BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Table of Contents

Table of Contents

Table of Contents

Control and Web Service API Reference

The Control APIs include the base interfaces for the Platform controls as well as context interfaces that provide access to a control's execution context. The Web Service APIs provide access to a web service's execution context, including conversation lifetime control.

Topics Included in This Section

The following links lead to the summary page for each API package.

com.bea.control

Contains interfaces and classes for Java controls and web services.

com.bea.jws

Contains interfaces and classes for web services.

com.bea.wlw.util

Contains utility classes.

webogic.jws.control

Deprecated. Use com.bea.control instead.

weblogic.jws.util

Deprecated. Use com.bea.wlw.util instead.

Related Topics

Using Built−In Java Controls

Building Web Services

com.bea.control Package

Interface Summary

B2BCallback	
B2BCallbackControl	
Control	The base interface for all controls.
ControlContext	The ControlContext interface defines the container services and events that controls can use at run time.
ControlContext.Callback	Provides a way for controls to receive callbacks from their environment.
ControlExtension	A marker interface indicating that the implementing source (in a JCX file) extends a Java control.
ControlFactory	This is a marker interface for all control factories.
ControlSource	A marker interface indicating that the implementing source (in a JCS file) is control source.
DatabaseControl	Simplifies access to a relational database from your Java code using SQL commands.
EBXMLControl	WebLogic Integration Control for ebXML protocol This is a base ebXML control that can be used to send and receive messages in the form of callback.
EBXMLControl.Callback	
EmailControl	Email control base interface.
EntityEJBControl	As part of the EJB control, this interface simplifies access to entity Enterprise JavaBeans (EJBs).
Extensible	A control implementation must implement the Extensible interface if it can be extended by a control extension (through a JCX file).
ExternalCallbackTarget	The ExternalCallbackTarget interface is a marker interface that should be placed on any control implementation (JCS) class that can be the target of an asynchronous external callback.
FileControl	File Control base interface
Issue	Implement this interface to create an issue class that your control can return to have the IDE display errors or messages.
JMSControl	Simplifies access to the Java Message Service.
JwsContext	Provides access to container services that support web services (JWS files).
JwsContext.Callback	The Callback interface defines events that can be received through the `Context` object.
ProcessControl	The process control is used to call a sub−process from a parent process.
PublishControl	
RosettaNetControl	Basic RosettaNet control.
RosettaNetControl.Callback	The following methods are available after a JCX instance is created.
ServiceBrokerControl	
ServiceControl	Provides simplified access to web services.
ServiceControl.Callback	Provides a way for the Service control to pass callback events to its clients.
SessionEJBControl	As part of the EJB control, this interface simplifies access to session

	Enterprise JavaBeans (EJBs).
SubscriptionControl	
SubscriptionControl.Callback	
TaskControl	Task control interface.
TaskControl.Callback	
TaskWorkerControl	Task control interface.
TimerControl	Notifies your application when a specified period of time has elapsed or when a specified absolute time has been reached.
TimerControl.Callback	Provides a way for the Timer control to receive callbacks.
TimerControlFactory	The base factory interface for the TimerControl.
TPMControl	
ValidateAttribute	The ValidateAttribute interface provides methods with which you can validate a property tag attribute.
ValidateControl	The ValidateControl interface provides methods with which you can validate a control JCX file.
ValidateMethod	The ValidateMethod interface provides methods with which you can validate a control method.
ValidateTag	The ValidateTag interface can be used to validate a property annotation as a whole.
WliJMSControl	
WliJMSControl.Callback	

Class Summary

DatabaseControl.SQLFragment	
DatabaseControl.SQLParameter	
DatabaseFilter	The DatabaseFilter class is a helper class for SQL generation.
DatabaseFilter.FilterTerm	
DatabaseFilter.SortTerm	
DefaultIssue	The DefaultIssue class provides a simple implementation of the Issue interface.
SoapFault	Web service developers can use the SoapFault class to control the shape of the detail of SOAP fault data.

Exception Summary

ControlException	An exception class for use when throwing exceptions from a control.
ProcessControlException	
ServiceControlException	The ServiceControlException class can be used to throw exceptions from a web service control.
TPMControlException	

com.bea.control

Control Interface

public interface ***Control***

extends `Serializable`

The base interface for all controls. Every control must extend this interface.

Related Topics

Building Custom Java Controls

All Superinterfaces

`Serializable`

All Known Subinterfaces

`B2BCallback`, `B2BCallbackControl`, `ClickContentEventControl`, `Control`, `CreateUserControl`, `DatabaseControl`, `DatabaseControl`, `DisplayContentEventControl`, `EBXMLControl`, `EJBControl`, `EmailControl`, `EntityEJBControl`, `EntityEJBControl`, `EntityPropertyManager`, `EventService`, `ExtendedServiceControl`, `FileControl`, `GenericEventControl`, `GenericTrackingControl`, `GroupManager`, `GroupProfileManager`, `GroupProviderControl`, `JMSControl`, `JMSControl`, `ProcessControl`, `ProfileControl`, `PropertyControl`, `PropertySetManager`, `PublishControl`, `RealmConfiguration`, `RosettaNetControl`, `RuleEventControl`, `RulesExecutorControl`, `RulesManagerControl`, `ServiceBrokerControl`, `ServiceControl`, `ServiceControl`, `SessionEJBControl`, `SessionEJBControl`, `SessionLoginEventControl`, `SubscriptionControl`, `TaskControl`, `TaskWorkerControl`, `TimerControl`, `TimerControl`, `TPMControl`, `UserInfoControl`, `UserLoginControl`, `UserManager`, `UserProfileControl`, `UserProfileManager`, `UserProviderControl`, `UserRegistrationEventControl`, `WliJMSControl`,

---

com.bea.control

ControlContext Interface

public interface ***ControlContext***

extends `Context`

The ControlContext interface defines the container services and events that controls can use at run time. For example, through methods of this interface, a Java control can access its own property attribute values, handle lifecycle events, send callbacks, and so on. **Note:** The ControlContext interface is provided as part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Related Topics

`Context`

All Superinterfaces

`Context`, `Serializable`

Nested Class Summary

`public static interface`	**`ControlContext.Callback`** Provides a way for controls to receive callbacks from their environment.

Nested classes from interface weblogic.jws.control.`Context`

`Context.Callback`

Method Summary

`public void` ***`cancelEvents`***`(String` *eventName)*
Cancels callback events that have the specified event name.

`public Class` ***`getCallbackInterface`***`()`
Returns the java.lang.Class object for this component's callback interface.

`public String` ***`getControlAttribute`***`(String` *tagName,* `String` *attrName)*
Returns the value for the specified control tag attribute.

public ***getControlAttributes***(String *tagName)*
List Returns a list of the attributes and values for the specified property tag annotating a control instance.

public Class ***getControlInterface***()
Returns the java.lang.Class object for the component call interface.

public Object ***getMethodArgument***(String *argName)*
Returns the value for the specified method argument.

public String[] ***getMethodArgumentNames***()
Returns the argument names associated with the current invocation context.

public String ***getMethodAttribute***(String *tagName*, String *attrName)*
Returns the value for the specified method tag attribute.

public List ***getMethodAttributes***(String *tagName)*
Returns a list of the attributes and values for the specified property tag annotating a control method.

public Object ***raiseEvent***()
Forwards to this control's client a callback that the control has received.

public void ***scheduleEvent***(String *eventName*, Object[] *eventArgs, long time, boolean ignoreIfFinished)*
Schedules a callback to occur on this control instance at a specified time.

public Object ***sendEvent***(String *eventName*, Object[] *args)*
Sends a callback event that is declared as part of this control's callback interface.

Methods from interface `weblogic.jws.control.Context`

`finishConversation, getCallerPrincipal, getCurrentAge, getCurrentIdleTime, getLogger, getMaxAge, getMaxIdleTime, getService, isCallerInRole, isFinished, resetIdleTime, setMaxAge, setMaxAge, setMaxIdleTime, setMaxIdleTime`

Method Detail

cancelEvents(String) Method

```
public void cancelEvents(String eventName)
     throws SchedulerException
```

Cancels callback events that have the specified event name. For example, if the *eventName* callbacks have been scheduled through the scheduleEvent method, this method removes them from the schedule.

Parameters

eventName
The name of the callbacks to cancel.

Exceptions

SchedulerException

---

getCallbackInterface() Method

```
public Class getCallbackInterface()
```

Returns the java.lang.Class object for this component's callback interface. For example, for a Java control that is customizable by generating a JCX file, the callback interface can be defined in the JCX itself. In other words, it may not be known until design time, when a developer is customizing it. You can use this method to discover information about the interface.

Returns

The Class object for the callback interface.

---

getControlAttribute(String, String) Method

```
public String getControlAttribute(String tagName,
                                  String attrName)
```

Returns the value for the specified control tag attribute. Use this method to retrieve at run time the value of a property attribute set for a control instance. For properties set on a control instance, property tags annotate the control's variable declaration or the control interface definition. Note that while a property tag's XML file can define attributes as types other than String, the values are always returned at run time as a String. Your code should convert them to other types as needed.

Parameters

tagName
The tag whose attribute should be returned.

attrName
The attribute whose value should be returned.

Returns

The attribute value, or null if undefined.

---

getControlAttributes(String) Method

```
public List getControlAttributes(String tagName)
```

Returns a list of the attributes and values for the specified property tag annotating a control instance. This method returns the attributes as a `List` of `Map` objects. Each entry in the list maps an attribute name to the attribute's current value (as a String) for a single occurrence of the tag.

Parameters

tagName
The tag whose attributes should be returned.

Returns

A list of maps of attribute values

---

getControlInterface() Method

```
public Class getControlInterface()
```

Returns the java.lang.Class object for the component call interface.

Returns

The Class object for the component call interface.

---

getMethodArgument(String) Method

```
public Object getMethodArgument(String argName)
     throws IllegalArgumentException
```

Returns the value for the specified method argument. Use the `getMethodArgumentNames` method to retrieve a list of arguments for the current method.

Parameters

argName
The name of the method argument to return.

Returns

The value for the specified argument.

Exceptions

`IllegalArgumentException`
If there is no argument whose name matches *argName*.

---

getMethodArgumentNames() Method

```
public String[] getMethodArgumentNames()
```

Returns the argument names associated with the current invocation context. The names are returned in order based on the order of declaration. Use the getMethodArgument method to retrieve the value for a particular argument.

Returns

An array of the argument names.

---

getMethodAttribute(String, String) Method

```
public String getMethodAttribute(String tagName,
                                 String attrName)
```

Returns the value for the specified method tag attribute. Use this method to retrieve at run time the value of a property attribute set on a method. A method attribute is expressed through an annotation on a method declaration in a JCX file. Note that while a property tag's XML file can define attributes as types other than String, the values are always returned at run time as a String. Your code should convert them to other types as needed.

Parameters

tagName
The name of the property tag that exposes *attrName*.

attrName
The name of the attribute whose value should be retrieved.

Returns

The attribute's value; null if the attribute is undefined.

---

getMethodAttributes(String) Method

```
public List getMethodAttributes(String tagName)
```

Returns a list of the attributes and values for the specified property tag annotating a control method. This method returns the attributes as a `List` of `Map` objects. Each entry in the list maps an attribute name to the attribute's current value (as a String) for a single occurrence of the tag.

Parameters

tagName
The tag whose attributes should be returned.

Returns

A list of Map objects in which keys are attribute names and values are attribute values.

---

raiseEvent() Method

```
public Object raiseEvent()
     throws Exception
```

Forwards to this control's client a callback that the control has received. Use this method when your control will intercept callbacks and send them on to its client unmodified.

Returns

The callback's return value.

Exceptions

Exception

---

scheduleEvent(String, Object[], long, boolean) Method

```
public void scheduleEvent(String eventName,
                          Object[] eventArgs,
                          long time,
                          boolean ignoreIfFinished)
     throws SchedulerException
```

Schedules a callback to occur on this control instance at a specified time.

Parameters

eventName
: The name of the callback event to schedule.

eventArgs
: An array of values to use as callback arguments.

time
: The time at which the callback event should occur. The event is guaranteed to happen on or after this time.

ignoreIfFinished
: `true` if no exception should be thrown if the callback is sent to a finished instance; `false` if an exception should be thrown.

Exceptions

SchedulerException

---

sendEvent(String, Object[]) Method

```
public Object sendEvent(String eventName,
                        Object[] args)
     throws Exception
```

Sends a callback event that is declared as part of this control's callback interface.

Parameters

eventName
The callback event to send.

args
An array of values to be used for the callback's arguments.

Returns

The callback's return value.

Exceptions

Exception

com.bea.control

ControlContext.Callback Interface

public static interface ***ControlContext.Callback***

extends `Context.Callback`

Provides a way for controls to receive callbacks from their environment. In particular, this interface defines lifecycle event callbacks. A control receives these at run time, during its use in a container, such as a web service or pageflow. You can handle these callbacks to, for example, acquire and release resources in a manner that helps your application's performance by holding resources only when they're needed.

Related Topics

`Context.Callback`

All Superinterfaces

`Context.Callback`

Enclosing interface

`ControlContext`

Method Summary

`public void` ***`onAcquire`***`()`
Received just before any of this control's top−level container's methods execute.

`public void` ***`onRelease`***`()`
Received after this control's top level container's conversational context has finished.

`public void` ***`onReset`***`()`
Received after execution if this control's containing component (or a parent) is stateless.

Methods from interface `weblogic.jws.control.Context.Callback`

`onAgeTimeout, onAsyncFailure, onCreate, onException, onFinish, onIdleTimeout`

Method Detail

onAcquire() Method

```
public void onAcquire()
```

Received just before any of this control's top−level container's methods execute. Handle this callback to acquire any external resources or state that will be held through the lifetime of the top−level execution context. This callback is intended for advanced control development. For information on using advanced control features, see the ControlDevKit sample application available with WebLogic Workshop.

---

onRelease() Method

```
public void onRelease()
```

Received after this control's top level container's conversational context has finished. Handle this callback to release resources or state the control acquired when handling the onAcquire callback. This callback is intended for advanced control development. For information on using advanced control features, see the ControlDevKit sample application available with WebLogic Workshop.

---

onReset() Method

```
public void onReset()
```

Received after execution if this control's containing component (or a parent) is stateless. Handle this callback to reset internal fields to a default state. This callback is intended for advanced control development. For information on using advanced control features, see the ControlDevKit sample application available with WebLogic Workshop.

com.bea.control

ControlException Class

public class ***ControlException***

extends `RuntimeException`

An exception class for use when throwing exceptions from a control. It is recommended that you wrap exceptions thrown by a control in a ControlException. This class provides a means for adding exceptions that may have been received by your control. For more information on building Java controls, see Building Custom Java Controls.

Hierarchy

```
Object
 Throwable
  Exception
   RuntimeException
    ControlException
```

All Implemented Interfaces

`Serializable`

Direct Known Subclasses

`ControlException`, `ProcessControlException`

Constructor Summary

ControlException*(*`String` *message,* `Throwable` *t)*

Constructs a ControlException object using the specified String as a message, and the specified Throwable as a nested exception.

ControlException*(*`String` *message)*

Constructs a ControlException object with the specified String as a message.

Method Summary

`public Throwable` ***`getNestedException`****()*

Retrieves an exception nested within this exception.

Methods from `java.lang.Throwable`

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

Methods from class `java.lang.Object`

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

Constructor Detail

ControlException

```
public ControlException(String message,
                        Throwable t)
```

Constructs a ControlException object using the specified String as a message, and the specified Throwable as a nested exception.

ControlException

```
public ControlException(String message)
```

Constructs a ControlException object with the specified String as a message.

Method Detail

getNestedException() Method

```
public Throwable getNestedException()
```

Retrieves an exception nested within this exception.

Returns

The nested exception.

com.bea.control

ControlExtension Interface

public interface ***ControlExtension***

extends `Serializable`

A marker interface indicating that the implementing source (in a JCX file) extends a Java control. JCX files must extend this interface.

All Superinterfaces

`Serializable`

---

com.bea.control

ControlFactory Interface

public interface ***ControlFactory***

extends `Serializable`

This is a marker interface for all control factories. All factories must be serializable to support persistance of conversations containing factories.

All Superinterfaces

`Serializable`

All Known Subinterfaces

`ControlFactory`, `TimerControlFactory`, `TimerControlFactory`

---

com.bea.control

ControlSource Interface

public interface ***ControlSource***

extends `Serializable`

A marker interface indicating that the implementing source (in a JCS file) is control source. All JCS implementation classes must implement this interface. For more information on building Java controls, see Building Custom Java Controls.

