



IBM Software Group

# IBM Rational Application Developer V7.5

## *Features for EJB 3.0 development*



@business on demand.

© 2008 IBM Corporation  
Updated April 21, 2015

This presentation will discuss the new features of Rational® Application Developer version 7.5 that support development of enterprise Java™ beans version 3.0.

## Agenda

- Getting started with EJB 3.0 projects
- Creating EJB 3.0 beans
- Working with EJB 3.0 bean annotations
- The EJB 3.0 deployment descriptor
- The EJB binding editor



This presentation will start by discussing how to get started with EJB 3.0 projects. It will then discuss creating EJB 3.0 beans and working with EJB 3.0 annotations. Finally, it will discuss the now optional EJB 3.0 deployment descriptor, and the EJB binding editor.

## Creating an EJB 3.0 project

- EJB 3.0 projects are created using the new project wizard
- The EJB project creation wizard now defaults to EJB 3.0 support
- EJB 1.1, 2.0, 2.1 are still supported and available as options in the drop down list
- Do not create an EJB 1.1, 2.0, 2.1 bean in an EJB 3.0 project
  - ▶ You cannot create an EJB 3.0 bean in a non 3.0 compliant project



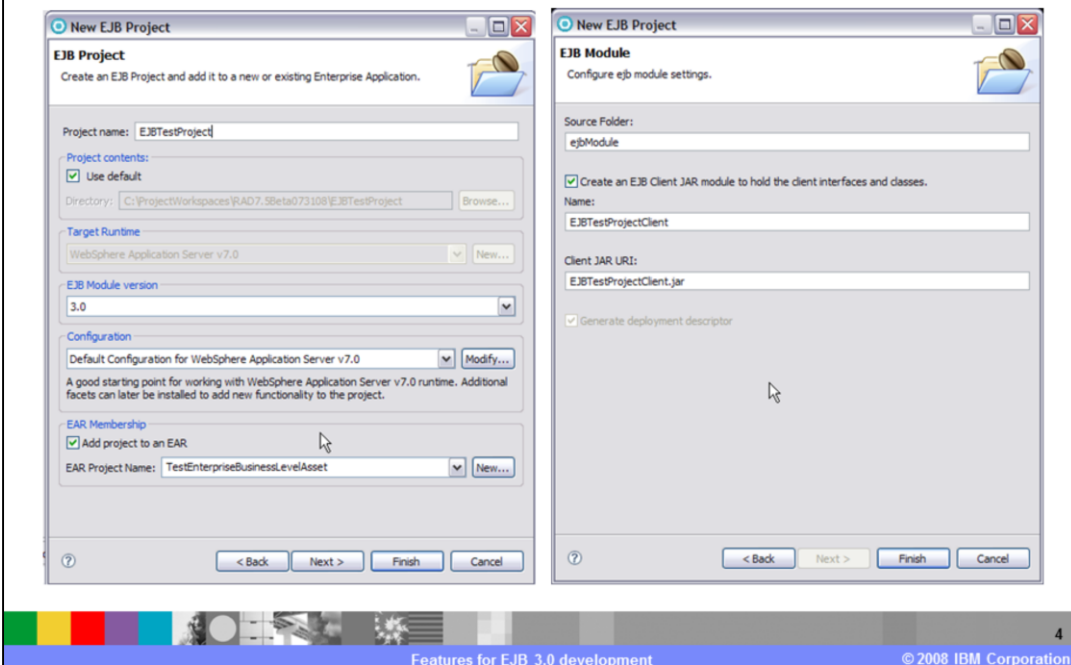
Creating an EJB 3.0 project in Rational Application Developer version 7.5 is very similar to previous versions of Rational Application Developer.

EJB 3.0 projects are created using the new project wizard.

The EJB project creation wizard now defaults to create an EJB project to the 3.0 specification, however a dropdown menu exists if you choose to create an EJB project that conforms to a previous version of EJBs.

It is worth noting that it is not recommended that you create a bean of version 1 or 2 in an EJB 3.0 project. Furthermore, you cannot create an enterprise bean that conforms to the 3.0 specification in a project that is designated for version 1 or 2 EJBs.

