EJB 3.0 Metadata Annotations Reference

The following topics provide reference information about the EJB 3.0 metadata annotations:

Overview of EJB 3.0 Annotations

The new EJB 3.0 programming model uses the JDK 5.0 metadata annotations feature in which you create an annotated EJB 3.0 bean file and then use the WebLogic compile tool `weblogic.appc` (or its Ant equivalent `wlappc`) to compile the bean file into a Java class file and generate the associated EJB artifacts, such as the required EJB interfaces and deployment descriptors.

The following sections provide reference information for the metadata annotations you can specify in the EJB bean file. Some of the annotations are in the `javax.ejb` package, and are thus specific to EJBs; others are more common and are used by other Java Platform, Enterprise Edition (Java EE) Version 5 components, and are thus in more generic packages, such as `javax.annotation`.

Annotations for Stateless, Stateful, and Message-Driven Beans

This section provides reference information for the following annotations:

- javax.ejb.Remote
- javax.ejb.RemoteHome
- javax.ejb.Remove
- javax.ejb.Stateful
- javax.ejb.Stateless
- javax.ejb.Timeout
- javax.ejb.TransactionAttribute
- javax.ejb.TransactionManagement

javax.ejb.ActivationConfigProperty

Description

Target: Any

Specifies properties used to configure a message-driven bean in its operational environment. This may include information about message acknowledgement modes, message selectors, expected destination or endpoint types, and so on.

This annotation is used only as a value to the `activationConfig` attribute of the `@javax.ejb.MessageDriven` annotation.

Attributes

Table A-1 Attributes of the javax.ejb.ActivationConfigProperty Annotation

Name	Description	Data Type	Required?
propertyName	Specifies the name of the activation property.	String	Yes
propertyValue	Specifies the value of the activation property.	String	Yes

javax.ejb.ApplicationException

Description

Target: Class

Specifies that an exception is an application exception and that it should be reported to the client application directly, or unwrapped.

This annotation can be applied to both checked and unchecked exceptions.

Attributes

Table A-2 Attributes of the javax.ejb.ApplicationException Annotation

Name	Description	Data Type	Required?
rollback	Specifies whether the EJB container should rollback the transaction, if the bean is currently being invoked inside of one, if the exception is thrown. Valid values for this attribute are `true` and `false`. Default value is `false`, or the transaction should *not* be rolled back.	boolean	No.

javax.ejb.EJB

Description

Target: Class, Method, Field

Specifies a dependency or reference to an EJB business or home interface.

You annotate a bean's instance variable with the `@EJB` annotation to specify a dependence on another EJB. WebLogic Server automatically initializes the annotated variable with the reference to the EJB on which it depends; this is also called *dependency injection*. This initialization occurs before any of the bean's business methods are invoked and after the bean's `EJBContext` is set.

You can also annotate a setter method in the bean class; in this case WebLogic Server uses the setter method itself when performing dependency injection. This is an alternative to instance variable dependency injection.

If you apply the annotation to a class, the annotation declares the EJB that the bean will look up at runtime.

Whether using variable or setter method injection, WebLogic Server determines the name of the referenced EJB by either the name or data type of the annotated instance variable or setter method parameter. If there is any ambiguity, you should use the `beanName` or `mappedName` attributes of the `@EJB` annotation to explicitly name the dependent EJB.

Attributes

Table A-3 Attributes of the javax.ejb.EJB Annotation

Name	Description	Data Type	Required?
name	Specifies the name by which the referenced EJB is to be looked up in the environment.	String	No
beanInterface	Specifies the interface type of the referenced EJB (either a business or home interface). Default value for this attribute is `Object.class`	Class	No
beanName	Specifies the name of the referenced EJB. This attribute corresponds to the `name` element of the `@Stateless` or `@Stateful` annotation in the referenced EJB, which by default is the unqualified name of the referenced bean class. This attribute is most useful when multiple session beans in an EJB JAR file implement the same interface, because the name of each bean must be unique.	String	No
mappedName	Specifies the global JNDI name of the referenced EJB. For example: `mappedName="bank.Account"` specifies that the referenced EJB has a global JNDI name of `bank.Account` and is deployed in the WebLogic Server JNDI tree. **Note:** EJBs that use mapped names may not be portable.	String	No
description	Describes the EJB reference.	String	No

javax.ejb.EJBs

Description

Target: Class

Specifies an array of `@javax.ejb.EJB` annotations.

