`IllegalStateException` - if this method is called after the container-initiated dispatch, during which one of the `ServletRequest#startAsync` methods was called, has returned to the container

createListener

```
<T extends AsyncListener> T createListener(java.lang.Class<T> clazz)
                                       throws ServletException
```

Instantiates the given `AsyncListener` class.

The returned AsyncListener instance may be further customized before it is registered with this AsyncContext via a call to one of the `addListener` methods.

The given AsyncListener class must define a zero argument constructor, which is used to instantiate it.

This method supports resource injection if the given `clazz` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

This method supports any annotations applicable to AsyncListener.

Parameters:
`clazz` - the AsyncListener class to instantiate
Returns:
the new AsyncListener instance
Throws:
`ServletException` - if the given `clazz` fails to be instantiated

---

setTimeout

```
void setTimeout(long timeout)
```

Sets the timeout (in milliseconds) for this AsyncContext.

The timeout applies to this AsyncContext once the container-initiated dispatch during which one of the `ServletRequest#startAsync` methods was called has returned to the container.

The timeout will expire if neither the `complete()` method nor any of the dispatch methods are called. A timeout value of zero or less indicates no timeout.

If `setTimeout(long)` is not called, then the container's default timeout, which is available via a call to `getTimeout()`, will apply.

Parameters:
`timeout` - the timeout in milliseconds
Throws:
`IllegalStateException` - if this method is called after the container-initiated dispatch, during which one of the `ServletRequest#startAsync` methods was called, has returned to the container

---

getTimeout

```
long getTimeout()
```

Gets the timeout (in milliseconds) for this AsyncContext.

This method returns the container's default timeout for asynchronous operations, or the timeout value passed to the most recent invocation of `setTimeout(long)`.

A timeout value of zero or less indicates no timeout.

Returns:
the timeout in milliseconds

Submit a bug or feature

javax.servlet

Class AsyncEvent

```
java.lang.Object
  └ javax.servlet.AsyncEvent
```

```
public class AsyncEvent
extends java.lang.Object
```

Event that gets fired when the asynchronous operation initiated on a ServletRequest (via a call to `ServletRequest#startAsync` or `ServletRequest#startAsync(ServletRequest, ServletResponse)`) has completed, timed out, or produced an error.

Since:
Servlet 3.0

Constructor Summary

`AsyncEvent(AsyncContext context)`   Constructs an AsyncEvent from the given AsyncContext.
`AsyncEvent(AsyncContext context, ServletRequest request, ServletResponse response)`   Constructs an AsyncEvent from the given AsyncContext, ServletRequest, and ServletResponse.
`AsyncEvent(AsyncContext context, ServletRequest request, ServletResponse response, java.lang.Throwable throwable)`   Constructs an AsyncEvent from the given AsyncContext, ServletRequest, ServletResponse, and Throwable.
`AsyncEvent(AsyncContext context, java.lang.Throwable throwable)`   Constructs an AsyncEvent from the given AsyncContext and Throwable.

Method Summary

`AsyncContext`	`getAsyncContext()` Gets the AsyncContext from this AsyncEvent.
`ServletRequest`	`getSuppliedRequest()` Gets the ServletRequest from this AsyncEvent.
`ServletResponse`	`getSuppliedResponse()` Gets the ServletResponse from this AsyncEvent.
`java.lang.Throwable`	`getThrowable()` Gets the Throwable from this AsyncEvent.

Methods inherited from class java.lang.Object

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

Constructor Detail

AsyncEvent

```
public AsyncEvent(AsyncContext context)
```

Constructs an AsyncEvent from the given AsyncContext.

Parameters:

`context` - the AsyncContex to be delivered with this AsyncEvent

AsyncEvent

```
public AsyncEvent(AsyncContext context,
                  ServletRequest request,
                  ServletResponse response)
```

Constructs an AsyncEvent from the given AsyncContext, ServletRequest, and ServletResponse.

Parameters:

`context` - the AsyncContex to be delivered with this AsyncEvent
`request` - the ServletRequest to be delivered with this AsyncEvent
`response` - the ServletResponse to be delivered with this AsyncEvent

---

AsyncEvent

```
public AsyncEvent(AsyncContext context,
                  java.lang.Throwable throwable)
```

Constructs an AsyncEvent from the given AsyncContext and Throwable.

Parameters:

`context` - the AsyncContex to be delivered with this AsyncEvent
`throwable` - the Throwable to be delivered with this AsyncEvent

AsyncEvent

```
public AsyncEvent(AsyncContext context,
                  ServletRequest request,
                  ServletResponse response,
                  java.lang.Throwable throwable)
```

Constructs an AsyncEvent from the given AsyncContext, ServletRequest, ServletResponse, and Throwable.

Parameters:

`context` - the AsyncContex to be delivered with this AsyncEvent
`request` - the ServletRequest to be delivered with this AsyncEvent
`response` - the ServletResponse to be delivered with this AsyncEvent
`throwable` - the Throwable to be delivered with this AsyncEvent

Method Detail

getAsyncContext

`public AsyncContext getAsyncContext()`

Gets the AsyncContext from this AsyncEvent.

Returns:
the AsyncContext that was used to initialize this AsyncEvent

---

getSuppliedRequest

`public ServletRequest getSuppliedRequest()`

Gets the ServletRequest from this AsyncEvent.

If the AsyncListener to which this AsyncEvent is being delivered was added using `AsyncContext#addListener(AsyncListener, ServletRequest, ServletResponse)`, the returned ServletRequest will be the same as the one supplied to the above method. If the AsyncListener was added via `AsyncContext#addListener(AsyncListener)`, this method must return null.

Returns:
the ServletRequest that was used to initialize this AsyncEvent, or null if this AsyncEvent was initialized without any ServletRequest

---

getSuppliedResponse

`public ServletResponse getSuppliedResponse()`

Gets the ServletResponse from this AsyncEvent.

If the AsyncListener to which this AsyncEvent is being delivered was added using `AsyncContext#addListener(AsyncListener, ServletRequest, ServletResponse)`, the returned ServletResponse will be the same as the one supplied to the above method. If the AsyncListener was added via `AsyncContext#addListener(AsyncListener)`, this method must return null.

Returns:
the ServletResponse that was used to initialize this AsyncEvent, or null if this AsyncEvent was initialized without any ServletResponse

---

getThrowable

`public java.lang.Throwable getThrowable()`

Gets the Throwable from this AsyncEvent.

Returns:

the Throwable that was used to initialize this AsyncEvent, or null if this AsyncEvent was initialized without any Throwable

---

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface AsyncListener

All Superinterfaces:

java.util.EventListener

```
public interface AsyncListener
extends java.util.EventListener
```

Listener that will be notified in the event that an asynchronous operation initiated on a ServletRequest to which the listener had been added has completed, timed out, or resulted in an error.

Since:

Servlet 3.0

Method Summary

void	`onComplete(AsyncEvent event)` Notifies this AsyncListener that an asynchronous operation has been completed.
void	`onError(AsyncEvent event)` Notifies this AsyncListener that an asynchronous operation has failed to complete.
void	`onStartAsync(AsyncEvent event)` Notifies this AsyncListener that a new asynchronous cycle is being initiated via a call to one of the `ServletRequest#startAsync` methods.
void	`onTimeout(AsyncEvent event)` Notifies this AsyncListener that an asynchronous operation has timed out.

Method Detail

onComplete

```
void onComplete(AsyncEvent event)
                throws java.io.IOException
```

Notifies this AsyncListener that an asynchronous operation has been completed.

The `AsyncContext` corresponding to the asynchronous operation that has been completed may be obtained by calling `getAsyncContext` on the given `event`.

In addition, if this AsyncListener had been registered via a call to `AsyncContext#addListener(AsyncListener, ServletRequest, ServletResponse)`, the supplied ServletRequest and ServletResponse objects may be retrieved by calling `getSuppliedRequest` and `getSuppliedResponse`, respectively, on the given `event`.

Parameters:

`event` - the AsyncEvent indicating that an asynchronous operation has been completed

Throws:

`java.io.IOException` - if an I/O related error has occurred during the processing of the given AsyncEvent

---

onTimeout

```
void onTimeout(AsyncEvent event)
               throws java.io.IOException
```

Notifies this AsyncListener that an asynchronous operation has timed out.

The `AsyncContext` corresponding to the asynchronous operation that has timed out may be obtained by calling `getAsyncContext` on the given `event`.

In addition, if this AsyncListener had been registered via a call to `AsyncContext#addListener(AsyncListener, ServletRequest, ServletResponse)`, the supplied ServletRequest and ServletResponse objects may be retrieved by calling `getSuppliedRequest` and `getSuppliedResponse`, respectively, on the given `event`.

Parameters:

`event` - the AsyncEvent indicating that an asynchronous operation has timed out

Throws:

`java.io.IOException` - if an I/O related error has occurred during the processing of the given AsyncEvent

---

onError

```
void onError(AsyncEvent event)
             throws java.io.IOException
```

Notifies this AsyncListener that an asynchronous operation has failed to complete.

The `AsyncContext` corresponding to the asynchronous operation that failed to complete may be obtained by calling `getAsyncContext` on the given `event`.

In addition, if this AsyncListener had been registered via a call to `AsyncContext#addListener(AsyncListener, ServletRequest, ServletResponse)`, the supplied ServletRequest and ServletResponse objects may be retrieved by calling `getSuppliedRequest` and `getSuppliedResponse`, respectively, on the given `event`.

Parameters:

`event` - the AsyncEvent indicating that an asynchronous operation has failed to complete

Throws:

`java.io.IOException` - if an I/O related error has occurred during the processing of the given AsyncEvent

---

onStartAsync

```
void onStartAsync(AsyncEvent event)
```

```
                throws java.io.IOException
```

Notifies this AsyncListener that a new asynchronous cycle is being initiated via a call to one of the `ServletRequest#startAsync` methods.

The `AsyncContext` corresponding to the asynchronous operation that is being reinitialized may be obtained by calling `getAsyncContext` on the given `event`.

In addition, if this AsyncListener had been registered via a call to `AsyncContext#addListener(AsyncListener, ServletRequest, ServletResponse)`, the supplied ServletRequest and ServletResponse objects may be retrieved by calling `getSuppliedRequest` and `getSuppliedResponse`, respectively, on the given `event`.

This AsyncListener will not receive any events related to the new asynchronous cycle unless it registers itself (via a call to `AsyncContext#addListener`) with the AsyncContext that is delivered as part of the given AsyncEvent.

Parameters:
`event` - the AsyncEvent indicating that a new asynchronous cycle is being initiated

Throws:
`java.io.IOException` - if an I/O related error has occurred during the processing of the given AsyncEvent

javax.servlet

Enum DispatcherType

```
java.lang.Object
  └java.lang.Enum<DispatcherType>
      └javax.servlet.DispatcherType
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable<DispatcherType>

```
public enum DispatcherType
extends java.lang.Enum<DispatcherType>
```

Enumeration of filter dispatcher types.

Since:

Servlet 3.0

Enum Constant Summary

ASYNC
ERROR
FORWARD
INCLUDE
REQUEST

Method Summary

static DispatcherType	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.
static DispatcherType[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

`clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class java.lang.Object

```
getClass, notify, notifyAll, wait, wait, wait
```

Enum Constant Detail

FORWARD

```
public static final DispatcherType FORWARD
```

INCLUDE

```
public static final DispatcherType INCLUDE
```

REQUEST

```
public static final DispatcherType REQUEST
```

ASYNC

```
public static final DispatcherType ASYNC
```

ERROR

```
public static final DispatcherType ERROR
```

Method Detail

values

```
public static DispatcherType[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (DispatcherType c : DispatcherType.values())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

---

valueOf

```
public static DispatcherType valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

`name` - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

`IllegalArgumentException` - if this enum type has no constant with the specified name

`java.lang.NullPointerException` - if the argument is null

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface Filter

`public interface **Filter**`

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

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

Filters are configured in the deployment descriptor of a web application.

Examples that have been identified for this design are:

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

Since:

Servlet 2.3

Method Summary

`void`	**`destroy`**`()` Called by the web container to indicate to a filter that it is being taken out of service.
`void`	**`doFilter`**`(ServletRequest request, ServletResponse response, FilterChain chain)` The `doFilter` method of the Filter is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
`void`	**`init`**`(FilterConfig filterConfig)` Called by the web container to indicate to a filter that it is being placed into service.

Method Detail

init

```
void init(FilterConfig filterConfig)
          throws ServletException
```

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

The servlet container calls the init method exactly once after instantiating the filter. The init method must complete successfully before the filter is asked to do any filtering work.

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

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

Throws:

`ServletException`

---

doFilter

```
void doFilter(ServletRequest request,
              ServletResponse response,
              FilterChain chain)
              throws java.io.IOException,
                     ServletException
```

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

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

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

Throws:

`java.io.IOException`
`ServletException`

---

destroy

```
void destroy()
```

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

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

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

Submit a bug or feature

javax.servlet
Interface FilterChain

```
public interface FilterChain
```

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

Since:
Servlet 2.3

See Also:
`Filter`

Method Summary

`void`	`doFilter(ServletRequest request, ServletResponse response)` Causes the next filter in the chain to be invoked, or if the calling filter is the last filter in the chain, causes the resource at the end of the chain to be invoked.

Method Detail

doFilter

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

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

Parameters:
`request` - the request to pass along the chain.
`response` - the response to pass along the chain.

Throws:
`java.io.IOException`
`ServletException`

Submit a bug or feature

javax.servlet

Interface FilterConfig

`public interface` **`FilterConfig`**

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

Since:
Servlet 2.3

See Also:
`Filter`

Method Summary

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

Method Detail

getFilterName

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

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

getServletContext

`ServletContext` **`getServletContext`**`()`

Returns a reference to the `ServletContext` in which the caller is executing.

Returns:
a `ServletContext` object, used by the caller to interact with its servlet container

See Also:
`ServletContext`

---

getInitParameter

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

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

Parameters:
`name` - a `String` specifying the name of the initialization parameter

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

---

getInitParameterNames

`java.util.Enumeration<java.lang.String>` **`getInitParameterNames`**`()`

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

Returns:
an `Enumeration` of `String` objects containing the names of the filter's initialization parameters

---

Submit a bug or feature

javax.servlet

Interface FilterRegistration.Dynamic

All Superinterfaces:

FilterRegistration, Registration, Registration.Dynamic

Enclosing interface:

FilterRegistration

```
public static interface FilterRegistration.Dynamic
extends FilterRegistration, Registration.Dynamic
```

Interface through which a `Filter` registered via one of the `addFilter` methods on `ServletContext` may be further configured.

Nested Class Summary

Nested classes/interfaces inherited from interface javax.servlet.FilterRegistration
`FilterRegistration.Dynamic`

Method Summary

Methods inherited from interface javax.servlet.FilterRegistration
`addMappingForServletNames, addMappingForUrlPatterns, getServletNameMappings, getUrlPatternMappings`

Methods inherited from interface javax.servlet.Registration.Dynamic
`setAsyncSupported`

Methods inherited from interface javax.servlet.Registration
`getClassName, getInitParameter, getInitParameters, getName, setInitParameter, setInitParameters`

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class GenericServlet

```
java.lang.Object
  └ javax.servlet.GenericServlet
```

All Implemented Interfaces:

java.io.Serializable, Servlet, ServletConfig

Direct Known Subclasses:

HttpServlet

```
public abstract class GenericServlet
extends java.lang.Object
implements Servlet, ServletConfig, java.io.Serializable
```

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

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

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

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

Author:

Various

See Also:

Serialized Form

Constructor Summary

`GenericServlet``()` Does nothing.

Method Summary

`void`	**`destroy`**`()` Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
`java.lang.String`	**`getInitParameter`**`(java.lang.String name)` Returns a `String` containing the value of the named

	initialization parameter, or `null` if the parameter does not exist.
`java.util.Enumeration<java.lang.String>`	**`getInitParameterNames`**`()` Returns the names of the servlet's initialization parameters as an `Enumeration` of `String` objects, or an empty `Enumeration` if the servlet has no initialization parameters.
`ServletConfig`	**`getServletConfig`**`()` Returns this servlet's `ServletConfig` object.
`ServletContext`	**`getServletContext`**`()` Returns a reference to the `ServletContext` in which this servlet is running.
`java.lang.String`	**`getServletInfo`**`()` Returns information about the servlet, such as author, version, and copyright.
`java.lang.String`	**`getServletName`**`()` Returns the name of this servlet instance.
`void`	**`init`**`()` A convenience method which can be overridden so that there's no need to call `super.init(config)`.
`void`	**`init`**`(ServletConfig config)` Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
`void`	**`log`**`(java.lang.String msg)` Writes the specified message to a servlet log file, prepended by the servlet's name.
`void`	**`log`**`(java.lang.String message, java.lang.Throwable t)` Writes an explanatory message and a stack trace for a given `Throwable` exception to the servlet log file, prepended by the servlet's name.
`abstract void`	**`service`**`(ServletRequest req, ServletResponse res)` Called by the servlet container to allow the servlet to respond to a request.

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

Constructor Detail

GenericServlet

```
public GenericServlet()
```

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

Method Detail

destroy

```
public void destroy()
```

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service. See `Servlet#destroy`.

Specified by:
`destroy` in interface `Servlet`

---

getInitParameter

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

Returns a `String` containing the value of the named initialization parameter, or `null` if the parameter does not exist. See `ServletConfig#getInitParameter`.

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

Specified by:
`getInitParameter` in interface `ServletConfig`

Parameters:
`name` - a `String` specifying the name of the initialization parameter

Returns:
String a `String` containing the value of the initialization parameter

---

getInitParameterNames

```
public java.util.Enumeration<java.lang.String> getInitParameterNames()
```

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

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

Specified by:
`getInitParameterNames` in interface `ServletConfig`

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

---

getServletConfig

```
public ServletConfig getServletConfig()
```

Returns this servlet's `ServletConfig` object.

Specified by:

`getServletConfig` in interface `Servlet`

Returns:

ServletConfig the `ServletConfig` object that initialized this servlet

See Also:

`Servlet.init(javax.servlet.ServletConfig)`

---

getServletContext

```
public ServletContext getServletContext()
```

Returns a reference to the `ServletContext` in which this servlet is running. See `ServletConfig#getServletContext`.

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

Specified by:

`getServletContext` in interface `ServletConfig`

Returns:

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

See Also:

`ServletContext`

---

getServletInfo

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

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

Specified by:

`getServletInfo` in interface `Servlet`

Returns:

String information about this servlet, by default an empty string

---

init

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

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

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

Specified by:

`init` in interface `Servlet`

Parameters:

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

Throws:

`ServletException` - if an exception occurs that interrupts the servlet's normal operation

See Also:

`UnavailableException`

---

init

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

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

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

Throws:

`ServletException` - if an exception occurs that interrupts the servlet's normal operation

---

log

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

Writes the specified message to a servlet log file, prepended by the servlet's name. See `ServletContext#log(String)`.

Parameters:

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

---

log

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

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

Parameters:

`message` - a `String` that describes the error or exception

`t` - the `java.lang.Throwable` error or exception

---

service

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

Called by the servlet container to allow the servlet to respond to a request. See `Servlet#service`.

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

Specified by:
`service` in interface `Servlet`

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

Throws:
`ServletException` - if an exception occurs that interferes with the servlet's normal operation occurred
`java.io.IOException` - if an input or output exception occurs

---

getServletName

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

Returns the name of this servlet instance. See `ServletConfig#getServletName`.

Specified by:
`getServletName` in interface `ServletConfig`

Returns:
the name of this servlet instance

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Submit a bug or feature

javax.servlet

Class HttpConstraintElement

```
java.lang.Object
  └ javax.servlet.HttpConstraintElement
```

Direct Known Subclasses:

HttpMethodConstraintElement, ServletSecurityElement

```
public class HttpConstraintElement
extends java.lang.Object
```

Java Class representation of an `HttpConstraint` annotation value.

Since:

Servlet 3.0

Constructor Summary

`HttpConstraintElement``()` Constructs a default HTTP constraint element
`HttpConstraintElement``(ServletSecurity.EmptyRoleSemantic semantic)` Convenience constructor to establish `EmptyRoleSemantic.DENY`
`HttpConstraintElement``(ServletSecurity.EmptyRoleSemantic semantic, ServletSecurity.TransportGuarantee guarantee, java.lang.String... roleNames)` Constructor to establish all of getEmptyRoleSemantic, getRolesAllowed, and getTransportGuarantee.
`HttpConstraintElement``(ServletSecurity.TransportGuarantee guarantee, java.lang.String... roleNames)` Constructor to establish non-empty getRolesAllowed and/or `TransportGuarantee.CONFIDENTIAL`.

Method Summary

`ServletSecurity.EmptyRoleSemantic`	**`getEmptyRoleSemantic`**`()` Gets the default authorization semantic.
`java.lang.String[]`	**`getRolesAllowed`**`()` Gets the names of the authorized roles.
`ServletSecurity.TransportGuarantee`	**`getTransportGuarantee`**`()` Gets the data protection requirement (i.e., whether or not SSL/TLS is required) that must be satisfied by the transport connection.

Methods inherited from class java.lang.Object

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

Constructor Detail

HttpConstraintElement

```
public HttpConstraintElement()
```

Constructs a default HTTP constraint element

HttpConstraintElement

```
public HttpConstraintElement(ServletSecurity.EmptyRoleSemantic semantic)
```

Convenience constructor to establish `EmptyRoleSemantic.DENY`

Parameters:

`semantic` - should be EmptyRoleSemantic.DENY

HttpConstraintElement

```
public HttpConstraintElement(ServletSecurity.TransportGuarantee guarantee,
                             java.lang.String... roleNames)
```

Constructor to establish non-empty getRolesAllowed and/or `TransportGuarantee.CONFIDENTIAL`.

Parameters:

`guarantee` - `TransportGuarantee.NONE` or `TransportGuarantee.CONFIDENTIAL`

`roleNames` - the names of the roles that are to be allowed access

HttpConstraintElement

```
public HttpConstraintElement(ServletSecurity.EmptyRoleSemantic semantic,
                             ServletSecurity.TransportGuarantee guarantee,
                             java.lang.String... roleNames)
```

Constructor to establish all of getEmptyRoleSemantic, getRolesAllowed, and getTransportGuarantee.

Parameters:

`semantic` - `EmptyRoleSemantic.DENY` or `EmptyRoleSemantic.PERMIT`

`guarantee` - `TransportGuarantee.NONE` or `TransportGuarantee.CONFIDENTIAL`

`roleNames - the names of the roles that are to be allowed access, or missing if the semantic is EmptyRoleSemantic.DENY`

Method Detail

getEmptyRoleSemantic

public ServletSecurity.EmptyRoleSemantic **getEmptyRoleSemantic**()

Gets the default authorization semantic.

This value is insignificant when getRolesAllowed returns a non-empty array, and should not be specified when a non-empty array is specified for getRolesAllowed.

Returns:
the ServletSecurity.EmptyRoleSemantic to be applied when getRolesAllowed returns an empty (that is, zero-length) array

---

getTransportGuarantee

public ServletSecurity.TransportGuarantee **getTransportGuarantee**()

Gets the data protection requirement (i.e., whether or not SSL/TLS is required) that must be satisfied by the transport connection.

Returns:
the TransportGuarantee indicating the data protection that must be provided by the connection

---

getRolesAllowed

public java.lang.String[] **getRolesAllowed**()

Gets the names of the authorized roles.

Duplicate role names appearing in getRolesAllowed are insignificant and may be discarded. The String "*" has no special meaning as a role name (should it occur in getRolesAllowed).

Returns:
a (possibly empty) array of role names. When the array is empty, its meaning depends on the value of getEmptyRoleSemantic(). If its value is DENY, and getRolesAllowed returns an empty array, access is to be denied independent of authentication state and identity. Conversely, if its value is PERMIT, it indicates that access is to be allowed independent of authentication state and identity. When the array contains the names of one or more roles, it indicates that access is contingent on membership in at least one of the named roles (independent of the value of getEmptyRoleSemantic()).

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class HttpMethodConstraintElement