All Superinterfaces

`Serializable`

---

com.bea.control

DatabaseControl Interface

public interface ***DatabaseControl***

extends `Control`

Simplifies access to a relational database from your Java code using SQL commands. The Database control handles the work of connecting to the database, which makes it simpler to use than JDBC. Note that only one of the methods of this interface (acceptChanges) is visible in Design View for a Database control in your application. You will typically not use the others. The best way to use a Database control is to add it to your design, then add to the control methods that capture the specific SQL expressions you want to execute against the database. Tasks such as getting a database connection are handled for you as you use the control. For more information about using the Database control, see Database Control.

All Superinterfaces

`Control`, `Control`, `Serializable`

All Known Subinterfaces

`DatabaseControl`

Nested Class Summary

public static class	**DatabaseControl.SQLFragment**
public static class	**DatabaseControl.SQLParameter**

Method Summary

public void	***acceptChanges****(RowSet r)* Wrapper for RowSet.acceptChanges().
public Connection	***getConnection****()* Returns a database connection to the server associated with the control.
public Calendar	***getDataSourceCalendar****()* Gets the Calendar instance used when setting and getting `Date`, `Time`, and `Timestamp` values.
public	

`void` ***`setDataSourceCalendar`***`(Calendar cal)`
Sets the Calendar instance that should be used when setting and getting `Date`, `Time`, and `Timestamp` values.

Method Detail

acceptChanges(RowSet) Method

```
public void acceptChanges(RowSet r)
     throws SQLException, OptimisticConflictException
```

Wrapper for RowSet.acceptChanges().

Exceptions

`SQLException`
`OptimisticConflictException`

---

getConnection() Method

```
public Connection getConnection()
     throws SQLException
```

Returns a database connection to the server associated with the control. It is typically not necessary to call this method when using the control.

Exceptions

`SQLException`

---

getDataSourceCalendar() Method

```
public Calendar getDataSourceCalendar()
```

Gets the Calendar instance used when setting and getting `Date`, `Time`, and `Timestamp` values. This is the Calendar set by the setDataSourceCalendar method.

Returns

The Calendar instance.

---

setDataSourceCalendar(Calendar) Method

```
public void setDataSourceCalendar(Calendar cal)
```

Sets the Calendar instance that should be used when setting and getting `Date`, `Time`, and `Timestamp` values.

Related Topics

```
java.sql.ResultSet#getDate(int, Calendar)
java.sql.ResultSet#getTime(int, Calendar)
java.sql.ResultSet#getTimestamp(int, Calendar)
java.sql.PreparedStatement#setDate(int, Date, Calendar)
java.sql.PreparedStatement#setTime(int, Time, Calendar)
java.sql.PreparedStatement#setTimestamp(int, Timestamp, Calendar)
```

---

com.bea.control

DatabaseControl.SQLFragment Class

public static class ***DatabaseControl.SQLFragment***

extends `Object`

Hierarchy

`Object`
`DatabaseControl.SQLFragment`

Enclosing interface

`DatabaseControl`

Field Summary

`protected List` ***`parameters`***
List

`protected CharSequence` ***`sql`***
CharSequence

Constructor Summary

DatabaseControl.SQLFragment*()*

DatabaseControl.SQLFragment*(`String` sql, `DatabaseControl.SQLParameter` parameters)*

Method Summary

`public DatabaseControl.SQLParameter[]` ***`getParameters`****()*
zero−based array (NOTE: ResultSet is one−based)

`public String` ***`getSQL`****()*

`public String`

toString*()*

Methods from class `java.lang.Object`

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

Field Detail

parameters

`protected List parameters`

---

sql

`protected CharSequence sql`

Constructor Detail

DatabaseControl.SQLFragment

`public DatabaseControl.SQLFragment()`

DatabaseControl.SQLFragment

`public DatabaseControl.SQLFragment(String` *`sql`*`,`
`DatabaseControl.SQLParameter[]` *`parameters`***`)`**

Method Detail

getParameters() Method

`public DatabaseControl.SQLParameter[] getParameters()`

zero−based array (NOTE: ResultSet is one−based)

---

getSQL() Method

`public String getSQL()`

---

toString() Method

```
public String toString()
```

Overrides

`Object.toString()`

com.bea.control

DatabaseControl.SQLParameter Class

public static class ***DatabaseControl.SQLParameter***

extends `Object`

Hierarchy

```
Object
 DatabaseControl.SQLParameter
```

Enclosing interface

`DatabaseControl`

Field Summary

public int	***dir***
public static final int	***IN***
public static final int	***INOUT***
public static final int	***OUT***
public int	***type***
public Object	***value***

Constructor Summary

DatabaseControl.SQLParameter*(*`Object` *value, int type, int dir)*

***DatabaseControl.SQLParameter**(`Object` value, int type)*

***DatabaseControl.SQLParameter**(`Object` value)*

Method Summary

`public Object` ***`clone`**()*

Methods from class `java.lang.Object`

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

Field Detail

dir

`public int dir`

IN

`public static final int IN`

INOUT

`public static final int INOUT`

OUT

`public static final int OUT`

---

type

`public int type`

---

value

`public Object value`

Constructor Detail

DatabaseControl.SQLParameter

```
public DatabaseControl.SQLParameter(Object value,
                                    int type,
                                    int dir)
```

DatabaseControl.SQLParameter

```
public DatabaseControl.SQLParameter(Object value,
                                    int type)
```

DatabaseControl.SQLParameter

```
public DatabaseControl.SQLParameter(Object value)
```

Method Detail

clone() Method

```
public Object clone()
```

Overrides

`Object.clone()`

com.bea.control

DatabaseFilter Class

public class ***DatabaseFilter***

extends `Object`
implements `Serializable`

The DatabaseFilter class is a helper class for SQL generation. NOTE: Depending on the database, column names may be case−sensitive. The most reliable way to avoid column name mistakes is to use setResultSetMetaData(). SQL Generation helper class

---

Hierarchy

`Object`
`DatabaseFilter`

All Implemented Interfaces

`Serializable`

Direct Known Subclasses

`DatabaseFilter`

Nested Class Summary

`public static class`	**`DatabaseFilter.FilterTerm`**
`public static class`	**`DatabaseFilter.SortTerm`**

Field Summary

`protected char`
`_chFilter`
char

`protected HashMap`
`_columns`
HashMap

`protected DatabaseFilter.FilterTerm`
`_filter`
DatabaseFilter.FilterTerm

`protected int`

_identifierOptions
int

protected int
_maxSortCols
int

protected DatabaseMetaData
_mdDatabase
DatabaseMetaData

protected ResultSetMetaData
_mdResultSet
ResultSetMetaData

protected String
_sIdentifierQuote
String

protected DatabaseFilter.SortTerm
_sort
DatabaseFilter.SortTerm

public static final int
IDENTIFIER_ASIS
int

public static final int
IDENTIFIER_CHANGECASE
int

public static final int
IDENTIFIER_DEFAULT
int

public static final int
IDENTIFIER_QUOTE
int

public static final int
IDENTIFIER_TOLOWER
int

public static final int
IDENTIFIER_TOUPPER
int

public static final int
opAsc
int

public static final int
opContains
int

public static final int
opDesc
int

public static final int
opEmpty
int

public static final int **opEqual**
int

public static final int **opGreater**
int

public static final int **opGreaterEqual**
int

public static final int **opIn**
int

public static final int **opInvalid**
int

public static final int **opIs**
int

public static final int **opIsNot**
int

public static final int **opLess**
int

public static final int **opLessEqual**
int

public static final int **opNotEqual**
int

public static final int **opStartsWith**
int

public static final String **sContains**
String

public static final String **sEmpty**
String

public static final String **sEquals**
String

public static final String **sFilterChar**
String

public static final String **sGreaterEqual**
String

public static final String **sGreaterThan**
String

public static final String **sIn**
String

public static final String **sIsEmpty**
String

public static final String **sIsNotEmpty**
String

public static final String **sLessEqual**
String

public static final String **sLessThan**
String

public static final String **sNotEqual**
String

public static final String **sStartsWith**
String

public static final String **sUnitDate**
String

public static final String **sUnitMonth**
String

public static final String **sUnitYear**
String

public static final int **unitDate**
int

public static final int **unitDefault**
int

public static final int **unitMonth**
int

public static final int **unitYear**
int

Constructor Summary

DatabaseFilter()

DatabaseFilter(`DatabaseFilter.FilterTerm` *filter*, `DatabaseFilter.SortTerm` *sort*)

Method Summary

public DatabaseControl.SQLFragment	***getFilterExpression****()*
public DatabaseControl.SQLFragment	***getOrderByClause****()*
public DatabaseControl.SQLFragment	***getSortExpression****()*
public DatabaseControl.SQLFragment	***getWhereClause****()*
public Calendar	***parseDate***(`CharSequence` *buf*, `Calendar` *calDefault*) Parse a date in ISO8601 or SOAP format.
public void	***parseQueryString***(`String` *query*)
public void	***parseQueryString***(`String` *query*, `String` *prefix*, `String` *enc*)
public void	***setDatabaseMetaData***(`DatabaseMetaData` *md*)
public void	***setIdentifierOptions***(*int flags*) control handling of identifiers in generated SQL.
public void	***setResultSetMetaData***(`ResultSetMetaData` *md*)

Methods from class `java.lang.Object`

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

Field Detail

_chFilter

```
protected char _chFilter
```

_columns

```
protected HashMap _columns
```

_filter

```
protected DatabaseFilter.FilterTerm _filter
```

_identifierOptions

```
protected int _identifierOptions
```

_maxSortCols

```
protected int _maxSortCols
```

_mdDatabase

```
protected DatabaseMetaData _mdDatabase
```

_mdResultSet

```
protected ResultSetMetaData _mdResultSet
```

_sIdentifierQuote

```
protected String _sIdentifierQuote
```

_sort

```
protected DatabaseFilter.SortTerm _sort
```

IDENTIFIER_ASIS

```
public static final int IDENTIFIER_ASIS
```

IDENTIFIER_CHANGECASE

```
public static final int IDENTIFIER_CHANGECASE
```

IDENTIFIER_DEFAULT

```
public static final int IDENTIFIER_DEFAULT
```

IDENTIFIER_QUOTE

```
public static final int IDENTIFIER_QUOTE
```

IDENTIFIER_TOLOWER

```
public static final int IDENTIFIER_TOLOWER
```

IDENTIFIER_TOUPPER

```
public static final int IDENTIFIER_TOUPPER
```

---

opAsc

```
public static final int opAsc
```

---

opContains

```
public static final int opContains
```

---

opDesc

```
public static final int opDesc
```

---

opEmpty

```
public static final int opEmpty
```

---

opEqual

```
public static final int opEqual
```

---

opGreater

```
public static final int opGreater
```

---

opGreaterEqual

```
public static final int opGreaterEqual
```

---

opIn

```
public static final int opIn
```

---

opInvalid

```
public static final int opInvalid
```

---

opIs

```
public static final int opIs
```

---

opIsNot

```
public static final int opIsNot
```

---

opLess

```
public static final int opLess
```

---

opLessEqual

```
public static final int opLessEqual
```

---

opNotEqual

```
public static final int opNotEqual
```

---

opStartsWith

```
public static final int opStartsWith
```

---

sContains

public static final String sContains

---

sEmpty

public static final String sEmpty

---

sEquals

public static final String sEquals

---

sFilterChar

public static final String sFilterChar

---

sGreaterEqual

public static final String sGreaterEqual

---

sGreaterThan

public static final String sGreaterThan

---

sIn

public static final String sIn

---

sIsEmpty

public static final String sIsEmpty

---

sIsNotEmpty

public static final String sIsNotEmpty

---

sLessEqual

public static final String sLessEqual

---

sLessThan

```
public static final String sLessThan
```

---

sNotEqual

```
public static final String sNotEqual
```

---

sStartsWith

```
public static final String sStartsWith
```

---

sUnitDate

```
public static final String sUnitDate
```

---

sUnitMonth

```
public static final String sUnitMonth
```

---

sUnitYear

```
public static final String sUnitYear
```

---

unitDate

```
public static final int unitDate
```

---

unitDefault

```
public static final int unitDefault
```

---

unitMonth

```
public static final int unitMonth
```

---

unitYear

```
public static final int unitYear
```

Constructor Detail

DatabaseFilter

```
public DatabaseFilter()
```

DatabaseFilter

```
public DatabaseFilter(DatabaseFilter.FilterTerm[] filter,
                      DatabaseFilter.SortTerm[] sort)
```

Method Detail

getFilterExpression() Method

```
public DatabaseControl.SQLFragment getFilterExpression()
```

---

getOrderByClause() Method

```
public DatabaseControl.SQLFragment getOrderByClause()
```

---

getSortExpression() Method

```
public DatabaseControl.SQLFragment getSortExpression()
```

---

getWhereClause() Method

```
public DatabaseControl.SQLFragment getWhereClause()
```

---

parseDate(CharSequence, Calendar) Method

```
public Calendar parseDate(CharSequence buf,
                          Calendar calDefault)
```

Parse a date in ISO8601 or SOAP format. `1999-05-31 13:20:00.000 1999-05-31T13:20:00Z 1999-05-31T13:20:00-05:00`

Returns

Calendar if the timezone is explicit (as in 2nd and 3rd) example the TimeZone of the returned calendar will be UTC. Otherwise the TimeZone will be as indicated by `calDefault` *if specified, or the system default otherwize.*

parseQueryString(String) Method

```
public void parseQueryString(String query)
```

parseQueryString(String, String, String) Method

```
public void parseQueryString(String query,
                             String prefix,
                             String enc)
     throws UnsupportedEncodingException
```

Exceptions

UnsupportedEncodingException

setDatabaseMetaData(DatabaseMetaData) Method

```
public void setDatabaseMetaData(DatabaseMetaData md)
```

setIdentifierOptions(int) Method

```
public void setIdentifierOptions(int flags)
```

control handling of identifiers in generated SQL. If setResultSetMetaData() is called IDENTIFIER_TOUPPER and IDENTIFIER_TOLOWER are ignored and the case will be as specificied by metadata.getColumnName()

setResultSetMetaData(ResultSetMetaData) Method

```
public void setResultSetMetaData(ResultSetMetaData md)
     throws SQLException
```

Exceptions

SQLException

com.bea.control

DatabaseFilter.FilterTerm Class

public static class ***DatabaseFilter.FilterTerm***

extends `DatabaseFilter.SortTerm`
implements `Serializable`

Hierarchy

```
Object
 DatabaseFilter.SortTerm
  DatabaseFilter.FilterTerm
```

All Implemented Interfaces

`Serializable`

Enclosing class

`DatabaseFilter`

Field Summary

`public int`
`unit`
int

`public Object`
`value`
Object

Fields from com.bea.control.`DatabaseFilter.SortTerm`

`op, sColumnName`

Constructor Summary

DatabaseFilter.FilterTerm()

Method Summary

Methods from class `java.lang.Object`

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

Field Detail

unit

`public int unit`

---

value

`public Object value`

Constructor Detail

DatabaseFilter.FilterTerm

`public DatabaseFilter.FilterTerm()`

com.bea.control

DatabaseFilter.SortTerm Class

public static class ***DatabaseFilter.SortTerm***

extends `Object`
implements `Serializable`

Hierarchy

`Object`
`DatabaseFilter.SortTerm`

All Implemented Interfaces

`Serializable`

Direct Known Subclasses

`DatabaseFilter.FilterTerm`

Enclosing class

`DatabaseFilter`

Field Summary

`public int`
op
int

`public String`
sColumnName
String

Constructor Summary

DatabaseFilter.SortTerm()

Method Summary

Methods from class `java.lang.Object`

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

Field Detail

op

`public int op`

---

sColumnName

`public String sColumnName`

Constructor Detail

DatabaseFilter.SortTerm

`public DatabaseFilter.SortTerm()`

com.bea.control

DefaultIssue Class

public class ***DefaultIssue***

extends `Object`
implements `Issue`

The DefaultIssue class provides a simple implementation of the Issue interface. Use this class as an alternative to implementing the Issue interface in your own class. Use this class for writing custom validators for controls. **Note:** The DefaultIssue class is provided as part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Hierarchy

```
Object
 DefaultIssue
```

All Implemented Interfaces

`Issue`

Constructor Summary

DefaultIssue*(*`String` *description,* `String` *prescription, boolean isError)*

Creates a new DefaultIssue instance using the specified description, prescription, and specifying whether the issue is an error.

DefaultIssue*(*`String` *description,* `String` *prescription)*

Creates a new DefaultIssue instance as an error, using the specified description and prescription.

DefaultIssue*(*`String` *description)*

Creates a new DefaultIssue instance as an error, using the specified description.

Method Summary

`public String` ***`getDescription`****()*
Returns to the IDE a description of this issue.

public ***getPrescription***()
String — Returns to the IDE a prescription for handling this issue.

public boolean ***isError***()
Notifies the IDE whether or not this issue is an error.