Attribute

Table A-4 Attribute of the javax.ejb.EJBs Annotation

Name	Description	Data Type	Required?
value	Specifies the array of `@javax.ejb.EJB` annotations	EJB[]	No

javax.ejb.Init

Description

Target: Method

Specifies the correspondence of a method in the bean class with a `createMETHOD` method for an adapted EJB 2.1 `EJBHome` or `EJBLocalHome` client view.

This annotation is used only in conjunction with stateful session beans, or those that have been annotated with the `@javax.ejb.Stateful` class-level annotation,

The return type of a method annotated with the `@javax.ejb.Init` annotation must be `void`, and its parameter types must be exactly the same as those of the referenced `createMETHOD` method or methods.

The @Init annotation is required only for stateful session beans that provide a Remote-Home or LocalHome interface. You must specify the name of the adapted create method of the Home or LocalHome interface, using the `value` attribute, if there is any ambiguity.

Attributes

Table A-5 Attributes of the javax.ejb.Init Annotation

Name	Description	Data Type	Required?
value	Specifies the name of the corresponding `createMETHOD` method. This attribute is required only when the `@Init` annotation is used to associate an adapted `Home` interface of a stateful session bean that has more than one `create<METHOD>` method.	String	No.

javax.ejb.Local

Description

Target: Class

Specifies the local interface or interfaces of a session bean. The local interface exposes business logic to local clients—those running in the same application as the EJB. It defines the business methods a local client can call.

You are required to specify this annotation if your bean class implements more than a single interface, not including the following:

- `java.io.Serializable`
- `java.io.Externalizable`
- `javax.ejb.*`

This annotation applies only to stateless or stateful session beans.

Attributes

Table A-6 Attributes of the javax.ejb.Local Annotation

Name	Description	Data Type	Required?
value	Specifies the list of local interfaces as an array of classes. You are required to specify this attribute only if your bean class implements more than a single interface, not including the following: ▪ `java.io.Serializable` ▪ `java.io.Externalizable` ▪ `javax.ejb.*`	Class[]	No.

javax.ejb.LocalHome

Description

Target: Class

Specifies the local home interface of the bean class.

The local home interface provides methods that local clients—those running in the same application as the EJB—can use to create, remove, and in the case of an entity bean, find instances of the bean. The local home interface also has *home methods*—business logic that is not specific to a particular bean instance.

This attribute applies only to stateless and stateful session beans.

You typically specify this attribute only if you are going to provide an adapted EJB 2.1 component view of the EJB 3.0 bean. You can also use this annotation with bean classes that have been written to the EJB 2.1 APIs.

Attributes

Table A-7 Attributes of the javax.ejb.LocalHome Annotation

Name	Description	Data Type	Required?
value	Specifies the local home class.	Class	Yes.

javax.ejb.MessageDriven

Description

Target: Class

Specifies that the Enterprise JavaBean is a message-driven bean.