```
java.lang.Object
  └ javax.servlet.HttpConstraintElement
      └ javax.servlet.HttpMethodConstraintElement
```

```
public class HttpMethodConstraintElement
extends HttpConstraintElement
```

Java Class represntation of an `HttpMethodConstraint` annotation value.

Since:
Servlet 3.0

Constructor Summary

`HttpMethodConstraintElement``(java.lang.String methodName)` Constructs an instance with default `HttpConstraintElement` value.
`HttpMethodConstraintElement``(java.lang.String methodName, HttpConstraintElement constraint)` Constructs an instance with specified `HttpConstraintElement` value.

Method Summary

`java.lang.String`	**`getMethodName`**`()` Gets the HTTP method name.

Methods inherited from class javax.servlet.HttpConstraintElement

`getEmptyRoleSemantic, getRolesAllowed, getTransportGuarantee`

Methods inherited from class java.lang.Object

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

Constructor Detail

HttpMethodConstraintElement

```
public HttpMethodConstraintElement(java.lang.String methodName)
```

Constructs an instance with default `HttpConstraintElement` value.

Parameters:

`methodName` - the name of an HTTP protocol method. The name must not be null, or the empty string, and must be a legitimate HTTP Method name as defined by RFC 2616

HttpMethodConstraintElement

```
public HttpMethodConstraintElement(java.lang.String methodName,
                                   HttpConstraintElement constraint)
```

Constructs an instance with specified `HttpConstraintElement` value.

Parameters:

`methodName` - the name of an HTTP protocol method. The name must not be null, or the empty string, and must be a legitimate HTTP Method name as defined by RFC 2616
`constraint` - the HTTPconstraintElement value to assign to the named HTTP method

Method Detail

getMethodName

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

Gets the HTTP method name.

Returns:

the Http method name

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class MultipartConfigElement

```
java.lang.Object
  └ javax.servlet.MultipartConfigElement
```

```
public class MultipartConfigElement
extends java.lang.Object
```

Java Class represntation of an `MultipartConfig` annotation value.

Since:
Servlet 3.0

Constructor Summary

`MultipartConfigElement``(MultipartConfig annotation)` Constructs an instance from a `MultipartConfig` annotation value.
`MultipartConfigElement``(java.lang.String location)` Constructs an instance with defaults for all but location.
`MultipartConfigElement``(java.lang.String location, long maxFileSize, long maxRequestSize, int fileSizeThreshold)` Constructs an instance with all values specified.

Method Summary

`int`	**`getFileSizeThreshold`**`()` Gets the size threshold after which files will be written to disk.
`java.lang.String`	**`getLocation`**`()` Gets the directory location where files will be stored.
`long`	**`getMaxFileSize`**`()` Gets the maximum size allowed for uploaded files.
`long`	**`getMaxRequestSize`**`()` Gets the maximum size allowed for multipart/form-data requests.

Methods inherited from class java.lang.Object

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

Constructor Detail

MultipartConfigElement

```
public MultipartConfigElement(java.lang.String location)
```

Constructs an instance with defaults for all but location.

Parameters:

`location` - defualts to "" if values is null.

MultipartConfigElement

```
public MultipartConfigElement(java.lang.String location,
                              long maxFileSize,
                              long maxRequestSize,
                              int fileSizeThreshold)
```

Constructs an instance with all values specified.

Parameters:

`location` - the directory location where files will be stored
`maxFileSize` - the maximum size allowed for uploaded files
`maxRequestSize` - the maximum size allowed for multipart/form-data requests
`fileSizeThreshold` - the size threshold after which files will be written to disk

---

MultipartConfigElement

```
public MultipartConfigElement(MultipartConfig annotation)
```

Constructs an instance from a `MultipartConfig` annotation value.

Parameters:

`annotation` - the annotation value

Method Detail

getLocation

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

Gets the directory location where files will be stored.

Returns:

the directory location where files will be stored

---

getMaxFileSize

```
public long getMaxFileSize()
```

Gets the maximum size allowed for uploaded files.

Returns:

the maximum size allowed for uploaded files

---

getMaxRequestSize

```
public long getMaxRequestSize()
```

Gets the maximum size allowed for multipart/form-data requests.

Returns:
the maximum size allowed for multipart/form-data requests

---

getFileSizeThreshold

```
public int getFileSizeThreshold()
```

Gets the size threshold after which files will be written to disk.

Returns:
the size threshold after which files will be written to disk

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface Registration

All Known Subinterfaces:

FilterRegistration, FilterRegistration.Dynamic, Registration.Dynamic, ServletRegistration, ServletRegistration.Dynamic

`public interface` **`Registration`**

Interface through which a `Servlet` or `Filter` may be further configured.

A Registration object whose `getClassName()` method returns null is considered *preliminary*. Servlets and Filters whose implementation class is container implementation specific may be declared without any `servlet-class` or `filter-class` elements, respectively, and will be represented as preliminary Registration objects. Preliminary registrations must be completed by calling one of the `addServlet` or `addFilter` methods on `ServletContext`, and passing in the Servlet or Filter name (obtained via `getName()`) along with the supporting Servlet or Filter implementation class name, Class object, or instance, respectively. In most cases, preliminary registrations will be completed by an appropriate, container-provided `ServletContainerInitializer`.

Since:

Servlet 3.0

Nested Class Summary

`static interface`	**`Registration.Dynamic`** Interface through which a `Servlet` or `Filter` registered via one of the `addServlet` or `addFilter` methods, respectively, on `ServletContext` may be further configured.

Method Summary

`java.lang.String`	**`getClassName`**`()` Gets the fully qualified class name of the Servlet or Filter that is represented Registration.
`java.lang.String`	**`getInitParameter`**`(java.lang.String name)` Gets the value of the initialization parameter with the given name that will b initialize the Servlet or Filter represented by this Registration object.
`java.util.Map<java.lang.String,java.lang.String>`	**`getInitParameters`**`()` Gets an immutable (and possibly empty) Map containing the currently availa parameters that will be used to initialize the Servlet or Filter represented by this Re
`java.lang.String`	**`getName`**`()` Gets the name of the Servlet or Filter that is represented by this Registration.
`boolean`	**`setInitParameter`**`(java.lang.String name, java.lang.String value)` Sets the initialization parameter with the given name and value on the Servle represented by this Registration.
`java.util.Set<java.lang.String>`	**`setInitParameters`**`(java.util.Map<java.lang.String,java.lang.String> i` Sets the given initialization parameters on the Servlet or Filter that is represe Registration.

Method Detail

getName

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

Gets the name of the Servlet or Filter that is represented by this Registration.

Returns:

the name of the Servlet or Filter that is represented by this Registration

getClassName

```
java.lang.String getClassName()
```

Gets the fully qualified class name of the Servlet or Filter that is represented by this Registration.

Returns:
the fully qualified class name of the Servlet or Filter that is represented by this Registration, or null if this Registration is preliminary

---

setInitParameter

```
boolean setInitParameter(java.lang.String name,
                         java.lang.String value)
```

Sets the initialization parameter with the given name and value on the Servlet or Filter that is represented by this Registration.

Parameters:
`name` - the initialization parameter name
`value` - the initialization parameter value
Returns:
true if the update was successful, i.e., an initialization parameter with the given name did not already exist for the Servlet or Filter represented by this Registration, and false otherwise
Throws:
`IllegalStateException` - if the ServletContext from which this Registration was obtained has already been initialized
`IllegalArgumentException` - if the given name or value is `null`

---

getInitParameter

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

Gets the value of the initialization parameter with the given name that will be used to initialize the Servlet or Filter represented by this Registration object.

Parameters:
`name` - the name of the initialization parameter whose value is requested
Returns:
the value of the initialization parameter with the given name, or `null` if no initialization parameter with the given name exists

---

setInitParameters

```
java.util.Set<java.lang.String> setInitParameters(java.util.Map<java.lang.String,java.lang.String> initParameters)
```

Sets the given initialization parameters on the Servlet or Filter that is represented by this Registration.

The given map of initialization parameters is processed *by-value*, i.e., for each initialization parameter contained in the map, this method calls `setInitParameter(String,String)`. If that method would return false for any of the initialization parameters in the given map, no updates will be performed, and false will be returned. Likewise, if the map contains an initialization parameter with a `null` name or value, no updates will be performed, and an IllegalArgumentException will be thrown.

Parameters:
`initParameters` - the initialization parameters
Returns:
the (possibly empty) Set of initialization parameter names that are in conflict
Throws:
`IllegalStateException` - if the ServletContext from which this Registration was obtained has already been initialized
`IllegalArgumentException` - if the given map contains an initialization parameter with a `null` name or value

---

getInitParameters

```
java.util.Map<java.lang.String,java.lang.String> getInitParameters()
```

Gets an immutable (and possibly empty) Map containing the currently available initialization parameters that will be used to initialize the Servlet or Filter represented by this Registration object.

Returns:

Map containing the currently available initialization parameters that will be used to initialize the Servlet or Filter represented by this Registration object

Submit a bug or feature

javax.servlet

Interface Registration.Dynamic

All Superinterfaces:
Registration

All Known Subinterfaces:
FilterRegistration.Dynamic, ServletRegistration.Dynamic

Enclosing interface:
Registration

```
public static interface Registration.Dynamic
extends Registration
```

Interface through which a `Servlet` or `Filter` registered via one of the `addServlet` or `addFilter` methods, respectively, on `ServletContext` may be further configured.

Nested Class Summary

Nested classes/interfaces inherited from interface javax.servlet.Registration
`Registration.Dynamic`

Method Summary

`void`	`setAsyncSupported(boolean isAsyncSupported)` Configures the Servlet or Filter represented by this dynamic Registration as supporting asynchronous operations or not.

Methods inherited from interface javax.servlet.Registration
`getClassName, getInitParameter, getInitParameters, getName, setInitParameter, setInitParameters`

Method Detail

setAsyncSupported

```
void setAsyncSupported(boolean isAsyncSupported)
```

Configures the Servlet or Filter represented by this dynamic Registration as supporting asynchronous operations or not.

By default, servlet and filters do not support asynchronous operations.

A call to this method overrides any previous setting.

Parameters:

`isAsyncSupported` - true if the Servlet or Filter represented by this dynamic Registration supports asynchronous operations, false otherwise

Throws:

`IllegalStateException` - if the ServletContext from which this dynamic Registration was obtained has already been initialized

---

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface RequestDispatcher

```
public interface RequestDispatcher
```

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server. The servlet container creates the `RequestDispatcher` object, which is used as a wrapper around a server resource located at a particular path or given by a particular name.

This interface is intended to wrap servlets, but a servlet container can create `RequestDispatcher` objects to wrap any type of resource.

Author:
Various

See Also:
`ServletContext#getRequestDispatcher(java.lang.String)`, `ServletContext#getNamedDispatcher(java.lang.String)`, `ServletRequest#getRequestDispatcher(java.lang.String)`

Field Summary

`static java.lang.String`	**`ERROR_EXCEPTION`** The name of the request attribute under which the exception object is propagated during an error dispatch
`static java.lang.String`	**`ERROR_EXCEPTION_TYPE`** The name of the request attribute under which the type of the exception object is propagated during an error dispatch
`static java.lang.String`	**`ERROR_MESSAGE`** The name of the request attribute under which the exception message is propagated during an error dispatch
`static java.lang.String`	**`ERROR_REQUEST_URI`** The name of the request attribute under which the request URI whose processing caused the error is propagated during an error dispatch
`static java.lang.String`	**`ERROR_SERVLET_NAME`** The name of the request attribute under which the name of the servlet in which the error occurred is propagated during an error dispatch
`static java.lang.String`	**`ERROR_STATUS_CODE`** The name of the request attribute under which the response status is propagated during an error dispatch
`static java.lang.String`	**`FORWARD_CONTEXT_PATH`** The name of the request attribute under which the original context path is made available to the target of a `forward`

`static java.lang.String`	**`FORWARD_PATH_INFO`** The name of the request attribute under which the original path info is made available to the target of a `forward`
`static java.lang.String`	**`FORWARD_QUERY_STRING`** The name of the request attribute under which the original query string is made available to the target of a `forward`
`static java.lang.String`	**`FORWARD_REQUEST_URI`** The name of the request attribute under which the original request URI is made available to the target of a `forward`
`static java.lang.String`	**`FORWARD_SERVLET_PATH`** The name of the request attribute under which the original servlet path is made available to the target of a `forward`
`static java.lang.String`	**`INCLUDE_CONTEXT_PATH`** The name of the request attribute under which the context path of the target of an `include` is stored
`static java.lang.String`	**`INCLUDE_PATH_INFO`** The name of the request attribute under which the path info of the target of an `include` is stored
`static java.lang.String`	**`INCLUDE_QUERY_STRING`** The name of the request attribute under which the query string of the target of an `include` is stored
`static java.lang.String`	**`INCLUDE_REQUEST_URI`** The name of the request attribute under which the request URI of the target of an `include` is stored
`static java.lang.String`	**`INCLUDE_SERVLET_PATH`** The name of the request attribute under which the servlet path of the target of an `include` is stored

Method Summary

`void`	**`forward`**`(ServletRequest request, ServletResponse response)` Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
`void`	**`include`**`(ServletRequest request, ServletResponse response)` Includes the content of a resource (servlet, JSP page, HTML file) in the response.

Field Detail

FORWARD_REQUEST_URI

`static final java.lang.String` **`FORWARD_REQUEST_URI`**

The name of the request attribute under which the original request URI is made available to the target of a `forward`

See Also:
Constant Field Values

FORWARD_CONTEXT_PATH

`static final java.lang.String` **`FORWARD_CONTEXT_PATH`**

The name of the request attribute under which the original context path is made available to the target of a `forward`

See Also:
Constant Field Values

FORWARD_PATH_INFO

`static final java.lang.String` **`FORWARD_PATH_INFO`**

The name of the request attribute under which the original path info is made available to the target of a `forward`

See Also:
Constant Field Values

FORWARD_SERVLET_PATH

`static final java.lang.String` **`FORWARD_SERVLET_PATH`**

The name of the request attribute under which the original servlet path is made available to the target of a `forward`

See Also:
Constant Field Values

FORWARD_QUERY_STRING

`static final java.lang.String` **`FORWARD_QUERY_STRING`**

The name of the request attribute under which the original query string is made available to the target of a `forward`

See Also:
Constant Field Values

INCLUDE_REQUEST_URI

`static final java.lang.String` **`INCLUDE_REQUEST_URI`**

The name of the request attribute under which the request URI of the target of an `include` is stored

See Also:
Constant Field Values

INCLUDE_CONTEXT_PATH

`static final java.lang.String` **`INCLUDE_CONTEXT_PATH`**

The name of the request attribute under which the context path of the target of an `include` is stored

See Also:
Constant Field Values

INCLUDE_PATH_INFO

`static final java.lang.String` **`INCLUDE_PATH_INFO`**

The name of the request attribute under which the path info of the target of an `include` is stored

See Also:
Constant Field Values

INCLUDE_SERVLET_PATH

`static final java.lang.String` **`INCLUDE_SERVLET_PATH`**

The name of the request attribute under which the servlet path of the target of an `include` is stored

See Also:
Constant Field Values

INCLUDE_QUERY_STRING

`static final java.lang.String` **`INCLUDE_QUERY_STRING`**

The name of the request attribute under which the query string of the target of an `include` is stored

See Also:
Constant Field Values

ERROR_EXCEPTION

`static final java.lang.String` **`ERROR_EXCEPTION`**

The name of the request attribute under which the exception object is propagated during an error dispatch

See Also:
Constant Field Values

ERROR_EXCEPTION_TYPE

`static final java.lang.String` **`ERROR_EXCEPTION_TYPE`**

The name of the request attribute under which the type of the exception object is propagated during an error dispatch

See Also:
Constant Field Values

ERROR_MESSAGE

`static final java.lang.String **ERROR_MESSAGE**`

The name of the request attribute under which the exception message is propagated during an error dispatch

See Also:
Constant Field Values

ERROR_REQUEST_URI

`static final java.lang.String **ERROR_REQUEST_URI**`

The name of the request attribute under which the request URI whose processing caused the error is propagated during an error dispatch

See Also:
Constant Field Values

ERROR_SERVLET_NAME

`static final java.lang.String **ERROR_SERVLET_NAME**`

The name of the request attribute under which the name of the servlet in which the error occurred is propagated during an error dispatch

See Also:
Constant Field Values

ERROR_STATUS_CODE

`static final java.lang.String **ERROR_STATUS_CODE**`

The name of the request attribute under which the response status is propagated during an error dispatch

See Also:
Constant Field Values

Method Detail

forward

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

Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server. This method allows one servlet to do preliminary processing of a request and another resource to generate the response.

For a `RequestDispatcher` obtained via `getRequestDispatcher()`, the `ServletRequest` object has its path elements and parameters adjusted to match the path of the target resource.

`forward` should be called before the response has been committed to the client (before response body output has been flushed). If the response already has been committed, this method throws an `IllegalStateException`. Uncommitted output in the response buffer is automatically cleared before the forward.

The request and response parameters must be either the same objects as were passed to the calling servlet's service method or be subclasses of the `ServletRequestWrapper` or `ServletResponseWrapper` classes that wrap them.

This method sets the dispatcher type of the given request to `DispatcherType.FORWARD`.

Parameters:

`request` - a `ServletRequest` object that represents the request the client makes of the servlet

`response` - a `ServletResponse` object that represents the response the servlet returns to the client

Throws:

`ServletException` - if the target resource throws this exception

`java.io.IOException` - if the target resource throws this exception

`IllegalStateException` - if the response was already committed

See Also:

`ServletRequest#getDispatcherType`

---

include

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

Includes the content of a resource (servlet, JSP page, HTML file) in the response. In essence, this method enables programmatic server-side includes.

The `ServletResponse` object has its path elements and parameters remain unchanged from the caller's. The included servlet cannot change the response status code or set headers; any attempt to make a change is ignored.

The request and response parameters must be either the same objects as were passed to the calling servlet's service method or be subclasses of the `ServletRequestWrapper` or `ServletResponseWrapper` classes that wrap them.

This method sets the dispatcher type of the given request to `DispatcherType.INCLUDE`.

Parameters:

`request` - a `ServletRequest` object that contains the client's request

`response` - a `ServletResponse` object that contains the servlet's response

Throws:

`ServletException` - if the included resource throws this exception

`java.io.IOException` - if the included resource throws this exception

See Also:

`ServletRequest#getDispatcherType`

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface Servlet

All Known Subinterfaces:

HttpJspPage, JspPage

All Known Implementing Classes:

FacesServlet, GenericServlet, HttpServlet

`public interface` **`Servlet`**

Defines methods that all servlets must implement.

A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.

To implement this interface, you can write a generic servlet that extends `javax.servlet.GenericServlet` or an HTTP servlet that extends `javax.servlet.http.HttpServlet`.

This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods and are called in the following sequence:

1. The servlet is constructed, then initialized with the `init` method.
2. Any calls from clients to the `service` method are handled.
3. The servlet is taken out of service, then destroyed with the `destroy` method, then garbage collected and finalized.

In addition to the life-cycle methods, this interface provides the `getServletConfig` method, which the servlet can use to get any startup information, and the `getServletInfo` method, which allows the servlet to return basic information about itself, such as author, version, and copyright.

Author:

Various

See Also:

`GenericServlet`, `HttpServlet`

Method Summary

`void`	**`destroy`**`()` Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
`ServletConfig`	**`getServletConfig`**`()` Returns a `ServletConfig` object, which contains initialization and startup parameters for this servlet.

`java.lang.String`	**`getServletInfo`**`()` Returns information about the servlet, such as author, version, and copyright.
`void`	**`init`**`(ServletConfig config)` Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
`void`	**`service`**`(ServletRequest req, ServletResponse res)` Called by the servlet container to allow the servlet to respond to a request.

Method Detail

init

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

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

The servlet container calls the `init` method exactly once after instantiating the servlet. The `init` method must complete successfully before the servlet can receive any requests.

The servlet container cannot place the servlet into service if the `init` method

1. Throws a `ServletException`
2. Does not return within a time period defined by the Web server

Parameters:

`config` - a `ServletConfig` object containing the servlet's configuration and initialization parameters

Throws:

`ServletException` - if an exception has occurred that interferes with the servlet's normal operation

See Also:

`UnavailableException`, `getServletConfig()`

---

getServletConfig

```
ServletConfig getServletConfig()
```

Returns a `ServletConfig` object, which contains initialization and startup parameters for this servlet. The `ServletConfig` object returned is the one passed to the `init` method.

Implementations of this interface are responsible for storing the `ServletConfig` object so that this method can return it. The `GenericServlet` class, which implements this interface, already does this.

Returns:

the `ServletConfig` object that initializes this servlet

See Also:

`init(javax.servlet.ServletConfig)`

---

service

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

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

This method is only called after the servlet's `init()` method has completed successfully.

The status code of the response always should be set for a servlet that throws or sends an error.

Servlets typically run inside multithreaded servlet containers that can handle multiple requests concurrently. Developers must be aware to synchronize access to any shared resources such as files, network connections, and as well as the servlet's class and instance variables. More information on multithreaded programming in Java is available in the Java tutorial on multi-threaded programming.

Parameters:
- `req` - the `ServletRequest` object that contains the client's request
- `res` - the `ServletResponse` object that contains the servlet's response

Throws:
- `ServletException` - if an exception occurs that interferes with the servlet's normal operation
- `java.io.IOException` - if an input or output exception occurs

---

getServletInfo

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

Returns information about the servlet, such as author, version, and copyright.

The string that this method returns should be plain text and not markup of any kind (such as HTML, XML, etc.).

Returns:
a `String` containing servlet information

---

destroy

```
void destroy()
```

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

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

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

javax.servlet

Interface ServletConfig

All Known Implementing Classes:

GenericServlet, HttpServlet

`public interface **ServletConfig**`

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

Method Summary

`java.lang.String`	`getInitParameter(java.lang.String name)` Gets the value of the initialization parameter with the given name.
`java.util.Enumeration<java.lang.String>`	`getInitParameterNames()` Returns the names of the servlet's initialization parameters as an `Enumeration` of `String` objects, or an empty `Enumeration` if the servlet has no initialization parameters.
`ServletContext`	`getServletContext()` Returns a reference to the `ServletContext` in which the caller is executing.
`java.lang.String`	`getServletName()` Returns the name of this servlet instance.

Method Detail

getServletName