## The EJB project creation wizard



Here are screen captures of the EJB project creation wizard. You can see in the left screen capture that the default EJB level is 3.0, with a dropdown menu available to change to a previous version of EJB level if needed. You can also see that in the right pane by default the check box is checked to create an EJB Client JAR module to hold client interfaces and classes. Since this is checked, the deployment descriptor will be created by default for the EJB project. If you uncheck the option to create a client JAR, the “Generate Deployment Descriptor” option becomes enabled and you can uncheck the box to prevent a deployment descriptor from being created.

For EJB 3.0, deployment descriptors have become optional. You are not required to create a deployment descriptor in order to use and deploy an EJB project, and if you uncheck the option to create a deployment descriptor, you can have one generated later using the pop up menu options on the EJB project in the project explorer view.

## Creating and working with EJBs

- Enterprise beans can be created in several ways using the EJB 3.0 support in Rational Application Developer 7.5
  - ▶ POJOs
    - EJB 3.0 is based on the plain old Java™ object model
    - You can create a regular Java class and add annotations to make it an EJB
  - ▶ The new pop-up menu
    - Right clicking on the EJB 3.0 compliant project provides access to the **new > session bean** and **new > message driven bean** options
    - Options in the menu for creating entity beans for 1.X and 2.X specifications are available as well



There are several options to create an Enterprise Java Bean using Rational Application Developer 7.5.

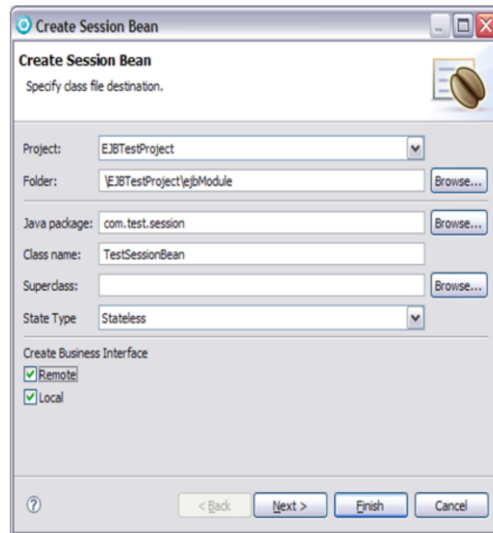
You can use the traditional method of using the pop up menu on an EJB project to create a new Session or Message driven bean by right clicking on the EJB project and selecting new session bean or new message driven bean.

There is also a new way of creating an EJB. You can begin by creating a regular Java object in the EJB project and add annotations to it to make it an EJB.

This is allowed now since EJB 3.0 is based on the plain old Java object model.

## New session bean wizard

- Right clicking on the EJB 3.0 project in the project explorer view and selecting **new > session bean** opens the new session bean wizard
- The option for stateless or Stateful bean creation is available in the dropdown menu
- The option to create remote and local business interfaces are available as check boxes



6

Features for EJB 3.0 development

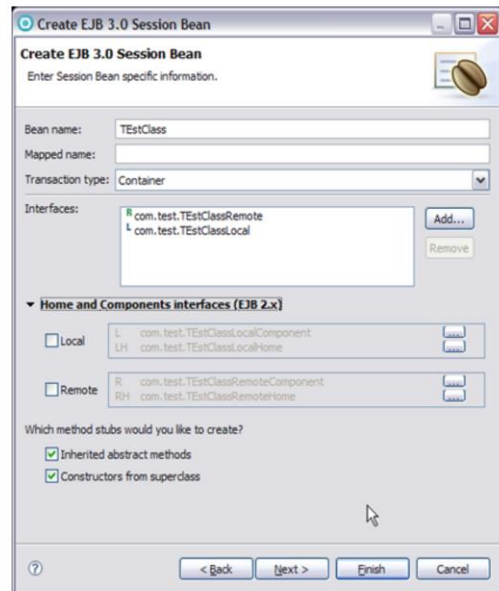
© 2008 IBM Corporation

The new session bean wizard has not changed very much from previous versions of Rational Application Developer.

In the session bean creation wizard you still have the option to name the EJB class, specify the package in which it will reside, and what the state type will be, either stateless or stateful beans. Check boxes exist at the bottom of the first page of the wizard to allow you to specify whether to create remote and local business interfaces.