Attributes

Table A-8 Attributes of the javax.ejb.MessageDriven Annotation

Name	Description	Data Type	Required?
name	Specifies the name of the message-driven bean. If you do not specify this attribute, the default value is the unqualified name of the bean class.	String	No.
messageListenerInterface	Specifies the message-listener interface of the bean class. You must specify this attribute if the bean class does not explicitly implement the message-	Class	No.

	listener interface, or if the bean class implements more than one interface other than `java.io.Serializable`, `java.io.Externalizable`, or any of the interfaces in the `javax.ejb` package. The default value for this attribute is `Object.class`.		
activationConfig	Specifies the configuration of the message-driven bean in its operational environment. This may include information about message acknowledgement modes, message selectors, expected destination or endpoint types, and so on. You specify activation configuration information using an Array of `@javax.ejb.ActivationConfigProperty` annotation, specify the property name and value.	ActivationConfigProperty[]	No.
mappedName	Specifies the product-specific name to which the message-driven bean should be mapped. You can also use this attribute to specify the JNDI name of the message destination of this message-driven bean. For example: `mappedName="my.Queue"` specifies that this message-driven bean is associated with a JMS queue, whose JNDI name is `my.Queue` and is deployed in the WebLogic Server JNDI tree. **Note:** If you specify this attribute, the message-driven bean may not be portable.	String	No
description	Specifies a description of the message-driven bean.	String	No

javax.ejb.PostActivate

Description

Target: Method

Specifies the lifecycle callback method that signals that the EJB container has just reactivated the bean instance.

This annotation applies only to stateful session beans. Because the EJB container automatically maintains the conversational state of a stateful session bean instance when it is passivated, you do not need to specify this annotation for most stateful session beans. You only need to use this annotation, along with its partner `@PrePassivate`, if you want to allow your stateful session bean to maintain the open resources that need to be closed prior to a bean instance's passivation and then reopened during the bean instance's activation.

Only one method in the bean class can be annotated with this annotation. If you annotate more than one method with this annotations, the EJB will not deploy.

The method annotated with `@PostActivate` must follow these requirements:

- The return type of the method must be `void`.
- The method must not throw a checked exception.
- The method may be `public`, `protected`, `package private` or `private`.
- The method must not be `static`.
- The method must not be `final`.

This annotation does not have any attributes.

javax.ejb.PrePassivate

Description

Target: Method

Specifies the lifecycle callback method that signals that the EJB container is about to passivate the bean instance.

This annotation applies only to stateful session beans. Because the EJB container automatically maintains the conversational state of a stateful session bean instance when it is passivated, you do not need to specify this annotation for most stateful session beans. You only need to use this annotation, along with its partner `@PostActivate`, if you want to allow your stateful session bean to maintain the open resources that need to be closed prior to a bean instance's passivation and then reopened during the bean instance's activation.

Only one method in the bean class can be annotated with this annotation. If you annotate more than one method with this annotations, the EJB will not deploy.

The method annotated with `@PrePassivate` must follow these requirements:

- The return type of the method must be `void`.
- The method must not throw a checked exception.
- The method may be `public`, `protected`, `package private` or `private`.
- The method must not be `static`.
- The method must not be `final`.

This annotation does not have any attributes.

javax.ejb.Remote

Description

Target: Class

Specifies the remote interface or interfaces of a session bean. The remote interface exposes business logic to remote clients—clients running in a separate application from the EJB. It defines the business methods a remote client can call.

This annotation applies only to stateless or stateful session beans.

Attributes

Table A-9 Attributes of the javax.ejb.Remote Annotation

Name	Description	Data Type	Required?
value	Specifies the list of remote interfaces as an array of classes. You are required to specify this attribute only if your bean class implements more than a single interface, not including the following: ▪ `java.io.Serializable` ▪ `java.io.Externalizable` ▪ `javax.ejb.*`	Class[]	No.

javax.ejb.RemoteHome

Description

Target: Class

Specifies the remote home interface of the bean class.

The remote home interface provides methods that remote clients—those running in a separate application from the EJB—can use to create, remove, and find instances of the bean.

This attribute applies only to stateless and stateful session beans.

You typically specify this attribute only if you are going to provide an adapted EJB 2.1 component view of the EJB 3.0 bean. You can also use this annotation with bean classes that have been written to the EJB 2.1 APIs.