Methods from class `java.lang.Object`

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

Methods from interface `com.bea.control.Issue`

`getDescription, getPrescription, isError`

Constructor Detail

DefaultIssue

```
public DefaultIssue(String description,
                    String prescription,
                    boolean isError)
```

Creates a new DefaultIssue instance using the specified description, prescription, and specifying whether the issue is an error.

DefaultIssue

```
public DefaultIssue(String description,
                    String prescription)
```

Creates a new DefaultIssue instance as an error, using the specified description and prescription.

DefaultIssue

```
public DefaultIssue(String description)
```

Creates a new DefaultIssue instance as an error, using the specified description.

Method Detail

getDescription() Method

```
public String getDescription()
```

Returns to the IDE a description of this issue. This is the value specified in this instance's constructor.

Returns

A description of the current issue.

---

getPrescription() Method

```
public String getPrescription()
```

Returns to the IDE a prescription for handling this issue. This is the value specified in this instance's constructor.

Returns

A message describing how this issue can be solved or avoided.

---

isError() Method

```
public boolean isError()
```

Notifies the IDE whether or not this issue is an error. This is the value specified in this instance's constructor.

Returns

`true` if this issue is an error; `false` if it isn't.

EntityEJBControl Interface

public interface ***EntityEJBControl***

extends `EJBControl`

As part of the EJB control, this interface simplifies access to entity Enterprise JavaBeans (EJBs). You do not need to call methods of this interface. The EJB control is actually made up of two main interfaces, one for access to entity EJBs and another for access to session EJBs. The presence of these two interfaces is invisible when you use the EJB control; their methods are called behind the scenes. Typically, you use the EJB control by adding the control to a component design (such as a web service or pageflow design), then calling the methods it provides. Those methods are not exposed by these control interfaces, but rather are extensions of the EJB itself that are generated when you add the EJB control. For more information about using the EJB control, see EJB Control.

All Superinterfaces

`Control`, `Control`, `EJBControl`, `Serializable`

All Known Subinterfaces

`EntityEJBControl`

Method Summary

`public Object` ***`getEJBNextBeanInstance`****`()`*

Supports iteration through a Collection of entity bean instances returned by a multi−select finder method.

Methods from interface `weblogic.jws.control.EJBControl`

`getEJBBeanInstance, getEJBException, getEJBHomeInstance, getJNDIName, hasEJBBeanInstance, setJNDIName`

Method Detail

getEJBNextBeanInstance() Method

`public Object getEJBNextBeanInstance()`

Supports iteration through a Collection of entity bean instances returned by a multi−select finder method. This method selects the next bean instance in the collection as the internal control instance, and returns the bean instance. The method will return null if no additional instances remain to be processed.

Returns

The next bean instance if any remain; otherwise, null.

Extensible Interface

public interface ***Extensible***

A control implementation must implement the Extensible interface if it can be extended by a control extension (through a JCX file). Through a JCX file, the control interface may be customized at design time. Implementing this interface serves two purposes for a customizable control. First, it is a marker interface; a control must implement Extensible in order to be recognized as customizable. Second, you implement this interface's invoke method to provide the logic behind methods on the control's JCX file. **Note:** The Extensible interface is part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Method Summary

`public Object` ***`invoke`***`(Method method, Object[] args)`
Called by the run time to handle calls to methods of a customized control.

Method Detail

invoke(Method, Object[]) Method

```
public Object invoke(Method method,
                     Object[] args)
     throws Throwable
```

Called by the run time to handle calls to methods of a customized control. A customized control is represented in WebLogic Workshop as a JCX file. Methods on a JCX file represent an extension of the control's interface; calls to those methods are actually passed to the implementation of this invoke method. Implementations of invoke should anticipate the *method* parameter as a method of the control extension, with the *args* parameter as an array of the extention method's parameters. An implementation should return the expected return value.

Parameters

method
The JCX method that was called.

args
Parameters of the JCX method that was called.

Returns

The value that should be returned by the JCX method.

Exceptions

Throwable

com.bea.control

ExternalCallbackTarget Interface

public interface ***ExternalCallbackTarget***

The ExternalCallbackTarget interface is a marker interface that should be placed on any control implementation (JCS) class that can be the target of an asynchronous external callback. An external callback is an event generated from outside the Workshop runtime. An example of this type of callback is an asynchronous callback on a web service proxy. Declaring the interface indicates that the control expects to receive these events, and will also cause a top−level policy hook to be placed on any containing components so security for these callbacks can be configured. **Note:** The ExternalCallbackTarget interface is part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Issue Interface

public interface ***Issue***

Implement this interface to create an issue class that your control can return to have the IDE display errors or messages. The values you return from the getDescription and getPrescription methods will be displayed to the control's user in an IDE dialog. As an alternative implementing this interface in your own class, you can use the `DefaultIssue` class provided with this API. **Note:** The Issue interface is part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

All Known Implementing Classes

`DefaultIssue`

Method Summary

`public String` ***`getDescription`***`()`
Returns to the IDE a description of this issue.

`public String` ***`getPrescription`***`()`
Returns to the IDE a prescription for handling this issue.

`public boolean` ***`isError`***`()`
Notifies the IDE whether or not this issue is an error.

Method Detail

getDescription() Method

`public String getDescription()`

Returns to the IDE a description of this issue.

Returns

A description of the current issue.

---

getPrescription() Method

`public String getPrescription()`

Returns to the IDE a prescription for handling this issue.

Returns

A message describing how this issue can be solved or avoided.

---

isError() Method

```
public boolean isError()
```

Notifies the IDE whether or not this issue is an error. In general, warnings will be noted in an alert or in the build output, but will not stop processing; errors will require the user to address the problem before proceeding.

Returns

`true` *if this issue is an error;* `false` *if it isn't.*

com.bea.control

JMSControl Interface

public interface ***JMSControl***

extends XMLControl

Simplifies access to the Java Message Service. Use this control to send and receive JMS messages. Incoming messages are delivered asynchronously through callbacks. By default, JMS controls you add to your application will provide an interface that includes built−in and customizable members. You use the customizable members to send or publish messages, and to receive messages through callback handlers. For more information about using the EJB control, see JMS Control.

All Superinterfaces

`Control`, `Serializable`,

All Known Subinterfaces

`JMSControl`, `WliJMSControl`

Field Summary

`public static final String` ***HEADER_CORRELATIONID***
Indicates the JMSCorrelationID message header.

`public static final String` ***HEADER_DELIVERYMODE***
Indicates the JMSDeliveryMode message header.

`public static final String` ***HEADER_EXPIRATION***
Indicates the JMSExpiration message header.

`public static final String` ***HEADER_MESSAGEID***
Indicates the JMSMessageID message header.

`public static final String` ***HEADER_PRIORITY***
Indicates the JMSPriority message header.

`public static final String` ***HEADER_REDELIVERED***
Indicates the JMSRedelivered message header.

`public static final String` ***HEADER_TIMESTAMP***
Indicates the JMSTimestamp message header.

`public`

static final String **HEADER_TYPE**
Indicates the JMSType message header.

Method Summary

public Map **getHeaders**()
Gets the JMS headers of the last message received.

public Map **getProperties**()
Gets the JMS properties of the last message received.

public Session **getSession**()
Returns the JMS session used by `this` control.

public void **setHeaders**(Map *headers*)
Sets the JMS headers to be assigned to the next JMS message sent.

public void **setProperties**(Map *properties*)
Sets the JMS properties to be assigned to the next JMS message sent.

public void **subscribe**()
Indicates that this control is now interested in receiving incoming messages published to the topic.

public void **unsubscribe**()
Indicates that this control is no longer interested in receiving incoming messages published to the topic.

Field Detail

HEADER_CORRELATIONID

public static final String HEADER_CORRELATIONID

Indicates the JMSCorrelationID message header. Use with the getHeaders and setHeaders methods.

Related Topics

```
JMSControl.getHeaders()
JMSControl.setHeaders(Map)
```

HEADER_DELIVERYMODE

public static final String HEADER_DELIVERYMODE

Indicates the JMSDeliveryMode message header. Use with the getHeaders and setHeaders methods.

Related Topics

```
JMSControl.getHeaders()
JMSControl.setHeaders(Map)
```

HEADER_EXPIRATION

public static final String HEADER_EXPIRATION

Indicates the JMSExpiration message header. Use with the getHeaders and setHeaders methods.

Related Topics

```
JMSControl.getHeaders()
JMSControl.setHeaders(Map)
```

HEADER_MESSAGEID

public static final String HEADER_MESSAGEID

Indicates the JMSMessageID message header. Use with the getHeaders and setHeaders methods.

Related Topics

```
JMSControl.getHeaders()
JMSControl.setHeaders(Map)
```

HEADER_PRIORITY

public static final String HEADER_PRIORITY

Indicates the JMSPriority message header. Use with the getHeaders and setHeaders methods.

Related Topics

```
JMSControl.getHeaders()
JMSControl.setHeaders(Map)
```

HEADER_REDELIVERED

public static final String HEADER_REDELIVERED

Indicates the JMSRedelivered message header. Use with the getHeaders and setHeaders methods.

Related Topics

`JMSControl.getHeaders()`
`JMSControl.setHeaders(Map)`

HEADER_TIMESTAMP

`public static final String HEADER_TIMESTAMP`

Indicates the JMSTimestamp message header. Use with the getHeaders and setHeaders methods.

Related Topics

`JMSControl.getHeaders()`
`JMSControl.setHeaders(Map)`

HEADER_TYPE

`public static final String HEADER_TYPE`

Indicates the JMSType message header. Use with the getHeaders and setHeaders methods.

Related Topics

`JMSControl.getHeaders()`
`JMSControl.setHeaders(Map)`

Method Detail

getHeaders() Method

`public Map getHeaders()`

Gets the JMS headers of the last message received. If no message has been received then `null` is returned. The return value maps header names (Strings) to header values.

Returns

The headers of the last message received; `null` if no message has been received.

---

getProperties() Method

`public Map getProperties()`

Gets the JMS properties of the last message received. If no message has been received then `null` is returned. The return value maps property names (Strings) to property values.

Returns

The properties of the last message received; `null` if no message has been received.

---

getSession() Method

`public Session getSession()`

Returns the JMS session used by `this` control.

Returns

The session object.

---

setHeaders(Map) Method

`public void setHeaders(Map` *`headers`***`)`**

Sets the JMS headers to be assigned to the next JMS message sent. Note that these headers are set only on the next message, subsequent messages will not get these headers. Also note that if the next message is sent through a publish method, then any header set through this map will override headers set in the message itself.

Parameters

headers
: A map of header names (Strings) to header values.

---

setProperties(Map) Method

`public void setProperties(Map` *`properties`***`)`**

Sets the JMS properties to be assigned to the next JMS message sent. Note that these properties are set only on the next message, subsequent messages will not get these properties. Also note that if the next message is sent through a publish method, then any property set through this map will override properties set in the message itself.

Parameters

properties
: A map of property names (Strings) to property values.

---

subscribe() Method

`public void subscribe()`

Indicates that this control is now interested in receiving incoming messages published to the topic. Note that

when the control is first created it will not receive messages on the topic until it is explicitly requested so. Also note that this method applies only to controls which listen to topics. Invoking this method on a control that does not listen on a topic has no effect.

---

unsubscribe() Method

```
public void unsubscribe()
```

Indicates that this control is no longer interested in receiving incoming messages published to the topic. This method cancels a subscription previously registered by `JMSControl.subscribe()`. Note that this method applies only to controls which listen to topics. Invoking this method on a control that does not listen on a topic has no effect.

com.bea.control

JwsContext Interface

public interface ***JwsContext***

extends `Context`

Provides access to container services that support web services (JWS files). Represents the execution context of the web service. Methods in this interface can be used to access out−of−band data for communication with other web service architectures and to manage conversations. For more information on building web services, see Building Web Services.

Related Topics

`Context`

All Superinterfaces

`Context`, `Serializable`

All Known Subinterfaces

`JpdContext`, `JwsContext`

Nested Class Summary

public static interface	**`JwsContext.Callback`** The Callback interface defines events that can be received through the `Context` object.

Nested classes from interface weblogic.jws.control.`Context`

`Context.Callback`

Method Summary

public String	***`getCallbackLocation`****()* Retrieves the URL set by a setCallbackLocation method, or retrieves the callback URL if it was set via SOAP headers.
public String	***`getCallbackPassword`****()* Gets the password used for credentials in callbacks.

public ***getCallbackUsername()***
String — Gets the user name used for credentials in callbacks.

public Element[] ***getInputHeaders()***
Returns the SOAP headers that arrived with the current method invocation message.

public Protocol ***getProtocol()***
Gets the protocol of the current request.

public boolean ***getUnderstoodInputHeaders()***
Returns the value most recently set by a call to `setUnderstoodInputHeaders`.

public void ***setCallbackLocation****(String url)*
Specifies the URL to which a web service callback should be sent.

public void ***setCallbackLocation****(URL url)*
Specifies the URL to which a web service callback should be sent.

public void ***setCallbackPassword****(String password)*
Sets the password to use for credentials in callbacks.

public void ***setCallbackUsername****(String username)*
Sets the user name to use for credentials in callbacks.

public void ***setOutputHeaders****(Element[] headers)*
Set the SOAP headers to be sent with outgoing messages to the client.

public void ***setUnderstoodInputHeaders****(boolean understood)*
Indicates whether input headers were understood.

Methods from interface `weblogic.jws.control.Context`

`finishConversation, getCallerPrincipal, getCurrentAge, getCurrentIdleTime, getLogger, getMaxAge, getMaxIdleTime, getService, isCallerInRole, isFinished, resetIdleTime, setMaxAge, setMaxAge, setMaxIdleTime, setMaxIdleTime`

Method Detail

getCallbackLocation() Method

DEPRECATED Supported for backward compatibility only; as of WebLogic Platform 8.1, replaced by `com.bea.control.ServiceControl.getEndPoint()`.

`public String getCallbackLocation()`

Retrieves the URL set by a setCallbackLocation method, or retrieves the callback URL if it was set via SOAP headers. You can call getCallbackLocation() to retrieve the URL set with JwsContext.setCallbackLocation, or

to discover the URL if it was set via SOAP headers when a conversation start method was called. Deprecated: JWS callback interfaces should extend ServiceControl, and then ServiceControl.getEndpoint() can be called on a callback instance to set the callback destination.

Returns

The URL to which callbacks should be sent.

---

getCallbackPassword() Method

DEPRECATED Supported for backward compatibility only; as of WebLogic Platform 8.1, replaced by `com.bea.control.ServiceControl.getPassword()`.

```
public String getCallbackPassword()
```

Gets the password used for credentials in callbacks. The password may have been specified by the client as part of the callbackLocation conversation start SOAP header, or it may have been set by a call to JwsContext.setCallbackPassword(). Deprecated: JWS callback interfaces should extend ServiceControl, and then call ServiceControl.getPassword() on a callback instance.

Returns

The password that will be used for callbacks to the client in the current conversation.

---

getCallbackUsername() Method

DEPRECATED Supported for backward compatibility only; as of WebLogic Platform 8.1, replaced by `com.bea.control.ServiceControl.getUsername()`.

```
public String getCallbackUsername()
```

Gets the user name used for credentials in callbacks. The username may have been specified by the client as part of the callbackLocation conversation start SOAP header, or it may have been set by a call to JwsContext.setCallbackUsername(). Deprecated: JWS callback interfaces should extend ServiceControl, and then call ServiceControl.getUsername() on a callback instance.

Returns

The username that will be used for callbacks to the client in the current conversation.

---

getInputHeaders() Method

```
public Element[] getInputHeaders()
```

Returns the SOAP headers that arrived with the current method invocation message. The SOAP headers used by WebLogic Workshop to manage conversations are included in the list of headers returned.

Returns

An array of org.w3c.dom.Element objects containing the SOAP headers that arrived with the current method invocation.

---

getProtocol() Method

```
public Protocol getProtocol()
```

Gets the protocol of the current request.

Returns

The protocol of the current request.

---

getUnderstoodInputHeaders() Method

```
public boolean getUnderstoodInputHeaders()
```

Returns the value most recently set by a call to `setUnderstoodInputHeaders`.

Returns

The value most recently set by a call to JwsContext.setUnderstoodInputHeaders(); `false` if JwsContext.setUnderstoodInputHeaders() has not been called.