## New session bean wizard

- The second screen in the bean creation wizard allows the option of overriding the default mapped name
  - ▶ By default the name is the same as the class name
- This can also be done later while editing the EJB 3.0 annotations
- The option to create home and component interfaces for EJB 2.X compliance are also available



Shown here in the screen capture is the second page of the session bean creation wizard.

You will notice that there is an entry box to allow you to specify the mapped name for the EJB if you want to override the default mapped name.

You may specify the transaction type here using the dropdown menu to select either container or bean transaction types.

By default the options to create local and remote interfaces for EJB 2 compliance are not checked, but they can be created by checking the options under the home and components interfaces section title.

By default you can see in the screen capture that the options to create method stubs for inherited abstract methods and constructors from the superclass are selected.

## New session bean generated code

```
TestSessionBean.java
package com.duck.ejb.session.slide;

import javax.ejb.Stateless;

/**
 * Session Bean implementation class for: TestSessionBean
 */
@Stateless
public class TestSessionBean implements TestSessionBeanRemote, TestSessionBeanLocal {
}
```

- When the stateless bean is created, an annotation `@Stateless` is added to mark it as a stateless bean
- The local and remote interfaces are created and the bean implements them automatically

Shown here is a screen capture of the generated EJB code. You can see the `@stateless` annotation above the class definition which makes this Java object a stateless enterprise java bean. The local and remote interfaces have been created and the bean implements them.



## Remote and local generated interfaces

```
package com.duck.ejb.session.slide;
import javax.ejb.Local;
@Local
public interface TestSessionBeanLocal {
}

package com.duck.ejb.session.slide;
import javax.ejb.Remote;
@Remote
public interface TestSessionBeanRemote {
}
```

- Since the option to create local and remote Interfaces was selected in the wizard two interfaces were created
  - ▶ Note that these interfaces also have annotations, **@Remote** and **@Local** that are generated into the code by the wizard
  - ▶ These annotations remove the need to declare the remote and local interfaces in the EJB deployment descriptor



The option in this case for local and remote interfaces was selected, so the interfaces have also been generated in the EJB project.

Opening the two files you will see generated interface code as shown above in the two screen captures.

On the left you will see the at local annotation has been added to the local interface, while the at remote annotation has been added to the remote interface.

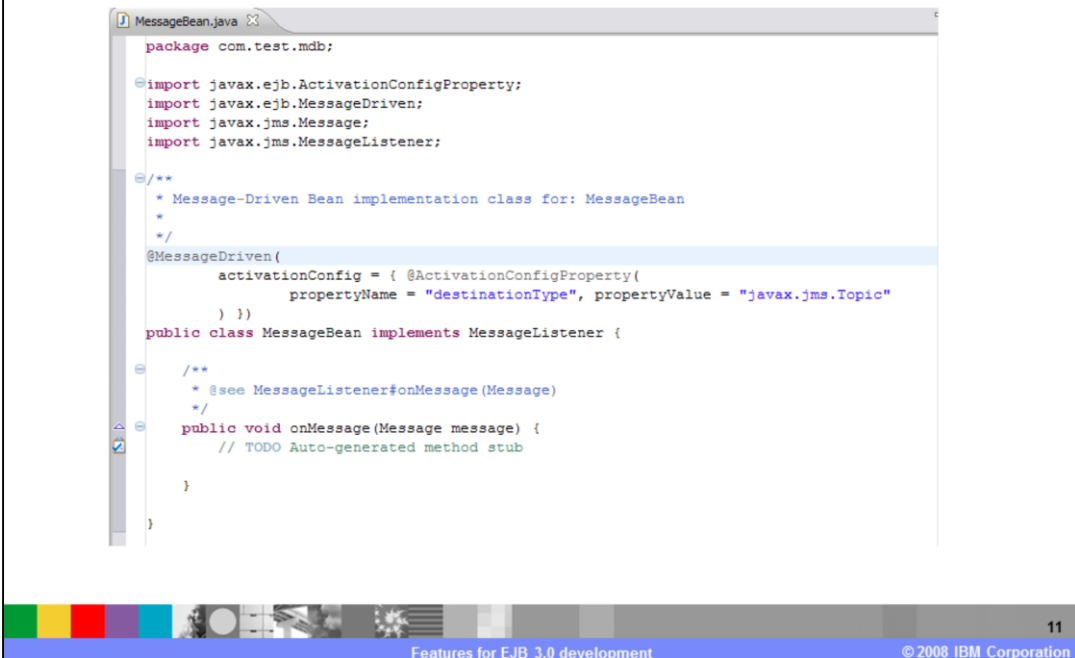
## Message driven beans

- The message driven bean wizard has entry fields to specify several options
  - ▶ The destination name
  - ▶ If the MDB is a JMS bean that subscribes to a topic or a queue
  - ▶ The transaction type
  - ▶ Interfaces that the bean implements
- The generated MDB is now updated for EJB 3.0 and the source that is generated is annotated MDB code