```
java.lang.String getServletName()
```

Returns the name of this servlet instance. The name may be provided via server administration, assigned in the web application deployment descriptor, or for an unregistered (and thus unnamed) servlet instance it will be the servlet's class name.

Returns:

the name of the servlet instance

---

getServletContext

`ServletContext` **`getServletContext`**`()`

Returns a reference to the `ServletContext` in which the caller is executing.

Returns:
a `ServletContext` object, used by the caller to interact with its servlet container

See Also:
`ServletContext`

---

getInitParameter

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

Gets the value of the initialization parameter with the given name.

Parameters:
`name` - the name of the initialization parameter whose value to get

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

---

getInitParameterNames

`java.util.Enumeration<java.lang.String>` **`getInitParameterNames`**`()`

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

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

---

Overview Package Class Tree Deprecated Index Help

PREV CLASS   NEXT CLASS   FRAMES   NO FRAMES   All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet
Interface ServletContainerInitializer

`public interface` **`ServletContainerInitializer`**

Interface which allows a library/runtime to be notified of a web application's startup phase and perform any required programmatic registration of servlets, filters, and listeners in response to it.

Implementations of this interface may be annotated with `HandlesTypes`, in order to receive (at their `onStartup(java.util.Set>, javax.servlet.ServletContext)` method) the Set of application classes that implement, extend, or have been annotated with the class types specified by the annotation.

If an implementation of this interface does not use this annotation, or none of the application classes match the ones specified by the annotation, the container must pass a `null` Set of classes to `onStartup(java.util.Set>, javax.servlet.ServletContext)`.

When examining the classes of an application to see if they match any of the criteria specified by the HandlesTypes annotation of a ServletContainerInitializer, the container may run into classloading problems if any of the application's optional JAR files are missing. Because the container is not in a position to decide whether these types of classloading failures will prevent the application from working correctly, it must ignore them, while at the same time providing a configuration option that would log them.

Implementations of this interface must be declared by a JAR file resource located inside the `META-INF/services` directory and named for the fully qualified class name of this interface, and will be discovered using the runtime's service provider lookup mechanism or a container specific mechanism that is semantically equivalent to it. In either case, ServletContainerInitializer services from web fragment JAR files excluded from an absolute ordering must be ignored, and the order in which these services are discovered must follow the application's classloading delegation model.

Since:
Servlet 3.0

See Also:
`HandlesTypes`

Method Summary

`void`	**`onStartup`**`(java.util.Set<java.lang.Class<?>> c, ServletContext ctx)` Notifies this `ServletContainerInitializer` of the startup of the application represented by the given `ServletContext`.

Method Detail

onStartup

```
void onStartup(java.util.Set<java.lang.Class<?>> c,
               ServletContext ctx)
               throws ServletException
```

Notifies this `ServletContainerInitializer` of the startup of the application represented by the given `ServletContext`.

If this `ServletContainerInitializer` is bundled in a JAR file inside the `WEB-INF/lib` directory of an application, its `onStartup` method will be invoked only once during the startup of the bundling application. If this `ServletContainerInitializer` is bundled inside a JAR file outside of any `WEB-INF/lib` directory, but still discoverable as described above, its `onStartup` method will be invoked every time an application is started.

Parameters:

`c` - the Set of application classes that extend, implement, or have been annotated with the class types specified by the `HandlesTypes` annotation, or `null` if there are no matches, or this `ServletContainerInitializer` has not been annotated with `HandlesTypes`

`ctx` - the `ServletContext` of the web application that is being started and in which the classes contained in `c` were found

Throws:

`ServletException` - if an error has occurred

Submit a bug or feature

javax.servlet

Interface ServletContext

`public interface` **`ServletContext`**

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

There is one context per "web application" per Java Virtual Machine. (A "web application" is a collection of servlets and content installed under a specific subset of the server's URL namespace such as `/catalog` and possibly installed via a `.war` file.)

In the case of a web application marked "distributed" in its deployment descriptor, there will be one context instance for each virtual machine. In this situation, the context cannot be used as a location to share global information (because the information won't be truly global). Use an external resource like a database instead.

The `ServletContext` object is contained within the `ServletConfig` object, which the Web server provides the servlet when the servlet is initialized.

Author:
Various

See Also:
`Servlet#getServletConfig`, `ServletConfig#getServletContext`

Field Summary

`static java.lang.String`	**`ORDERED_LIBS`** The name of the `ServletContext` attribute whose value (of type `java.util.List<java.lang.String>`) contains the list of names of JAR files in `WEB-INF/lib` ordered by their web fragment names (with possible exclusions if `<absolute-ordering>` without any `<others/>` is being used), or null if no absolute or relative ordering has been specified
`static java.lang.String`	**`TEMPDIR`** The name of the `ServletContext` attribute which stores the private temporary directory (of type `java.io.File`) provided by the servlet container for the `ServletContext`

Method Summary

`FilterRegistration.Dynamic`	**`addFilter`**`(java.lang.String filterName, java.lang.Class<? extends Filter> filterClass)` Adds the filter with the given name and class type to this servlet context.
`FilterRegistration.Dynamic`	**`addFilter`**`(java.lang.String filterName, Filter filter)` Registers the given filter instance with this ServletContext under the given `filterName`.
`FilterRegistration.Dynamic`	**`addFilter`**`(java.lang.String filterName, java.lang.String className)` Adds the filter with the given name and class name to this servlet context.
`void`	**`addListener`**`(java.lang.Class<? extends java.util.EventListener> listenerClass)` Adds a listener of the given class type to this ServletContext.
`void`	**`addListener`**`(java.lang.String className)` Adds the listener with the given class name to this ServletContext.
`<T extends java.util.EventListener> void`	**`addListener`**`(T t)` Adds the given listener to this ServletContext.
`ServletRegistration.Dynamic`	**`addServlet`**`(java.lang.String servletName, java.lang.Class<? extends Servlet> servletClass)` Adds the servlet with the given name and class type to this servlet context.
`ServletRegistration.Dynamic`	**`addServlet`**`(java.lang.String servletName, Servlet servlet)` Registers the given servlet instance with this ServletContext under the given `servletName`.
`ServletRegistration.Dynamic`	**`addServlet`**`(java.lang.String servletName, java.lang.String className)` Adds the servlet with the given name and class name to this servlet context.
`<T extends Filter> T`	**`createFilter`**`(java.lang.Class<T> clazz)` Instantiates the given Filter class.
`<T extends java.util.EventListener> T`	**`createListener`**`(java.lang.Class<T> clazz)` Instantiates the given EventListener class.
`<T extends Servlet> T`	**`createServlet`**`(java.lang.Class<T> clazz)` Instantiates the given Servlet class.

`void`	**`declareRoles`**`(java.lang.String... roleNames)` Declares role names that are tested using `isUserInRole`.
`java.lang.Object`	**`getAttribute`**`(java.lang.String name)` Returns the servlet container attribute with the given name, or `null` if there is no attribute by that name.
`java.util.Enumeration<java.lang.String>`	**`getAttributeNames`**`()` Returns an `Enumeration` containing the attribute names available within this ServletContext.
`java.lang.ClassLoader`	**`getClassLoader`**`()` Gets the class loader of the web application represented by this ServletContext.
`ServletContext`	**`getContext`**`(java.lang.String uripath)` Returns a `ServletContext` object that corresponds to a specified URL on the server.
`java.lang.String`	**`getContextPath`**`()` Returns the context path of the web application.
`java.util.Set<SessionTrackingMode>`	**`getDefaultSessionTrackingModes`**`()` Gets the session tracking modes that are supported by default for this `ServletContext`.
`int`	**`getEffectiveMajorVersion`**`()` Gets the major version of the Servlet specification that the application represented by this ServletContext is based on.
`int`	**`getEffectiveMinorVersion`**`()` Gets the minor version of the Servlet specification that the application represented by this ServletContext is based on.
`java.util.Set<SessionTrackingMode>`	**`getEffectiveSessionTrackingModes`**`()` Gets the session tracking modes that are in effect for this `ServletContext`.
`FilterRegistration`	**`getFilterRegistration`**`(java.lang.String filterName)` Gets the FilterRegistration corresponding to the filter with the given `filterName`.
`java.util.Map<java.lang.String,? extends FilterRegistration>`	**`getFilterRegistrations`**`()` Gets a (possibly empty) Map of the FilterRegistration objects (keyed by filter name) corresponding to all filters registered with this ServletContext.
`java.lang.String`	**`getInitParameter`**`(java.lang.String name)` Returns a `String` containing the value of the named context-wide initialization parameter, or `null` if the parameter does not exist.
`java.util.Enumeration<java.lang.String>`	**`getInitParameterNames`**`()` Returns the names of the context's initialization parameters as an `Enumeration` of `String` objects, or an empty `Enumeration` if the context has no initialization parameters.
`JspConfigDescriptor`	**`getJspConfigDescriptor`**`()` Gets the `<jsp-config>` related configuration that was aggregated from the `web.xml` and `web-fragment.xml` descriptor files of the web application represented by this ServletContext.
`int`	**`getMajorVersion`**`()` Returns the major version of the Servlet API that this servlet container supports.
`java.lang.String`	**`getMimeType`**`(java.lang.String file)` Returns the MIME type of the specified file, or `null` if the MIME type is not known.
`int`	**`getMinorVersion`**`()` Returns the minor version of the Servlet API that this servlet container supports.
`RequestDispatcher`	**`getNamedDispatcher`**`(java.lang.String name)` Returns a `RequestDispatcher` object that acts as a wrapper for the named servlet.
`java.lang.String`	**`getRealPath`**`(java.lang.String path)` Gets the *real* path corresponding to the given *virtual* path.
`RequestDispatcher`	**`getRequestDispatcher`**`(java.lang.String path)` Returns a `RequestDispatcher` object that acts as a wrapper for the resource located at the given path.
`java.net.URL`	**`getResource`**`(java.lang.String path)` Returns a URL to the resource that is mapped to the given path.
`java.io.InputStream`	**`getResourceAsStream`**`(java.lang.String path)` Returns the resource located at the named path as an `InputStream` object.
`java.util.Set<java.lang.String>`	**`getResourcePaths`**`(java.lang.String path)` Returns a directory-like listing of all the paths to resources within the web application whose longest sub-path matches the supplied path argument.
`java.lang.String`	**`getServerInfo`**`()` Returns the name and version of the servlet container on which the servlet is running.
`Servlet`	**`getServlet`**`(java.lang.String name)` **Deprecated.** *As of Java Servlet API 2.1, with no direct replacement.* *This method was originally defined to retrieve a servlet from a `ServletContext`. In this version, this method always returns `null` and remains only to preserve binary compatibility. This method*

	will be permanently removed in a future version of the Java Servlet API. *In lieu of this method, servlets can share information using the `ServletContext` class and can perform shared business logic by invoking methods on common non-servlet classes.*
`java.lang.String`	**`getServletContextName`**`()` Returns the name of this web application corresponding to this ServletContext as specified in the deployment descriptor for this web application by the display-name element.
`java.util.Enumeration<java.lang.String>`	**`getServletNames`**`()` **Deprecated.** *As of Java Servlet API 2.1, with no replacement.* *This method was originally defined to return an `Enumeration` of all the servlet names known to this context. In this version, this method always returns an empty `Enumeration` and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.*
`ServletRegistration`	**`getServletRegistration`**`(java.lang.String servletName)` Gets the ServletRegistration corresponding to the servlet with the given `servletName`.
`java.util.Map<java.lang.String,? extends ServletRegistration>`	**`getServletRegistrations`**`()` Gets a (possibly empty) Map of the ServletRegistration objects (keyed by servlet name) corresponding to all servlets registered with this ServletContext.
`java.util.Enumeration<Servlet>`	**`getServlets`**`()` **Deprecated.** *As of Java Servlet API 2.0, with no replacement.* *This method was originally defined to return an `Enumeration` of all the servlets known to this servlet context. In this version, this method always returns an empty enumeration and remains only to preserve binary compatibility. This method will be permanently removed in a future version of the Java Servlet API.*
`SessionCookieConfig`	**`getSessionCookieConfig`**`()` Gets the `SessionCookieConfig` object through which various properties of the session tracking cookies created on behalf of this `ServletContext` may be configured.
`void`	**`log`**`(java.lang.Exception exception, java.lang.String msg)` **Deprecated.** *As of Java Servlet API 2.1, use `log(String message, Throwable throwable)` instead.* *This method was originally defined to write an exception's stack trace and an explanatory error message to the servlet log file.*
`void`	**`log`**`(java.lang.String msg)` Writes the specified message to a servlet log file, usually an event log.
`void`	**`log`**`(java.lang.String message, java.lang.Throwable throwable)` Writes an explanatory message and a stack trace for a given `Throwable` exception to the servlet log file.
`void`	**`removeAttribute`**`(java.lang.String name)` Removes the attribute with the given name from this ServletContext.
`void`	**`setAttribute`**`(java.lang.String name, java.lang.Object object)` Binds an object to a given attribute name in this ServletContext.
`boolean`	**`setInitParameter`**`(java.lang.String name, java.lang.String value)` Sets the context initialization parameter with the given name and value on this ServletContext.
`void`	**`setSessionTrackingModes`**`(java.util.Set<SessionTrackingMode> sessionTrackingModes)` Sets the session tracking modes that are to become effective for this `ServletContext`.

Field Detail

TEMPDIR

`static final java.lang.String` **`TEMPDIR`**

The name of the `ServletContext` attribute which stores the private temporary directory (of type `java.io.File`) provided by the servlet container for the `ServletContext`

See Also:
Constant Field Values

ORDERED_LIBS

```
static final java.lang.String ORDERED_LIBS
```

The name of the `ServletContext` attribute whose value (of type `java.util.List<java.lang.String>`) contains the list of names of JAR files in `WEB-INF/lib` ordered by their web fragment names (with possible exclusions if `<absolute-ordering>` without any `<others/>` is being used), or null if no absolute or relative ordering has been specified

See Also:
Constant Field Values

Method Detail

getContextPath

```
java.lang.String getContextPath()
```

Returns the context path of the web application.

The context path is the portion of the request URI that is used to select the context of the request. The context path always comes first in a request URI. The path starts with a `/` character but does not end with a `/` character. For servlets in the default (root) context, this method returns "".

It is possible that a servlet container may match a context by more than one context path. In such cases the `HttpServletRequest.getContextPath()` will return the actual context path used by the request and it may differ from the path returned by this method. The context path returned by this method should be considered as the prime or preferred context path of the application.

Returns:
The context path of the web application, or "" for the default (root) context

Since:
Servlet 2.5

See Also:
`HttpServletRequest.getContextPath()`

---

getContext

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

Returns a `ServletContext` object that corresponds to a specified URL on the server.

This method allows servlets to gain access to the context for various parts of the server, and as needed obtain `RequestDispatcher` objects from the context. The given path must be begin with `/`, is interpreted relative to the server's document root and is matched against the context roots of other web applications hosted on this container.

In a security conscious environment, the servlet container may return `null` for a given URL.

Parameters:
`uripath` - a `String` specifying the context path of another web application in the container.

Returns:
the `ServletContext` object that corresponds to the named URL, or null if either none exists or the container wishes to restrict this access.

See Also:
`RequestDispatcher`

---

getMajorVersion

```
int getMajorVersion()
```

Returns the major version of the Servlet API that this servlet container supports. All implementations that comply with Version 3.0 must have this method return the integer 3.

Returns:
3

---

getMinorVersion

```
int getMinorVersion()
```

Returns the minor version of the Servlet API that this servlet container supports. All implementations that comply with Version 3.0 must have this method return the integer 0.

Returns:
0

---

getEffectiveMajorVersion

`int getEffectiveMajorVersion()`

Gets the major version of the Servlet specification that the application represented by this ServletContext is based on.

The value returned may be different from `getMajorVersion()`, which returns the major version of the Servlet specification supported by the Servlet container.

Returns:
the major version of the Servlet specification that the application represented by this ServletContext is based on

Throws:
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

getEffectiveMinorVersion

`int getEffectiveMinorVersion()`

Gets the minor version of the Servlet specification that the application represented by this ServletContext is based on.

The value returned may be different from `getMinorVersion()`, which returns the minor version of the Servlet specification supported by the Servlet container.

Returns:
the minor version of the Servlet specification that the application xrepresented by this ServletContext is based on

Throws:
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

getMimeType

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

Returns the MIME type of the specified file, or `null` if the MIME type is not known. The MIME type is determined by the configuration of the servlet container, and may be specified in a web application deployment descriptor. Common MIME types include `text/html` and `image/gif`.

Parameters:
`file` - a `String` specifying the name of a file

Returns:
a `String` specifying the file's MIME type

---

getResourcePaths

`java.util.Set<java.lang.String> getResourcePaths(java.lang.String path)`

Returns a directory-like listing of all the paths to resources within the web application whose longest sub-path matches the supplied path argument.

Paths indicating subdirectory paths end with a `/`.

The returned paths are all relative to the root of the web application, or relative to the `/META-INF/resources` directory of a JAR file inside the web application's `/WEB-INF/lib` directory, and have a leading `/`.

For example, for a web application containing:

```
/welcome.html
```

```
   /catalog/index.html
   /catalog/products.html
   /catalog/offers/books.html
   /catalog/offers/music.html
   /customer/login.jsp
   /WEB-INF/web.xml
   /WEB-INF/classes/com.example.OrderServlet.class
   /WEB-INF/lib/catalog.jar!/META-INF/resources/catalog/moreOffers/books.html
```

`getResourcePaths("/")` would return `{"/welcome.html", "/catalog/", "/customer/", "/WEB-INF/"}`, and `getResourcePaths("/catalog/")` would return `{"/catalog/index.html", "/catalog/products.html", "/catalog/offers/", "/catalog/moreOffers/"}`.

Parameters:
`path` - the partial path used to match the resources, which must start with a `/`

Returns:
a Set containing the directory listing, or null if there are no resources in the web application whose path begins with the supplied path.

Since:
Servlet 2.3

getResource

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

Returns a URL to the resource that is mapped to the given path.

The path must begin with a `/` and is interpreted as relative to the current context root, or relative to the `/META-INF/resources` directory of a JAR file inside the web application's `/WEB-INF/lib` directory. This method will first search the document root of the web application for the requested resource, before searching any of the JAR files inside `/WEB-INF/lib`. The order in which the JAR files inside `/WEB-INF/lib` are searched is undefined.

This method allows the servlet container to make a resource available to servlets from any source. Resources can be located on a local or remote file system, in a database, or in a `.war` file.

The servlet container must implement the URL handlers and `URLConnection` objects that are necessary to access the resource.

This method returns `null` if no resource is mapped to the pathname.

Some containers may allow writing to the URL returned by this method using the methods of the URL class.

The resource content is returned directly, so be aware that requesting a `.jsp` page returns the JSP source code. Use a `RequestDispatcher` instead to include results of an execution.

This method has a different purpose than `java.lang.Class.getResource`, which looks up resources based on a class loader. This method does not use class loaders.

Parameters:
`path` - a `String` specifying the path to the resource

Returns:
the resource located at the named path, or `null` if there is no resource at that path

Throws:
`java.net.MalformedURLException` - if the pathname is not given in the correct form

getResourceAsStream

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

Returns the resource located at the named path as an `InputStream` object.

The data in the `InputStream` can be of any type or length. The path must be specified according to the rules given in `getResource`. This method returns `null` if no resource exists at the specified path.

Meta-information such as content length and content type that is available via `getResource` method is lost when using this method.

The servlet container must implement the URL handlers and `URLConnection` objects necessary to access the resource.

This method is different from `java.lang.Class.getResourceAsStream`, which uses a class loader. This method allows servlet containers to make a resource available to a servlet from any location, without using a class loader.

Parameters:

`path` - a `String` specifying the path to the resource

Returns:

the `InputStream` returned to the servlet, or `null` if no resource exists at the specified path

---

getRequestDispatcher

```
RequestDispatcher getRequestDispatcher(java.lang.String path)
```

Returns a `RequestDispatcher` object that acts as a wrapper for the resource located at the given path. A `RequestDispatcher` object can be used to forward a request to the resource or to include the resource in a response. The resource can be dynamic or static.

The pathname must begin with a `/` and is interpreted as relative to the current context root. Use `getContext` to obtain a `RequestDispatcher` for resources in foreign contexts.

This method returns `null` if the `ServletContext` cannot return a `RequestDispatcher`.

Parameters:

`path` - a `String` specifying the pathname to the resource

Returns:

a `RequestDispatcher` object that acts as a wrapper for the resource at the specified path, or `null` if the `ServletContext` cannot return a `RequestDispatcher`

See Also:

`RequestDispatcher`, `ServletContext#getContext`

---

getNamedDispatcher

```
RequestDispatcher getNamedDispatcher(java.lang.String name)
```

Returns a `RequestDispatcher` object that acts as a wrapper for the named servlet.

Servlets (and JSP pages also) may be given names via server administration or via a web application deployment descriptor. A servlet instance can determine its name using `ServletConfig#getServletName`.

This method returns `null` if the `ServletContext` cannot return a `RequestDispatcher` for any reason.

Parameters:

`name` - a `String` specifying the name of a servlet to wrap

Returns:

a `RequestDispatcher` object that acts as a wrapper for the named servlet, or `null` if the `ServletContext` cannot return a `RequestDispatcher`

See Also:

`RequestDispatcher`, `ServletContext#getContext`, `ServletConfig#getServletName`

---

getServlet

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

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

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

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

Throws:

`ServletException`

---

getServlets

```
java.util.Enumeration<Servlet> getServlets()
```

Deprecated. *As of Java Servlet API 2.0, with no replacement.*

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

---

getServletNames

```
java.util.Enumeration<java.lang.String> getServletNames()
```

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

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

---

log

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

Writes the specified message to a servlet log file, usually an event log. The name and type of the servlet log file is specific to the servlet container.

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

---

log

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

Deprecated. *As of Java Servlet API 2.1, use `log(String message, Throwable throwable)` instead.*

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

---

log

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

Writes an explanatory message and a stack trace for a given `Throwable` exception to the servlet log file. The name and type of the servlet log file is specific to the servlet container, usually an event log.

Parameters:
`message` - a `String` that describes the error or exception
`throwable` - the `Throwable` error or exception

---

getRealPath

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

Gets the *real* path corresponding to the given *virtual* path.

For example, if `path` is equal to `/index.html`, this method will return the absolute file path on the server's filesystem to which a request of the form `http://<host>:<port>/<contextPath>/index.html` would be mapped, where `<contextPath>` corresponds to the context path of this ServletContext.

The real path returned will be in a form appropriate to the computer and operating system on which the servlet container is running, including the proper path separators.

Resources inside the `/META-INF/resources` directories of JAR files bundled in the application's `/WEB-INF/lib` directory must be considered only if the container has unpacked them from their containing JAR file, in which case the path to the unpacked location must be returned.

This method returns `null` if the servlet container is unable to translate the given *virtual* path to a *real* path.

Parameters:
`path` - the *virtual* path to be translated to a *real* path
Returns:
the *real* path, or `null` if the translation cannot be performed

---

getServerInfo

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

Returns the name and version of the servlet container on which the servlet is running.

The form of the returned string is *servername*/*versionnumber*. For example, the JavaServer Web Development Kit may return the string `JavaServer Web Dev Kit/1.0`.

The servlet container may return other optional information after the primary string in parentheses, for example, `JavaServer Web Dev Kit/1.0 (JDK 1.1.6; Windows NT 4.0 x86)`.

Returns:
a `String` containing at least the servlet container name and version number

---

getInitParameter

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

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

This method can make available configuration information useful to an entire web application. For example, it can provide a webmaster's email address or the name of a system that holds critical data.

Parameters:
`name` - a `String` containing the name of the parameter whose value is requested
Returns:
a `String` containing at least the servlet container name and version number
See Also:
`ServletConfig#getInitParameter`

---

getInitParameterNames