Attributes

Table A-10 Attributes of the javax.ejb.RemoteHome Annotation

Name	Description	Data Type	Required?
value	Specifies the remote home class.	Class	Yes.

javax.ejb.Remove

Description

Target: Method

Use the `@javax.ejb.Remove` annotation to denote a remove method of a stateful session bean.

When the method completes, the EJB container will invoke the method annotated with the `@javax.annotation.PreDestroy` annotation, if any, and then destroy the stateful session bean.

Attributes

Table A-11 Attributes of the javax.ejb.Remove Annotation

Name	Description	Data Type	Required?
retainIfException	Specifies that the container should not remove the stateful session bean if the annotated method terminates abnormally with an application exception. Valid values are `true` and `false`. Default value is `false`.	boolean	No.

javax.ejb.Stateful

Description

Target: Class

Specifies that the Enterprise JavaBean is a stateful session bean.

Attributes

Table A-12 Attributes of the javax.ejb.Stateful Annotation

Name	Description	Data Type	Required?
name	Specifies the name of the stateful session bean.	String	No.

	If you do not specify this attribute, the default value is the unqualified name of the bean class.		
mappedName	Specifies the product-specific name to which the stateful session bean should be mapped. You can also use this attribute to specify the JNDI name of this stateful session bean. WebLogic Server uses the value of the `mappedName` attribute when creating the bean's global JNDI name. In particular, the JNDI name will be: *`mappedName#name_of_businessInterface`* where *`name_of_businessInterface`* is the fully qualified name of the business interface of this session bean. For example, if you specify `mappedName="bank"` and the fully qualified name of the business interface is `com.CheckingAccount`, then the JNDI of the business interface is `bank#com.CheckingAccount`. **Note:** If you specify this attribute, the stateful session bean may not be portable.	String	No.
description	Describes the stateful session bean.	String	No.

javax.ejb.Stateless

Description

Target: Class

Specifies that the Enterprise JavaBean is a stateless session bean.

Attributes

Table A-13 Attributes of the javax.ejb.Stateless Annotation

Name	Description	Data Type	Required?
name	Specifies the name of the stateless session bean. If you do not specify this attribute, the default value is the unqualified name of the bean class.	String	No.
mappedName	Specifies the product-specific name to which the stateless session bean should be mapped. You can also use this attribute to specify the JNDI name of this stateless session bean. WebLogic Server uses the value of the `mappedName` attribute when creating the bean's global JNDI name. In particular, the JNDI name will be: *`mappedName#name_of_businessInterface`* where *`name_of_businessInterface`* is the fully qualified name of the business interface of this session bean. For example, if you specify `mappedName="bank"` and the fully qualified name of the business interface is `com.CheckingAccount`, then the JNDI of the business interface is `bank#com.CheckingAccount`. **Note:** If you specify this attribute, the stateless session bean may not be portable.	String	No.
description	Describes the stateless session bean.	String	No.

javax.ejb.Timeout

Description

Target: Method

Specifies the timeout method of the bean class.

This annotation makes it easy to program an EJB timer service in your bean class. The EJB timer service is an EJB-container provided service that allows you to create timers that schedule callbacks to occur when a timer object expires.

Previous to EJB 3.0, your bean class was required to implement `javax.ejb.TimedObject` if you wanted to program the timer service. Additionally, your bean class had to include a method with the exact name `ejbTimeout`. These requirements are relaxed in Version 3.0 of EJB. You no longer are required to implement the `javax.ejb.TimedObject` interface, and you can name your timeout method anything you want, as long as you annotate it with the `@Timeout` annotation. You can, however, continue to use the pre-3.0 way of programming the timer service if you want.

For details, see Programming the EJB Timer Service.

This annotation does not have any attributes.

javax.ejb.TransactionAttribute

Description

Target: Class, Method

Specifies whether the EJB container invokes an EJB business method within a transaction context.

WARNING: If you specify this annotation, you are also required to use the `@TransactionManagement` annotation to specify container-managed transaction demarcation.