The Message driven bean wizard is very similar to the session bean creation wizard. You can specify the class and package names, but for the MDB wizard you also have options to specify the MDB destination name, and the option to specify if it will use a queue or topic destination type.

## Example: MDB generated by wizard



```
MessageBean.java
package com.test.mdb;

import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.jms.Message;
import javax.jms.MessageListener;

/**
 * Message-Driven Bean implementation class for: MessageBean
 *
 */
@MessageDriven(
    activationConfig = { @ActivationConfigProperty(
        propertyName = "destinationType", propertyValue = "javax.jms.Topic"
    ) })
public class MessageBean implements MessageListener {

    /**
     * @see MessageListener#onMessage(Message)
     */
    public void onMessage(Message message) {
        // TODO Auto-generated method stub
    }
}
```

11  
Features for EJB 3.0 development © 2008 IBM Corporation

Shown here is a screen capture of generated message driven bean code. You can see that the `@MessageDriven` annotation is added to the top of the class definition. Annotations are added to specify the destination type, in this case you can see that the destination type is a topic.

The `onMessage` method stub is auto generated and ready to be completed.

## Entity beans

- Entity bean support has changed for EJB 3.0.
  - ▶ Entity Beans are based on the EJB 2.x specification
  - ▶ WebSphere Application Server currently does not support 2.x Entity beans in a EJB 3.0 Jar
- The Java Persistence API (JPA) was introduced with the EJB 3.0 specification
  - ▶ JPA provides a simplified model for working with persistent data
  - ▶ JPA is discussed in more depth in the JPA portion of this education session



Entity bean support for EJB 3.0 has changed from previous versions of EJBs. While entity beans are still based on the EJB 2.x specification, EJB 3.0 has introduced a new method of data persistence called the Java persistence API, or JPA for short. In depth information on JPA tools is available in the Rational Application Developer V7.5 JPA development tools section of the IBM education assistant.

## Section

# Annotations for EJB 3.0



This section will discuss annotation support for EJB 3.0 in Rational Application Developer version 7.5.

## Annotation support

- Annotations for EJB 3.0 are supported
- While editing Java files annotations can be added using the content assist function
  - ▶ Type '@' then hit control + space to display a list of available annotations including annotations for EJB 3.0
- The editor provides validation for annotations while you type
- Annotations are also able to be edited, added, and removed using the new annotation view