---

setCallbackLocation(String) Method

DEPRECATED Supported for backward compatibility only; as of WebLogic Platform 8.1, replaced by `com.bea.control.ServiceControl.setEndPoint(java.net.URL)`.

```
public void setCallbackLocation(String url)
     throws MalformedURLException
```

Specifies the URL to which a web service callback should be sent. Call this method before sending callbacks for conversations with clients that do not implicitly provide callback information via SOAP headers. You may provide credentials as part of the URL (if supported by the scheme) or by using the setCallbackUsername and setCallbackPassword methods. The callback location is typically provided by the client in the callbackLocation conversation start SOAP header, but may be set explicitly using this method. The callback location must be set before attempting to send a callback to the client. JwsContext.getCallbackLocation() may be used to retrieve the URL set by this method or to discover the callback URL if it was set via the SOAP header. Credentials may be provided as part of the URL (if supported by the scheme) or by using the JwsContext.setCallbackUsername() and JwsContext.setCallbackPassword() methods. Deprecated: JWS callback interfaces should extend ServiceControl, and then ServiceControl.setEndpoint(URL) can be called on a callback instance to set the callback destination.

Parameters

url
The URL to which callbacks should be sent.

Exceptions

`MalformedURLException`
If the URL did not specify a protocol, or if it could not be parsed.

---

setCallbackLocation(URL) Method

DEPRECATED Supported for backward compatibility only; as of WebLogic Platform 8.1, replaced by `com.bea.control.ServiceControl.setEndPoint(java.net.URL)`.

`public void setCallbackLocation(URL` *`url`***`)`**

Specifies the URL to which a web service callback should be sent. Call this method before sending callbacks for conversations with clients that do not implicitly provide callback information via SOAP headers. You may provide credentials as part of the URL (if supported by the scheme) or by using the setCallbackUsername and setCallbackPassword methods. The callback location is typically provided by the client in the callbackLocation conversation start SOAP header, but may be set explicitly using this method. The callback location must be set before attempting to send a callback to the client. JwsContext.getCallbackLocation() may be used to retrieve the URL set by this method or to discover the callback URL if it was set via the SOAP header. Credentials may be provided as part of the URL (if supported by the scheme) or by using the JwsContext.setCallbackUsername() and JwsContext.setCallbackPassword() methods. Deprecated: JWS callback interfaces should extend ServiceControl, and then ServiceControl.setEndpoint(URL) can be called on a callback instance to set the callback destination.

Parameters

url
The URL to which callbacks should be sent.

---

setCallbackPassword(String) Method

DEPRECATED Supported for backward compatibility only; as of WebLogic Platform 8.1, replaced by `com.bea.control.ServiceControl.setPassword(java.lang.String)`.

`public void setCallbackPassword(String` *`password`***`)`**

Sets the password to use for credentials in callbacks. Use the setCallbackUsername and setCallbackPassword methods if the callback's recipient is secure, and if no credentials were specified in the callback URL. Overrides the current password if it was specified by the client as part of the callbackLocation conversation start SOAP header. Deprecated: JWS callback interfaces should extend ServiceControl, and then call ServiceControl.setPassword() on a callback instance.

Parameters

password
The password to use in credentials.

---

setCallbackUsername(String) Method

DEPRECATED Supported for backward compatibility only; as of WebLogic Platform 8.1, replaced by `com.bea.control.ServiceControl.setUsername(java.lang.String)`.

`public void setCallbackUsername(String` *`username`***`)`**

Sets the user name to use for credentials in callbacks. Use the setCallbackUsername and setCallbackPassword methods if the callback's recipient is secure, and if no credentials were specified in the callback URL. Overrides the current username if it was specified by the client as part of the callbackLocation conversation start SOAP header. Deprecated: JWS callback interfaces should extend ServiceControl, and then call ServiceControl.setUsername() on a callback instance.

Parameters

username
The user name to use in credentials.

---

setOutputHeaders(Element[]) Method

`public void setOutputHeaders(Element[]` *`headers`***`)`**

Set the SOAP headers to be sent with outgoing messages to the client.

Parameters

headers
An array of org.w3c.dom.Element objects containing valid SOAP headers.

---

setUnderstoodInputHeaders(boolean) Method

`public void setUnderstoodInputHeaders(boolean` *`understood`***`)`**

Indicates whether input headers were understood. If any input headers are marked with the mustUnderstand attribute value of "1" or "true" and this method is not called with the value `true`, then a SOAP fault will be generated.

Parameters

understood
`true` to indicate that "mustUnderstand" headers were understood; `false` to indicate they weren't.

JwsContext.Callback Interface

public static interface ***JwsContext.Callback***

extends `Context.Callback`

The Callback interface defines events that can be received through the `Context` object. See `Context.Callback` for more information about these events.

All Superinterfaces

`Context.Callback`

All Known Subinterfaces

`JpdContext.Callback`, `JwsContext.Callback`

Enclosing interface

`JwsContext`

Method Summary

Methods from interface `weblogic.jws.control.Context.Callback`

`onAgeTimeout, onAsyncFailure, onCreate, onException, onFinish, onIdleTimeout`

com.bea.control

ServiceControl Interface

public interface ***ServiceControl***

extends Asynchronous, ServiceProxy, XMLControl

Provides simplified access to web services. A Service control provides an interface between your application and a web service, which allows your application to invoke the methods and handle the callbacks of that web service. Using a Web Service control, you can connect to any web service for which a WSDL file is available, whether or not it was built using WebLogic Workshop. You typically use a Service control by creating the control from a WSDL file, or from a web service you created with WebLogic Workshop, then add the control to a design in your application. The target web service's operations are exposed as methods of the control. For more information on using the Service control, see Web Service Control.

All Superinterfaces

`Control`, `Serializable`,

All Known Subinterfaces

`B2BCallback`, `B2BCallbackControl`, `ExtendedServiceControl`, `ServiceBrokerControl`, `ServiceControl`

Nested Class Summary

`public static interface`	**`ServiceControl.Callback`** Provides a way for the Service control to pass callback events to its clients.

Method Summary

`public String`	***`getConversationID`****`()`* Retrieves the conversation ID of the current conversation with this Service control instance.
`public URL`	***`getEndPoint`****`()`* Gets the callback URL that the Service control instance will use as the base URL for callback invocations.
`public Element[]`	***`getInputHeaders`****`()`* Retrieves the SOAP headers that were included in the most recent arriving callback from this Service control.

public String **getPassword**()
Retrieves the password string that was set by the most recent call to the setPassword method.

public Protocol **getProtocol**()
Returns a Protocol object representing the protocol to use when sending messages to the target web service.

public String **getReliableMessageID**()
Retrieves the unique ID that will be used when invoking a Service control method in the context of reliable messaging.

public String **getUsername**()
Retrieves the username string that was set by the most recent call to setUsername.

public void **reset**()
Clears all parameters that were set by previous calls to the setConversationID, setOutputHeaders, setPassword, or setUsername methods.

public void **setClientCert**(`String` *alias*, `String` *password*)
Sets the client certificate alias and password when using client certificates with Secure Sockets Layer (SSL).

public void **setConversationID**(`String` *conversationID*)
Sets the unique key that will be proposed as the conversation ID when initiating a conversation with the Web Service control.

public void **setEndPoint**(*URL url*)
Sets the callback URL that the Service control instance will use as the base URL for callback invocations.

public void **setKeystore**(`String` *location*, `String` *password*, `String` *type*)
Specifies the keystore information (including type) to use when using client−certificates with Secure Sockets Layer (SSL).

public void **setKeystore**(`String` *location*, `String` *password*)
Specifies the keystore information to use when using client−certificates with Secure Sockets Layer (SSL).

public void **setOutputHeaders**(`Element[]` *headers*)
Sets the SOAP headers that will be included in the next outgoing method invocation message to the Service control.

public void **setPassword**(`String` *password*)
Sets the password that will be sent with the next outgoing Service control method invocation.

public void **setProtocol**(`Protocol` *protocol*)

Specifies the protocol to use for messages sent to the target web service.

public void ***setReliableMessageID***(String *messageID)*
Sets the message ID to use for reliable messaging.

public void ***setUsername***(String *username)*
Sets the username that will be sent with the next outgoing Service control method invocation.

public void ***useClientKeySSL***(*boolean b)*
Specifies whether a client certificate should be used with Secure Sockets Layer (SSL).

Method Detail

getConversationID() Method

public String getConversationID()

Retrieves the conversation ID of the current conversation with this Service control instance.

Returns

The conversation ID for this Service control's current conversation.

---

getEndPoint() Method

public URL getEndPoint()

Gets the callback URL that the Service control instance will use as the base URL for callback invocations.

Returns

The callback URL that will be used.

---

getInputHeaders() Method

public Element[] getInputHeaders()

Retrieves the SOAP headers that were included in the most recent arriving callback from this Service control.

Returns

An array of the SOAP input header elements for this control's most recently receive callback.

---

getPassword() Method

```
public String getPassword()
```

Retrieves the password string that was set by the most recent call to the setPassword method.

Returns

The password set by the setPassword method.

---

getProtocol() Method

```
public Protocol getProtocol()
```

Returns a Protocol object representing the protocol to use when sending messages to the target web service.

Returns

The protocol that will be used.

---

getReliableMessageID() Method

```
public String getReliableMessageID()
```

Retrieves the unique ID that will be used when invoking a Service control method in the context of reliable messaging.

Returns

The message ID.

---

getUsername() Method

```
public String getUsername()
```

Retrieves the username string that was set by the most recent call to setUsername.

Returns

The username set by the setUsername method.

---

reset() Method

```
public void reset()
```

Clears all parameters that were set by previous calls to the setConversationID, setOutputHeaders,

setPassword, or setUsername methods.

setClientCert(String, String) Method

```
public void setClientCert(String alias,
                          String password)
```

Sets the client certificate alias and password when using client certificates with Secure Sockets Layer (SSL).

Parameters

alias
The client certficate alias.

password
The client certificate password.

setConversationID(String) Method

```
public void setConversationID(String conversationID)
```

Sets the unique key that will be proposed as the conversation ID when initiating a conversation with the Web Service control. Note that WebLogic Workshop automatically computes a conversation ID when a WebLogic web service invokes a start method of a Service control. Use the setConversationID method to override the automatic value. The only case where it is useful to do so is if you supply the conversation ID of an existing conversation that is currently ongoing on the target web service. You may then invoke methods on the target service that will execute in the context of the specified conversation. However, only the client that originated the conversation may receive callbacks.

Parameters

conversationID
The new value for the conversation ID.

setEndPoint(URL) Method

```
public void setEndPoint(URL url)
```

Sets the callback URL that the Service control instance will use as the base URL for callback invocations. While this is set automatically by WebLogic Workshop, you can use this method to override the callback URL if you wish callbacks to be sent to a different destination.

Parameters

url
The new destination for callbacks.

setKeystore(String, String, String) Method

```
public void setKeystore(String location,
                        String password,
                        String type)
```

Specifies the keystore information (including type) to use when using client−certificates with Secure Sockets Layer (SSL). The default keystore is the WebLogic Server system−identity keystore. Use the *type* parameters to specify a keystore type other than the default, which is Java KeyStore (JKS).

Parameters

location
The path to the keystore (JKS) file.

password
The password for the keystore.

type
The type of keystore to use.

---

setKeystore(String, String) Method

```
public void setKeystore(String location,
                        String password)
```

Specifies the keystore information to use when using client−certificates with Secure Sockets Layer (SSL). The keystore type is implied to be "JKS".

Parameters

location
The path to the keystore (JKS) file.

password
The password for the keystore.

---

setOutputHeaders(Element[]) Method

```
public void setOutputHeaders(Element[] headers)
```

Sets the SOAP headers that will be included in the next outgoing method invocation message to the Service control.

Parameters

headers
An array of the new SOAP output header elements.

---

setPassword(String) Method

public void setPassword(String *password***)**

Sets the password that will be sent with the next outgoing Service control method invocation. Used if the Service control uses HTTP basic authentication.

Parameters

password
The password to send for authentication.

---

setProtocol(Protocol) Method

public void setProtocol(Protocol *protocol***)**

Specifies the protocol to use for messages sent to the target web service.

Parameters

protocol
The protocol to use.

---

setReliableMessageID(String) Method

public void setReliableMessageID(String *messageID***)**

Sets the message ID to use for reliable messaging. Use this method to set a unique ID for each invocation on the Service control. This ID can then be used to identify individual method invocations if message delivery fails and the Service control receives the onDeliveryFailure callback. The ID will be received with the callback. Note that this method sets a message ID for reliable messaging only.

Parameters

messageID
The unique ID to use.

---

setUsername(String) Method

public void setUsername(String *username***)**

Sets the username that will be sent with the next outgoing Service control method invocation. Used if the Service control uses HTTP basic authentication.

Parameters

username
The username to send for authentication.

---

useClientKeySSL(boolean) Method

`public void useClientKeySSL(boolean` *`b`***`)`**

Specifies whether a client certificate should be used with Secure Sockets Layer (SSL).

Parameters

b
`true` to use a client certificate; otherwise, `false`.

ServiceControl.Callback Interface

public static interface ***ServiceControl.Callback***

Provides a way for the Service control to pass callback events to its clients.

Enclosing interface

`ServiceControl`

Method Summary

`public`
`void` ***`onAsyncFailure`***`(String` *methodName,* `Object[]` *argv)*
Received when a service control's JCX file has a method marked with the annotation `message-buffer enable="true"`, and an exception occurs when trying to call out.

`public`
`void` ***`onDeliveryFailure`***`(String` *messageID,* `String` *methodName,* `String` *faultCode,* `String` *faultDetail)*
Received on failure of a method's attempt to send a message for reliable delivery.

Method Detail

onAsyncFailure(String, Object[]) Method

```
public void onAsyncFailure(String methodName,
                           Object[] argv)
```

Received when a service control's JCX file has a method marked with the annotation `message-buffer enable="true"`, and an exception occurs when trying to call out.

Parameters

methodName
The Service control method from which the exception was thrown.

argv
The Service control method's parameters.

---

onDeliveryFailure(String, String, String, String) Method

```
public void onDeliveryFailure(String messageID,
                              String methodName,
                              String faultCode,
                              String faultDetail)
```

Received on failure of a method's attempt to send a message for reliable delivery.

Parameters

messageID
The ReliableMessageID property

methodName
The method that attempted to deliver the message reliably.

faultCode
The fault code associated with the failure.

faultDetail
Details about the fault associated with the failure.

com.bea.control

ServiceControlException Class

public class ***ServiceControlException***

extends `ControlException`

The ServiceControlException class can be used to throw exceptions from a web service control.

Related Topics

`ControlException`

Hierarchy

```
Object
 Throwable
  Exception
   RuntimeException
    ControlException
     ControlException
      ServiceControlException
```

All Implemented Interfaces

`Serializable`

Direct Known Subclasses

`ServiceControlException`

Constructor Summary

ServiceControlException*(*`String` *msg,* `Throwable` *t)*

ServiceControlException*(*`String` *msg)*

Method Summary

`public SoapFault` ***`getSoapFault`****()*

public boolean ***hasSoapFault****()*

Methods from `com.bea.control.ControlException`

`getNestedException`

Methods from `java.lang.Throwable`

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

Methods from class `java.lang.Object`

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

Constructor Detail

ServiceControlException

```
public ServiceControlException(String msg,
                               Throwable t)
```

ServiceControlException

```
public ServiceControlException(String msg)
```

Method Detail

getSoapFault() Method

```
public SoapFault getSoapFault()
```

Related Topics

`SoapFault`

---

hasSoapFault() Method

```
public boolean hasSoapFault()
```

---

com.bea.control

SessionEJBControl Interface

public interface ***SessionEJBControl***

extends `EJBControl`

As part of the EJB control, this interface simplifies access to session Enterprise JavaBeans (EJBs). You do not need to use this interface directly. The EJB control is actually made up of two main interfaces, one for access to entity EJBs and another for access to session EJBs. The presence of these two interfaces is invisible when you use the EJB control; their methods are called behind the scenes. Typically, you use the EJB control by adding the control to a component design (such as a web service or pageflow design), then calling the methods it provides. Those methods are not exposed by these control interfaces, but rather are extensions of the EJB itself that are generated when you add the EJB control. For more information about using the EJB control, see EJB Control.

All Superinterfaces

`Control`, `Control`, `EJBControl`, `Serializable`

All Known Subinterfaces

`SessionEJBControl`

Method Summary

Methods from interface `weblogic.jws.control.EJBControl`

`getEJBBeanInstance, getEJBException, getEJBHomeInstance, getJNDIName, hasEJBBeanInstance, setJNDIName`

com.bea.control

SoapFault Class

public class ***SoapFault***

extends `Object`
implements `Serializable`

Web service developers can use the SoapFault class to control the shape of the detail of SOAP fault data.

Hierarchy

```
Object
 SoapFault
```

All Implemented Interfaces

`Serializable`

Constructor Summary

SoapFault*(`XmlObject` fault)*

Method Summary

`public Fault` ***`get11Fault`****()*

`public Fault` ***`get12Fault`****()*

`public XmlObject` ***`getDetail`****()*
Returns the entire content of Detail element.