You can specify this annotation on either the bean class, or a particular method of the class that is also a method of the business interface. If specified at the bean class, the annotation applies to all applicable business interface methods of the class. If specified for a particular method, the annotation applies to that method only. If the annotation is specified at both the class and the method level, the method value overrides if the two disagree.

If you do not specify the `@TransactionAttribute` annotation in your bean class, and the bean uses container managed transaction demarcation, the semantics of the REQUIRED transaction attribute are assumed.

Attributes

Table A-14 Attributes of the javax.ejb.TransactionAttribute Annotation

Name	Description	Data Type	Required?
value	Specifies how the EJB container manages the transaction boundaries when invoking a business method. For details about these values, see the description of the trans-attribute element in the Container-Managed Transactions Elements table. Valid values for this attribute are: ▪ `TransactionAttributeType.MANDATORY` ▪ `TransactionAttributeType.REQUIRED` ▪ `TransactionAttributeType.REQUIRED_NEW` ▪ `TransactionAttributeType.SUPPORTS` ▪ `TransactionAttributeType.NOT_SUPPORTED` ▪ `TransactionAttributeType.NEVER` Default value is `TransactionAttributeType.REQUIRED`.	TransactionAttributeType	No.

javax.ejb.TransactionManagement

Description

Target: Class

Specifies the transaction management demarcation type of the session bean or message-driven bean.

A transaction is a unit of work that changes application state—whether on disk, in memory or in a database—that, once started, is completed entirely, or not at all. Transactions can be demarcated—started, and ended with a commit or rollback—by the EJB container, by bean code, or by client code. This annotation specifies whether the EJB container or the user-written bean code manages the demarcation of a transaction.

If you do not specify this annotation in your bean class, it is assumed that the bean has container-managed transaction demarcation.

For additional information about transactions, see Transaction Design and Management Options.

Attributes

Table A-15 Attributes of the javax.ejb.TransactionManagement Annotation

Name	Description	Data Type	Required?
value	Specifies the transaction management demarcation type used by the bean class. Valid values for this attribute are: ▪ `TransactionManagementType.CONTAINER` ▪ `TransactionManagementType.BEAN` Default value is `TransactionManagementType.CONTAINER`	TransacationManagementType	No.

Annotations Used to Configure Interceptors

This section provides reference information for the following annotations:

- javax.interceptor.AroundInvoke
- javax.interceptor.ExcludeClassInterceptors
- javax.interceptor.ExcludeDefaultInterceptors
- javax.interceptor.Interceptors

javax.interceptor.AroundInvoke

Description

Target: Method

Specifies the business method interceptor for either a bean class or an interceptor class.

You can annotate only *one* method in the bean class or interceptor class with the `@AroundInvoke` annotation; the method cannot be a business method of the bean class.

This annotation does not have any attributes.

javax.interceptor.ExcludeClassInterceptors

Description

Target: Method

Specifies that any class-level interceptors should not be invoked for the annotated method. This does not include default interceptors, whose invocation are excluded only with the `@ExcludeDefaultInterceptors` annotation.

This annotation does not have any attributes.

javax.interceptor.ExcludeDefaultInterceptors

Description

Target: Class, Method

Specifies that any defined default interceptors (which can be specified only in the EJB deployment descriptors, and not with annotations) should not be invoked.

If defined at the class-level, the default interceptors are never invoked for any of the bean's business methods. If defined at the method-level, the default interceptors are never invoked for the particular business method, but they are invoked for all other business methods that do not have the `@ExludeDefaultInterceptors` annotation.

This annotation does not include any attributes.

javax.interceptor.Interceptors

Description

Target: Class, Method

Specifies the interceptor classes that are associated with the bean class or method. An interceptor class is a class —distinct from the bean class itself—whose methods are invoked in response to business method invocations and/or lifecycle events on the bean.