```
java.util.Enumeration<java.lang.String> getInitParameterNames()
```

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

Returns:
an `Enumeration` of `String` objects containing the names of the context's initialization parameters
See Also:
`ServletConfig#getInitParameter`

---

setInitParameter

```
boolean setInitParameter(java.lang.String name,
                         java.lang.String value)
```

Sets the context initialization parameter with the given name and value on this ServletContext.

Parameters:
`name` - the name of the context initialization parameter to set
`value` - the value of the context initialization parameter to set
Returns:
true if the context initialization parameter with the given name and value was set successfully on this ServletContext, and false if it was not set because this ServletContext already contains a context initialization parameter with a matching name
Throws:
`IllegalStateException` - if this ServletContext has already been initialized
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`
Since:
Servlet 3.0

---

getAttribute

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

Returns the servlet container attribute with the given name, or `null` if there is no attribute by that name.

An attribute allows a servlet container to give the servlet additional information not already provided by this interface. See your server documentation for information about its attributes. A list of supported attributes can be retrieved using `getAttributeNames`.

The attribute is returned as a `java.lang.Object` or some subclass.

Attribute names should follow the same convention as package names. The Java Servlet API specification reserves names matching `java.*`, `javax.*`, and `sun.*`.

Parameters:
`name` - a `String` specifying the name of the attribute

Returns:
an `Object` containing the value of the attribute, or `null` if no attribute exists matching the given name

See Also:
`ServletContext#getAttributeNames`

---

getAttributeNames

```
java.util.Enumeration<java.lang.String> getAttributeNames()
```

Returns an `Enumeration` containing the attribute names available within this ServletContext.

Use the `getAttribute(java.lang.String)` method with an attribute name to get the value of an attribute.

Returns:
an `Enumeration` of attribute names

See Also:
`getAttribute(java.lang.String)`

---

setAttribute

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

Binds an object to a given attribute name in this ServletContext. If the name specified is already used for an attribute, this method will replace the attribute with the new to the new attribute.

If listeners are configured on the `ServletContext` the container notifies them accordingly.

If a null value is passed, the effect is the same as calling `removeAttribute()`.

Attribute names should follow the same convention as package names. The Java Servlet API specification reserves names matching `java.*`, `javax.*`, and `sun.*`.

Parameters:
`name` - a `String` specifying the name of the attribute
`object` - an `Object` representing the attribute to be bound

---

removeAttribute

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

Removes the attribute with the given name from this ServletContext. After removal, subsequent calls to `getAttribute(java.lang.String)` to retrieve the attribute's value will return `null`.

If listeners are configured on the `ServletContext` the container notifies them accordingly.

Parameters:
`name` - a `String` specifying the name of the attribute to be removed

---

getServletContextName

```
java.lang.String getServletContextName()
```

Returns the name of this web application corresponding to this ServletContext as specified in the deployment descriptor for this web application by the display-name element.

Returns:
The name of the web application or null if no name has been declared in the deployment descriptor.

Since:
Servlet 2.3

---

addServlet

```
ServletRegistration.Dynamic addServlet(java.lang.String servletName,
                                       java.lang.String className)
```

Adds the servlet with the given name and class name to this servlet context.

The registered servlet may be further configured via the returned `ServletRegistration` object.

The specified `className` will be loaded using the classloader associated with the application represented by this ServletContext.

If this ServletContext already contains a preliminary ServletRegistration for a servlet with the given `servletName`, it will be completed (by assigning the given `className` to it) and returned.

This method introspects the class with the given `className` for the `ServletSecurity`, `MultipartConfig`, `javax.annotation.security.RunAs`, and `javax.annotation.security.DeclareRoles` annotations. In addition, this method supports resource injection if the class with the given `className` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
`servletName` - the name of the servlet
`className` - the fully qualified class name of the servlet
Returns:
a ServletRegistration object that may be used to further configure the registered servlet, or `null` if this ServletContext already contains a complete ServletRegistration for a servlet with the given `servletName`
Throws:
`IllegalStateException` - if this ServletContext has already been initialized
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`
Since:
Servlet 3.0

---

addServlet

```
ServletRegistration.Dynamic addServlet(java.lang.String servletName,
                                       Servlet servlet)
```

Registers the given servlet instance with this ServletContext under the given `servletName`.

The registered servlet may be further configured via the returned `ServletRegistration` object.

If this ServletContext already contains a preliminary ServletRegistration for a servlet with the given `servletName`, it will be completed (by assigning the class name of the given servlet instance to it) and returned.

Parameters:
`servletName` - the name of the servlet
`servlet` - the servlet instance to register
Returns:
a ServletRegistration object that may be used to further configure the given servlet, or `null` if this ServletContext already contains a complete ServletRegistration for a servlet with the given `servletName` or if the same servlet instance has already been registered with this or another ServletContext in the same container
Throws:
`IllegalStateException` - if this ServletContext has already been initialized
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`
`IllegalArgumentException` - if the given servlet instance implements `SingleThreadModel`
Since:
Servlet 3.0

---

addServlet

```
ServletRegistration.Dynamic addServlet(java.lang.String servletName,
                                       java.lang.Class<? extends Servlet> servletClass)
```

Adds the servlet with the given name and class type to this servlet context.

The registered servlet may be further configured via the returned `ServletRegistration` object.

If this ServletContext already contains a preliminary ServletRegistration for a servlet with the given `servletName`, it will be completed (by assigning the name of the given `servletClass` to it) and returned.

This method introspects the given `servletClass` for the `ServletSecurity`, `MultipartConfig`, `javax.annotation.security.RunAs`, and `javax.annotation.security.DeclareRoles` annotations. In addition, this method supports resource injection if the given `servletClass` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
`servletName` - the name of the servlet
`servletClass` - the class object from which the servlet will be instantiated
Returns:
a ServletRegistration object that may be used to further configure the registered servlet, or `null` if this ServletContext already contains a complete ServletRegistration for the given `servletName`
Throws:
`IllegalStateException` - if this ServletContext has already been initialized
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`
Since:
Servlet 3.0

---

createServlet

```
<T extends Servlet> T createServlet(java.lang.Class<T> clazz)
                                throws ServletException
```

Instantiates the given Servlet class.

The returned Servlet instance may be further customized before it is registered with this ServletContext via a call to `addServlet(String,Servlet)`.

The given Servlet class must define a zero argument constructor, which is used to instantiate it.

This method introspects the given `clazz` for the following annotations: `ServletSecurity`, `MultipartConfig`, `javax.annotation.security.RunAs`, and `javax.annotation.security.DeclareRoles`. In addition, this method supports resource injection if the given `clazz` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
`clazz` - the Servlet class to instantiate
Returns:
the new Servlet instance
Throws:
`ServletException` - if the given `clazz` fails to be instantiated
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`
Since:
Servlet 3.0

---

getServletRegistration

```
ServletRegistration getServletRegistration(java.lang.String servletName)
```

Gets the ServletRegistration corresponding to the servlet with the given `servletName`.

Returns:
the (complete or preliminary) ServletRegistration for the servlet with the given `servletName`, or null if no ServletRegistration exists under that name
Throws:
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`
Since:
Servlet 3.0

---

getServletRegistrations

```
java.util.Map<java.lang.String,? extends ServletRegistration> getServletRegistrations()
```

Gets a (possibly empty) Map of the ServletRegistration objects (keyed by servlet name) corresponding to all servlets registered with this ServletContext.

The returned Map includes the ServletRegistration objects corresponding to all declared and annotated servlets, as well as the ServletRegistration objects corresponding to all servlets that have been added via one of the `addServlet` methods.

If permitted, any changes to the returned Map must not affect this ServletContext.

Returns:
Map of the (complete and preliminary) ServletRegistration objects corresponding to all servlets currently registered with this ServletContext

Throws:
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

addFilter

```
FilterRegistration.Dynamic addFilter(java.lang.String filterName,
                                     java.lang.String className)
```

Adds the filter with the given name and class name to this servlet context.

The registered filter may be further configured via the returned `FilterRegistration` object.

The specified `className` will be loaded using the classloader associated with the application represented by this ServletContext.

If this ServletContext already contains a preliminary FilterRegistration for a filter with the given `filterName`, it will be completed (by assigning the given `className` to it) and returned.

This method supports resource injection if the class with the given `className` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
`filterName` - the name of the filter
`className` - the fully qualified class name of the filter

Returns:
a FilterRegistration object that may be used to further configure the registered filter, or `null` if this ServletContext already contains a complete FilterRegistration for a filter with the given `filterName`

Throws:
`IllegalStateException` - if this ServletContext has already been initialized
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

addFilter

```
FilterRegistration.Dynamic addFilter(java.lang.String filterName,
                                     Filter filter)
```

Registers the given filter instance with this ServletContext under the given `filterName`.

The registered filter may be further configured via the returned `FilterRegistration` object.

If this ServletContext already contains a preliminary FilterRegistration for a filter with the given `filterName`, it will be completed (by assigning the class name of the given filter instance to it) and returned.

Parameters:
`filterName` - the name of the filter
`filter` - the filter instance to register

Returns:
a FilterRegistration object that may be used to further configure the given filter, or `null` if this ServletContext already contains a complete FilterRegistration for a filter with the given `filterName` or if the same filter instance has already been registered with this or another ServletContext in the same container

Throws:
`IllegalStateException` - if this ServletContext has already been initialized

`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

addFilter

```
FilterRegistration.Dynamic addFilter(java.lang.String filterName,
                                     java.lang.Class<? extends Filter> filterClass)
```

Adds the filter with the given name and class type to this servlet context.

The registered filter may be further configured via the returned `FilterRegistration` object.

If this ServletContext already contains a preliminary FilterRegistration for a filter with the given `filterName`, it will be completed (by assigning the name of the given `filterClass` to it) and returned.

This method supports resource injection if the given `filterClass` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
`filterName` - the name of the filter
`filterClass` - the class object from which the filter will be instantiated

Returns:
a FilterRegistration object that may be used to further configure the registered filter, or `null` if this ServletContext already contains a complete FilterRegistration for a filter with the given `filterName`

Throws:
`IllegalStateException` - if this ServletContext has already been initialized
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

createFilter

```
<T extends Filter> T createFilter(java.lang.Class<T> clazz)
                              throws ServletException
```

Instantiates the given Filter class.

The returned Filter instance may be further customized before it is registered with this ServletContext via a call to `addFilter(String,Filter)`.

The given Filter class must define a zero argument constructor, which is used to instantiate it.

This method supports resource injection if the given `clazz` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
`clazz` - the Filter class to instantiate

Returns:
the new Filter instance

Throws:
`ServletException` - if the given `clazz` fails to be instantiated
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

getFilterRegistration

```
FilterRegistration getFilterRegistration(java.lang.String filterName)
```

Gets the FilterRegistration corresponding to the filter with the given `filterName`.

Returns:
the (complete or preliminary) FilterRegistration for the filter with the given `filterName`, or null if no FilterRegistration exists

under that name

Throws:

`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:

Servlet 3.0

---

getFilterRegistrations

```
java.util.Map<java.lang.String,? extends FilterRegistration> getFilterRegistrations()
```

Gets a (possibly empty) Map of the FilterRegistration objects (keyed by filter name) corresponding to all filters registered with this ServletContext.

The returned Map includes the FilterRegistration objects corresponding to all declared and annotated filters, as well as the FilterRegistration objects corresponding to all filters that have been added via one of the `addFilter` methods.

Any changes to the returned Map must not affect this ServletContext.

Returns:

Map of the (complete and preliminary) FilterRegistration objects corresponding to all filters currently registered with this ServletContext

Throws:

`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:

Servlet 3.0

---

getSessionCookieConfig

```
SessionCookieConfig getSessionCookieConfig()
```

Gets the `SessionCookieConfig` object through which various properties of the session tracking cookies created on behalf of this `ServletContext` may be configured.

Repeated invocations of this method will return the same `SessionCookieConfig` instance.

Returns:

the `SessionCookieConfig` object through which various properties of the session tracking cookies created on behalf of this `ServletContext` may be configured

Throws:

`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:

Servlet 3.0

---

setSessionTrackingModes

```
void setSessionTrackingModes(java.util.Set<SessionTrackingMode> sessionTrackingModes)
```

Sets the session tracking modes that are to become effective for this `ServletContext`.

The given `sessionTrackingModes` replaces any session tracking modes set by a previous invocation of this method on this `ServletContext`.

Parameters:

`sessionTrackingModes` - the set of session tracking modes to become effective for this `ServletContext`

Throws:

`IllegalStateException` - if this ServletContext has already been initialized

`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

`IllegalArgumentException` - if `sessionTrackingModes` specifies a combination of `SessionTrackingMode.SSL` with a session tracking mode other than `SessionTrackingMode.SSL`, or if `sessionTrackingModes` specifies a session tracking mode that is not supported by the servlet container

Since:

Servlet 3.0

---

getDefaultSessionTrackingModes

`java.util.Set<SessionTrackingMode> **getDefaultSessionTrackingModes**()`

Gets the session tracking modes that are supported by default for this `ServletContext`.

Returns:
set of the session tracking modes supported by default for this `ServletContext`

Throws:
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

getEffectiveSessionTrackingModes

`java.util.Set<SessionTrackingMode> **getEffectiveSessionTrackingModes**()`

Gets the session tracking modes that are in effect for this `ServletContext`.

The session tracking modes in effect are those provided to `setSessionTrackingModes`.

By default, the session tracking modes returned by `getDefaultSessionTrackingModes` are in effect.

Returns:
set of the session tracking modes in effect for this `ServletContext`

Throws:
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

addListener

`void **addListener**(java.lang.String className)`

Adds the listener with the given class name to this ServletContext.

The class with the given name will be loaded using the classloader associated with the application represented by this ServletContext, and must implement one or more of the following interfaces:

- `ServletContextAttributeListener`
- `ServletRequestListener`
- `ServletRequestAttributeListener`
- `HttpSessionListener`
- `HttpSessionAttributeListener`

If this ServletContext was passed to `ServletContainerInitializer#onStartup`, then the class with the given name may also implement `ServletContextListener`, in addition to the interfaces listed above.

As part of this method call, the container must load the class with the specified class name to ensure that it implements one of the required interfaces.

If the class with the given name implements a listener interface whose invocation order corresponds to the declaration order (i.e., if it implements `ServletRequestListener`, `ServletContextListener`, or `HttpSessionListener`), then the new listener will be added to the end of the ordered list of listeners of that interface.

This method supports resource injection if the class with the given `className` represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
`className` - the fully qualified class name of the listener

Throws:
`IllegalArgumentException` - if the class with the given name does not implement any of the above interfaces, or if it implements `ServletContextListener` and this ServletContext was not passed to `ServletContainerInitializer#onStartup`
`IllegalStateException` - if this ServletContext has already been initialized

`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

addListener

```
<T extends java.util.EventListener> void addListener(T t)
```

Adds the given listener to this ServletContext.

The given listener must be an instance of one or more of the following interfaces:

- `ServletContextAttributeListener`
- `ServletRequestListener`
- `ServletRequestAttributeListener`
- `HttpSessionListener`
- `HttpSessionAttributeListener`

If this ServletContext was passed to `ServletContainerInitializer#onStartup`, then the given listener may also be an instance of `ServletContextListener`, in addition to the interfaces listed above.

If the given listener is an instance of a listener interface whose invocation order corresponds to the declaration order (i.e., if it is an instance of `ServletRequestListener`, `ServletContextListener`, or `HttpSessionListener`), then the listener will be added to the end of the ordered list of listeners of that interface.

Parameters:
`t` - the listener to be added

Throws:
`IllegalArgumentException` - if the given listener is not an instance of any of the above interfaces, or if it is an instance of `ServletContextListener` and this ServletContext was not passed to `ServletContainerInitializer#onStartup`
`IllegalStateException` - if this ServletContext has already been initialized
`UnsupportedOperationException` - if this ServletContext was passed to the `ServletContextListener#contextInitialized` method of a `ServletContextListener` that was neither declared in `web.xml` or `web-fragment.xml`, nor annotated with `WebListener`

Since:
Servlet 3.0

---

addListener

```
void addListener(java.lang.Class<? extends java.util.EventListener> listenerClass)
```

Adds a listener of the given class type to this ServletContext.

The given `listenerClass` must implement one or more of the following interfaces:

- `ServletContextAttributeListener`
- `ServletRequestListener`
- `ServletRequestAttributeListener`
- `HttpSessionListener`
- `HttpSessionAttributeListener`

If this ServletContext was passed to `ServletContainerInitializer#onStartup`, then the given `listenerClass` may also implement `ServletContextListener`, in addition to the interfaces listed above.

If the given `listenerClass implements a listener interface whose invocation order corresponds to the declaration order (i.e., if it implements ServletRequestListener, ServletContextListener, or HttpSessionListener), then the new listener will be added to the end of the ordered list of listeners of that interface.`

`This method supports resource injection if the given listenerClass represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.`

`Parameters:`
`listenerClass - the listener class to be instantiated`

`Throws:`
`IllegalArgumentException - if the given listenerClass does not implement any of the above interfaces, or if it implements ServletContextListener and this ServletContext was not passed to ServletContainerInitializer#onStartup`
`IllegalStateException - if this ServletContext has already been initialized`
`UnsupportedOperationException - if this ServletContext was passed to the ServletContextListener#contextInitialized method of a ServletContextListener that was neither declared in web.xml or web-fragment.xml, nor annotated with WebListener`

`Since:`

Servlet 3.0

---

createListener

```
<T extends java.util.EventListener> T createListener(java.lang.Class<T> clazz)
                                                 throws ServletException
```

Instantiates the given EventListener class.

The specified EventListener class must implement at least one of the ServletContextListener, ServletContextAttributeListener, ServletRequestListener, ServletRequestAttributeListener, HttpSessionListener, or HttpSessionAttributeListener interfaces.

The returned EventListener instance may be further customized before it is registered with this ServletContext via a call to addListener(EventListener).

The given EventListener class must define a zero argument constructor, which is used to instantiate it.

This method supports resource injection if the given clazz represents a Managed Bean. See the Java EE platform and JSR 299 specifications for additional details about Managed Beans and resource injection.

Parameters:
clazz - the EventListener class to instantiate
Returns:
the new EventListener instance
Throws:
ServletException - if the given clazz fails to be instantiated
UnsupportedOperationException - if this ServletContext was passed to the ServletContextListener#contextInitialized method of a ServletContextListener that was neither declared in web.xml or web-fragment.xml, nor annotated with WebListener
IllegalArgumentException - if the specified EventListener class does not implement any of the ServletContextListener, ServletContextAttributeListener, ServletRequestListener, ServletRequestAttributeListener, HttpSessionListener, or HttpSessionAttributeListener interfaces.
Since:
Servlet 3.0

---

getJspConfigDescriptor

JspConfigDescriptor **getJspConfigDescriptor**()

Gets the <jsp-config> related configuration that was aggregated from the web.xml and web-fragment.xml descriptor files of the web application represented by this ServletContext.

Returns:
the <jsp-config> related configuration that was aggregated from the web.xml and web-fragment.xml descriptor files of the web application represented by this ServletContext, or null if no such configuration exists
Throws:
UnsupportedOperationException - if this ServletContext was passed to the ServletContextListener#contextInitialized method of a ServletContextListener that was neither declared in web.xml or web-fragment.xml, nor annotated with WebListener
Since:
Servlet 3.0
See Also:
JspConfigDescriptor

---

getClassLoader

java.lang.ClassLoader **getClassLoader**()

Gets the class loader of the web application represented by this ServletContext.

If a security manager exists, and the caller's class loader is not the same as, or an ancestor of the requested class loader, then the security manager's checkPermission method is called with a RuntimePermission("getClassLoader") permission to check whether access to the requested class loader should be granted.

Returns:
the class loader of the web application represented by this ServletContext
Throws:
UnsupportedOperationException - if this ServletContext was passed to the ServletContextListener#contextInitialized method of a ServletContextListener that was neither declared in web.xml or web-fragment.xml, nor annotated with WebListener
SecurityException - if a security manager denies access to the requested class loader
Since:
Servlet 3.0

---

declareRoles

void **declareRoles**(java.lang.String... roleNames)

Declares role names that are tested using isUserInRole.

Roles that are implicitly declared as a result of their use within the setServletSecurity or setRunAsRole methods of the ServletRegistration interface need not be declared.

Parameters:
roleNames - the role names being declared

Throws:

`UnsupportedOperationException` - if this ServletContext was passed to the ServletContextListener#contextInitialized method of a ServletContextListener that was neither declared in web.xml or web-fragment.xml, nor annotated with WebListener
`IllegalArgumentException` - if any of the argument roleNames is null or the empty string
`IllegalStateException` - if the ServletContext has already been initialized

Since:

Servlet 3.0

Submit a bug or feature

javax.servlet

Class ServletContextAttributeEvent

```
java.lang.Object
  └java.util.EventObject
      └javax.servlet.ServletContextEvent
          └javax.servlet.ServletContextAttributeEvent
```

All Implemented Interfaces:
java.io.Serializable

```
public class ServletContextAttributeEvent
extends ServletContextEvent
```

Event class for notifications about changes to the attributes of the ServletContext of a web application.

Since:
Servlet 2.3

See Also:
`ServletContextAttributeListener`, Serialized Form

Field Summary

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

Constructor Summary

`ServletContextAttributeEvent``(ServletContext source, java.lang.String name, java.lang.Object value)` Constructs a ServletContextAttributeEvent from the given ServletContext, attribute name, and attribute value.

Method Summary

`java.lang.String`	**`getName`**`()` Gets the name of the ServletContext attribute that changed.
`java.lang.Object`	**`getValue`**`()` Gets the value of the ServletContext attribute that changed.

Methods inherited from class javax.servlet.ServletContextEvent
`getServletContext`

Methods inherited from class java.util.EventObject
`getSource, toString`

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

Constructor Detail

ServletContextAttributeEvent

```
public ServletContextAttributeEvent(ServletContext source,
                                    java.lang.String name,
                                    java.lang.Object value)
```

Constructs a ServletContextAttributeEvent from the given ServletContext, attribute name, and attribute value.

Parameters:

`source` - the ServletContext whose attribute changed
`name` - the name of the ServletContext attribute that changed
`value` - the value of the ServletContext attribute that changed

Method Detail

getName

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

Gets the name of the ServletContext attribute that changed.

Returns:

the name of the ServletContext attribute that changed

---

getValue

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

Gets the value of the ServletContext attribute that changed.

If the attribute was added, this is the value of the attribute. If the attribute was removed, this is the value of the removed attribute. If the attribute was replaced, this is the old value of the attribute.

Returns:

the value of the ServletContext attribute that changed

---

javax.servlet

Interface ServletContextAttributeListener

All Superinterfaces:

java.util.EventListener

```
public interface ServletContextAttributeListener
extends java.util.EventListener
```

Interface for receiving notification events about ServletContext attribute changes.

In order to receive these notification events, the implementation class must be either declared in the deployment descriptor of the web application, annotated with `WebListener`, or registered via one of the addListener methods defined on `ServletContext`.

The order in which implementations of this interface are invoked is unspecified.

Since:

Servlet 2.3

See Also:

`ServletContextAttributeEvent`

Method Summary

`void`	`attributeAdded(ServletContextAttributeEvent event)` Receives notification that an attribute has been added to the ServletContext.
`void`	`attributeRemoved(ServletContextAttributeEvent event)` Receives notification that an attribute has been removed from the ServletContext.
`void`	`attributeReplaced(ServletContextAttributeEvent event)`

Method Detail

attributeAdded

```
void attributeAdded(ServletContextAttributeEvent event)
```

Receives notification that an attribute has been added to the ServletContext.

Parameters:

`event` - the ServletContextAttributeEvent containing the ServletContext to which the attribute was added, along with the attribute name and value

attributeRemoved

```
void attributeRemoved(ServletContextAttributeEvent event)
```

Receives notification that an attribute has been removed from the ServletContext.

Parameters:

`event` - the ServletContextAttributeEvent containing the ServletContext from which the attribute was removed, along with the attribute name and value

attributeReplaced

```
void attributeReplaced(ServletContextAttributeEvent event)
```

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class ServletContextEvent

```
java.lang.Object
  └java.util.EventObject
      └javax.servlet.ServletContextEvent
```

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

ServletContextAttributeEvent

```
public class ServletContextEvent
extends java.util.EventObject
```

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

Since:

Servlet 2.3

See Also:

`ServletContextListener`, Serialized Form

Field Summary

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

Constructor Summary

`ServletContextEvent(ServletContext source)` Construct a ServletContextEvent from the given context.

Method Summary

`ServletContext`	`getServletContext()` Return the ServletContext that changed.

Methods inherited from class java.util.EventObject
`getSource, toString`

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

Constructor Detail

ServletContextEvent

```
public ServletContextEvent(ServletContext source)
```

Construct a ServletContextEvent from the given context.

Parameters:

`source` - - the ServletContext that is sending the event.

Method Detail

getServletContext

```
public ServletContext getServletContext()
```

Return the ServletContext that changed.

Returns:

the ServletContext that sent the event.

Submit a bug or feature

javax.servlet

Class ServletException

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └javax.servlet.ServletException
```

All Implemented Interfaces:
java.io.Serializable

Direct Known Subclasses:
UnavailableException

```
public class ServletException
extends java.lang.Exception
```

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

Author:
Various

See Also:
Serialized Form

Constructor Summary

`ServletException()` Constructs a new servlet exception.
`ServletException(java.lang.String message)` Constructs a new servlet exception with the specified message.
`ServletException(java.lang.String message, java.lang.Throwable rootCause)` Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation, including a description message.
`ServletException(java.lang.Throwable rootCause)` Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation.