`public XmlObject[]` ***`getDetailContents`****()*
Returns an array of XmlObjects that are the **immediate** children of the SOAP Fault Detail element.

`public boolean` ***`hasDetail`****()*

`public boolean` ***`isSoap11`****()*

`public boolean` ***`isSoap12`****()*

Methods from class `java.lang.Object`

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

Constructor Detail

SoapFault

`public SoapFault(XmlObject` *`fault`*`)`

Method Detail

get11Fault() Method

`public Fault get11Fault()`

---

get12Fault() Method

`public Fault get12Fault()`

---

getDetail() Method

`public XmlObject getDetail()`

Returns the entire content of Detail element. Clients can use the XmlCursor API to extract complex content from the Detail, including attributes.

Returns

the SOAP Fault Detail content

---

getDetailContents() Method

`public XmlObject[] getDetailContents()`

Returns an array of XmlObjects that are the **immediate** children of the SOAP Fault Detail element. If the schema for the elements has been compiled as XBeans, the XmlObjects can be casted to the Java type associated with the XML Schema type. If there are no immediate children, then an empty XmlObject array is returned.

Returns

the immediate children of the SOAP Faul Detail element

---

hasDetail() Method

```
public boolean hasDetail()
```

---

isSoap11() Method

```
public boolean isSoap11()
```

---

isSoap12() Method

```
public boolean isSoap12()
```

TimerControl Interface

public interface ***TimerControl***

extends Asynchronous, `Control`

Notifies your application when a specified period of time has elapsed or when a specified absolute time has been reached. When using the TimerControl, you typically begin by setting its onTimeout attribute to the amount of time that should pass from the time the timer starts to the time its onTimeout callback fires. You then write code in an onTimeout callback handler; this code will execute when the timer expires. You can also specify that the timer should continue to fire its onTimeout callback at a specific interval. For more information about using the EJB control, see Timer Control.

All Superinterfaces

`Control`, `Control`, `Serializable`

All Known Subinterfaces

`TimerControl`

Nested Class Summary

`public static interface`	**`TimerControl.Callback`** Provides a way for the Timer control to receive callbacks.

Method Summary

`public boolean`	***`getCoalesceEvents`****()* Returns the current value of the coalesce−events attribute.
`public String`	***`getRepeatsEvery`****()* Returns the current interval specified by the repeats−every attribute of the @jc:timer tag or the most recent call to the setRepeatsEvery method.
`public long`	***`getTimeout`****()* Returns the current duration specified by the timeout attribute of the @jc:timer tag of the most recent call to the setTimeout method.
`public Date`	***`getTimeoutAt`****()* Returns the time at which the timer is next scheduled to fire, if the control's

repeats−every attribute is set to a value greater than zero.

public void ***restart****()*
Resets the timer.

public void ***setCoalesceEvents****(boolean coalesce)*
Enables or disables the coalesce−events behavior.

public void ***setRepeatsEvery****(long seconds)*
Sets the repeat interval for the timer using seconds since the epoch.

public void ***setRepeatsEvery****(String interval)*
Sets the repeat interval using an xsd:duration string.

public void ***setTimeout****(long seconds)*
Sets the time between start or restart and the first expiration of the timer, in seconds.

public void ***setTimeout****(String delay)*
Sets the time between start or restart and the first expiration of the timer, as an xsd:duration string.

public void ***setTimeoutAt****(Date time)*
Sets the absolute date and time at which the timer will expire the first time after being started or restarted.

public void ***start****()*
Starts the timer.

public void ***stop****()*
Stops the timer.

Method Detail

getCoalesceEvents() Method

public boolean getCoalesceEvents()

Returns the current value of the coalesce−events attribute.

Returns

The current value of the coalesce−events attribute.

---

getRepeatsEvery() Method

public String getRepeatsEvery()

Returns the current interval specified by the repeats−every attribute of the @jc:timer tag or the most recent call to the setRepeatsEvery method.

Returns

The current repetition interval.

---

getTimeout() Method

```
public long getTimeout()
```

Returns the current duration specified by the timeout attribute of the @jc:timer tag of the most recent call to the setTimeout method.

Returns

The current timeout value.

---

getTimeoutAt() Method

```
public Date getTimeoutAt()
```

Returns the time at which the timer is next scheduled to fire, if the control's repeats−every attribute is set to a value greater than zero. If the repeats−every attribute is set to zero, then the getTimeoutAt method returns the value set by the setTimeoutAt method or the value set in the timeout attribute.

Returns

The time at which the timer is next scheduled to fire its onTimeout callback.

---

restart() Method

```
public void restart()
```

Resets the timer. Any pending events are canceled. The timer will subsequently expire after the repeats−every period has elapsed after this call.

---

setCoalesceEvents(boolean) Method

```
public void setCoalesceEvents(boolean coalesce)
```

Enables or disables the coalesce−events behavior.

Parameters

coalesce
`true` to coalesce events; otherwise, `false`.

---

setRepeatsEvery(long) Method

```
public void setRepeatsEvery(long seconds)
     throws IllegalArgumentException
```

Sets the repeat interval for the timer using seconds since the epoch.

Parameters

seconds
The repetition interval after which the onTimeout callback should fire after its first expiration.

Exceptions

`IllegalArgumentException`

---

setRepeatsEvery(String) Method

```
public void setRepeatsEvery(String interval)
     throws IllegalArgumentException
```

Sets the repeat interval using an xsd:duration string.

Parameters

interval
The repetition interval after which the onTimeout callback should fire after its first expiration.

Exceptions

`IllegalArgumentException`

---

setTimeout(long) Method

```
public void setTimeout(long seconds)
     throws IllegalArgumentException
```

Sets the time between start or restart and the first expiration of the timer, in seconds.

Parameters

seconds
The duration after which the timer should expire.

Exceptions

IllegalArgumentException

---

setTimeout(String) Method

```
public void setTimeout(String delay)
     throws IllegalArgumentException
```

Sets the time between start or restart and the first expiration of the timer, as an xsd:duration string.

Parameters

delay
The duration after which the timer should expire.

Exceptions

IllegalArgumentException

---

setTimeoutAt(Date) Method

```
public void setTimeoutAt(Date time)
```

Sets the absolute date and time at which the timer will expire the first time after being started or restarted.

Parameters

time
The date and time at which the timer should first expire after being started or restarted.

---

start() Method

```
public void start()
     throws IllegalStateException
```

Starts the timer. The first timer expiration will occur after the period specified by the timeout attribute has elapsed.

Exceptions

IllegalStateException

---

stop() Method

```
public void stop()
```

Stops the timer. No further timer expiration callbacks will be invoked.

TimerControl.Callback Interface

public static interface ***TimerControl.Callback***

Provides a way for the Timer control to receive callbacks.

Enclosing interface

`TimerControl`

Method Summary

`public`
`void` ***`onTimeout`****`(long time)`*
Received when the timer expires.

Method Detail

onTimeout(long) Method

`public void onTimeout(long` *`time`***`)`**

Received when the timer expires. The time at which the timer expired is passed as the *time* parameter. Note that this may be some time in the past if the `onTimeout` callback could not be invoked due to high system load.

Parameters

time
The time at which the timer expired.

com.bea.control

TimerControlFactory Interface

public interface ***TimerControlFactory***

extends `ControlFactory`

The base factory interface for the TimerControl.

All Superinterfaces

`ControlFactory`, `ControlFactory`, `Serializable`

All Known Subinterfaces

`TimerControlFactory`

Method Summary

`public TimerControl` ***`create`****`()`*

Method Detail

create() Method

`public TimerControl create()`

ValidateAttribute Interface

public interface ***ValidateAttribute***

The ValidateAttribute interface provides methods with which you can validate a property tag attribute. This will be used by the WebLogic Workshop compiler to show errors in source for an attribute of custom type, and by the IDE property view for any attribute. To expose your implementation to the IDE, you specify the implementation with the class−name attribute of the <custom> element in the control's tags XML file. To use it in the property view, an instance of this class should be returned when appropriate by the EditorSupport.getBehavior() call for your control. **Note:** The ValidateAttribute interface is part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Method Summary

`public Issue[]` ***`validateDuringCompile`***`(String` *attributeType,* `String` *value,* `Map` *context)*

Provides a way for property attributes to be validated during compilation, and for you to display specific information in Source View for invalid attribute values.

`public Issue[]` ***`validateDuringEdit`***`(String` *attributeType,* `String` *value)*

Provides a way for property attributes to be validated while they are being edited.

Method Detail

validateDuringCompile(String, String, Map) Method

```
public Issue[] validateDuringCompile(String attributeType,
                                     String value,
                                     Map context)
```

Provides a way for property attributes to be validated during compilation, and for you to display specific information in Source View for invalid attribute values. This method is called by the IDE under many circumstances, including attempts by the control's user to build a component that contains the control. Invalid attributes will be displayed in Source View with a red underline.

Parameters

attributeType

The name of the attribute being validated.

value

The attribute's current value.

context
State that the control author can use to accumulate information during

Returns

An array of issues. If there are no issues the array should be empty.

---

validateDuringEdit(String, String) Method

```
public Issue[] validateDuringEdit(String attributeType,
                                  String value)
```

Provides a way for property attributes to be validated while they are being edited. In particular, this method is called by the IDE when the control's user edits the specified attribute's value and attempts to save the value by navigating past the value in the Property Editor. You can also call this method from your own custom property editing dialog.

Parameters

attributeType
The name of the attribute being validated.

value
The attribute's current value.

Returns

An array of issues. If there are no issues the array should be empty.

com.bea.control

ValidateControl Interface

public interface ***ValidateControl***

The ValidateControl interface provides methods with which you can validate a control JCX file. The WebLogic Workshop IDE calls these methods in your implementation of this interface. To expose your implementation to the IDE, you specify the implementation with the control−validator attribute of the <control−tags> element in the control's tags XML file. **Note:** The ValidateControl interface is part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Method Summary

`public Issue[]` ***`validateDuringCompile`****(boolean start,* `Map` *context)*

`public Issue[]` ***`validateDuringEdit`****(boolean start)*
Not yet supported.

Method Detail

validateDuringCompile(boolean, Map) Method

```
public Issue[] validateDuringCompile(boolean start,
                                     Map context)
```

Parameters

start
− true if the "control" type is being entered.

context
− state that the control author can use to accumulate information during compile.

Returns

− an array of issues. If there are no issues the array should be empty.

---

validateDuringEdit(boolean) Method

```
public Issue[] validateDuringEdit(boolean start)
```

Not yet supported.

ValidateMethod Interface

public interface ***ValidateMethod***

The ValidateMethod interface provides methods with which you can validate a control method. The WebLogic Workshop IDE calls these methods in your implementation of this interface. To expose your implementation to the IDE, you specify the implementation with the method−validator attribute of the <control−tags> element in the control's tags XML file. **Note:** The ValidateMethod interface is part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Method Summary

`public Issue[]` ***`validateDuringCompile`****(`String` methodName, boolean callback, `String` returnType, `Map` parameters, `Map` context)*

Provides a way for control method declarations to be validated during compilation, and for you to display specific information in Source View for invalid methods.

`public Issue[]` ***`validateDuringEdit`****(`String` methodName, boolean callback, `String` returnType, `Map` parameters)*

Not yet supported.

Method Detail

validateDuringCompile(String, boolean, String, Map, Map) Method

```
public Issue[] validateDuringCompile(String methodName,
                                     boolean callback,
                                     String returnType,
                                     Map parameters,
                                     Map context)
```

Provides a way for control method declarations to be validated during compilation, and for you to display specific information in Source View for invalid methods. This method is called by the IDE under many circumstances, including attempts by the control's user to build a component that contains the control. Invalid methods will be displayed in Source View with a red underline.

Parameters

methodName

The name of the method being validated.

callback

`true` if the method is a callback; otherwise, `false`.

returnType
The type returned by this method.

parameters
Map where the key is the parameter name, and the value is the parameter type (Mark −− param value?).

context
State that the control author can use to accumulate information during validation.

Returns

An array of issues. If there are no issues the array should be empty.

---

validateDuringEdit(String, boolean, String, Map) Method

```
public Issue[] validateDuringEdit(String methodName,
                                  boolean callback,
                                  String returnType,
                                  Map parameters)
```

Not yet supported.

com.bea.control

ValidateTag Interface

public interface ***ValidateTag***

The ValidateTag interface can be used to validate a property annotation as a whole. Having an implementation scoped to the annotation is useful when the property exposes multiple attributes and the property isn't valid unless attributes are valid together. For example, you might have a property with "user−name", "password" and "server−name" attributes. If your control's code requires all three to authenticate someone, then the property might be invalid if one of the three attributes is missing. Note that this is invoked per instance of the property annotation. For tags that allow multiple occurence, each occurence will be called individually. There is no support for cross instance validation at compile time. **Note:** The ValidateTag interface is part of the WebLogic Workshop Control Development Kit. The Control Development Kit is for users who are creating advanced controls. It is available in your BEA installation in the following directory: BEA_HOME\weblogic81\samples\workshop\ControlDevKit\

Method Summary

`public Issue[]` ***`validateDuringCompile`***`(String` *tagName*`, Map` *attributes*`, Map` *context*`)`
Provides a way for property attribute annotations to be validated during compilation, and for you to display specific information in Source View for invalid annotations.

`public Issue[]` ***`validateDuringEdit`***`(String` *tagName*`, Map` *attributes*`)`
Not yet supported.

Method Detail

validateDuringCompile(String, Map, Map) Method

```
public Issue[] validateDuringCompile(String tagName,
                                     Map attributes,
                                     Map context)
```

Provides a way for property attribute annotations to be validated during compilation, and for you to display specific information in Source View for invalid annotations. This method is called by the IDE under many circumstances, including attempts by the control's user to build a component that contains the control. Invalid annotations will be displayed in Source View with a red underline.

Parameters

tagName
The name of the annotation being validated.

attributes
The annotation's attributes in name/value pairs.

context
State that the control you can use to accumulate information.

Returns

An array of issues. If there are no issues the array should be empty.

---

validateDuringEdit(String, Map) Method

```
public Issue[] validateDuringEdit(String tagName,
                                  Map attributes)
```

Not yet supported.

com.bea.jws Package

Interface Summary

ServiceHandle	The ServiceHandle interface provides a persistable reference to a JWS service object.
WebService	The WebService interface is a marker interface for web services.

Class Summary

Protocol	The Protocol class is an enumerated type that defines the set of supported invocation protocols available for Weblogic Workshop component invocation.

Exception Summary

RetryException	This class can be thrown by JWS applications on non−blocking ports to indicate that the method should be called again at a later point in time.
SoapFaultException	The SoapFaultException is a wrapper exception that enables the web service developer to generate a specific SOAP fault.

Protocol Class

public class ***Protocol***

extends `Object`
implements `Serializable`

The Protocol class is an enumerated type that defines the set of supported invocation protocols available for Weblogic Workshop component invocation. A Protocol instance has an ID, which is guaranteed to be unique and can be safely used in switch statement against the various ID_xxx values, and a name, which corresponds to the protocol name used to enable/disable the protocol in the protocol tag. It is also safe to use object equality to compare protocol values obtained as parameters or via deserialization to the defined constants. For example: `Protocol.getProtocol(ID_HTTP_GET) == Protocol.HTTP_GET` is always guaranteed to be true.