The interceptor class can include both an business interceptor method (annotated with the `@javax.interceptor.AroundInvoke` annotation) and lifecycle callback methods (annotated with the `@javax.annotation.PostConstruct`, `@javax.annotation.PreDestroy`, `@javax.ejb.PostActivate`, and `@javax.ejb.PrePassivate` annotations).

Any number of interceptor classes may be defined for a bean class. If more than one interceptor class is defined, they are invoked in the order they are specified in the annotation.

If the annotation is specified at the class-level, the interceptors apply to all business methods of the EJB. If specified at the method-level, the interceptors apply to just that method. You can specify the same interceptor class to more than one method of the bean class. By default, method-level interceptors are invoked after all applicable interceptors (default interceptors, class-level interceptors, and so on).

Attributes

Table A-16 Attributes of the javax.interceptor.Interceptors Annotation

Name	Description	Data Type	Required?
value	Specifies the array of interceptor classes. If there is more than one interceptor class in the array, the order in which they are listed defines the order in which they are invoked.	Class[]	Yes

Annotations Used to Interact With Entity Beans

This section provides reference information about the following annotations:

- javax.persistence.PersistenceContext
- javax.persistence.PersistenceContexts
- javax.persistence.PersistenceUnit

- javax.persistence.PersistenceUnits

javax.persistence.PersistenceContext

Description

Target: Class, Method, Field

Specifies a dependency on a container-managed `EntityManager` persistence context.

You use this annotation to interact with a 3.0 entity bean, typically by performing dependency injection into an `EntityManager` instance.

The `EntityManager` interface defines the methods that are used to interact with the persistence context. A persistence context is a set of entity instances; an entity is a lightweight persistent domain object. The `EntityManager` API is used to create and remove persistent entity instances, to find entities by their primary key, and to query over entities.

Attributes

Table A-17 Attributes of the javax.pesistence.PersistenceContext Annotation

Name	Description	Data Type	Required?
name	Specifies the name by which the `EntityManager` and its persistence unit are to be known within the context of the session or message-driven bean. You only need to specify this attribute if you use a JNDI lookup to obtain an `EntityManager`; if you use dependency injection, then you do not need to specify this attribute.	String	No.
unitName	Specifies the name of the persistence unit. If you specify a value for this attribute that is the same as the name of a persistence unit in the `persistence.xml` file, the EJB container automatically deploys the persistence unit and sets its JNDI name to its persistence unit name. Similarly, if you do not specify this attribute, but the name of the variable into which you are injecting the persistence context information is the same as the name of a persistence unit in the `persistence.xml` file, then the EJB container again automatically deploys the persistence unit with its JNDI name equal to its unit name. **Note:** The `persistence.xml` file is an XML file, located in the `META-INF` directory of the EJB JAR file, that specifies the database used with the entity beans and specifies the default behavior of the `EntityManager`. You must specify this attribute if there is more than one persistence unit within the referencing scope.	String	No.
type	Specifies whether the lifetime of the persistence context is scoped to a transaction or whether it extends beyond that of a single transaction. Valid values for this attribute are: ▪ `PersistenceContextType.TRANSACTION` ▪ `PersistenceContextType.EXTENDED` Default value is `PersistenceContextType.TRANSACTION`.	`PersistenceContextType`	No.

javax.persistence.PersistenceContexts

Description

Target: Class

Specifies an array of `@javax.persistence.PersistencContext` annotations.

Attributes

Table A-18 Attributes of the javax.pesistence.PersistenceContexts Annotation

Name	Description	Data Type	Required?
value	Specifies the array of `@javax.persistence.PersistencContext` annotations.	`PersistenceContext[]`	Yes.

javax.persistence.PersistenceUnit

Description

Target: Class, Method, Field

Specifies a dependency on an `EntityManagerFactory` object.

You use this annotation to interact with a 3.0 entity bean, typically by performing dependency injection into an `EntityManagerFactory` instance. You can then use the `EntityManagerFactory` to create one or more `EntityManager` instances. This annotation is similar to the `@PersistenceContext` annotation, except that it gives you more control over the life of the `EntityManager` because you create and destroy it yourself, rather than let the EJB container do it for you.