Method Summary

`java.lang.Throwable`	`getRootCause()` Returns the exception that caused this servlet exception.

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

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

Constructor Detail

ServletException

```
public ServletException()
```

Constructs a new servlet exception.

ServletException

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

Constructs a new servlet exception with the specified message. The message can be written to the server log and/or displayed for the user.

Parameters:

`message` - a `String` specifying the text of the exception message

ServletException

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

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

Parameters:

`message` - a `String` containing the text of the exception message

`rootCause` - the `Throwable` exception that interfered with the servlet's normal operation, making this servlet exception necessary

---

ServletException

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

Constructs a new servlet exception when the servlet needs to throw an exception and include a message about the "root cause" exception that interfered with its normal operation. The exception's message is based on the localized message of the underlying exception.

This method calls the `getLocalizedMessage` method on the `Throwable` exception to get a localized exception message. When subclassing `ServletException`, this method can be overridden to create an exception message designed for a specific locale.

Parameters:

`rootCause` - the `Throwable` exception that interfered with the servlet's normal operation, making the servlet exception necessary

Method Detail

getRootCause

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

Returns the exception that caused this servlet exception.

Returns:

the `Throwable` that caused this servlet exception

javax.servlet

Class ServletInputStream

```
java.lang.Object
  └java.io.InputStream
      └javax.servlet.ServletInputStream
```

All Implemented Interfaces:

java.io.Closeable

```
public abstract class ServletInputStream
extends java.io.InputStream
```

Provides an input stream for reading binary data from a client request, including an efficient `readLine` method for reading data one line at a time. With some protocols, such as HTTP POST and PUT, a `ServletInputStream` object can be used to read data sent from the client.

A `ServletInputStream` object is normally retrieved via the `ServletRequest#getInputStream` method.

This is an abstract class that a servlet container implements. Subclasses of this class must implement the `java.io.InputStream.read()` method.

Author:

Various

See Also:

`ServletRequest`

Constructor Summary

`protected`	`ServletInputStream()` Does nothing, because this is an abstract class.

Method Summary

`int`	`readLine(byte[] b, int off, int len)` Reads the input stream, one line at a time.

Methods inherited from class java.io.InputStream

`available, close, mark, markSupported, read, read, read, reset, skip`

Methods inherited from class java.lang.Object

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

Constructor Detail

ServletInputStream

```
protected ServletInputStream()
```

Does nothing, because this is an abstract class.

Method Detail

readLine

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

Reads the input stream, one line at a time. Starting at an offset, reads bytes into an array, until it reads a certain number of bytes or reaches a newline character, which it reads into the array as well.

This method returns -1 if it reaches the end of the input stream before reading the maximum number of bytes.

Parameters:

`b` - an array of bytes into which data is read

`off` - an integer specifying the character at which this method begins reading

`len` - an integer specifying the maximum number of bytes to read

Returns:

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

Throws:

`java.io.IOException` - if an input or output exception has occurred

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class ServletOutputStream

```
java.lang.Object
  └java.io.OutputStream
      └javax.servlet.ServletOutputStream
```

All Implemented Interfaces:

java.io.Closeable, java.io.Flushable

```
public abstract class ServletOutputStream
extends java.io.OutputStream
```

Provides an output stream for sending binary data to the client. A `ServletOutputStream` object is normally retrieved via the `ServletResponse#getOutputStream` method.

This is an abstract class that the servlet container implements. Subclasses of this class must implement the `java.io.OutputStream.write(int)` method.

Author:

Various

See Also:

`ServletResponse`

Constructor Summary

`protected`	`ServletOutputStream()` Does nothing, because this is an abstract class.

Method Summary

`void`	`print(boolean b)` Writes a `boolean` value to the client, with no carriage return-line feed (CRLF) character at the end.
`void`	`print(char c)` Writes a character to the client, with no carriage return-line feed (CRLF) at the end.
`void`	`print(double d)` Writes a `double` value to the client, with no carriage return-line feed (CRLF) at the end.
`void`	`print(float f)` Writes a `float` value to the client, with no carriage return-line feed (CRLF) at the end.
`void`	`print(int i)` Writes an int to the client, with no carriage return-line feed (CRLF) at the end.
`void`	`print(long l)` Writes a `long` value to the client, with no carriage return-line feed (CRLF) at the end.

`void`	**`print`**`(java.lang.String s)` Writes a `String` to the client, without a carriage return-line feed (CRLF) character at the end.
`void`	**`println`**`()` Writes a carriage return-line feed (CRLF) to the client.
`void`	**`println`**`(boolean b)` Writes a `boolean` value to the client, followed by a carriage return-line feed (CRLF).
`void`	**`println`**`(char c)` Writes a character to the client, followed by a carriage return-line feed (CRLF).
`void`	**`println`**`(double d)` Writes a `double` value to the client, followed by a carriage return-line feed (CRLF).
`void`	**`println`**`(float f)` Writes a `float` value to the client, followed by a carriage return-line feed (CRLF).
`void`	**`println`**`(int i)` Writes an int to the client, followed by a carriage return-line feed (CRLF) character.
`void`	**`println`**`(long l)` Writes a `long` value to the client, followed by a carriage return-line feed (CRLF).
`void`	**`println`**`(java.lang.String s)` Writes a `String` to the client, followed by a carriage return-line feed (CRLF).

Methods inherited from class java.io.OutputStream
`close, flush, write, write, write`

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

Constructor Detail

ServletOutputStream

```
protected ServletOutputStream()
```

Does nothing, because this is an abstract class.

Method Detail

print

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

Writes a `String` to the client, without a carriage return-line feed (CRLF) character at the end.

Parameters:

`s` - the `String` to send to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

print

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

Writes a `boolean` value to the client, with no carriage return-line feed (CRLF) character at the end.

Parameters:

`b` - the `boolean` value to send to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

print

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

Writes a character to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:

`c` - the character to send to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

print

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

Writes an int to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:

`i` - the int to send to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

print

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

Writes a `long` value to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:

`l` - the `long` value to send to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

print

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

Writes a `float` value to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:
`f` - the `float` value to send to the client
Throws:
`java.io.IOException` - if an input or output exception occurred

---

print

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

Writes a `double` value to the client, with no carriage return-line feed (CRLF) at the end.

Parameters:
`d` - the `double` value to send to the client
Throws:
`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes a carriage return-line feed (CRLF) to the client.

Throws:
`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes a `String` to the client, followed by a carriage return-line feed (CRLF).

Parameters:
`s` - the `String` to write to the client
Throws:
`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes a `boolean` value to the client, followed by a carriage return-line feed (CRLF).

Parameters:

`b` - the `boolean` value to write to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes a character to the client, followed by a carriage return-line feed (CRLF).

Parameters:

`c` - the character to write to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes an int to the client, followed by a carriage return-line feed (CRLF) character.

Parameters:

`i` - the int to write to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes a `long` value to the client, followed by a carriage return-line feed (CRLF).

Parameters:

`l` - the `long` value to write to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes a `float` value to the client, followed by a carriage return-line feed (CRLF).

Parameters:

`f` - the `float` value to write to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

---

println

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

Writes a `double` value to the client, followed by a carriage return-line feed (CRLF).

Parameters:
`d` - the `double` value to write to the client

Throws:
`java.io.IOException` - if an input or output exception occurred

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface ServletRequest

All Known Subinterfaces:

HttpServletRequest

All Known Implementing Classes:

HttpServletRequestWrapper, ServletRequestWrapper

`public interface` **`ServletRequest`**

Defines an object to provide client request information to a servlet. The servlet container creates a `ServletRequest` object and passes it as an argument to the servlet's `service` method.

A `ServletRequest` object provides data including parameter name and values, attributes, and an input stream. Interfaces that extend `ServletRequest` can provide additional protocol-specific data (for example, HTTP data is provided by `HttpServletRequest`.

Author:

Various

See Also:

`HttpServletRequest`

Method Summary

`AsyncContext`	**`getAsyncContext`**`()` Gets the AsyncContext that was created or reinitialized by the most recent invocation of `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on this request.
`java.lang.Object`	**`getAttribute`**`(java.lang.String name)` Returns the value of the named attribute as an `Object`, or `null` if no attribute of the given name exists.
`java.util.Enumeration<java.lang.String>`	**`getAttributeNames`**`()` Returns an `Enumeration` containing the names of the attributes available to this request.
`java.lang.String`	**`getCharacterEncoding`**`()` Returns the name of the character encoding used in the body of this request.
`int`	**`getContentLength`**`()` Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known.

`java.lang.String`	**`getContentType`**`()` Returns the MIME type of the body of the request, or `null` if the type is not known.
`DispatcherType`	**`getDispatcherType`**`()` Gets the dispatcher type of this request.
`ServletInputStream`	**`getInputStream`**`()` Retrieves the body of the request as binary data using a `ServletInputStream`.
`java.lang.String`	**`getLocalAddr`**`()` Returns the Internet Protocol (IP) address of the interface on which the request was received.
`java.util.Locale`	**`getLocale`**`()` Returns the preferred `Locale` that the client will accept content in, based on the Accept-Language header.
`java.util.Enumeration<java.util.Locale>`	**`getLocales`**`()` Returns an `Enumeration` of `Locale` objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header.
`java.lang.String`	**`getLocalName`**`()` Returns the host name of the Internet Protocol (IP) interface on which the request was received.
`int`	**`getLocalPort`**`()` Returns the Internet Protocol (IP) port number of the interface on which the request was received.
`java.lang.String`	**`getParameter`**`(java.lang.String name)` Returns the value of a request parameter as a `String`, or `null` if the parameter does not exist.
`java.util.Map<java.lang.String,java.lang.String[]>`	**`getParameterMap`**`()` Returns a java.util.Map of the parameters of this request.
`java.util.Enumeration<java.lang.String>`	**`getParameterNames`**`()` Returns an `Enumeration` of `String` objects containing the names of the parameters contained in this request.
`java.lang.String[]`	**`getParameterValues`**`(java.lang.String name)` Returns an array of `String` objects containing all of the values the given request parameter has, or `null` if the parameter does not exist.
`java.lang.String`	**`getProtocol`**`()` Returns the name and version of the protocol the request uses in the form *protocol/majorVersion.minorVersion*, for example, HTTP/1.1.
`java.io.BufferedReader`	**`getReader`**`()` Retrieves the body of the request as character data using a `BufferedReader`.

`java.lang.String`	**`getRealPath`**`(java.lang.String path)` **Deprecated.** *As of Version 2.1 of the Java Servlet API, use `ServletContext#getRealPath` instead.*
`java.lang.String`	**`getRemoteAddr`**`()` Returns the Internet Protocol (IP) address of the client or last proxy that sent the request.
`java.lang.String`	**`getRemoteHost`**`()` Returns the fully qualified name of the client or the last proxy that sent the request.
`int`	**`getRemotePort`**`()` Returns the Internet Protocol (IP) source port of the client or last proxy that sent the request.
`RequestDispatcher`	**`getRequestDispatcher`**`(java.lang.String path)` Returns a `RequestDispatcher` object that acts as a wrapper for the resource located at the given path.
`java.lang.String`	**`getScheme`**`()` Returns the name of the scheme used to make this request, for example, `http`, `https`, or `ftp`.
`java.lang.String`	**`getServerName`**`()` Returns the host name of the server to which the request was sent.
`int`	**`getServerPort`**`()` Returns the port number to which the request was sent.
`ServletContext`	**`getServletContext`**`()` Gets the servlet context to which this ServletRequest was last dispatched.
`boolean`	**`isAsyncStarted`**`()` Checks if this request has been put into asynchronous mode.
`boolean`	**`isAsyncSupported`**`()` Checks if this request supports asynchronous operation.
`boolean`	**`isSecure`**`()` Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.
`void`	**`removeAttribute`**`(java.lang.String name)` Removes an attribute from this request.
`void`	**`setAttribute`**`(java.lang.String name, java.lang.Object o)` Stores an attribute in this request.
`void`	**`setCharacterEncoding`**`(java.lang.String env)` Overrides the name of the character encoding used in the body of this request.
`AsyncContext`	**`startAsync`**`()` Puts this request into asynchronous mode, and initializes its `AsyncContext` with the original (unwrapped) ServletRequest and ServletResponse

	objects.
AsyncContext	**startAsync**(ServletRequest servletRequest, ServletResponse servletResponse) Puts this request into asynchronous mode, and initializes its `AsyncContext` with the given request and response objects.

Method Detail

getAttribute

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

Returns the value of the named attribute as an `Object`, or `null` if no attribute of the given name exists.

Attributes can be set two ways. The servlet container may set attributes to make available custom information about a request. For example, for requests made using HTTPS, the attribute `javax.servlet.request.X509Certificate` can be used to retrieve information on the certificate of the client. Attributes can also be set programatically using `ServletRequest#setAttribute`. This allows information to be embedded into a request before a `RequestDispatcher` call.

Attribute names should follow the same conventions as package names. This specification reserves names matching `java.*`, `javax.*`, and `sun.*`.

Parameters:
`name` - a `String` specifying the name of the attribute

Returns:
an `Object` containing the value of the attribute, or `null` if the attribute does not exist

---

getAttributeNames

`java.util.Enumeration<java.lang.String> **getAttributeNames**()`

Returns an `Enumeration` containing the names of the attributes available to this request. This method returns an empty `Enumeration` if the request has no attributes available to it.

Returns:
an `Enumeration` of strings containing the names of the request's attributes

---

getCharacterEncoding

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

Returns the name of the character encoding used in the body of this request. This method returns `null` if the request does not specify a character encoding

Returns:
a `String` containing the name of the character encoding, or `null` if the request does not specify a character encoding

---

setCharacterEncoding

```
void setCharacterEncoding(java.lang.String env)
                          throws java.io.UnsupportedEncodingException
```

Overrides the name of the character encoding used in the body of this request. This method must be called prior to reading request parameters or reading input using getReader(). Otherwise, it has no effect.

Parameters:
`env` - `String` containing the name of the character encoding.

Throws:
`java.io.UnsupportedEncodingException` - if this ServletRequest is still in a state where a character encoding may be set, but the specified encoding is invalid

---

getContentLength

```
int getContentLength()
```

Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known. For HTTP servlets, same as the value of the CGI variable CONTENT_LENGTH.

Returns:
an integer containing the length of the request body or -1 if the length is not known

---

getContentType

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

Returns the MIME type of the body of the request, or `null` if the type is not known. For HTTP servlets, same as the value of the CGI variable CONTENT_TYPE.

Returns:
a `String` containing the name of the MIME type of the request, or null if the type is not known

---

getInputStream

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

Retrieves the body of the request as binary data using a `ServletInputStream`. Either this method or `getReader()` may be called to read the body, not both.

Returns:
a `ServletInputStream` object containing the body of the request

Throws:
`IllegalStateException` - if the `getReader()` method has already been called for this request
`java.io.IOException` - if an input or output exception occurred

---

getParameter

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

Returns the value of a request parameter as a `String`, or `null` if the parameter does not exist. Request parameters are extra information sent with the request. For HTTP servlets, parameters are contained in the query string or posted form data.

You should only use this method when you are sure the parameter has only one value. If the parameter might have more than one value, use `getParameterValues(java.lang.String)`.

If you use this method with a multivalued parameter, the value returned is equal to the first value in the array returned by `getParameterValues`.

If the parameter data was sent in the request body, such as occurs with an HTTP POST request, then reading the body directly via `getInputStream()` or `getReader()` can interfere with the execution of this method.

Parameters:
`name` - a `String` specifying the name of the parameter

Returns:
a `String` representing the single value of the parameter

See Also:
`getParameterValues(java.lang.String)`

---

getParameterNames

```
java.util.Enumeration<java.lang.String> getParameterNames()
```

Returns an `Enumeration` of `String` objects containing the names of the parameters contained in this request. If the request has no parameters, the method returns an empty `Enumeration`.

Returns:
an `Enumeration` of `String` objects, each `String` containing the name of a request parameter; or an empty `Enumeration` if the request has no parameters

---

getParameterValues

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

Returns an array of `String` objects containing all of the values the given request parameter has, or `null` if the parameter does not exist.

If the parameter has a single value, the array has a length of 1.

Parameters:
`name` - a `String` containing the name of the parameter whose value is requested

Returns:
an array of `String` objects containing the parameter's values

See Also:
`getParameter(java.lang.String)`

---

getParameterMap

```
java.util.Map<java.lang.String,java.lang.String[]> getParameterMap()
```

Returns a java.util.Map of the parameters of this request.

Request parameters are extra information sent with the request. For HTTP servlets, parameters are contained in the query string or posted form data.

Returns:

an immutable java.util.Map containing parameter names as keys and parameter values as map values. The keys in the parameter map are of type String. The values in the parameter map are of type String array.

---

getProtocol

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

Returns the name and version of the protocol the request uses in the form *protocol/majorVersion.minorVersion*, for example, HTTP/1.1. For HTTP servlets, the value returned is the same as the value of the CGI variable `SERVER_PROTOCOL`.

Returns:

a `String` containing the protocol name and version number

---

getScheme

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

Returns the name of the scheme used to make this request, for example, `http`, `https`, or `ftp`. Different schemes have different rules for constructing URLs, as noted in RFC 1738.

Returns:

a `String` containing the name of the scheme used to make this request

---

getServerName

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

Returns the host name of the server to which the request was sent. It is the value of the part before ":" in the `Host` header value, if any, or the resolved server name, or the server IP address.

Returns:

a `String` containing the name of the server

---

getServerPort

```
int getServerPort()
```

Returns the port number to which the request was sent. It is the value of the part after ":" in the `Host` header value, if any, or the server port where the client connection was accepted on.

Returns:

an integer specifying the port number

---

getReader

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

Retrieves the body of the request as character data using a `BufferedReader`. The reader translates the character data according to the character encoding used on the body. Either this method or `getInputStream()` may be called to read the body, not both.

Returns:

a `BufferedReader` containing the body of the request

Throws:

`java.io.UnsupportedEncodingException` - if the character set encoding used is not supported and the text cannot be decoded

`IllegalStateException` - if `getInputStream()` method has been called on this request

`java.io.IOException` - if an input or output exception occurred

See Also:

`getInputStream()`

---

getRemoteAddr

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

Returns the Internet Protocol (IP) address of the client or last proxy that sent the request. For HTTP servlets, same as the value of the CGI variable `REMOTE_ADDR`.

Returns:

a `String` containing the IP address of the client that sent the request

---

getRemoteHost

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

Returns the fully qualified name of the client or the last proxy that sent the request. If the engine cannot or chooses not to resolve the hostname (to improve performance), this method returns the dotted-string form of the IP address. For HTTP servlets, same as the value of the CGI variable `REMOTE_HOST`.

Returns:

a `String` containing the fully qualified name of the client

---

setAttribute

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

Stores an attribute in this request. Attributes are reset between requests. This method is most often used in conjunction with `RequestDispatcher`.

Attribute names should follow the same conventions as package names. Names beginning with `java.*`, `javax.*`, and `com.sun.*`, are reserved for use by Sun Microsystems.
If the object passed in is null, the effect is the same as calling `removeAttribute(java.lang.String)`.
It is warned that when the request is dispatched from the servlet resides in a different web application by `RequestDispatcher`, the object set by this method may not be correctly retrieved in the caller servlet.

Parameters:

`name` - a `String` specifying the name of the attribute

`o` - the `Object` to be stored

---

removeAttribute

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

Removes an attribute from this request. This method is not generally needed as attributes only persist as long as the request is being handled.

Attribute names should follow the same conventions as package names. Names beginning with `java.*`, `javax.*`, and `com.sun.*`, are reserved for use by Sun Microsystems.

Parameters:

`name` - a `String` specifying the name of the attribute to remove

---

getLocale

```
java.util.Locale getLocale()
```

Returns the preferred `Locale` that the client will accept content in, based on the Accept-Language header. If the client request doesn't provide an Accept-Language header, this method returns the default locale for the server.

Returns:

the preferred `Locale` for the client

---

getLocales

```
java.util.Enumeration<java.util.Locale> getLocales()
```

Returns an `Enumeration` of `Locale` objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header. If the client request doesn't provide an Accept-Language header, this method returns an `Enumeration` containing one `Locale`, the default locale for the server.