Hierarchy

```
Object
 Protocol
```

All Implemented Interfaces

`Serializable`

Field Summary

`public static final Protocol`	***HTTP_GET*** Protocol
`public static final Protocol`	***HTTP_POST*** Protocol
`public static final Protocol`	***HTTP_SOAP*** Protocol
`public static final Protocol`	***HTTP_SOAP12*** Protocol
`public static final Protocol`	***HTTP_XML*** Protocol
`public static final int`	***ID_HTTP_GET*** Value indicating query−style invocation using HTTP GET

public static final int **ID_HTTP_POST**
Value indicating query−style invocation using HTTP POST

public static final int **ID_HTTP_SOAP**
Value indicating SOAP invocation using HTTP (POST)

public static final int **ID_HTTP_SOAP12**
Value indicating SOAP12 invocation using HTTP (POST)

public static final int **ID_HTTP_XML**
Value indicating raw XML using HTTP (POST)

public static final int **ID_JMS_SOAP**
Value indicating SOAP invocation using JMS

public static final int **ID_JMS_SOAP12**
Value indicating SOAP invocation using JMS

public static final int **ID_JMS_XML**
Value indicating raw XML using HTTP (POST)

public static final Protocol **JAVA**
Protocol

public static final Protocol **JMS_SOAP**
Protocol

public static final Protocol **JMS_SOAP12**
Protocol

public static final Protocol **JMS_XML**
Protocol

Method Summary

public int **getID**()
Returns the integer identifier associated with the protocol.

public String **getName**()
Returns the protocol name for the protocol.

public static Protocol **getProtocolByID**(int id)
Returns the Protocol object selected by protocol id

public static **getProtocolByName**(String name)

Protocol	Returns the Protocol object selected by protocol name

public static boolean	***isSoap***(Protocol p)
public static boolean	***isSoap11***(Protocol p)
public static boolean	***isSoap12***(Protocol p)
public String	***toString***()

Methods from class `java.lang.Object`

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

Field Detail

HTTP_GET

`public static final Protocol HTTP_GET`

HTTP_POST

`public static final Protocol HTTP_POST`

HTTP_SOAP

`public static final Protocol HTTP_SOAP`

HTTP_SOAP12

`public static final Protocol HTTP_SOAP12`

HTTP_XML

`public static final Protocol HTTP_XML`

ID_HTTP_GET

public static final int ID_HTTP_GET

Value indicating query−style invocation using HTTP GET

ID_HTTP_POST

public static final int ID_HTTP_POST

Value indicating query−style invocation using HTTP POST

ID_HTTP_SOAP

public static final int ID_HTTP_SOAP

Value indicating SOAP invocation using HTTP (POST)

ID_HTTP_SOAP12

public static final int ID_HTTP_SOAP12

Value indicating SOAP12 invocation using HTTP (POST)

ID_HTTP_XML

public static final int ID_HTTP_XML

Value indicating raw XML using HTTP (POST)

ID_JMS_SOAP

public static final int ID_JMS_SOAP

Value indicating SOAP invocation using JMS

ID_JMS_SOAP12

public static final int ID_JMS_SOAP12

Value indicating SOAP invocation using JMS

ID_JMS_XML

public static final int ID_JMS_XML

Value indicating raw XML using HTTP (POST)

JAVA

public static final Protocol JAVA

JMS_SOAP

public static final Protocol JMS_SOAP

JMS_SOAP12

public static final Protocol JMS_SOAP12

JMS_XML

public static final Protocol JMS_XML

Method Detail

getID() Method

public int getID()

Returns the integer identifier associated with the protocol.

---

getName() Method

public String getName()

Returns the protocol name for the protocol. This name corresponds to the tag name associated with the protocol for the protocol tag.

---

getProtocolByID(int) Method

public static Protocol getProtocolByID(int *id*)

Returns the Protocol object selected by protocol id

getProtocolByName(String) Method

public static Protocol getProtocolByName(String *name***)**

Returns the Protocol object selected by protocol name

isSoap(Protocol) Method

public static boolean isSoap(Protocol *p***)**

isSoap11(Protocol) Method

public static boolean isSoap11(Protocol *p***)**

isSoap12(Protocol) Method

public static boolean isSoap12(Protocol *p***)**

toString() Method

public String toString()

Overrides

`Object.toString()`

com.bea.jws

RetryException Class

public class ***RetryException***

extends `Exception`

This class can be thrown by JWS applications on non−blocking ports to indicate that the method should be called again at a later point in time.

The `retryDelay` value can be one of:

- DEFAULT_DELAY − use delay specified declaratively in JWS @Nonblocking
- 0 − no delay on retry
- n − delay for 'n' seconds before retry
- duration − a duration string in the format used for the retryDelay attribute of the \@jws:message−buffer tag.

Hierarchy

```
Object
 Throwable
  Exception
   RetryException
```

All Implemented Interfaces

`Serializable`

Direct Known Subclasses

`RetryException`

Field Summary

`public static final String` ***DEFAULT_DELAY***
String

Constructor Summary

RetryException*(`String` s, `String` retryString)*

***RetryException**(`String` s, long retrySeconds)*

***RetryException**(`String` s)*

Method Summary

public
String ***getRetryDelay**()*

Methods from `java.lang.Throwable`

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

Methods from class `java.lang.Object`

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

Field Detail

DEFAULT_DELAY

```
public static final String DEFAULT_DELAY
```

Constructor Detail

RetryException

```
public RetryException(String s,
                      String retryString)
```

RetryException

```
public RetryException(String s,
                      long retrySeconds)
```

RetryException

```
public RetryException(String s)
```

Method Detail

getRetryDelay() Method

```
public String getRetryDelay()
```

com.bea.jws

ServiceHandle Interface

public interface ***ServiceHandle***

extends `Serializable`

The ServiceHandle interface provides a persistable reference to a JWS service object. A ServiceHandle can refer to both a JWS service instance or to a JWS control instance that is associated with a JWS service instance.

All Superinterfaces

`Serializable`

All Known Subinterfaces

`ServiceHandle`

Field Summary

`public static final int` ***SCHEME_DEFAULT***
int

`public static final int` ***SCHEME_FILE***
int

`public static final int` ***SCHEME_FTP***
int

`public static final int` ***SCHEME_HTTP***
int

`public static final int` ***SCHEME_JMS***
int

`public static final int` ***SCHEME_SMTP***
int

Method Summary

`public String` ***getContextURI****()*

Returns the base context URI that references the root of the application containing this component.

public String ***getControlID()***

Returns the identity of a control instance that the handle refers to.

public String ***getConversationID()***

Returns the identity of a JWS conversation instance that the handle refers to.

public String ***getJNDIBaseName()***

Returns a period−separated string based on the URI, which uniquely identifies this service on this server and is used to generate unique JNDI names for objects associated with this service.

public int ***getScheme()***

Returns the protocol scheme that was used to construct the ServiceHandle.

public String ***getURI()***

Returns a protocol−indendent URI which can be used to refer to this service.

public URL ***getURL****(int scheme)*

Returns a URL which defines a reference to this service using the specified scheme.

public URL ***getURL()***

Returns a URL which defines a reference to this service using the same scheme that was used to construct the service handle.

Field Detail

SCHEME_DEFAULT

public static final int SCHEME_DEFAULT

SCHEME_FILE

public static final int SCHEME_FILE

SCHEME_FTP

public static final int SCHEME_FTP

SCHEME_HTTP

public static final int SCHEME_HTTP

SCHEME_JMS

```
public static final int SCHEME_JMS
```

SCHEME_SMTP

```
public static final int SCHEME_SMTP
```

Method Detail

getContextURI() Method

```
public String getContextURI()
```

Returns the base context URI that references the root of the application containing this component.

---

getControlID() Method

```
public String getControlID()
```

Returns the identity of a control instance that the handle refers to. This may be null if the handle is associated with a service instance instead of a control instance.

---

getConversationID() Method

```
public String getConversationID()
```

Returns the identity of a JWS conversation instance that the handle refers to. This may be null if referring to stateless service or constructed as a result of processing a stateless operation on a service.

---

getJNDIBaseName() Method

```
public String getJNDIBaseName()
```

Returns a period−separated string based on the URI, which uniquely identifies this service on this server and is used to generate unique JNDI names for objects associated with this service.

---

getScheme() Method

```
public int getScheme()
```

Returns the protocol scheme that was used to construct the ServiceHandle.

getURI() Method

```
public String getURI()
```

Returns a protocol−indendent URI which can be used to refer to this service.

getURL(int) Method

```
public URL getURL(int scheme)
```

Returns a URL which defines a reference to this service using the specified scheme.

getURL() Method

```
public URL getURL()
```

Returns a URL which defines a reference to this service using the same scheme that was used to construct the service handle.

com.bea.jws

SoapFaultException Class

public class ***SoapFaultException***

extends `RuntimeException`

The SoapFaultException is a wrapper exception that enables the web service developer to generate a specific SOAP fault.

Related Topics

`SoapFault`

Hierarchy

```
Object
 Throwable
  Exception
   RuntimeException
    SoapFaultException
```

All Implemented Interfaces

`Serializable`

Field Summary

public static final int	***FAULT_SOAP11*** int
public static final int	***FAULT_SOAP12*** int
public static final int	***FAULT_UNKNOWN*** int

Constructor Summary

SoapFaultException*(`XmlObject` detailContent, `String` s)*

Assumes the user just wants to generate the fault document defined in the WSDL for the current

operation.

SoapFaultException*(*`XmlObject` *xmlObj)*

User wants to own the shape of the fault.

SoapFaultException*(*`XmlObject` *detailContent)*

Assumes the user just wants to generate the fault document defined in the WSDL for the current operation.

Method Summary

`public XmlObject[]` ***getDetail****()*

`public XmlObject` ***getFault****()*

`public boolean` ***hasDetail****()*

`public boolean` ***hasFault****()*

`public boolean` ***isCausedBySender****()*

`public void` ***setCausedBySender****(boolean senderIsCause)*
Settes the code in the SoapFault: for Soap 1.2 Sender/Receiver, for Soap 1.1 Client/Server

`public int` ***soapFaultVersion****()*

Methods from `java.lang.Throwable`

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

Methods from class `java.lang.Object`

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

Field Detail

FAULT_SOAP11

```
public static final int FAULT_SOAP11
```

FAULT_SOAP12

```
public static final int FAULT_SOAP12
```

FAULT_UNKNOWN

```
public static final int FAULT_UNKNOWN
```

Constructor Detail

SoapFaultException

```
public SoapFaultException(XmlObject detailContent,
                          String s)
```

Assumes the user just wants to generate the fault document defined in the WSDL for the current operation. This form defaults to server/receiver as the cause.

SoapFaultException

```
public SoapFaultException(XmlObject xmlObj)
```

User wants to own the shape of the fault. The user has to know which version of the envelop to use.

Related Topics

`Protocol`

SoapFaultException

```
public SoapFaultException(XmlObject[] detailContent)
```

Assumes the user just wants to generate the fault document defined in the WSDL for the current operation. This form defaults to server/receiver as the cause.

Method Detail

getDetail() Method

```
public XmlObject[] getDetail()
```

Returns

detail

---

getFault() Method

```
public XmlObject getFault()
```

Returns

fault

---

hasDetail() Method

```
public boolean hasDetail()
```

Returns

whether the fault contains detail

---

hasFault() Method

```
public boolean hasFault()
```

Returns

whether this holds a fault

---

isCausedBySender() Method

```
public boolean isCausedBySender()
```

Returns

senderIsCause

---

setCausedBySender(boolean) Method

public void setCausedBySender(boolean *senderIsCause***)**

Settes the code in the SoapFault: for Soap 1.2 Sender/Receiver, for Soap 1.1 Client/Server

Parameters

senderIsCause
if true: Sender (Client for Soap 1.1) else: Receiver (Server for Soap 1.1)

---

soapFaultVersion() Method

public int soapFaultVersion()

Returns

soapFaultVersion

com.bea.jws

WebService Interface

public interface ***WebService***

extends `Serializable`

The WebService interface is a marker interface for web services.

All Superinterfaces
`Serializable`

---

com.bea.wlw.util Package

Interface Summary

Logger	The Logger interface defines application logging functions.

com.bea.wlw.util

Logger Interface

public interface ***Logger***

The Logger interface defines application logging functions. A Logger can be obtained from a `JwsContext` or `ControlContext` instance. Logging may be enabled at one of four levels: "debug" (lowest level, results in highest volume of output), "info", "warn" or "error" (highest level, results in least output). Messages emitted below the current logging level are not logged.

All Known Subinterfaces

`Logger`

Method Summary

`public void` ***`debug`***`(String message)`
Emit a "debug" level message.

`public void` ***`debug`***`(String message, Throwable throwable)`
Emit a "debug" level message with an associated `Throwable`.

`public void` ***`error`***`(String message)`
Emit a "error" level message.

`public void` ***`error`***`(String message, Throwable throwable)`
Emit a "error" level message with an associated `Throwable`.

`public void` ***`info`***`(String message)`
Emit a "info" level message.

`public void` ***`info`***`(String message, Throwable throwable)`
Emit a "info" level message with an associated `Throwable`.

`public boolean` ***`isDebugEnabled`***`()`
returns true if "debug" level logging is enabled.

`public boolean` ***`isErrorEnabled`***`()`
returns true if "error" level logging is enabled.

`public boolean` ***`isInfoEnabled`***`()`
returns true if "info" level logging is enabled.

public ***isWarnEnabled****()*
boolean — returns true if "warn" level logging is enabled.

public void ***warn****(String message)*
Emit a "warn" level message.

public void ***warn****(String message, Throwable throwable)*
Emit a "warn" level message with an associated `Throwable`.

Method Detail

debug(String) Method

```
public void debug(String message)
```

Emit a "debug" level message.

Parameters

message
the message to write to the log.

---

debug(String, Throwable) Method

```
public void debug(String message,
                  Throwable throwable)
```

Emit a "debug" level message with an associated `Throwable`.

Parameters

message
the message to write to the log.

throwable
the `Throwable` (typically an `Exception`) to write to the log.

---

error(String) Method

```
public void error(String message)
```

Emit a "error" level message.

Parameters

message
the message to write to the log.

---

error(String, Throwable) Method

```
public void error(String message,
                  Throwable throwable)
```

Emit a "error" level message with an associated `Throwable`.

Parameters

message
: the message to write to the log.

throwable
: the `Throwable` (typically an `Exception`) to write to the log.

---

info(String) Method

```
public void info(String message)
```

Emit a "info" level message.

Parameters

message
: the message to write to the log.

---

info(String, Throwable) Method

```
public void info(String message,
                 Throwable throwable)
```

Emit a "info" level message with an associated `Throwable`.

Parameters

message
: the message to write to the log.

throwable
: the `Throwable` (typically an `Exception`) to write to the log.

---

isDebugEnabled() Method

```
public boolean isDebugEnabled()
```

returns true if "debug" level logging is enabled. "debug" level is only enabled if the current logging level is "debug".

---

isErrorEnabled() Method

```
public boolean isErrorEnabled()
```

returns true if "error" level logging is enabled. "error" level is enabled if the current logging level is "error", "warn", "info" or "debug".

---

isInfoEnabled() Method

```
public boolean isInfoEnabled()
```

returns true if "info" level logging is enabled. "info" level is enabled if the current logging level is "info" or "debug".

---

isWarnEnabled() Method

```
public boolean isWarnEnabled()
```

returns true if "warn" level logging is enabled. "warn" level is enabled if the current logging level is "warn", "info" or "debug".

---

warn(String) Method

```
public void warn(String message)
```

Emit a "warn" level message.

Parameters

message
: the message to write to the log.

---

warn(String, Throwable) Method

```
public void warn(String message,
                 Throwable throwable)
```

Emit a "warn" level message with an associated `Throwable`.

Parameters

message
: the message to write to the log.

throwable
: the `Throwable` (typically an `Exception`) to write to the log.

weblogic.jws Package

Interface Summary

ServiceHandle	The ServiceHandle object represents a persistable reference to a JWS service object.

Exception Summary

RetryException	This class can be thrown by JWS applications on non−blocking ports to indicate that the method should be reattempted at a later point in time.

DEPRECATED

RetryException Class

public class ***RetryException***

extends `com.bea.jws.RetryException`

This class can be thrown by JWS applications on non−blocking ports to indicate that the method should be reattempted at a later point in time.

The `retryDelay` value can be one of:

- DEFAULT_DELAY − use delay specified declaratively in JWS @Nonblocking
- 0 − no delay on retry
- n − delay for 'n' seconds before retry
- duration − a duration string in the format used for the retryDelay attribute of the \@jws:message−buffer tag.

Related Topics

`RetryException`

Hierarchy

```
Object
 Throwable
  Exception
   com.bea.jws.RetryException
    RetryException
```

All Implemented Interfaces

`Serializable`

Field Summary

Fields from com.bea.jws.`com.bea.jws.RetryException`

`DEFAULT_DELAY`

Constructor Summary

RetryException*(`String` s, `String` retryString)*

***RetryException**(`String` s, long retrySeconds)*

***RetryException**(`String` s)*

Method Summary

Methods from `com.bea.jws.com.bea.jws.RetryException`

`getRetryDelay`

Methods from `java.lang.Throwable`

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

Methods from class `java.lang.Object`

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

Constructor Detail

RetryException

```
public RetryException(String s,
                      String retryString)
```

RetryException

```
public RetryException(String s,
                      long retrySeconds)
```

RetryException

```
public RetryException(String s)
```

DEPRECATED

ServiceHandle Interface

public interface ***ServiceHandle***

extends `com.bea.jws.ServiceHandle`

The ServiceHandle object represents a persistable reference to a JWS service object. A Service Handle can refer to both a JWS service instance or to a JWS control instance that is associated with a JWS service instance.