The `EntityManager` interface defines the methods that are used to interact with the persistence context. A persistence context is a set of entity instances; an entity is a lightweight persistent domain object. The `EntityManager` API is used to create and remove persistent entity instances, to find entities by their primary key, and to query over entities.

Attributes

Table A-19 Attributes of the javax.pesistence.PersistenceUnit Annotation

Name	Description	Data Type	Required?
name	Specifies the name by which the `EntityManagerFactory` is to be known within the context of the session or message-driven bean You are not required to specify this attribute if you use dependency injection, only if you also use JNDI to look up information.	String	No
unitName	Refers to the name of the persistence unit as defined in the `persistence.xml` file. This file is an XML file, located in the `META-INF` directory of the EJB JAR file, that specifies the database used with the entity beans and specifies the default behavior of the `EntityManager`. If you set this attribute, the EJB container automatically deploys the referenced persistence unit and sets its JNDI name to its persistence unit name. Similarly, if you do not specify this attribute, but the name of the variable into which you are injecting the persistence context information is the same as the name of a persistence unit in the `persistence.xml` file, then the EJB container again automatically deploys the persistence unit with its JNDI name equal to its unit name. You are required to specify this attribute only if there is more than one persistence unit in the referencing scope.	String	No

javax.persistence.PersistenceUnits

Description

Target: Class

Specifies an array of `@javax.persistence.PersistenceUnit` annotations.

Attributes

Table A-20 Attributes of the javax.pesistence.PersistenceUnits Annotation

Name	Description	Data Type	Required?
value	Specifies the array of `@javax.persistence.PersistenceUnit` annotations.	`PersistenceUnit[]`	Yes

Standard JDK Annotations Used By EJB 3.0

This section provides reference information about the following annotations:

- javax.annotation.PostConstruct
- javax.annotation.PreDestroy
- javax.annotation.Resource
- javax.annotation.Resources

javax.annotation.PostConstruct

Description

Target: Method

Specifies the lifecycle callback method that the EJB container should execute before the first business method invocation and after dependency injection is done to perform any initialization.

You may specify a `@PostConstruct` method in any bean class that includes dependency injection.

Only one method in the bean class can be annotated with this annotation. If you annotate more than one method with this annotations, the EJB will not deploy.

The method annotated with `@PostConstruct` must follow these requirements:

- The return type of the method must be `void`.
- The method must not throw a checked exception.
- The method may be `public`, `protected`, `package private` or `private`.
- The method must not be `static`.
- The method must not be `final`.

This annotation does not have any attributes.

javax.annotation.PreDestroy

Description

Target: Method

Specifies the lifecycle callback method that signals that the bean class instance is about to be destroyed by the EJB container. You typically apply this annotation to methods that release resources that the bean class has been holding.

Only one method in the bean class can be annotated with this annotation. If you annotate more than one method with this annotations, the EJB will not deploy.

The method annotated with `@PreDestroy` must follow these requirements:

- The return type of the method must be `void`.
- The method must not throw a checked exception.
- The method may be `public`, `protected`, `package private` or `private`.
- The method must not be `static`.
- The method must not be `final`.

This annotation does not have any attributes.

javax.annotation.Resource

Description

Target: Class, Method, Field

Specifies a dependence on an external resource, such as a JDBC data source or a JMS destination or connection factory.

If you specify the annotation on a field or method, the EJB container injects an instance of the requested resource into the bean when the bean is initialized. If you apply the annotation to a class, the annotation declares a resource that the bean will look up at runtime.