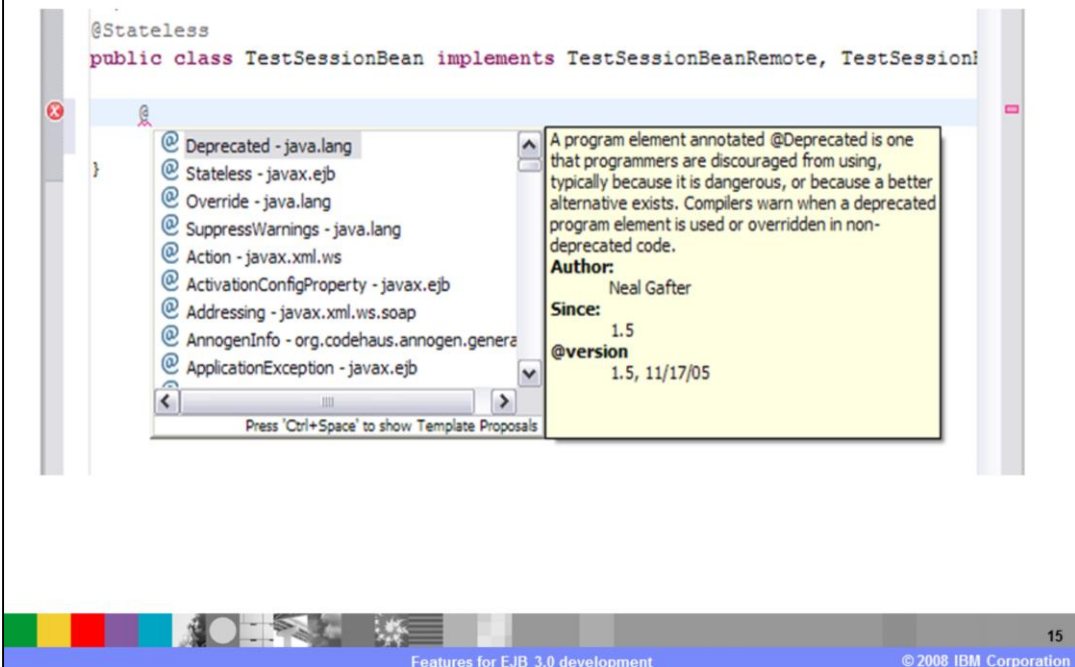


Annotation support is available in several locations in Rational Application Developer 7.5.

First, annotations can be added in a Java editor using the content assist feature. Typing an at symbol and pressing control and space will pop up a list of available annotations in the editor. Annotations that are typed into the Java editor are validated on the fly and typing in a wrong or unsupported annotation will yield an error.

A new view called the Annotation view is also available to help edit, add, and delete annotations while working on a Java class.

## Content assist helps add annotations



You can see in the screen capture here an example of using content assist to add an annotation to a session bean EJB class.

An at symbol has been typed and control space has been pressed to bring up the list of available annotations.

## Section

# The Annotation view



This section will discuss the Annotation view in Rational Application Developer 7.5.



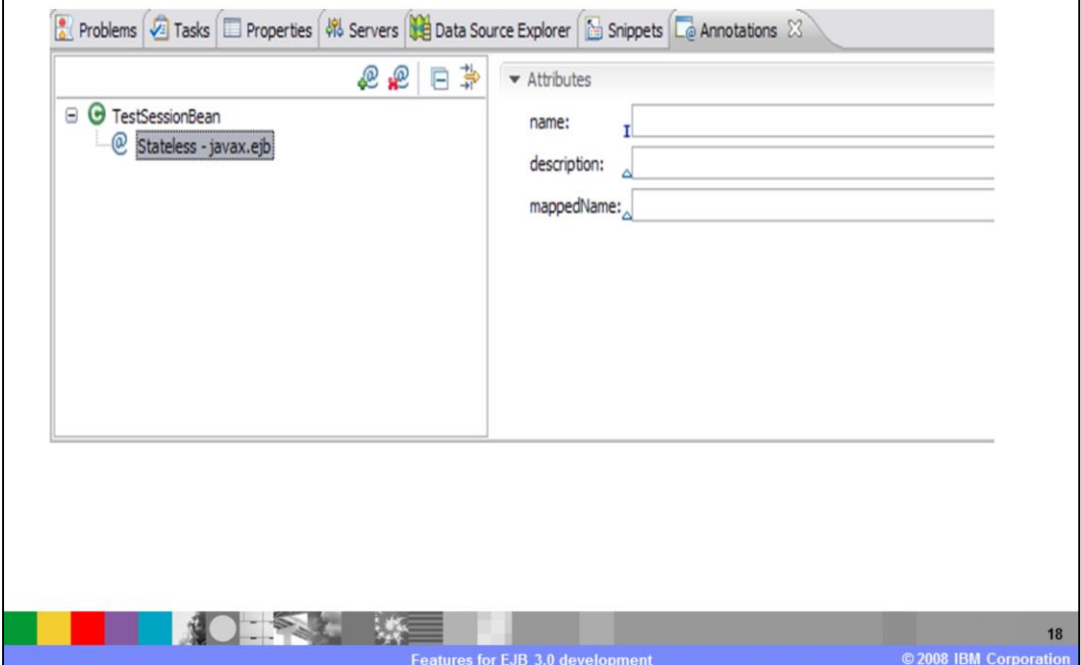
## The Annotation view

- The Annotation view is new
- It is available in the default Java EE perspective along the bottom right of the IDE workspace
- While working on a Java file the Annotation view will display the annotations that the file contains
- The Annotation view has several features including:
  - ▶ Adding a new annotation to the file
  - ▶ Removing existing annotations
  - ▶ Editing annotations and their attributes



The Annotation view is new in Rational Application Developer. Using the Annotation view while editing a Java class, you can add new annotations to the file, remove existing annotations, and edit existing annotations and their attributes.