Related Topics

`ServiceHandle`

All Superinterfaces

`Serializable`, `com.bea.jws.ServiceHandle`

Field Summary

Fields from interface com.bea.jws.`com.bea.jws.ServiceHandle`

`SCHEME_DEFAULT, SCHEME_FILE, SCHEME_FTP, SCHEME_HTTP, SCHEME_JMS, SCHEME_SMTP`

Method Summary

Methods from interface `com.bea.jws.com.bea.jws.ServiceHandle`

`getContextURI, getControlID, getConversationID, getJNDIBaseName, getScheme, getURI, getURL, getURL`

weblogic.jws.control Package

Interface Summary

Context	Provides access to services and information specific to this component's context.
Context.Callback	Defines callback events that may be received by the container.
Control	This is a base interface for all controls.
ControlFactory	This is a marker interface for all control factories.
DatabaseControl	Database Control base interface
EJBControl	Enterprise Java Bean Control base interface
EntityEJBControl	Entity Enterprise Java Bean Control base interface
JMSControl	Control used to send and receive JMS messages.
JwsContext	The JwsContext interface provides access to JWS container services.
JwsContext.Callback	
ServiceControl	Server Proxy Control base interface
SessionEJBControl	Session Enterprise Java Bean Control base interface
TimerControl	Timer Control base interface
TimerControlFactory	The base factory interface for TimerControl

Class Summary

DatabaseFilter

Exception Summary

ControlException	
SchedulerException	The SchedulerException is thrown to indicate a system error in control scheduling events.
ServiceControlException	

Context Interface

public interface ***Context***

extends `Serializable`

Provides access to services and information specific to this component's context. You will never need to extend or implement this interface. You will use this interface only through more specific subinterfaces, such as `JwsContext` and `ControlContext`.

All Superinterfaces
`Serializable`

All Known Subinterfaces

`ControlContext`, `JpdContext`, `JwsContext`, `JwsContext`

Nested Class Summary

public static interface	**`Context.Callback`** Defines callback events that may be received by the container.

Method Summary

public void	***`finishConversation`***`()` Marks the current conversation instance as requiring removal after the currently executing method or event handler returns.
public Principal	***`getCallerPrincipal`***`()` Returns the security principal associated with the current method invocation if authentication was performed.
public long	***`getCurrentAge`***`()` Returns the conversation's current age (in seconds).
public long	***`getCurrentIdleTime`***`()` Gets the number of seconds since the last client request, or since the conversation's maximum idle time was reset.
public Logger	***`getLogger`***`(String name)`

Gets an instance of the Logger class, which you can use to send messages from your code to a log file.

`public long` **`getMaxAge`**`()`
Gets the time representing the longest the conversation may remain active before finishing.

`public long` **`getMaxIdleTime`**`()`
Gets the number of seconds that the conversation can remain idle before finishing due to client inactivity.

`public ServiceHandle` **`getService`**`()`
Returns a ServiceHandle instance for the currently active service instance.

`public boolean` **`isCallerInRole`**`(String roleName)`
Returns true if the authenticated principal is within the specified security role.

`public boolean` **`isFinished`**`()`
Returns whether or not this conversation instance has had finish() called on it (indicating that code has requested termination of this conversation instance).

`public void` **`resetIdleTime`**`()`
Resets the timer measuring the number of seconds since the last activity for the current conversation.

`public void` **`setMaxAge`**`(Date date)`
Sets a new maximum age for the conversation to an absolute Date.

`public void` **`setMaxAge`**`(String duration)`
Sets a conversation's maximum age by specifying a duration as a string.

`public void` **`setMaxIdleTime`**`(long seconds)`
Sets the number of seconds that the conversation can remain idle before finishing due to client inactivity.

`public void` **`setMaxIdleTime`**`(String duration)`
Sets the number of seconds (as a String) that the conversation can remain idle before finishing due to client inactivity.

Method Detail

finishConversation() Method

`public void finishConversation()`

Marks the current conversation instance as requiring removal after the currently executing method or event handler returns. Call this method to force removal of the conversation instance as an alternative to waiting for it to be removed by WebLogic Server when the conversation times out or its "finish" method is reached.

Calling this method from within another method is equivalent to marking the calling method with the `@conversation phase="finish"` tag.

---

getCallerPrincipal() Method

```
public Principal getCallerPrincipal()
```

Returns the security principal associated with the current method invocation if authentication was performed.

Returns

The Principal that was produced by authentication.

---

getCurrentAge() Method

```
public long getCurrentAge()
     throws IllegalStateException
```

Returns the conversation's current age (in seconds).

Returns

The number of seconds that have passed since the conversation started.

Exceptions

`IllegalStateException`
If the method is called from a service instance that is not conversational.

---

getCurrentIdleTime() Method

```
public long getCurrentIdleTime()
     throws IllegalStateException
```

Gets the number of seconds since the last client request, or since the conversation's maximum idle time was reset.

Returns

The number of seconds since the last activity affecting the conversation.

Exceptions

`IllegalStateException`
if the method is called from a service instance that is not conversational.

---

getLogger(String) Method

```
public Logger getLogger(String name)
```

Gets an instance of the Logger class, which you can use to send messages from your code to a log file. Use getLogger to log messages from your web service Java code to a text file. By default, for the samples domain installed with WebLogic Workshop web services, this text file is located at the following path of your WebLogic Workshop installation:

```
BEA_HOME/weblogic81/samples/workshop/jws.log
```

Use the *categoryName* parameter to specify category text that will be included with log entries. For example, you might specify the name of the JWS file so that you can more easily find relevant messages when scanning the log file. A log message might appear as follows for an entry in which *categoryName* is "MyService".

```
16:18:11 ERROR MyService: My log message.
```

Note: You can customize aspects of the logging configuration, including the name of the application log file, its size limit, and so on. You configure logging using the workshopLogCfg.xml file. For more information, search the WebLogic Workshop documentation for "workshopLogCfg.xml Configuration File". The Logger class returned by this method includes four methods that you can use to print log entries to a text file. For more information, see the `Logger` class.

Parameters

name
: The name of the category by which log messages should be grouped.

Returns

A Logger class that may be used to send messages to the application log.

---

getMaxAge() Method

```
public long getMaxAge()
     throws IllegalStateException
```

Gets the time representing the longest the conversation may remain active before finishing.

Returns

The number of seconds since the conversation started before which it will finish.

Exceptions

`IllegalStateException`
: if the method is called from a service instance that is not conversational.

---

getMaxIdleTime() Method

```
public long getMaxIdleTime()
     throws IllegalStateException
```

Gets the number of seconds that the conversation can remain idle before finishing due to client inactivity. A conversation is idle if the service is not receiving incoming messages through an operation method. Note that messages received through callback handlers do not reset a conversation idle time.

Returns

The number of seconds that the conversation can remain idle before finishing due to client inactivity.

Exceptions

`IllegalStateException`
: if the method is called from a service instance that is not conversational.

---

getService() Method

```
public ServiceHandle getService()
```

Returns a ServiceHandle instance for the currently active service instance. You can use this handle to query service information such as the conversation ID, the protocol scheme initiating the current request, and the URL of the service.

Returns

A weblogic.jws.ServiceHandle object representing the current web service.

---

isCallerInRole(String) Method

```
public boolean isCallerInRole(String roleName)
```

Returns true if the authenticated principal is within the specified security role.

Parameters

roleName
: The name of the security role against which to check the authenticated principal.

Returns

`true` *if the principal is within the specified security role;* `false` *if they are not.*

---

isFinished() Method

```
public boolean isFinished()
```

Returns whether or not this conversation instance has had finish() called on it (indicating that code has requested termination of this conversation instance). You can finish a conversation by calling the finishConversation method on the conversation instance.

Returns

`true` if the conversation has finished; `false` if it hasn't.

---

resetIdleTime() Method

```
public void resetIdleTime()
     throws IllegalStateException
```

Resets the timer measuring the number of seconds since the last activity for the current conversation. You can use this method to reset the timer for other service activity beyond client requests.

Returns

The number of seconds before which the conversation will be finished due to client inactivity.

Exceptions

`IllegalStateException`
if the method is called from a service instance that is not conversational.

---

setMaxAge(Date) Method

```
public void setMaxAge(Date date)
     throws IllegalStateException, IllegalArgumentException
```

Sets a new maximum age for the conversation to an absolute Date. If the date parameter is in the past, the conversation will finish immediately. Calling this method from a web service's code sets the maximum conversation age (relative to the current time) for that service instance. The *date* parameter marks the time after which WebLogic Server will be allowed to finish this conversation. Note that setting the maximum age with this method overrides the default setting or the setting given in the service's JWS file. This is an absolute age that isn't affected by network traffic.

Parameters

date
The time after which the conversation should finish; null to disable the age timeout.

Exceptions

IllegalStateException
: if the method is called from a service instance that is not conversational.

IllegalArgumentException
: If this method has passed an illegal or inappropriate argument.

---

setMaxAge(String) Method

```
public void setMaxAge(String duration)
     throws IllegalStateException, IllegalArgumentException
```

Sets a conversation's maximum age by specifying a duration as a string. Call this method to set the maximum conversation age (relative to the current time) for the conversation. The duration parameter marks the time after which the WebLogic Server will be allowed to finish this conversation. Note that setting the maximum age with this method overrides the default setting or the setting given in the service's JWS file. This is an absolute age that isn't affected by network traffic. If the *duration* value results in zero seconds, the maximum age timeout will be disabled. If the *duration* value is in the past, the conversation will finish immediately. The *duration* value is considered to be relative to the current time, not to the start of the conversation.

Parameters

duration
: The period after which the conversation will finish.

Exceptions

IllegalStateException
: if the method is called from a service instance that is not conversational.

IllegalArgumentException
: If this method has passed an illegal or inappropriate argument.

---

setMaxIdleTime(long) Method

```
public void setMaxIdleTime(long seconds)
     throws IllegalStateException, IllegalArgumentException
```

Sets the number of seconds that the conversation can remain idle before finishing due to client inactivity. Conversation idle time is the amount of time that can pass between incoming messages before the service instance is finished due to client inactivity. You can initialize a conversation's idle time to a default through the setting given at the top of the JWS file, but each instance can be set to a different idle time value through the `setMaxIdleTime` or `setMaxIdleTime` method. To disable idle time expiration, set the maximum idle time value to zero. The timer associated with tracking the idle time will automatically reset whenever a request is received from a web service's client or when `resetIdleTime` method is called.

Parameters

seconds
: The number of seconds the conversation can remain idle before it will expire.

Exceptions

IllegalStateException
: if the method is called from a service instance that is not conversational.

IllegalArgumentException
: If this method has passed an illegal or inappropriate argument.

---

setMaxIdleTime(String) Method

```
public void setMaxIdleTime(String duration)
     throws IllegalStateException, IllegalArgumentException
```

Sets the number of seconds (as a String) that the conversation can remain idle before finishing due to client inactivity. Conversation idle time is the amount of time that can pass between incoming messages before the service instance is finished due to client inactivity. You can initialize a conversation's idle time to a default through the setting given at the top of the JWS file, but each instance can be set to a different idle time value through the `setMaxIdleTime` or `setMaxIdleTime` method. To disable idle time expiration, set the maximum idle time value to zero. The timer associated with tracking the idle time will automatically reset whenever a request is received from a web service's client or when `resetIdleTime` method is called.

Parameters

duration
: The number of seconds the conversation can remain idle before it will expire.

Exceptions

IllegalStateException
: if the method is called from a service instance that is not conversational.

IllegalArgumentException
: If this method has passed an illegal or inappropriate argument.

Context.Callback Interface

public static interface ***Context.Callback***

Defines callback events that may be received by the container.

All Known Subinterfaces

`ControlContext.Callback`, `JpdContext.Callback`, `JwsContext.Callback`, `JwsContext.Callback`

Enclosing interface

`Context`

Method Summary

`public void` ***`onAgeTimeout`***`(long age)`
Received by conversational web services when the current instance has reached its maximum lifetime.

`public void` ***`onAsyncFailure`***`(String methodName, Object[] args)`
Received when an asynchronous (or, message−buffered) method is unable to be succesfully delivered.

`public void` ***`onCreate`***`()`
Received after a new instance has been created and system initialization (of context object and contained controls) has been completed.

`public void` ***`onException`***`(Exception e, String methodName, Object[] args)`
Received when a web service operation throws an uncaught exception.

`public void` ***`onFinish`***`(boolean expired)`
Received by conversational web services when the current instance is about to be finished.

`public void` ***`onIdleTimeout`***`(long time)`
Received by conversational web services when the current instance has reached its maximum idle timeout.

Method Detail

onAgeTimeout(long) Method

```
public void onAgeTimeout(long age)
```

Received by conversational web services when the current instance has reached its maximum lifetime. The conversation finishes when the handler for this callback has finished executing.

Parameters

age
The age of the conversation that has timed out.

---

onAsyncFailure(String, Object[]) Method

```
public void onAsyncFailure(String methodName,
                           Object[] args)
```

Received when an asynchronous (or, message−buffered) method is unable to be succesfully delivered.

Parameters

methodName
The method that could not be delivered.

args
An array containing the parameters of the method.

---

onCreate() Method

```
public void onCreate()
```

Received after a new instance has been created and system initialization (of context object and contained controls) has been completed.

---

onException(Exception, String, Object[]) Method

```
public void onException(Exception e,
                        String methodName,
                        Object[] args)
```

Received when a web service operation throws an uncaught exception. Implement a handler for this callback as a single place to catch exceptions thrown from operation methods. You implement a callback handler that will execute when your service receives this callback from WebLogic Server. The following code might be used for debugging. Prior for deployment, you might replace it with code that logs this information instead. Notice that thisÃŠcode also ends the conversation in which the service might have been participating. This frees no−longer−needed resources.

```
public void context_onException(Exception e, String methodName, Object[] arguments)
```

```
{
    System.out.println("MyService: exception in " + methodName + "(" +
        arguments + "). Exception: " + e);
        context.finishConversation();
}
```

Not all methods in a web service are necessarily operations. In web services, an operation is a method specifically exposed to clients. In WebLogic Workshop, the source code of an operation is preceded by an @common:operation annotation. Note that in previous releases, the @common:operation tag was known as the @jws:operation tag. This tag is still supported for backward compatibility.

Parameters

e
The exception object thrown from the method.

methodName
The name of the method from which the exception was thrown.

args
An array containing the parameters of the method that threw the exception.

---

onFinish(boolean) Method

`public void onFinish(boolean` *`expired`***`)`**

Received by conversational web services when the current instance is about to be finished. The conversation may be ending due to a call to the finishConversation method, a timeout, or successful completion of a method annotated with `@conversation phase="finish"`.

Parameters

expired
`true` if the conversation finished because it timed out; otherwise, `false`.

---

onIdleTimeout(long) Method

`public void onIdleTimeout(long` *`time`***`)`**

Received by conversational web services when the current instance has reached its maximum idle timeout. The conversation finishes when the handler for this callback has finished executing.

Parameters

time
The current time in milliseconds.

DEPRECATED The `com.bea.control.Control` interface replaces this interface.

Control Interface

public interface ***Control***

extends `com.bea.control.Control`

This is a base interface for all controls. All controls must be serializable to support persistance of conversations containing controls.

All Superinterfaces

`com.bea.control.Control`, `Serializable`

All Known Subinterfaces

`com.bea.p13n.controls.events.standard.ClickContentEventControl`,
`com.bea.p13n.controls.createUser.CreateUserControl`,
`com.bea.control.DatabaseControl`, `weblogic.jws.control.DatabaseControl`,
`com.bea.p13n.controls.events.standard.DisplayContentEventControl`,
`weblogic.jws.control.EJBControl`, `com.bea.control.EmailControl`,
`com.bea.control.EntityEJBControl`,
`weblogic.jws.control.EntityEJBControl`,
`com.bea.p13n.controls.ejb.property.EntityPropertyManager`,
`com.bea.p13n.controls.ejb.events.EventService`,
`com.bea.control.FileControl`,
`com.bea.p13n.controls.events.generic.GenericEventControl`,
`com.bea.p13n.controls.events.generic.GenericTrackingControl`,
`com.bea.p13n.controls.ejb.usermgmt.GroupManager`,
`com.bea.p13n.controls.ejb.usermgmt.profile.GroupProfileManager`,
`com.bea.p13n.controls.securityProvider.GroupProviderControl`,
`com.bea.p13n.controls.profile.ProfileControl`,
`com.bea.p13n.controls.profile.PropertyControl`,
`com.bea.p13n.controls.ejb.property.PropertySetManager`,
`com.bea.control.PublishControl`,
`com.bea.p13n.controls.ejb.usermgmt.RealmConfiguration`,
`com.bea.p13n.controls.events.standard.RuleEventControl`,
`com.bea.p13n.controls.rules.RulesExecutorControl`,
`com.bea.p13n.controls.rules.RulesManagerControl`,
`com.bea.control.SessionEJBControl`,
`weblogic.jws.control.SessionEJBControl`,
`com.bea.p13n.controls.events.standard.SessionLoginEventControl`,
`com.bea.control.SubscriptionControl`, `com.bea.control.TaskControl`,
`com.bea.control.TaskWorkerControl`, `com.bea.control.TimerControl`,
`weblogic.jws.control.TimerControl`, `com.bea.control.TPMControl`,
`com.bea.p13n.controls.userInfoQuery.UserInfoControl`,
`com.bea.p13n.controls.login.UserLoginControl`,
`com.bea.p13n.controls.ejb.usermgmt.UserManager`,

`com.bea.p13n.controls.profile.UserProfileControl`,
`com.bea.p13n.controls.ejb.usermgmt.profile.UserProfileManager`,
`com.bea.p13n.controls.securityProvider.UserProviderControl`,
`com.bea.p13n.controls.events.standard.UserRegistrationEventControl`

DEPRECATED The `com.bea.control.ControlException` class replaces this interface.