Attributes

Table A-21 Attributes of the javax.annotation.Resource Annotation

Name	Description	Data Type	Required?
name	Specifies the name of the resource reference. If you apply the `@Resource` annotation to a field, the default value of the `name` attribute is the field name, qualified by the class name. If you apply it to a method, the default value is the JavaBeans property name corresponding to the method, qualified by the class name. If you apply the annotation to class, there is no default value and thus you are required to specify the attribute.	String	No
type	Specifies the Java data type of the resource. If you apply the `@Resource` annotation to a field, the default value of the `type` attribute is the type of the field. If you apply it to a method, the default is the type of the JavaBeans property. If you apply it to a class, there is no default value and thus you are required to specify this attribute.	Class	No
authenticationType	Specifies the authentication type to use for the resource. You specify this attribute only for resources representing a connection factory of any supported type. Valid values for this attribute are: ▪ `AuthenticationType.CONTAINER` ▪ `AuthenticationType.APPLICATION` Default value is `AuthenticationType.CONTAINER`	AuthenticationType	No
shareable	Indicates whether a resource can be shared between this EJB and other EJBs. You specify this attribute only for resources representing a connection factory of any supported type or ORB object instances.	boolean	No.

	Valid values for this attribute are `true` and `false`. Default value is `true`.		
mappedName	Specifies the global JNDI name of the dependent resource. For example: `mappedName="my.Datasource"` specifies that the JNDI name of the dependent resources is `my.Datasource` and is deployed in the WebLogic Server JNDI tree.	String	No.
description	Specifies a description of the resource.	String	No.

javax.annotation.Resources

Description

Target: Class

Specifies an array of `@Resource` annotations.

Attributes

Table A-22 Attributes of the javax.annotation.Resources Annotation

Name	Description	Data Type	Required?
value	Specifies the array of `@Resource` annotations.	`Resource[]`	Yes.

Standard Security-Related JDK Annotations Used by EJB 3.0

This section provides reference information about the following annotations:

- javax.annotation.security.DeclareRoles
- javax.annotation.security.DenyAll
- javax.annotation.security.PermitAll
- javax.annotation.security.RolesAllowed
- javax.annotation.security.RunAs

javax.annotation.security.DeclareRoles

Description

Target: Class

Defines the security roles that will be used in the EJB.

You typically use this annotation to define roles that can be tested from within the methods of the annotated class, such as using the `isUserInRole` method. You can also use the annotation to explicitly declare roles that are implicitly declared if you use the `@RolesAllowed` annotation on the class or a method of the class.

You create security roles in WebLogic Server using the Administration Console. For details, see Manage Security Roles.

Attributes

Table A-23 Attributes of the javax.annotation.security.DeclareRoles Annotation

Name	Description	Data Type	Required?
value	Specifies an array of security roles that will be used in the bean class.	String[]	Yes.

javax.annotation.security.DenyAll

Description

Target: Method

Specifies that no security role is allowed to access the annotated method, or in other words, the method is excluded from execution in the EJB container.

This annotation does not have any attributes.

javax.annotation.security.PermitAll

Description

Target: Method

Specifies that all security roles currently defined for WebLogic Server are allowed to access the annotated method.

This annotation does not have any attributes.

javax.annotation.security.RolesAllowed

Description

Target: Class, Method

Specifies the list of security roles that are allowed to access methods in the EJB.

If you specify it at the class-level, then it applies to all methods in the bean class. If you specify it at the method-level, then it only applies to that method. If you specify the annotation at both the class- and method-level, the method value overrides the class value.

You create security roles in WebLogic Server using the Administration Console. For details, see Manage Security Roles.

Attributes

Table A-24 Attributes of the javax.annotation.security.RolesAllowed Annotation

Name	Description	Data Type	Required?
value	List of security roles that are allowed to access methods of the bean class.	String[]	Yes.

javax.annotation.security.RunAs

Description

Target: Class

Specifies the security role which actually executes the EJB in the EJB container.

The security role must exist in the WebLogic Server security realm and map to a user or group. For details, see Manage Security Roles.

Attributes

Table A-25 Attributes of the javax.annotation.security.RunAs Annotation

Name	Description	Data Type	Required?
value	Specifies the security role which the EJB should run as.	String	Yes.

back to top previous next