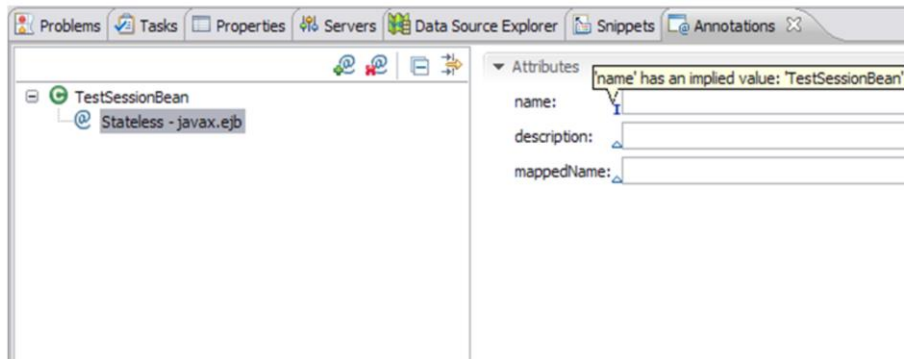
## The Annotation view



Shown here is a screen capture of the Annotation view while editing the properties available for the @Stateless EJB 3.0 annotation.

You can see that once the at stateless annotation is highlighted in the left pane of the view that the right hand pane displays entry fields for the attributes available to be edited for this annotation.

## The Annotation view



- Hovering over the icons by the entry fields in the annotation view shows pop up help

The icons that appear by the entry boxes for the annotation attributes provide help text when they are hovered over with the mouse.

These descriptions help you decide what information to add in the available fields.

## Section

# ***The EJB 3.0 deployment descriptor***



This section will discuss the EJB 3.0 deployment descriptor in Rational Application Developer version 7.5.

## EJB 3.0 deployment descriptor

- The EJB deployment descriptor is now optional with EJB 3.0
- Adding elements to the deployment descriptor will override any annotations that are added to the beans in the application
- The deployment descriptor is only generated into the project if the option to generate a descriptor was selected during the creation of the EJB 3.0 project
  - ▶ As an alternative you may right click the EJB module and select Java EE > Generate Deployment Descriptor Stub
- The EJB-jar.xml file is available to be edited with the deployment descriptor editor
  - ▶ This editor allows developers to add, remove, and modify EJB descriptions and properties