ControlException Class

public class ***ControlException***

extends `com.bea.control.ControlException`

Hierarchy

```
Object
 Throwable
  Exception
   RuntimeException
    com.bea.control.ControlException
     ControlException
```

All Implemented Interfaces

`Serializable`

Direct Known Subclasses

`com.bea.control.ServiceControlException`,
`com.bea.control.TPMControlException`

Constructor Summary

ControlException*(*`String` *message,* `Throwable` *t)*

ControlException*(*`String` *message)*

Method Summary

Methods from `com.bea.control.com.bea.control.ControlException`

`getNestedException`

Methods from `java.lang.Throwable`

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

printStackTrace, setStackTrace, toString

Methods from class `java.lang.Object`

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

Constructor Detail

ControlException

```
public ControlException(String message,
                        Throwable t)
```

ControlException

```
public ControlException(String message)
```

DEPRECATED The `com.bea.control.ControlFactory` interface replaces this interface.

ControlFactory Interface

public interface ***ControlFactory***

extends `com.bea.control.ControlFactory`

This is a marker interface for all control factories. All factories must be serializable to support persistance of conversations containing factories.

All Superinterfaces

`com.bea.control.ControlFactory`, `Serializable`

All Known Subinterfaces

`com.bea.control.TimerControlFactory`,
`weblogic.jws.control.TimerControlFactory`

DEPRECATED The `com.bea.control.DatabaseControl` interface replaces this interface.

weblogic.jws.control

DatabaseControl Interface

public interface ***DatabaseControl***

extends `com.bea.control.DatabaseControl`

Database Control base interface

All Superinterfaces

`com.bea.control.Control`, `weblogic.jws.control.Control`, `com.bea.control.DatabaseControl`, `Serializable`

Nested Class Summary

Nested classes from interface com.bea.control.`com.bea.control.DatabaseControl`

`DatabaseControl.SQLFragment, DatabaseControl.SQLParameter`

Method Summary

Methods from interface `com.bea.control.com.bea.control.DatabaseControl`

`acceptChanges, getConnection, getDataSourceCalendar, setDataSourceCalendar,`

DEPRECATED The `com.bea.control.DatabaseFilter` class replaces this class.

weblogic.jws.control

DatabaseFilter Class

public class ***DatabaseFilter***

extends `com.bea.control.DatabaseFilter`

Hierarchy

```
Object
 com.bea.control.DatabaseFilter
  DatabaseFilter
```

All Implemented Interfaces

`Serializable`

Nested Class Summary

Nested classes from com.bea.control.`com.bea.control.DatabaseFilter`

`DatabaseFilter.FilterTerm, DatabaseFilter.SortTerm`

Field Summary

Fields from com.bea.control.`com.bea.control.DatabaseFilter`

`_chFilter, _columns, _filter, _identifierOptions, _maxSortCols, _mdDatabase, _mdResultSet, _sIdentifierQuote, _sort, IDENTIFIER_ASIS, IDENTIFIER_CHANGECASE, IDENTIFIER_DEFAULT, IDENTIFIER_QUOTE, IDENTIFIER_TOLOWER, IDENTIFIER_TOUPPER, opAsc, opContains, opDesc, opEmpty, opEqual, opGreater, opGreaterEqual, opIn, opInvalid, opIs, opIsNot, opLess, opLessEqual, opNotEqual, opStartsWith, sContains, sEmpty, sEquals, sFilterChar, sGreaterEqual, sGreaterThan, sIn, sIsEmpty, sIsNotEmpty, sLessEqual, sLessThan, sNotEqual, sStartsWith, sUnitDate, sUnitMonth, sUnitYear, unitDate, unitDefault, unitMonth, unitYear`

Constructor Summary

DatabaseFilter()

Method Summary

Methods from `com.bea.control.com.bea.control.DatabaseFilter`

`getFilterExpression, getOrderByClause, getSortExpression, getWhereClause, parseDate, parseQueryString, parseQueryString, setDatabaseMetaData, setIdentifierOptions, setResultSetMetaData`

Methods from class `java.lang.Object`

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

Constructor Detail

DatabaseFilter

```
public DatabaseFilter()
```

EJBControl Interface

public interface ***EJBControl***

extends `Control`

Enterprise Java Bean Control base interface

All Superinterfaces

`Control`, `Control`, `Serializable`

All Known Subinterfaces

`EntityEJBControl`, `EntityEJBControl`, `SessionEJBControl`, `SessionEJBControl`

Method Summary

`public Object` ***`getEJBBeanInstance`****()*
Returns the current target instance of the EJB business interface used for business interface method invocations.

`public Throwable` ***`getEJBException`****()*
Returns the last EJB exception serviced by this control on the developer's behalf.

`public Object` ***`getEJBHomeInstance`****()*
Returns an instance of the home interface associated with the target EJB.

`public String` ***`getJNDIName`****()*
Returns the JNDI name associated with the EJB which this control represents.

`public boolean` ***`hasEJBBeanInstance`****()*
Returns `true` if this control currently has a target bean instance upon which bean business interface methods may be invoked.

`public void` ***`setJNDIName`****(`String` jndiName)*
Sets the JNDI name associated with this control; in other words, this method effectively sets which EJB is associated with the control.

Method Detail

getEJBBeanInstance() Method

```
public Object getEJBBeanInstance()
```

Returns the current target instance of the EJB business interface used for business interface method invocations. This API is provided for advanced use cases were direct access to the local/ remote interfaces outside of the control is required. It will return `null` if no target instance is currently selected.

Returns

The business interface instance for the EJB that this control represents; `null` if no target instance is currently selected.

---

getEJBException() Method

```
public Throwable getEJBException()
```

Returns the last EJB exception serviced by this control on the developer's behalf. This can be used to discover or log additional information, such as when a create or find method is unable to locate a target bean instance.

Returns

The last EJB exception serviced by this control.

---

getEJBHomeInstance() Method

```
public Object getEJBHomeInstance()
```

Returns an instance of the home interface associated with the target EJB.

Returns

The home interface instance for the EJB with which this control is associated.

---

getJNDIName() Method

```
public String getJNDIName()
```

Returns the JNDI name associated with the EJB which this control represents. If the control was defined using the jndi−home−name attribute, the returned value will start with "jndi:" and will start with "java:comp/env" if the control was defined using an ejb−link attribute.

Returns

The JNDI name for the EJB represented by this control.

---

hasEJBBeanInstance() Method

```
public boolean hasEJBBeanInstance()
```

Returns `true` if this control currently has a target bean instance upon which bean business interface methods may be invoked. This will be true after successful execution of a create or single select find method, or in cases where implicit creation or find has occurred by the control on the control user's behalf. You can use this method as a simple way to procedurally check the status of explicit or implicit bean instance creation or find operations.

Returns

`true` if this control is associated with a target bean instance; otherwise, `false`.

---

setJNDIName(String) Method

```
public void setJNDIName(String jndiName)
```

Sets the JNDI name associated with this control; in other words, this method effectively sets which EJB is associated with the control. This method validates that *jndiName* is a valid JNDI name. The method then drops the EJB instance that was associated with the previous JNDI name and loads the EJB associated with the new JNDI name.

You can use this method to associate this control with an EJB that has the same interface as the previous EJB, but a different JNDI name.

Parameters

jndiName
: The JNDI name of the EJB which with this control should be associated.

DEPRECATED The `com.bea.control.EntityEJBControl` interface replaces this interface.

EntityEJBControl Interface

public interface ***EntityEJBControl***

extends `com.bea.control.EntityEJBControl`

Entity Enterprise Java Bean Control base interface

All Superinterfaces

`com.bea.control.Control`, `weblogic.jws.control.Control`, `weblogic.jws.control.EJBControl`, `com.bea.control.EntityEJBControl`, `Serializable`

Method Summary

Methods from interface `weblogic.jws.control.weblogic.jws.control.EJBControl`

`getEJBBeanInstance, getEJBException, getEJBHomeInstance, getJNDIName, hasEJBBeanInstance, setJNDIName`

Methods from interface `com.bea.control.com.bea.control.EntityEJBControl`

`getEJBNextBeanInstance`

DEPRECATED The `com.bea.control.JMSControl` interface replaces this interface.

JMSControl Interface

public interface ***JMSControl***

extends `com.bea.control.JMSControl`

Control used to send and receive JMS messages. Incoming messages are asynchronously delivered. JMSControl is an extensible control: developers may customize it via CTRL files.

All Superinterfaces
`com.bea.control.Control`, `com.bea.control.JMSControl`, `Serializable`,

Field Summary

Fields from interface com.bea.control.`com.bea.control.JMSControl`

`HEADER_CORRELATIONID, HEADER_DELIVERYMODE, HEADER_EXPIRATION, HEADER_MESSAGEID, HEADER_PRIORITY, HEADER_REDELIVERED, HEADER_TIMESTAMP, HEADER_TYPE`

Method Summary

Methods from interface `com.bea.control.com.bea.control.JMSControl`

`getHeaders, getProperties, getSession, setHeaders, setProperties, subscribe, unsubscribe`

DEPRECATED The `com.bea.control.JwsContext` interface replaces this interface.

JwsContext Interface

public interface ***JwsContext***

extends `com.bea.control.JwsContext`

The JwsContext interface provides access to JWS container services.

All Superinterfaces

`weblogic.jws.control.Context`, `com.bea.control.JwsContext`, `Serializable`

Nested Class Summary

public static interface	**`weblogic.jws.control.JwsContext.Callback`**

Nested classes from interface weblogic.jws.control.`weblogic.jws.control.Context`

`Context.Callback`

Nested classes from interface com.bea.control.`com.bea.control.JwsContext`

`JwsContext.Callback`

Method Summary

Methods from interface `weblogic.jws.control.weblogic.jws.control.Context`

`finishConversation, getCallerPrincipal, getCurrentAge, getCurrentIdleTime, getLogger, getMaxAge, getMaxIdleTime, getService, isCallerInRole, isFinished, resetIdleTime, setMaxAge, setMaxAge, setMaxIdleTime, setMaxIdleTime`

Methods from interface `com.bea.control.com.bea.control.JwsContext`

`getCallbackLocation, getCallbackPassword, getCallbackUsername, getInputHeaders, getProtocol, getUnderstoodInputHeaders, setCallbackLocation, setCallbackLocation, setCallbackPassword, setCallbackUsername, setOutputHeaders, setUnderstoodInputHeaders`

DEPRECATED Use the `com.bea.control.JwsContext.Callback` interface of the `com.bea.control.JwsContext` interface instead.

JwsContext.Callback Interface

public static interface ***JwsContext.Callback***

extends `com.bea.control.JwsContext.Callback`

All Superinterfaces

`weblogic.jws.control.Context.Callback`,
`com.bea.control.JwsContext.Callback`

Enclosing interface

`JwsContext`

Method Summary

Methods from interface `weblogic.jws.control.weblogic.jws.control.Context.Callback`

`onAgeTimeout, onAsyncFailure, onCreate, onException, onFinish, onIdleTimeout`

weblogic.jws.control

SchedulerException Class

public class ***SchedulerException***

extends `RuntimeException`

The SchedulerException is thrown to indicate a system error in control scheduling events.

Hierarchy

```
Object
 Throwable
  Exception
   RuntimeException
    SchedulerException
```

All Implemented Interfaces

`Serializable`

Constructor Summary

SchedulerException*(*`String` *message,* `Throwable` *t)*

SchedulerException*(*`String` *message)*

Method Summary

`public Throwable` ***`getNestedException`****()*

Methods from `java.lang.Throwable`

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

Methods from class `java.lang.Object`

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

Constructor Detail

SchedulerException

```
public SchedulerException(String message,
                          Throwable t)
```

SchedulerException

```
public SchedulerException(String message)
```

Method Detail

getNestedException() Method

```
public Throwable getNestedException()
```

DEPRECATED The `com.bea.control.ServiceControl` interface replaces this interface.

ServiceControl Interface

public interface ***ServiceControl***

extends `com.bea.control.ServiceControl`

Server Proxy Control base interface

All Superinterfaces

`com.bea.control.Control`, `Serializable`, `com.bea.control.ServiceControl`,

Nested Class Summary

Nested classes from interface com.bea.control.`com.bea.control.ServiceControl`

`ServiceControl.Callback`

Method Summary

Methods from interface `com.bea.control.com.bea.control.ServiceControl`

`getConversationID, getEndPoint, getInputHeaders, getPassword, getProtocol, getReliableMessageID, getUsername, reset, setClientCert, setConversationID, setEndPoint, setKeystore, setKeystore, setOutputHeaders, setPassword, setProtocol, setReliableMessageID, setUsername, useClientKeySSL`

DEPRECATED The `com.bea.control.ServiceControlException` class replaces this interface.

ServiceControlException Class

public class ***ServiceControlException***

extends `com.bea.control.ServiceControlException`

Hierarchy

```
Object
 Throwable
  Exception
   RuntimeException
    com.bea.control.ControlException
     weblogic.jws.control.ControlException
      com.bea.control.ServiceControlException
       ServiceControlException
```

All Implemented Interfaces

`Serializable`

Constructor Summary

ServiceControlException*(*`String` *msg,* `Throwable` *t)*

ServiceControlException*(*`String` *msg)*

Method Summary

Methods from `com.bea.control.com.bea.control.ServiceControlException`

`getSoapFault, hasSoapFault,`

Methods from `com.bea.control.com.bea.control.ControlException`

`getNestedException`

Methods from `java.lang.Throwable`

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

Methods from class `java.lang.Object`

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

Constructor Detail

ServiceControlException

```
public ServiceControlException(String msg,
                               Throwable t)
```

ServiceControlException

```
public ServiceControlException(String msg)
```

DEPRECATED The `com.bea.control.SessionEJBControl` interface replaces this interface.

SessionEJBControl Interface

public interface ***SessionEJBControl***

extends `com.bea.control.SessionEJBControl`

Session Enterprise Java Bean Control base interface

All Superinterfaces

`com.bea.control.Control`, `weblogic.jws.control.Control`, `weblogic.jws.control.EJBControl`, `Serializable`, `com.bea.control.SessionEJBControl`

Method Summary

Methods from interface `weblogic.jws.control.weblogic.jws.control.EJBControl`

`getEJBBeanInstance, getEJBException, getEJBHomeInstance, getJNDIName, hasEJBBeanInstance, setJNDIName`

DEPRECATED The `com.bea.control.TimerControl` interface replaces this interface.

TimerControl Interface

public interface ***TimerControl***

extends `com.bea.control.TimerControl`

Timer Control base interface

All Superinterfaces

`com.bea.control.Control`, `weblogic.jws.control.Control`, `Serializable`, `com.bea.control.TimerControl`

Nested Class Summary

Nested classes from interface com.bea.control.`com.bea.control.TimerControl`

`TimerControl.Callback`

Method Summary

Methods from interface `com.bea.control.com.bea.control.TimerControl`

`getCoalesceEvents, getRepeatsEvery, getTimeout, getTimeoutAt, restart, setCoalesceEvents, setRepeatsEvery, setRepeatsEvery, setTimeout, setTimeout, setTimeoutAt, start, stop`

DEPRECATED The `com.bea.control.TimerControlFactory` interface replaces this interface.

TimerControlFactory Interface

public interface ***TimerControlFactory***

extends `com.bea.control.TimerControlFactory`

The base factory interface for TimerControl

All Superinterfaces

`com.bea.control.ControlFactory`, `weblogic.jws.control.ControlFactory`, `Serializable`, `com.bea.control.TimerControlFactory`

Method Summary

Methods from interface `com.bea.control.com.bea.control.TimerControlFactory`

`create`

weblogic.jws.util Package

Interface Summary

Logger
DEPRECATED as of WebLogic Platform 8.1, use `com.bea.wlw.util.Logger`

Logger Interface

public interface ***Logger***

extends `com.bea.wlw.util.Logger`

Related Topics

`Logger`

All Superinterfaces

`com.bea.wlw.util.Logger`

Method Summary

Methods from interface `com.bea.wlw.util.com.bea.wlw.util.Logger`

`debug, debug, error, error, info, info, isDebugEnabled, isErrorEnabled, isInfoEnabled, isWarnEnabled, warn, warn`