Returns:

an `Enumeration` of preferred `Locale` objects for the client

---

isSecure

```
boolean isSecure()
```

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

Returns:

a boolean indicating if the request was made using a secure channel

---

getRequestDispatcher

```
RequestDispatcher getRequestDispatcher(java.lang.String path)
```

Returns a `RequestDispatcher` object that acts as a wrapper for the resource located at the given path. A `RequestDispatcher` object can be used to forward a request to the resource or to include the resource in a response. The resource can be dynamic or static.

The pathname specified may be relative, although it cannot extend outside the current servlet context. If the path begins with a "/" it is interpreted as relative to the current context root. This method returns `null` if the servlet container cannot return a `RequestDispatcher`.

The difference between this method and `ServletContext#getRequestDispatcher` is that this method can take a relative path.

Parameters:
`path` - a `String` specifying the pathname to the resource. If it is relative, it must be relative against the current servlet.

Returns:
a `RequestDispatcher` object that acts as a wrapper for the resource at the specified path, or `null` if the servlet container cannot return a `RequestDispatcher`

See Also:
`RequestDispatcher`, `ServletContext#getRequestDispatcher`

---

getRealPath

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

Deprecated. *As of Version 2.1 of the Java Servlet API, use `ServletContext#getRealPath` instead.*

---

getRemotePort

```
int getRemotePort()
```

Returns the Internet Protocol (IP) source port of the client or last proxy that sent the request.

Returns:
an integer specifying the port number

Since:
Servlet 2.4

---

getLocalName

```
java.lang.String getLocalName()
```

Returns the host name of the Internet Protocol (IP) interface on which the request was received.

Returns:
a `String` containing the host name of the IP on which the request was received.

Since:
Servlet 2.4

---

getLocalAddr

```
java.lang.String getLocalAddr()
```

Returns the Internet Protocol (IP) address of the interface on which the request was received.

Returns:
a `String` containing the IP address on which the request was received.

Since:
Servlet 2.4

---

getLocalPort

```
int getLocalPort()
```

Returns the Internet Protocol (IP) port number of the interface on which the request was received.

Returns:
an integer specifying the port number
Since:
Servlet 2.4

---

getServletContext

```
ServletContext getServletContext()
```

Gets the servlet context to which this ServletRequest was last dispatched.

Returns:
the servlet context to which this ServletRequest was last dispatched
Since:
Servlet 3.0

---

startAsync

```
AsyncContext startAsync()
                        throws java.lang.IllegalStateException
```

Puts this request into asynchronous mode, and initializes its `AsyncContext` with the original (unwrapped) ServletRequest and ServletResponse objects.

Calling this method will cause committal of the associated response to be delayed until `AsyncContext#complete` is called on the returned `AsyncContext`, or the asynchronous operation has timed out.

Calling `AsyncContext#hasOriginalRequestAndResponse()` on the returned AsyncContext will return `true`. Any filters invoked in the *outbound* direction after this request was put into asynchronous mode may use this as an indication that any request and/or response wrappers that they added during their *inbound* invocation need not stay around for the duration of the asynchronous operation, and therefore any of their associated resources may be released.

This method clears the list of `AsyncListener` instances (if any) that were registered with the AsyncContext returned by the previous call to one of the startAsync methods, after calling each AsyncListener at its `onStartAsync` method.

Subsequent invocations of this method, or its overloaded variant, will return the same AsyncContext instance, reinitialized as appropriate.

Returns:
the (re)initialized AsyncContext
Throws:

`IllegalStateException` - if this request is within the scope of a filter or servlet that does not support asynchronous operations (that is, `isAsyncSupported()` returns false), or if this method is called again without any asynchronous dispatch (resulting from one of the `AsyncContext#dispatch` methods), is called outside the scope of any such dispatch, or is called again within the scope of the same dispatch, or if the response has already been closed

Since:

Servlet 3.0

---

startAsync

```
AsyncContext startAsync(ServletRequest servletRequest,
                        ServletResponse servletResponse)
                        throws java.lang.IllegalStateException
```

Puts this request into asynchronous mode, and initializes its `AsyncContext` with the given request and response objects.

The ServletRequest and ServletResponse arguments must be the same instances, or instances of `ServletRequestWrapper` and `ServletResponseWrapper` that wrap them, that were passed to the `service` method of the Servlet or the `doFilter` method of the Filter, respectively, in whose scope this method is being called.

Calling this method will cause committal of the associated response to be delayed until `AsyncContext#complete` is called on the returned `AsyncContext`, or the asynchronous operation has timed out.

Calling `AsyncContext#hasOriginalRequestAndResponse()` on the returned AsyncContext will return `false`, unless the passed in ServletRequest and ServletResponse arguments are the original ones or do not carry any application-provided wrappers. Any filters invoked in the *outbound* direction after this request was put into asynchronous mode may use this as an indication that some of the request and/or response wrappers that they added during their *inbound* invocation may need to stay in place for the duration of the asynchronous operation, and their associated resources may not be released. A ServletRequestWrapper applied during the *inbound* invocation of a filter may be released by the *outbound* invocation of the filter only if the given `servletRequest`, which is used to initialize the AsyncContext and will be returned by a call to `AsyncContext#getRequest()`, does not contain said ServletRequestWrapper. The same holds true for ServletResponseWrapper instances.

This method clears the list of `AsyncListener` instances (if any) that were registered with the AsyncContext returned by the previous call to one of the startAsync methods, after calling each AsyncListener at its `onStartAsync` method.

Subsequent invocations of this method, or its zero-argument variant, will return the same AsyncContext instance, reinitialized as appropriate. If a call to this method is followed by a call to its zero-argument variant, the specified (and possibly wrapped) request and response objects will remain *locked in* on the returned AsyncContext.

Parameters:

`servletRequest` - the ServletRequest used to initialize the AsyncContext

`servletResponse` - the ServletResponse used to initialize the AsyncContext

Returns:

the (re)initialized AsyncContext

Throws:

`IllegalStateException` - if this request is within the scope of a filter or servlet that does not support asynchronous operations (that is, `isAsyncSupported()` returns false), or if this method is called again without any asynchronous dispatch (resulting from one of the `AsyncContext#dispatch` methods), is called outside the scope of any such dispatch, or is called again within the scope of the same dispatch, or if the response has already been closed

Since:
Servlet 3.0

---

isAsyncStarted

```
boolean isAsyncStarted()
```

Checks if this request has been put into asynchronous mode.

A ServletRequest is put into asynchronous mode by calling `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on it.

This method returns `false` if this request was put into asynchronous mode, but has since been dispatched using one of the `AsyncContext#dispatch` methods or released from asynchronous mode via a call to `AsyncContext#complete`.

Returns:
true if this request has been put into asynchronous mode, false otherwise

Since:
Servlet 3.0

---

isAsyncSupported

```
boolean isAsyncSupported()
```

Checks if this request supports asynchronous operation.

Asynchronous operation is disabled for this request if this request is within the scope of a filter or servlet that has not been annotated or flagged in the deployment descriptor as being able to support asynchronous handling.

Returns:
true if this request supports asynchronous operation, false otherwise

Since:
Servlet 3.0

---

getAsyncContext

```
AsyncContext getAsyncContext()
```

Gets the AsyncContext that was created or reinitialized by the most recent invocation of `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on this request.

Returns:
the AsyncContext that was created or reinitialized by the most recent invocation of `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on this request

Throws:
`IllegalStateException` - if this request has not been put into asynchronous mode, i.e., if neither `startAsync()` nor `startAsync(ServletRequest,ServletResponse)` has been called

Since:

Servlet 3.0

---

getDispatcherType

`DispatcherType` **`getDispatcherType`**`()`

Gets the dispatcher type of this request.

The dispatcher type of a request is used by the container to select the filters that need to be applied to the request: Only filters with matching dispatcher type and url patterns will be applied.

Allowing a filter that has been configured for multiple dispatcher types to query a request for its dispatcher type allows the filter to process the request differently depending on its dispatcher type.

The initial dispatcher type of a request is defined as `DispatcherType.REQUEST`. The dispatcher type of a request dispatched via `RequestDispatcher#forward(ServletRequest, ServletResponse)` or `RequestDispatcher#include(ServletRequest, ServletResponse)` is given as `DispatcherType.FORWARD` or `DispatcherType.INCLUDE`, respectively, while the dispatcher type of an asynchronous request dispatched via one of the `AsyncContext#dispatch` methods is given as `DispatcherType.ASYNC`. Finally, the dispatcher type of a request dispatched to an error page by the container's error handling mechanism is given as `DispatcherType.ERROR`.

Returns:
the dispatcher type of this request
Since:
Servlet 3.0
See Also:
`DispatcherType`

Submit a bug or feature

javax.servlet

Class ServletRequestAttributeEvent

```
java.lang.Object
  └java.util.EventObject
      └javax.servlet.ServletRequestEvent
          └javax.servlet.ServletRequestAttributeEvent
```

All Implemented Interfaces:

java.io.Serializable

```
public class ServletRequestAttributeEvent
extends ServletRequestEvent
```

This is the event class for notifications of changes to the attributes of the servlet request in an application.

Since:

Servlet 2.4

See Also:

`ServletRequestAttributeListener`, Serialized Form

Field Summary

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

Constructor Summary

`ServletRequestAttributeEvent``(ServletContext sc, ServletRequest request, java.lang.String name, java.lang.Object value)` Construct a ServletRequestAttributeEvent giving the servlet context of this web application, the ServletRequest whose attributes are changing and the name and value of the attribute.

Method Summary

`java.lang.String`	**`getName`**`()` Return the name of the attribute that changed on the ServletRequest.
`java.lang.Object`	**`getValue`**`()` Returns the value of the attribute that has been added, removed or replaced.

Methods inherited from class javax.servlet.ServletRequestEvent
`getServletContext, getServletRequest`

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

ServletRequestAttributeEvent

```
public ServletRequestAttributeEvent(ServletContext sc,
                                    ServletRequest request,
                                    java.lang.String name,
                                    java.lang.Object value)
```

Construct a ServletRequestAttributeEvent giving the servlet context of this web application, the ServletRequest whose attributes are changing and the name and value of the attribute.

Parameters:

`sc` - the ServletContext that is sending the event.
`request` - the ServletRequest that is sending the event.
`name` - the name of the request attribute.
`value` - the value of the request attribute.

Method Detail

getName

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

Return the name of the attribute that changed on the ServletRequest.

Returns:

the name of the changed request attribute

---

getValue

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

Returns the value of the attribute that has been added, removed or replaced. If the attribute was added, this is the value of the attribute. If the attribute was removed, this is the value of the removed attribute. If the attribute was replaced, this is the old value of the attribute.

Returns:

the value of the changed request attribute

---

javax.servlet

Interface ServletRequestAttributeListener

All Superinterfaces:

java.util.EventListener

```
public interface ServletRequestAttributeListener
extends java.util.EventListener
```

Interface for receiving notification events about ServletRequest attribute changes.

Notifications will be generated while the request is within the scope of the web application. A ServletRequest is defined as coming into scope of a web application when it is about to enter the first servlet or filter of the web application, and as going out of scope when it exits the last servlet or the first filter in the chain.

In order to receive these notification events, the implementation class must be either declared in the deployment descriptor of the web application, annotated with `WebListener`, or registered via one of the addListener methods defined on `ServletContext`.

The order in which implementations of this interface are invoked is unspecified.

Since:

Servlet 2.4

Method Summary

`void`	`attributeAdded(ServletRequestAttributeEvent srae)` Receives notification that an attribute has been added to the ServletRequest.
`void`	`attributeRemoved(ServletRequestAttributeEvent srae)` Receives notification that an attribute has been removed from the ServletRequest.
`void`	`attributeReplaced(ServletRequestAttributeEvent srae)` Receives notification that an attribute has been replaced on the ServletRequest.

Method Detail

attributeAdded

```
void attributeAdded(ServletRequestAttributeEvent srae)
```

Receives notification that an attribute has been added to the ServletRequest.

Parameters:

`srae` - the ServletRequestAttributeEvent containing the ServletRequest and the name and value of the attribute that was added

---

attributeRemoved

`void` **`attributeRemoved`**`(ServletRequestAttributeEvent srae)`

Receives notification that an attribute has been removed from the ServletRequest.

Parameters:

`srae` - the ServletRequestAttributeEvent containing the ServletRequest and the name and value of the attribute that was removed

---

attributeReplaced

`void` **`attributeReplaced`**`(ServletRequestAttributeEvent srae)`

Receives notification that an attribute has been replaced on the ServletRequest.

Parameters:

`srae` - the ServletRequestAttributeEvent containing the ServletRequest and the name and (old) value of the attribute that was replaced

---

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS **FRAMES NO FRAMES All Classes**

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Submit a bug or feature

javax.servlet

Class ServletRequestEvent

```
java.lang.Object
  └java.util.EventObject
      └javax.servlet.ServletRequestEvent
```

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

ServletRequestAttributeEvent

```
public class ServletRequestEvent
extends java.util.EventObject
```

Events of this kind indicate lifecycle events for a ServletRequest. The source of the event is the ServletContext of this web application.

Since:

Servlet 2.4

See Also:

`ServletRequestListener`, Serialized Form

Field Summary

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

Constructor Summary

`ServletRequestEvent(ServletContext sc, ServletRequest request)`
Construct a ServletRequestEvent for the given ServletContext and ServletRequest.

Method Summary

`ServletContext`	`getServletContext()` Returns the ServletContext of this web application.
`ServletRequest`	`getServletRequest()` Returns the ServletRequest that is changing.

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

ServletRequestEvent

```
public ServletRequestEvent(ServletContext sc,
                           ServletRequest request)
```

Construct a ServletRequestEvent for the given ServletContext and ServletRequest.

Parameters:

`sc` - the ServletContext of the web application.
`request` - the ServletRequest that is sending the event.

Method Detail

getServletRequest

```
public ServletRequest getServletRequest()
```

Returns the ServletRequest that is changing.

---

getServletContext

```
public ServletContext getServletContext()
```

Returns the ServletContext of this web application.

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface ServletRequestListener

All Superinterfaces:

java.util.EventListener

```
public interface ServletRequestListener
extends java.util.EventListener
```

Interface for receiving notification events about requests coming into and going out of scope of a web application.

A ServletRequest is defined as coming into scope of a web application when it is about to enter the first servlet or filter of the web application, and as going out of scope as it exits the last servlet or the first filter in the chain.

In order to receive these notification events, the implementation class must be either declared in the deployment descriptor of the web application, annotated with `WebListener`, or registered via one of the addListener methods defined on `ServletContext`.

Implementations of this interface are invoked at their `requestInitialized(javax.servlet.ServletRequestEvent)` method in the order in which they have been declared, and at their `requestDestroyed(javax.servlet.ServletRequestEvent)` method in reverse order.

Since:

Servlet 2.4

Method Summary

`void`	`requestDestroyed(ServletRequestEvent sre)` Receives notification that a ServletRequest is about to go out of scope of the web application.
`void`	`requestInitialized(ServletRequestEvent sre)` Receives notification that a ServletRequest is about to come into scope of the web application.

Method Detail

requestDestroyed

```
void requestDestroyed(ServletRequestEvent sre)
```

Receives notification that a ServletRequest is about to go out of scope of the web application.

Parameters:

`sre` - the ServletRequestEvent containing the ServletRequest and the ServletContext representing the web application

---

requestInitialized

```
void requestInitialized(ServletRequestEvent sre)
```

Receives notification that a ServletRequest is about to come into scope of the web application.

Parameters:

`sre` - the ServletRequestEvent containing the ServletRequest and the ServletContext representing the web application

---

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class ServletRequestWrapper

```
java.lang.Object
  └ javax.servlet.ServletRequestWrapper
```

All Implemented Interfaces:

ServletRequest

Direct Known Subclasses:

HttpServletRequestWrapper

```
public class ServletRequestWrapper
extends java.lang.Object
implements ServletRequest
```

Provides a convenient implementation of the ServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet. This class implements the Wrapper or Decorator pattern. Methods default to calling through to the wrapped request object.

Since:

Servlet 2.3

See Also:

`ServletRequest`

Constructor Summary

`ServletRequestWrapper``(ServletRequest request)` Creates a ServletRequest adaptor wrapping the given request object.

Method Summary

`AsyncContext`	**`getAsyncContext`**`()` Gets the AsyncContext that was created or reinitialized by the most recent invocation of `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on the wrapped request.
`java.lang.Object`	**`getAttribute`**`(java.lang.String name)` The default behavior of this method is to call getAttribute(String name) on the wrapped request object.
`java.util.Enumeration<java.lang.String>`	**`getAttributeNames`**`()` The default behavior of this method is to return getAttributeNames() on the wrapped request object.

`java.lang.String`	**`getCharacterEncoding`**`()` The default behavior of this method is to return getCharacterEncoding() on the wrapped request object.
`int`	**`getContentLength`**`()` The default behavior of this method is to return getContentLength() on the wrapped request object.
`java.lang.String`	**`getContentType`**`()` The default behavior of this method is to return getContentType() on the wrapped request object.
`DispatcherType`	**`getDispatcherType`**`()` Gets the dispatcher type of the wrapped request.
`ServletInputStream`	**`getInputStream`**`()` The default behavior of this method is to return getInputStream() on the wrapped request object.
`java.lang.String`	**`getLocalAddr`**`()` The default behavior of this method is to return getLocalAddr() on the wrapped request object.
`java.util.Locale`	**`getLocale`**`()` The default behavior of this method is to return getLocale() on the wrapped request object.
`java.util.Enumeration<java.util.Locale>`	**`getLocales`**`()` The default behavior of this method is to return getLocales() on the wrapped request object.
`java.lang.String`	**`getLocalName`**`()` The default behavior of this method is to return getLocalName() on the wrapped request object.
`int`	**`getLocalPort`**`()` The default behavior of this method is to return getLocalPort() on the wrapped request object.
`java.lang.String`	**`getParameter`**`(java.lang.String name)` The default behavior of this method is to return getParameter(String name) on the wrapped request object.
`java.util.Map<java.lang.String,java.lang.String[]>`	**`getParameterMap`**`()` The default behavior of this method is to return getParameterMap() on the wrapped request object.
`java.util.Enumeration<java.lang.String>`	**`getParameterNames`**`()` The default behavior of this method is to return getParameterNames() on the wrapped request object.
`java.lang.String[]`	**`getParameterValues`**`(java.lang.String name)` The default behavior of this method is to

	return getParameterValues(String name) on the wrapped request object.
`java.lang.String`	**`getProtocol`**`()` The default behavior of this method is to return getProtocol() on the wrapped request object.
`java.io.BufferedReader`	**`getReader`**`()` The default behavior of this method is to return getReader() on the wrapped request object.
`java.lang.String`	**`getRealPath`**`(java.lang.String path)` **Deprecated.** *As of Version 2.1 of the Java Servlet API, use `ServletContext#getRealPath` instead*
`java.lang.String`	**`getRemoteAddr`**`()` The default behavior of this method is to return getRemoteAddr() on the wrapped request object.
`java.lang.String`	**`getRemoteHost`**`()` The default behavior of this method is to return getRemoteHost() on the wrapped request object.
`int`	**`getRemotePort`**`()` The default behavior of this method is to return getRemotePort() on the wrapped request object.
`ServletRequest`	**`getRequest`**`()` Return the wrapped request object.
`RequestDispatcher`	**`getRequestDispatcher`**`(java.lang.String path)` The default behavior of this method is to return getRequestDispatcher(String path) on the wrapped request object.
`java.lang.String`	**`getScheme`**`()` The default behavior of this method is to return getScheme() on the wrapped request object.
`java.lang.String`	**`getServerName`**`()` The default behavior of this method is to return getServerName() on the wrapped request object.
`int`	**`getServerPort`**`()` The default behavior of this method is to return getServerPort() on the wrapped request object.
`ServletContext`	**`getServletContext`**`()` Gets the servlet context to which the wrapped servlet request was last dispatched.
`boolean`	**`isAsyncStarted`**`()` Checks if the wrapped request has been put into asynchronous mode.
`boolean`	**`isAsyncSupported`**`()` Checks if the wrapped request supports asynchronous operation.

boolean	**isSecure**() The default behavior of this method is to return isSecure() on the wrapped request object.
boolean	**isWrapperFor**(java.lang.Class wrappedType) Checks (recursively) if this ServletRequestWrapper wraps a `ServletRequest` of the given class type.
boolean	**isWrapperFor**(ServletRequest wrapped) Checks (recursively) if this ServletRequestWrapper wraps the given `ServletRequest` instance.
void	**removeAttribute**(java.lang.String name) The default behavior of this method is to call removeAttribute(String name) on the wrapped request object.
void	**setAttribute**(java.lang.String name, java.lang.Object o) The default behavior of this method is to return setAttribute(String name, Object o) on the wrapped request object.
void	**setCharacterEncoding**(java.lang.String enc) The default behavior of this method is to set the character encoding on the wrapped request object.
void	**setRequest**(ServletRequest request) Sets the request object being wrapped.
AsyncContext	**startAsync**() The default behavior of this method is to invoke `ServletRequest#startAsync` on the wrapped request object.
AsyncContext	**startAsync**(ServletRequest servletRequest, ServletResponse servletResponse) The default behavior of this method is to invoke `ServletRequest#startAsync(ServletRequest, ServletResponse)` on the wrapped request object.

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

Constructor Detail

ServletRequestWrapper

```
public ServletRequestWrapper(ServletRequest request)
```

Creates a ServletRequest adaptor wrapping the given request object.

Throws:

`java.lang.IllegalArgumentException` - if the request is null

Method Detail

getRequest

`public ServletRequest getRequest()`

Return the wrapped request object.

---

setRequest

`public void setRequest(ServletRequest request)`

Sets the request object being wrapped.

Throws:

`java.lang.IllegalArgumentException` - if the request is null.

---

getAttribute

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

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

Specified by:

`getAttribute` in interface `ServletRequest`

Parameters:

`name` - a `String` specifying the name of the attribute

Returns:

an `Object` containing the value of the attribute, or `null` if the attribute does not exist

---

getAttributeNames

`public java.util.Enumeration<java.lang.String> getAttributeNames()`

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

Specified by:

`getAttributeNames` in interface `ServletRequest`

Returns:

an `Enumeration` of strings containing the names of the request's attributes

---

getCharacterEncoding

`public java.lang.String getCharacterEncoding()`

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

Specified by:

`getCharacterEncoding` in interface `ServletRequest`

Returns:

a `String` containing the name of the character encoding, or `null` if the request does not specify a character encoding

setCharacterEncoding

```
public void setCharacterEncoding(java.lang.String enc)
                          throws java.io.UnsupportedEncodingException
```

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

Specified by:

`setCharacterEncoding` in interface `ServletRequest`

Parameters:

`enc` - `String` containing the name of the character encoding.