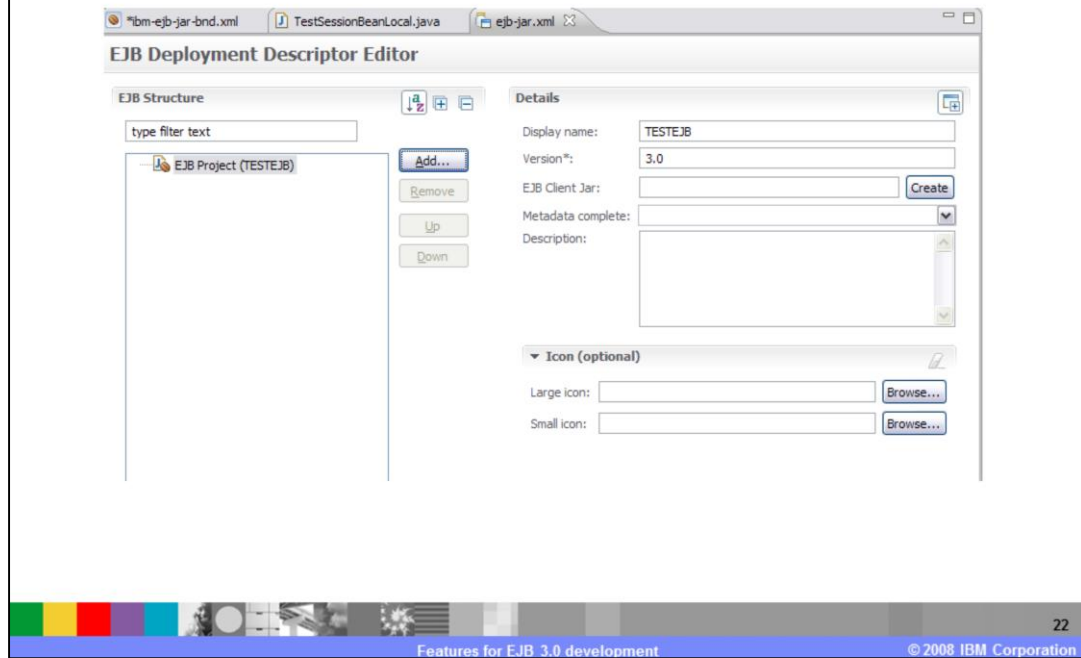


With the EJB 3.0 specification, the EJB deployment descriptor has become optional. Annotations in the code now specify the attributes of the EJB, and if you have an EJB deployment descriptor, attributes that are specified in the deployment descriptor will override the annotation information in the code.

If a deployment descriptor was not selected to be generated at the time the EJB project was created, a descriptor can be generated by selecting the EJB project, right clicking, and selecting the Java EE submenu where the option to generate deployment descriptor stub is available.

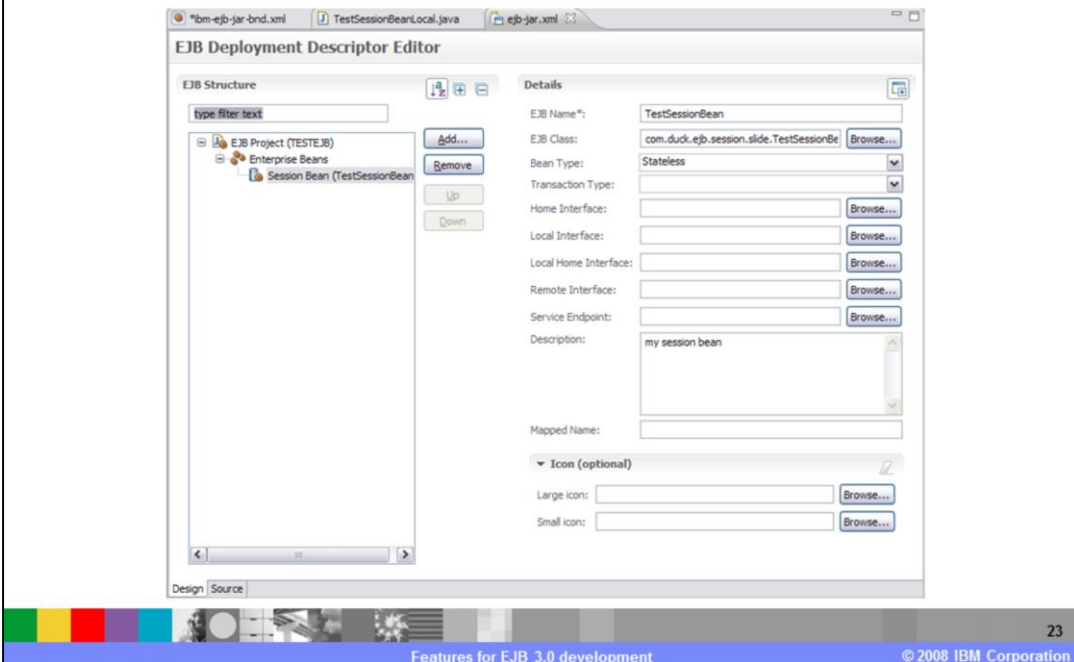
Once generated, the EJB jar.xml file can be edited with the deployment descriptor editor.

## Example: Deployment descriptor editor



The deployment descriptor editor has a GUI portion and a source view. Shown here is a screen capture of the editor in GUI mode.

## Example: Deployment descriptor



When you add EJB definitions to the deployment descriptor, any annotations that the EJB has do not get pre-filled in the descriptors annotation settings.

The deployment descriptor can be used to override annotations set in the code of the EJB.

## Section

# *The EJB binding file*



This section will discuss the EJB binding file in Rational Application Developer version 7.5.