Throws:

`java.io.UnsupportedEncodingException` - if this ServletRequest is still in a state where a character encoding may be set, but the specified encoding is invalid

getContentLength

```
public int getContentLength()
```

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

Specified by:

`getContentLength` in interface `ServletRequest`

Returns:

an integer containing the length of the request body or -1 if the length is not known

getContentType

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

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

Specified by:

`getContentType` in interface `ServletRequest`

Returns:

a `String` containing the name of the MIME type of the request, or null if the type is not known

getInputStream

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

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

Specified by:

`getInputStream` in interface `ServletRequest`

Returns:

a `ServletInputStream` object containing the body of the request

Throws:

`java.io.IOException` - if an input or output exception occurred

getParameter

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

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

Specified by:
`getParameter` in interface `ServletRequest`

Parameters:
`name` - a `String` specifying the name of the parameter

Returns:
a `String` representing the single value of the parameter

See Also:
`ServletRequest.getParameterValues(java.lang.String)`

---

getParameterMap

```
public java.util.Map<java.lang.String,java.lang.String[]> getParameterMap()
```

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

Specified by:
`getParameterMap` in interface `ServletRequest`

Returns:
an immutable java.util.Map containing parameter names as keys and parameter values as map values. The keys in the parameter map are of type String. The values in the parameter map are of type String array.

---

getParameterNames

```
public java.util.Enumeration<java.lang.String> getParameterNames()
```

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

Specified by:
`getParameterNames` in interface `ServletRequest`

Returns:
an `Enumeration` of `String` objects, each `String` containing the name of a request parameter; or an empty `Enumeration` if the request has no parameters

---

getParameterValues

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

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

Specified by:
`getParameterValues` in interface `ServletRequest`

Parameters:
`name` - a `String` containing the name of the parameter whose value is requested

Returns:

an array of `String` objects containing the parameter's values

See Also:

`ServletRequest.getParameter(java.lang.String)`

---

getProtocol

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

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

Specified by:

`getProtocol` in interface `ServletRequest`

Returns:

a `String` containing the protocol name and version number

---

getScheme

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

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

Specified by:

`getScheme` in interface `ServletRequest`

Returns:

a `String` containing the name of the scheme used to make this request

---

getServerName

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

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

Specified by:

`getServerName` in interface `ServletRequest`

Returns:

a `String` containing the name of the server

---

getServerPort

```
public int getServerPort()
```

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

Specified by:

`getServerPort` in interface `ServletRequest`

Returns:

an integer specifying the port number

---

getReader

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

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

Specified by:

`getReader` in interface `ServletRequest`

Returns:

a `BufferedReader` containing the body of the request

Throws:

`java.io.UnsupportedEncodingException` - if the character set encoding used is not supported and the text cannot be decoded

`java.io.IOException` - if an input or output exception occurred

See Also:

`ServletRequest.getInputStream()`

---

getRemoteAddr

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

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

Specified by:

`getRemoteAddr` in interface `ServletRequest`

Returns:

a `String` containing the IP address of the client that sent the request

---

getRemoteHost

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

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

Specified by:

`getRemoteHost` in interface `ServletRequest`

Returns:

a `String` containing the fully qualified name of the client

---

setAttribute

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

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

Specified by:

`setAttribute` in interface `ServletRequest`

Parameters:

`name` - a `String` specifying the name of the attribute

`o` - the `Object` to be stored

---

removeAttribute

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

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

Specified by:
`removeAttribute` in interface `ServletRequest`
Parameters:
`name` - a `String` specifying the name of the attribute to remove

---

getLocale

```
public java.util.Locale getLocale()
```

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

Specified by:
`getLocale` in interface `ServletRequest`
Returns:
the preferred `Locale` for the client

---

getLocales

```
public java.util.Enumeration<java.util.Locale> getLocales()
```

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

Specified by:
`getLocales` in interface `ServletRequest`
Returns:
an `Enumeration` of preferred `Locale` objects for the client

---

isSecure

```
public boolean isSecure()
```

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

Specified by:
`isSecure` in interface `ServletRequest`
Returns:
a boolean indicating if the request was made using a secure channel

---

getRequestDispatcher

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

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

Specified by:
`getRequestDispatcher` in interface `ServletRequest`
Parameters:
`path` - a `String` specifying the pathname to the resource. If it is relative, it must be relative against

the current servlet.

Returns:

a `RequestDispatcher` object that acts as a wrapper for the resource at the specified path, or `null` if the servlet container cannot return a `RequestDispatcher`

See Also:

`RequestDispatcher`, `ServletContext#getRequestDispatcher`

---

getRealPath

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

Deprecated. *As of Version 2.1 of the Java Servlet API, use `ServletContext#getRealPath` instead*

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

Specified by:

`getRealPath` in interface `ServletRequest`

---

getRemotePort

`public int getRemotePort()`

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

Specified by:

`getRemotePort` in interface `ServletRequest`

Returns:

an integer specifying the port number

Since:

Servlet 2.4

---

getLocalName

`public java.lang.String getLocalName()`

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

Specified by:

`getLocalName` in interface `ServletRequest`

Returns:

a `String` containing the host name of the IP on which the request was received.

Since:

Servlet 2.4

---

getLocalAddr

`public java.lang.String getLocalAddr()`

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

Specified by:

`getLocalAddr` in interface `ServletRequest`

Returns:
: a `String` containing the IP address on which the request was received.

Since:
: Servlet 2.4

---

getLocalPort

`public int getLocalPort()`

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

Specified by:
: `getLocalPort` in interface `ServletRequest`

Returns:
: an integer specifying the port number

Since:
: Servlet 2.4

---

getServletContext

`public ServletContext getServletContext()`

Gets the servlet context to which the wrapped servlet request was last dispatched.

Specified by:
: `getServletContext` in interface `ServletRequest`

Returns:
: the servlet context to which the wrapped servlet request was last dispatched

Since:
: Servlet 3.0

---

startAsync

```
public AsyncContext startAsync()
                        throws java.lang.IllegalStateException
```

The default behavior of this method is to invoke `ServletRequest#startAsync` on the wrapped request object.

Specified by:
: `startAsync` in interface `ServletRequest`

Returns:
: the (re)initialized AsyncContext

Throws:
: `IllegalStateException` - if the request is within the scope of a filter or servlet that does not support asynchronous operations (that is, `isAsyncSupported()` returns false), or if this method is called again without any asynchronous dispatch (resulting from one of the `AsyncContext#dispatch` methods), is called outside the scope of any such dispatch, or is called again within the scope of the same dispatch, or if the response has already been closed

Since:
: Servlet 3.0

See Also:
: `ServletRequest#startAsync`

startAsync

```
public AsyncContext startAsync(ServletRequest servletRequest,
                               ServletResponse servletResponse)
                        throws java.lang.IllegalStateException
```

The default behavior of this method is to invoke `ServletRequest#startAsync(ServletRequest, ServletResponse)` on the wrapped request object.

Specified by:
: `startAsync` in interface `ServletRequest`

Parameters:
: `servletRequest` - the ServletRequest used to initialize the AsyncContext
: `servletResponse` - the ServletResponse used to initialize the AsyncContext

Returns:
: the (re)initialized AsyncContext

Throws:
: `IllegalStateException` - if the request is within the scope of a filter or servlet that does not support asynchronous operations (that is, `isAsyncSupported()` returns false), or if this method is called again without any asynchronous dispatch (resulting from one of the `AsyncContext#dispatch` methods), is called outside the scope of any such dispatch, or is called again within the scope of the same dispatch, or if the response has already been closed

Since:
: Servlet 3.0

See Also:
: `ServletRequest#startAsync(ServletRequest, ServletResponse)`

---

isAsyncStarted

```
public boolean isAsyncStarted()
```

Checks if the wrapped request has been put into asynchronous mode.

Specified by:
: `isAsyncStarted` in interface `ServletRequest`

Returns:
: true if this request has been put into asynchronous mode, false otherwise

Since:
: Servlet 3.0

See Also:
: `ServletRequest#isAsyncStarted`

---

isAsyncSupported

```
public boolean isAsyncSupported()
```

Checks if the wrapped request supports asynchronous operation.

Specified by:
: `isAsyncSupported` in interface `ServletRequest`

Returns:
: true if this request supports asynchronous operation, false otherwise

Since:

Servlet 3.0

See Also:

`ServletRequest#isAsyncSupported`

---

getAsyncContext

```
public AsyncContext getAsyncContext()
```

Gets the AsyncContext that was created or reinitialized by the most recent invocation of `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on the wrapped request.

Specified by:

`getAsyncContext` in interface `ServletRequest`

Returns:

the AsyncContext that was created or reinitialized by the most recent invocation of `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on the wrapped request

Throws:

`IllegalStateException` - if this request has not been put into asynchronous mode, i.e., if neither `startAsync()` nor `startAsync(ServletRequest,ServletResponse)` has been called

Since:

Servlet 3.0

See Also:

`ServletRequest#getAsyncContext`

---

isWrapperFor

```
public boolean isWrapperFor(ServletRequest wrapped)
```

Checks (recursively) if this ServletRequestWrapper wraps the given `ServletRequest` instance.

Parameters:

`wrapped` - the ServletRequest instance to search for

Returns:

true if this ServletRequestWrapper wraps the given ServletRequest instance, false otherwise

Since:

Servlet 3.0

---

isWrapperFor

```
public boolean isWrapperFor(java.lang.Class wrappedType)
```

Checks (recursively) if this ServletRequestWrapper wraps a `ServletRequest` of the given class type.

Parameters:

`wrappedType` - the ServletRequest class type to search for

Returns:

true if this ServletRequestWrapper wraps a ServletRequest of the given class type, false otherwise

Throws:

`IllegalArgumentException` - if the given class does not implement `ServletRequest`

Since:

Servlet 3.0

---

getDispatcherType

```
public DispatcherType getDispatcherType()
```

Gets the dispatcher type of the wrapped request.

Specified by:
`getDispatcherType` in interface `ServletRequest`

Returns:
the dispatcher type of the wrapped request

Since:
Servlet 3.0

See Also:
`ServletRequest#getDispatcherType`

Submit a bug or feature

javax.servlet

Interface ServletResponse

All Known Subinterfaces:
HttpServletResponse

All Known Implementing Classes:
HttpServletResponseWrapper, ServletResponseWrapper

`public interface` **`ServletResponse`**

Defines an object to assist a servlet in sending a response to the client. The servlet container creates a `ServletResponse` object and passes it as an argument to the servlet's `service` method.

To send binary data in a MIME body response, use the `ServletOutputStream` returned by `getOutputStream()`. To send character data, use the `PrintWriter` object returned by `getWriter()`. To mix binary and text data, for example, to create a multipart response, use a `ServletOutputStream` and manage the character sections manually.

The charset for the MIME body response can be specified explicitly using the `setCharacterEncoding(java.lang.String)` and `setContentType(java.lang.String)` methods, or implicitly using the `setLocale(java.util.Locale)` method. Explicit specifications take precedence over implicit specifications. If no charset is specified, ISO-8859-1 will be used. The `setCharacterEncoding`, `setContentType`, or `setLocale` method must be called before `getWriter` and before committing the response for the character encoding to be used.

See the Internet RFCs such as RFC 2045 for more information on MIME. Protocols such as SMTP and HTTP define profiles of MIME, and those standards are still evolving.

Author:
Various

See Also:
`ServletOutputStream`

Method Summary

`void`	**`flushBuffer`**`()` Forces any content in the buffer to be written to the client.
`int`	**`getBufferSize`**`()` Returns the actual buffer size used for the response.
`java.lang.String`	**`getCharacterEncoding`**`()` Returns the name of the character encoding (MIME charset) used for the body sent in this response.
`java.lang.String`	**`getContentType`**`()` Returns the content type used for the MIME body sent in this response.

`java.util.Locale`	**`getLocale`**`()` Returns the locale specified for this response using the `setLocale(java.util.Locale)` method.
`ServletOutputStream`	**`getOutputStream`**`()` Returns a `ServletOutputStream` suitable for writing binary data in the response.
`java.io.PrintWriter`	**`getWriter`**`()` Returns a `PrintWriter` object that can send character text to the client.
`boolean`	**`isCommitted`**`()` Returns a boolean indicating if the response has been committed.
`void`	**`reset`**`()` Clears any data that exists in the buffer as well as the status code and headers.
`void`	**`resetBuffer`**`()` Clears the content of the underlying buffer in the response without clearing headers or status code.
`void`	**`setBufferSize`**`(int size)` Sets the preferred buffer size for the body of the response.
`void`	**`setCharacterEncoding`**`(java.lang.String charset)` Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8.
`void`	**`setContentLength`**`(int len)` Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
`void`	**`setContentType`**`(java.lang.String type)` Sets the content type of the response being sent to the client, if the response has not been committed yet.
`void`	**`setLocale`**`(java.util.Locale loc)` Sets the locale of the response, if the response has not been committed yet.

Method Detail

getCharacterEncoding

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

Returns the name of the character encoding (MIME charset) used for the body sent in this response. The character encoding may have been specified explicitly using the `setCharacterEncoding(java.lang.String)` or `setContentType(java.lang.String)` methods, or implicitly using the `setLocale(java.util.Locale)` method. Explicit specifications take precedence over implicit specifications. Calls made to these methods after `getWriter` has been called or after the response has been committed have no effect on the character encoding. If no character encoding has been specified, `ISO-8859-1` is returned.

See RFC 2047 (http://www.ietf.org/rfc/rfc2047.txt) for more information about character encoding and MIME.

Returns:

a `String` specifying the name of the character encoding, for example, `UTF-8`

---

getContentType

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

Returns the content type used for the MIME body sent in this response. The content type proper must have been specified using `setContentType(java.lang.String)` before the response is committed. If no content type has been specified, this method returns null. If a content type has been specified, and a character encoding has been explicitly or implicitly specified as described in `getCharacterEncoding()` or `getWriter()` has been called, the charset parameter is included in the string returned. If no character encoding has been specified, the charset parameter is omitted.

Returns:
a `String` specifying the content type, for example, `text/html; charset=UTF-8`, or null

Since:
Servlet 2.4

---

getOutputStream

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

Returns a `ServletOutputStream` suitable for writing binary data in the response. The servlet container does not encode the binary data.

Calling flush() on the ServletOutputStream commits the response. Either this method or `getWriter()` may be called to write the body, not both.

Returns:
a `ServletOutputStream` for writing binary data

Throws:
`IllegalStateException` - if the `getWriter` method has been called on this response
`java.io.IOException` - if an input or output exception occurred

See Also:
`getWriter()`

---

getWriter

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

Returns a `PrintWriter` object that can send character text to the client. The `PrintWriter` uses the character encoding returned by `getCharacterEncoding()`. If the response's character encoding has not been specified as described in `getCharacterEncoding` (i.e., the method just returns the default value `ISO-8859-1`), `getWriter` updates it to `ISO-8859-1`.

Calling flush() on the `PrintWriter` commits the response.

Either this method or `getOutputStream()` may be called to write the body, not both.

Returns:

a `PrintWriter` object that can return character data to the client

Throws:

`UnsupportedEncodingException` - if the character encoding returned by `getCharacterEncoding` cannot be used

`IllegalStateException` - if the `getOutputStream` method has already been called for this response object

`java.io.IOException` - if an input or output exception occurred

See Also:

`getOutputStream()`, `setCharacterEncoding(java.lang.String)`

---

setCharacterEncoding

```
void setCharacterEncoding(java.lang.String charset)
```

Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8. If the character encoding has already been set by `setContentType(java.lang.String)` or `setLocale(java.util.Locale)`, this method overrides it. Calling `setContentType(java.lang.String)` with the `String` of `text/html` and calling this method with the `String` of `UTF-8` is equivalent with calling `setContentType` with the `String` of `text/html; charset=UTF-8`.

This method can be called repeatedly to change the character encoding. This method has no effect if it is called after `getWriter` has been called or after the response has been committed.

Containers must communicate the character encoding used for the servlet response's writer to the client if the protocol provides a way for doing so. In the case of HTTP, the character encoding is communicated as part of the `Content-Type` header for text media types. Note that the character encoding cannot be communicated via HTTP headers if the servlet does not specify a content type; however, it is still used to encode text written via the servlet response's writer.

Parameters:

`charset` - a String specifying only the character set defined by IANA Character Sets (http://www.iana.org/assignments/character-sets)

Since:

Servlet 2.4

See Also:

`#setLocale`

---

setContentLength

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

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

Parameters:

`len` - an integer specifying the length of the content being returned to the client; sets the Content-Length header

---

setContentType

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

Sets the content type of the response being sent to the client, if the response has not been committed yet. The given content type may include a character encoding specification, for example, `text/html;charset=UTF-8`. The response's character encoding is only set from the given content type if this method is called before `getWriter` is called.

This method may be called repeatedly to change content type and character encoding. This method has no effect if called after the response has been committed. It does not set the response's character encoding if it is called after `getWriter` has been called or after the response has been committed.

Containers must communicate the content type and the character encoding used for the servlet response's writer to the client if the protocol provides a way for doing so. In the case of HTTP, the `Content-Type` header is used.

Parameters:
`type` - a `String` specifying the MIME type of the content

See Also:
`setLocale(java.util.Locale)`, `setCharacterEncoding(java.lang.String)`, `getOutputStream()`, `getWriter()`

---

setBufferSize

`void setBufferSize(int size)`

Sets the preferred buffer size for the body of the response. The servlet container will use a buffer at least as large as the size requested. The actual buffer size used can be found using `getBufferSize`.

A larger buffer allows more content to be written before anything is actually sent, thus providing the servlet with more time to set appropriate status codes and headers. A smaller buffer decreases server memory load and allows the client to start receiving data more quickly.

This method must be called before any response body content is written; if content has been written or the response object has been committed, this method throws an `IllegalStateException`.

Parameters:
`size` - the preferred buffer size

Throws:
`IllegalStateException` - if this method is called after content has been written

See Also:
`getBufferSize()`, `flushBuffer()`, `isCommitted()`, `reset()`

---

getBufferSize

`int getBufferSize()`

Returns the actual buffer size used for the response. If no buffering is used, this method returns 0.

Returns:
the actual buffer size used

See Also:
`setBufferSize(int)`, `flushBuffer()`, `isCommitted()`, `reset()`

---

flushBuffer

```
void flushBuffer()
                 throws java.io.IOException
```

Forces any content in the buffer to be written to the client. A call to this method automatically commits the response, meaning the status code and headers will be written.

Throws:
`java.io.IOException`
See Also:
`setBufferSize(int)`, `getBufferSize()`, `isCommitted()`, `reset()`

---

resetBuffer

```
void resetBuffer()
```

Clears the content of the underlying buffer in the response without clearing headers or status code. If the response has been committed, this method throws an `IllegalStateException`.

Since:
Servlet 2.3
See Also:
`setBufferSize(int)`, `getBufferSize()`, `isCommitted()`, `reset()`

---

isCommitted

```
boolean isCommitted()
```

Returns a boolean indicating if the response has been committed. A committed response has already had its status code and headers written.

Returns:
a boolean indicating if the response has been committed
See Also:
`setBufferSize(int)`, `getBufferSize()`, `flushBuffer()`, `reset()`

---

reset

```
void reset()
```

Clears any data that exists in the buffer as well as the status code and headers. If the response has been committed, this method throws an `IllegalStateException`.

Throws:
`IllegalStateException` - if the response has already been committed
See Also:

`setBufferSize(int)`, `getBufferSize()`, `flushBuffer()`, `isCommitted()`

---

setLocale

```
void setLocale(java.util.Locale loc)
```

Sets the locale of the response, if the response has not been committed yet. It also sets the response's character encoding appropriately for the locale, if the character encoding has not been explicitly set using `setContentType(java.lang.String)` or `setCharacterEncoding(java.lang.String)`, `getWriter` hasn't been called yet, and the response hasn't been committed yet. If the deployment descriptor contains a `locale-encoding-mapping-list` element, and that element provides a mapping for the given locale, that mapping is used. Otherwise, the mapping from locale to character encoding is container dependent.

This method may be called repeatedly to change locale and character encoding. The method has no effect if called after the response has been committed. It does not set the response's character encoding if it is called after `setContentType(java.lang.String)` has been called with a charset specification, after `setCharacterEncoding(java.lang.String)` has been called, after `getWriter` has been called, or after the response has been committed.

Containers must communicate the locale and the character encoding used for the servlet response's writer to the client if the protocol provides a way for doing so. In the case of HTTP, the locale is communicated via the `Content-Language` header, the character encoding as part of the `Content-Type` header for text media types. Note that the character encoding cannot be communicated via HTTP headers if the servlet does not specify a content type; however, it is still used to encode text written via the servlet response's writer.

Parameters:
`loc` - the locale of the response

See Also:
`getLocale()`, `setContentType(java.lang.String)`, `setCharacterEncoding(java.lang.String)`

---

getLocale

```
java.util.Locale getLocale()
```

Returns the locale specified for this response using the `setLocale(java.util.Locale)` method. Calls made to `setLocale` after the response is committed have no effect. If no locale has been specified, the container's default locale is returned.

See Also:
`setLocale(java.util.Locale)`

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class ServletResponseWrapper

```
java.lang.Object
  └javax.servlet.ServletResponseWrapper
```

All Implemented Interfaces:

ServletResponse

Direct Known Subclasses:

HttpServletResponseWrapper

```
public class ServletResponseWrapper
extends java.lang.Object
implements ServletResponse
```

Provides a convenient implementation of the ServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet. This class implements the Wrapper or Decorator pattern. Methods default to calling through to the wrapped response object.

Since:

Servlet 2.3

Author:

Various

See Also:

`ServletResponse`

Constructor Summary

`ServletResponseWrapper``(ServletResponse response)` Creates a ServletResponse adaptor wrapping the given response object.

Method Summary

`void`	**`flushBuffer`**`()` The default behavior of this method is to call flushBuffer() on the wrapped response object.
`int`	**`getBufferSize`**`()` The default behavior of this method is to return getBufferSize() on the wrapped response object.
`java.lang.String`	**`getCharacterEncoding`**`()` The default behavior of this method is to return getCharacterEncoding() on the wrapped response object.
`java.lang.String`	**`getContentType`**`()` The default behavior of this method is to return getContentType() on the

	wrapped response object.
`java.util.Locale`	**getLocale**() The default behavior of this method is to return getLocale() on the wrapped response object.
`ServletOutputStream`	**getOutputStream**() The default behavior of this method is to return getOutputStream() on the wrapped response object.
`ServletResponse`	**getResponse**() Return the wrapped ServletResponse object.
`java.io.PrintWriter`	**getWriter**() The default behavior of this method is to return getWriter() on the wrapped response object.
`boolean`	**isCommitted**() The default behavior of this method is to return isCommitted() on the wrapped response object.
`boolean`	**isWrapperFor**(java.lang.Class wrappedType) Checks (recursively) if this ServletResponseWrapper wraps a `ServletResponse` of the given class type.
`boolean`	**isWrapperFor**(ServletResponse wrapped) Checks (recursively) if this ServletResponseWrapper wraps the given `ServletResponse` instance.
`void`	**reset**() The default behavior of this method is to call reset() on the wrapped response object.
`void`	**resetBuffer**() The default behavior of this method is to call resetBuffer() on the wrapped response object.
`void`	**setBufferSize**(int size) The default behavior of this method is to call setBufferSize(int size) on the wrapped response object.
`void`	**setCharacterEncoding**(java.lang.String charset) The default behavior of this method is to call setCharacterEncoding(String charset) on the wrapped response object.
`void`	**setContentLength**(int len) The default behavior of this method is to call setContentLength(int len) on the wrapped response object.
`void`	**setContentType**(java.lang.String type) The default behavior of this method is to call setContentType(String type) on the wrapped response object.
`void`	**setLocale**(java.util.Locale loc) The default behavior of this method is to call setLocale(Locale loc) on the wrapped response object.
`void`	**setResponse**(ServletResponse response) Sets the response being wrapped.

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

Constructor Detail

ServletResponseWrapper

`public ServletResponseWrapper(ServletResponse response)`

Creates a ServletResponse adaptor wrapping the given response object.

Throws:
`java.lang.IllegalArgumentException` - if the response is null.

Method Detail

getResponse

`public ServletResponse getResponse()`

Return the wrapped ServletResponse object.

---

setResponse

`public void setResponse(ServletResponse response)`

Sets the response being wrapped.

Throws:
`java.lang.IllegalArgumentException` - if the response is null.

---

setCharacterEncoding

`public void setCharacterEncoding(java.lang.String charset)`

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

Specified by:
`setCharacterEncoding` in interface `ServletResponse`

Parameters:
`charset` - a String specifying only the character set defined by IANA Character Sets (http://www.iana.org/assignments/character-sets)

Since:
Servlet 2.4

See Also:
`#setLocale`

---

getCharacterEncoding

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

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

Specified by:

`getCharacterEncoding` in interface `ServletResponse`

Returns:

a `String` specifying the name of the character encoding, for example, `UTF-8`

---

getOutputStream

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

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

Specified by:

`getOutputStream` in interface `ServletResponse`

Returns:

a `ServletOutputStream` for writing binary data

Throws:

`java.io.IOException` - if an input or output exception occurred

See Also:

`ServletResponse.getWriter()`

---

getWriter

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

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

Specified by:

`getWriter` in interface `ServletResponse`

Returns:

a `PrintWriter` object that can return character data to the client

Throws:

`java.io.IOException` - if an input or output exception occurred

See Also:

`ServletResponse.getOutputStream()`, `ServletResponse.setCharacterEncoding(java.lang.String)`

---

setContentLength

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

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

Specified by:

`setContentLength` in interface `ServletResponse`

Parameters:

`len` - an integer specifying the length of the content being returned to the client; sets the Content-Length header

---

setContentType

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

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

Specified by:

`setContentType` in interface `ServletResponse`

Parameters:

`type` - a `String` specifying the MIME type of the content

See Also:

`ServletResponse.setLocale(java.util.Locale)`, `ServletResponse.setCharacterEncoding(java.lang.String)`, `ServletResponse.getOutputStream()`, `ServletResponse.getWriter()`

---

getContentType

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

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

Specified by:

`getContentType` in interface `ServletResponse`

Returns:

a `String` specifying the content type, for example, `text/html; charset=UTF-8`, or null

Since:

Servlet 2.4

---

setBufferSize

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

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

Specified by:

`setBufferSize` in interface `ServletResponse`

Parameters:

`size` - the preferred buffer size

See Also:

`ServletResponse.getBufferSize()`, `ServletResponse.flushBuffer()`, `ServletResponse.isCommitted()`, `ServletResponse.reset()`

---

getBufferSize

```
public int getBufferSize()
```

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

Specified by:
`getBufferSize` in interface `ServletResponse`

Returns:
the actual buffer size used

See Also:
`ServletResponse.setBufferSize(int)`, `ServletResponse.flushBuffer()`, `ServletResponse.isCommitted()`, `ServletResponse.reset()`

---

flushBuffer

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

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

Specified by:
`flushBuffer` in interface `ServletResponse`

Throws:
`java.io.IOException`

See Also:
`ServletResponse.setBufferSize(int)`, `ServletResponse.getBufferSize()`, `ServletResponse.isCommitted()`, `ServletResponse.reset()`

---

isCommitted

```
public boolean isCommitted()
```

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

Specified by:
`isCommitted` in interface `ServletResponse`

Returns:
a boolean indicating if the response has been committed

See Also:
`ServletResponse.setBufferSize(int)`, `ServletResponse.getBufferSize()`, `ServletResponse.flushBuffer()`, `ServletResponse.reset()`

---

reset

```
public void reset()
```

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

Specified by:
`reset` in interface `ServletResponse`

See Also:
`ServletResponse.setBufferSize(int)`, `ServletResponse.getBufferSize()`, `ServletResponse.flushBuffer()`, `ServletResponse.isCommitted()`

---

resetBuffer

```
public void resetBuffer()
```

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

Specified by:
`resetBuffer` in interface `ServletResponse`

See Also:
`ServletResponse.setBufferSize(int)`, `ServletResponse.getBufferSize()`, `ServletResponse.isCommitted()`, `ServletResponse.reset()`

---

setLocale

```
public void setLocale(java.util.Locale loc)
```

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

Specified by:
`setLocale` in interface `ServletResponse`

Parameters:
`loc` - the locale of the response

See Also:
`ServletResponse.getLocale()`, `ServletResponse.setContentType(java.lang.String)`, `ServletResponse.setCharacterEncoding(java.lang.String)`

---

getLocale

```
public java.util.Locale getLocale()
```

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

Specified by:
`getLocale` in interface `ServletResponse`

See Also:
`ServletResponse.setLocale(java.util.Locale)`

---

isWrapperFor

```
public boolean isWrapperFor(ServletResponse wrapped)
```

Checks (recursively) if this ServletResponseWrapper wraps the given `ServletResponse` instance.

Parameters:
`wrapped` - the ServletResponse instance to search for

Returns:

true if this ServletResponseWrapper wraps the given ServletResponse instance, false otherwise

Since:

Servlet 3.0

---

isWrapperFor

```
public boolean isWrapperFor(java.lang.Class wrappedType)
```

Checks (recursively) if this ServletResponseWrapper wraps a `ServletResponse` of the given class type.

Parameters:

`wrappedType` - the ServletResponse class type to search for

Returns:

true if this ServletResponseWrapper wraps a ServletResponse of the given class type, false otherwise

Throws:

`IllegalArgumentException` - if the given class does not implement `ServletResponse`

Since:

Servlet 3.0

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Submit a bug or feature

javax.servlet

Class ServletSecurityElement

```
java.lang.Object
  └javax.servlet.HttpConstraintElement
      └javax.servlet.ServletSecurityElement
```

```
public class ServletSecurityElement
extends HttpConstraintElement
```

Java Class represntation of a `ServletSecurity` annotation value.

Since:

Servlet 3.0

Constructor Summary

`ServletSecurityElement``()` Constructs an instance using the default `HttpConstraintElement` value as the default Constraint element and with no HTTP Method specific constraint elements.
`ServletSecurityElement``(java.util.Collection<HttpMethodConstraintElement> methodConstraints)` Constructs an instance using the default `HttpConstraintElement` value as the default Constraint element and with a collection of HTTP Method specific constraint elements.
`ServletSecurityElement``(HttpConstraintElement constraint)` Constructs an instance with a default Constraint element and with no HTTP Method specific constraint elements.
`ServletSecurityElement``(HttpConstraintElement constraint, java.util.Collection<HttpMethodConstraintElement> methodConstraints)` Constructs an instance with a default Constraint element and with a collection of HTTP Method specific constraint elements.
`ServletSecurityElement``(ServletSecurity annotation)` Constructs an instance from a `ServletSecurity` annotation value.

Method Summary

`java.util.Collection<HttpMethodConstraintElement>`	**`getHttpMethodConstraints`**`()` Gets the (possibly empty) collection of HTTP Method specific constraint elements.
`java.util.Collection<java.lang.String>`	**`getMethodNames`**`()` Gets the set of HTTP method names named by the HttpMethodConstraints.

Methods inherited from class javax.servlet.HttpConstraintElement
`getEmptyRoleSemantic, getRolesAllowed, getTransportGuarantee`

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

Constructor Detail

ServletSecurityElement

```
public ServletSecurityElement()
```

Constructs an instance using the default `HttpConstraintElement` value as the default Constraint element and with no HTTP Method specific constraint elements.

ServletSecurityElement

```
public ServletSecurityElement(HttpConstraintElement constraint)
```

Constructs an instance with a default Constraint element and with no HTTP Method specific constraint elements.

Parameters:

`constraint` - the HttpConstraintElement to be applied to all HTTP methods other than those represented in the `methodConstraints`

ServletSecurityElement

```
public ServletSecurityElement(java.util.Collection<HttpMethodConstraintElement> methodConstraints)
```

Constructs an instance using the default `HttpConstraintElement` value as the default Constraint element and with a collection of HTTP Method specific constraint elements.

Parameters:

`methodConstraints` - the collection of HTTP method specific constraint elements

Throws:

`IllegalArgumentException` - if duplicate method names are detected

ServletSecurityElement

```
public ServletSecurityElement(HttpConstraintElement constraint,
                              java.util.Collection<HttpMethodConstraintElement> methodConstraints)
```

Constructs an instance with a default Constraint element and with a collection of HTTP Method specific constraint elements.

Parameters:

`constraint` - the HttpConstraintElement to be applied to all HTTP methods other than those represented in the `methodConstraints`

`methodConstraints` - the collection of HTTP method specific constraint elements.

Throws:

`IllegalArgumentException` - if duplicate method names are detected

ServletSecurityElement

```
public ServletSecurityElement(ServletSecurity annotation)
```

Constructs an instance from a `ServletSecurity` annotation value.

Parameters:

`annotation` - the annotation value

Throws:

`IllegalArgumentException` - if duplicate method names are detected

Method Detail

getHttpMethodConstraints

```
public java.util.Collection<HttpMethodConstraintElement> getHttpMethodConstraints()
```

Gets the (possibly empty) collection of HTTP Method specific constraint elements.

If permitted, any changes to the returned `Collection` must not affect this `ServletSecurityElement`.

Returns:

the (possibly empty) collection of HttpMethodConstraintElement objects

---

getMethodNames

```
public java.util.Collection<java.lang.String> getMethodNames()
```

Gets the set of HTTP method names named by the HttpMethodConstraints.

If permitted, any changes to the returned `Collection` must not affect this `ServletSecurityElement`.

Returns:

the collection String method names

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS   FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

---

javax.servlet

Interface SessionCookieConfig

```
public interface SessionCookieConfig
```

Class that may be used to configure various properties of cookies used for session tracking purposes.

An instance of this class is acquired by a call to `ServletContext#getSessionCookieConfig`.

Since:

Servlet 3.0

Method Summary

`java.lang.String`	**`getComment`**`()` Gets the comment that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.
`java.lang.String`	**`getDomain`**`()` Gets the domain name that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.
`int`	**`getMaxAge`**`()` Gets the lifetime (in seconds) of the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.
`java.lang.String`	**`getName`**`()` Gets the name that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.
`java.lang.String`	**`getPath`**`()` Gets the path that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.
`boolean`	**`isHttpOnly`**`()` Checks if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired will be marked as *HttpOnly*.
`boolean`	**`isSecure`**`()` Checks if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired will be marked as *secure* even if the request that initiated the corresponding session is using plain HTTP instead of HTTPS.

void	**setComment**(java.lang.String comment) Sets the comment that will be assigned to any session tracking cookies created on behalf of the application represented by the ServletContext from which this SessionCookieConfig was acquired.
void	**setDomain**(java.lang.String domain) Sets the domain name that will be assigned to any session tracking cookies created on behalf of the application represented by the ServletContext from which this SessionCookieConfig was acquired.
void	**setHttpOnly**(boolean httpOnly) Marks or unmarks the session tracking cookies created on behalf of the application represented by the ServletContext from which this SessionCookieConfig was acquired as *HttpOnly*.
void	**setMaxAge**(int maxAge) Sets the lifetime (in seconds) for the session tracking cookies created on behalf of the application represented by the ServletContext from which this SessionCookieConfig was acquired.
void	**setName**(java.lang.String name) Sets the name that will be assigned to any session tracking cookies created on behalf of the application represented by the ServletContext from which this SessionCookieConfig was acquired.
void	**setPath**(java.lang.String path) Sets the path that will be assigned to any session tracking cookies created on behalf of the application represented by the ServletContext from which this SessionCookieConfig was acquired.
void	**setSecure**(boolean secure) Marks or unmarks the session tracking cookies created on behalf of the application represented by the ServletContext from which this SessionCookieConfig was acquired as *secure*.

Method Detail

setName

```
void setName(java.lang.String name)
```

Sets the name that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

NOTE: Changing the name of session tracking cookies may break other tiers (for example, a load balancing frontend) that assume the cookie name to be equal to the default `JSESSIONID`, and therefore should only be done cautiously.

Parameters:

`name` - the cookie name to use

Throws:

`IllegalStateException` - if the `ServletContext` from which this `SessionCookieConfig` was acquired has already been initialized

getName

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

Gets the name that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

By default, `JSESSIONID` will be used as the cookie name.

Returns:
the cookie name set via `setName(java.lang.String)`, or `null` if `setName(java.lang.String)` was never called

See Also:
`Cookie.getName()`

---

setDomain

`void` **`setDomain`**`(java.lang.String domain)`

Sets the domain name that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

Parameters:
`domain` - the cookie domain to use

Throws:
`IllegalStateException` - if the `ServletContext` from which this `SessionCookieConfig` was acquired has already been initialized

See Also:
`Cookie.setDomain(String)`

---

getDomain

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

Gets the domain name that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

Returns:
the cookie domain set via `setDomain(java.lang.String)`, or `null` if `setDomain(java.lang.String)` was never called

See Also:
`Cookie.getDomain()`

---

setPath

`void` **`setPath`**`(java.lang.String path)`

Sets the path that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

Parameters:
`path` - the cookie path to use

Throws:
`IllegalStateException` - if the `ServletContext` from which this `SessionCookieConfig` was acquired has already been initialized

See Also:
`Cookie.setPath(String)`

---

getPath

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

Gets the path that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

By default, the context path of the `ServletContext` from which this `SessionCookieConfig` was acquired will be used.

Returns:
the cookie path set via `setPath(java.lang.String)`, or `null` if `setPath(java.lang.String)` was never called

See Also:
`Cookie.getPath()`

---

setComment

```
void setComment(java.lang.String comment)
```

Sets the comment that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

As a side effect of this call, the session tracking cookies will be marked with a `Version` attribute equal to `1`.

Parameters:
`comment` - the cookie comment to use

Throws:
`IllegalStateException` - if the `ServletContext` from which this `SessionCookieConfig` was acquired has already been initialized

See Also:
`Cookie.setComment(String)`, `Cookie.getVersion()`

---

getComment

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

Gets the comment that will be assigned to any session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

Returns:
the cookie comment set via `setComment(java.lang.String)`, or `null` if `setComment(java.lang.String)` was never called

See Also:
`Cookie.getComment()`

---

setHttpOnly

```
void setHttpOnly(boolean httpOnly)
```

Marks or unmarks the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired as *HttpOnly*.

A cookie is marked as `HttpOnly` by adding the `HttpOnly` attribute to it. *HttpOnly* cookies are not supposed to be exposed to client-side scripting code, and may therefore help mitigate certain kinds of cross-site scripting attacks.

Parameters:
`httpOnly` - true if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired shall be marked as *HttpOnly*, false otherwise

Throws:
`IllegalStateException` - if the `ServletContext` from which this `SessionCookieConfig` was acquired has already been initialized

See Also:
`Cookie.setHttpOnly(boolean)`

---

isHttpOnly

```
boolean isHttpOnly()
```

Checks if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired will be marked as *HttpOnly*.

Returns:
true if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired will be marked as *HttpOnly*, false otherwise

See Also:
`Cookie.isHttpOnly()`

---

setSecure

```
void setSecure(boolean secure)
```

Marks or unmarks the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired as *secure*.

One use case for marking a session tracking cookie as `secure`, even though the request that initiated the session came over HTTP, is to support a topology where the web container is front-ended by an SSL offloading load balancer. In this case, the traffic between the client and the load balancer will be over HTTPS, whereas the traffic between the load balancer and the web container will be over HTTP.

Parameters:

`secure` - true if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired shall be marked as *secure* even if the request that initiated the corresponding session is using plain HTTP instead of HTTPS, and false if they shall be marked as *secure* only if the request that initiated the corresponding session was also secure

Throws:

`IllegalStateException` - if the `ServletContext` from which this `SessionCookieConfig` was acquired has already been initialized

See Also:

`Cookie.setSecure(boolean)`, `ServletRequest#isSecure()`

---

isSecure

`boolean **isSecure**()`

Checks if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired will be marked as *secure* even if the request that initiated the corresponding session is using plain HTTP instead of HTTPS.

Returns:

true if the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired will be marked as *secure* even if the request that initiated the corresponding session is using plain HTTP instead of HTTPS, and false if they will be marked as *secure* only if the request that initiated the corresponding session was also secure

See Also:

`Cookie.getSecure()`, `ServletRequest#isSecure()`

---

setMaxAge

`void **setMaxAge**(int maxAge)`

Sets the lifetime (in seconds) for the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

Parameters:

`maxAge` - the lifetime (in seconds) of the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

Throws:

`IllegalStateException` - if the `ServletContext` from which this `SessionCookieConfig` was acquired has already been initialized

See Also:

`Cookie.setMaxAge(int)`

---

getMaxAge

```
int getMaxAge()
```

Gets the lifetime (in seconds) of the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired.

By default, `-1` is returned.

Returns:

the lifetime (in seconds) of the session tracking cookies created on behalf of the application represented by the `ServletContext` from which this `SessionCookieConfig` was acquired, or `-1` (the default)

See Also:

`Cookie.getMaxAge()`

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS **FRAMES NO FRAMES All Classes**

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Enum SessionTrackingMode

```
java.lang.Object
  └java.lang.Enum<SessionTrackingMode>
      └javax.servlet.SessionTrackingMode
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable<SessionTrackingMode>

```
public enum SessionTrackingMode
extends java.lang.Enum<SessionTrackingMode>
```

Enumeration of session tracking modes.

Since:
Servlet 3.0

Enum Constant Summary

COOKIE
SSL
URL

Method Summary

static SessionTrackingMode	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.
static SessionTrackingMode[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

COOKIE

```
public static final SessionTrackingMode COOKIE
```

URL

```
public static final SessionTrackingMode URL
```

SSL

```
public static final SessionTrackingMode SSL
```

Method Detail

values

```
public static SessionTrackingMode[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (SessionTrackingMode c : SessionTrackingMode.values())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

---

valueOf

```
public static SessionTrackingMode valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

`name` - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

`IllegalArgumentException` - if this enum type has no constant with the specified name

`java.lang.NullPointerException` - if the argument is null

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Interface SingleThreadModel

Deprecated. *As of Java Servlet API 2.4, with no direct replacement.*

```
public interface SingleThreadModel
```

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

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

Note that SingleThreadModel does not solve all thread safety issues. For example, session attributes and static variables can still be accessed by multiple requests on multiple threads at the same time, even when SingleThreadModel servlets are used. It is recommended that a developer take other means to resolve those issues instead of implementing this interface, such as avoiding the usage of an instance variable or synchronizing the block of the code accessing those resources. This interface is deprecated in Servlet API version 2.4.

Author:
Various

Submit a bug or feature

Generated on 10-February-2011 12:41

javax.servlet

Class UnavailableException

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └javax.servlet.ServletException
              └javax.servlet.UnavailableException
```

All Implemented Interfaces:

java.io.Serializable

```
public class UnavailableException
extends ServletException
```

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

When a servlet or filter is permanently unavailable, something is wrong with it, and it cannot handle requests until some action is taken. For example, a servlet might be configured incorrectly, or a filter's state may be corrupted. The component should log both the error and the corrective action that is needed.

A servlet or filter is temporarily unavailable if it cannot handle requests momentarily due to some system-wide problem. For example, a third-tier server might not be accessible, or there may be insufficient memory or disk storage to handle requests. A system administrator may need to take corrective action.

Servlet containers can safely treat both types of unavailable exceptions in the same way. However, treating temporary unavailability effectively makes the servlet container more robust. Specifically, the servlet container might block requests to the servlet or filter for a period of time suggested by the exception, rather than rejecting them until the servlet container restarts.

Author:

Various

See Also:

Serialized Form

Constructor Summary

`UnavailableException(int seconds, Servlet servlet, java.lang.String msg)` **Deprecated.** *As of Java Servlet API 2.2, use* `UnavailableException(String, int)` *instead.*
`UnavailableException(Servlet servlet, java.lang.String msg)` **Deprecated.** *As of Java Servlet API 2.2, use* `UnavailableException(String)` *instead.*
`UnavailableException(java.lang.String msg)` Constructs a new exception with a descriptive message indicating that the servlet is permanently unavailable.

`UnavailableException``(java.lang.String msg, int seconds)` Constructs a new exception with a descriptive message indicating that the servlet is temporarily unavailable and giving an estimate of how long it will be unavailable.

Method Summary

`Servlet`	**`getServlet`**`()` **Deprecated.** *As of Java Servlet API 2.2, with no replacement. Returns the servlet that is reporting its unavailability.*
`int`	**`getUnavailableSeconds`**`()` Returns the number of seconds the servlet expects to be temporarily unavailable.
`boolean`	**`isPermanent`**`()` Returns a `boolean` indicating whether the servlet is permanently unavailable.

Methods inherited from class javax.servlet.ServletException
`getRootCause`

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

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

Constructor Detail

UnavailableException

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

Deprecated. *As of Java Servlet API 2.2, use `UnavailableException(String)` instead.*

Parameters:

`servlet` - the `Servlet` instance that is unavailable
`msg` - a `String` specifying the descriptive message

UnavailableException

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

Deprecated. *As of Java Servlet API 2.2, use `UnavailableException(String, int)` instead.*

Parameters:

`seconds` - an integer specifying the number of seconds the servlet expects to be unavailable; if zero or negative, indicates that the servlet can't make an estimate

`servlet` - the `Servlet` that is unavailable
`msg` - a `String` specifying the descriptive message, which can be written to a log file or displayed for the user.

UnavailableException

```
public UnavailableException(java.lang.String msg)
```

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

Parameters:
`msg` - a `String` specifying the descriptive message

UnavailableException

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

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

In some cases, the servlet cannot make an estimate. For example, the servlet might know that a server it needs is not running, but not be able to report how long it will take to be restored to functionality. This can be indicated with a negative or zero value for the `seconds` argument.

Parameters:
`msg` - a `String` specifying the descriptive message, which can be written to a log file or displayed for the user.
`seconds` - an integer specifying the number of seconds the servlet expects to be unavailable; if zero or negative, indicates that the servlet can't make an estimate

Method Detail

isPermanent

```
public boolean isPermanent()
```

Returns a `boolean` indicating whether the servlet is permanently unavailable. If so, something is wrong with the servlet, and the system administrator must take some corrective action.

Returns:
`true` if the servlet is permanently unavailable; `false` if the servlet is available or temporarily unavailable

---

getServlet

```
public Servlet getServlet()
```

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

Returns:

the `Servlet` object that is throwing the `UnavailableException`

---

getUnavailableSeconds

```
public int getUnavailableSeconds()
```

Returns the number of seconds the servlet expects to be temporarily unavailable.

If this method returns a negative number, the servlet is permanently unavailable or cannot provide an estimate of how long it will be unavailable. No effort is made to correct for the time elapsed since the exception was first reported.

Returns:

an integer specifying the number of seconds the servlet will be temporarily unavailable, or a negative number if the servlet is permanently unavailable or cannot make an estimate

---

Submit a bug or feature