## EJB WebSphere® bindings file

- The bindings file is new for EJB 3.0 support
  - ▶ Previously EJB bindings were set in an XMI file
  - ▶ Bindings are now set in an XML file
- An editor is provided to help create the EJB bindings
- It is not necessary to define bindings for EJB 3.0
  - ▶ However, for beans that implement the same Interface bindings must be set explicitly
- Default bindings are provided by the EJB container if bindings are not set



The EJB bindings file has changed in Rational Application developer version 7.5.

Previously the EJB bindings were set in an XMI file. This has changed, and the bindings now reside in a more common XML file type.

For EJB 3.0 it is not necessary to specify bindings for EJBs. Beans that implement the same interface must have bindings set explicitly.

If there are no bindings set for the EJBs, when they are deployed the EJB container will provide default bindings for the EJBs.

## WebSphere EJB bindings file

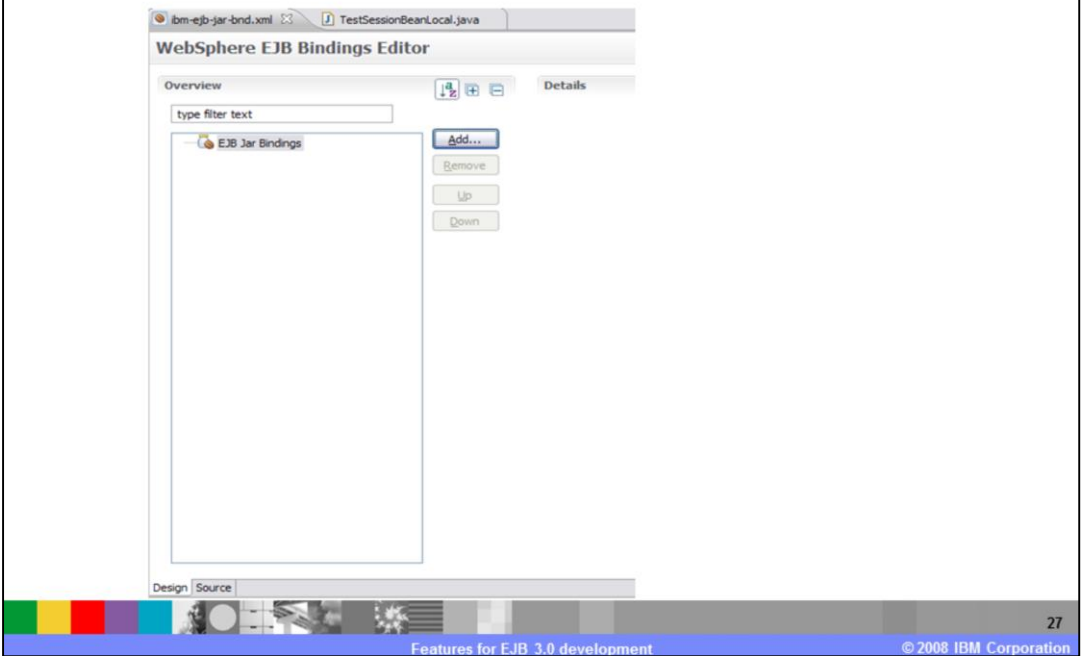
- To create a bindings file, right click the EJB module and select Java EE > Generate WebSphere Bindings File
- The bindings file will be created and the bindings editor will open
- Using the editor you can specify the JNDI bindings for your EJBs



A bindings file can be generated similar to creating an EJB deployment descriptor. You can do this by right clicking on the EJB project and selecting the Java EE submenu; followed by clicking the menu option to generate WebSphere bindings file.

When the bindings file is created, it will open automatically in the bindings editor. The editor helps to specify the JNDI bindings for your EJBs.

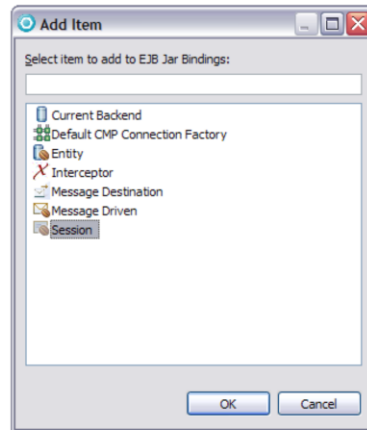
## WebSphere EJB bindings file



Shown here is a screen capture of the EJB bindings file editor that is opened when the bindings file is generated.

## WebSphere EJB bindings file

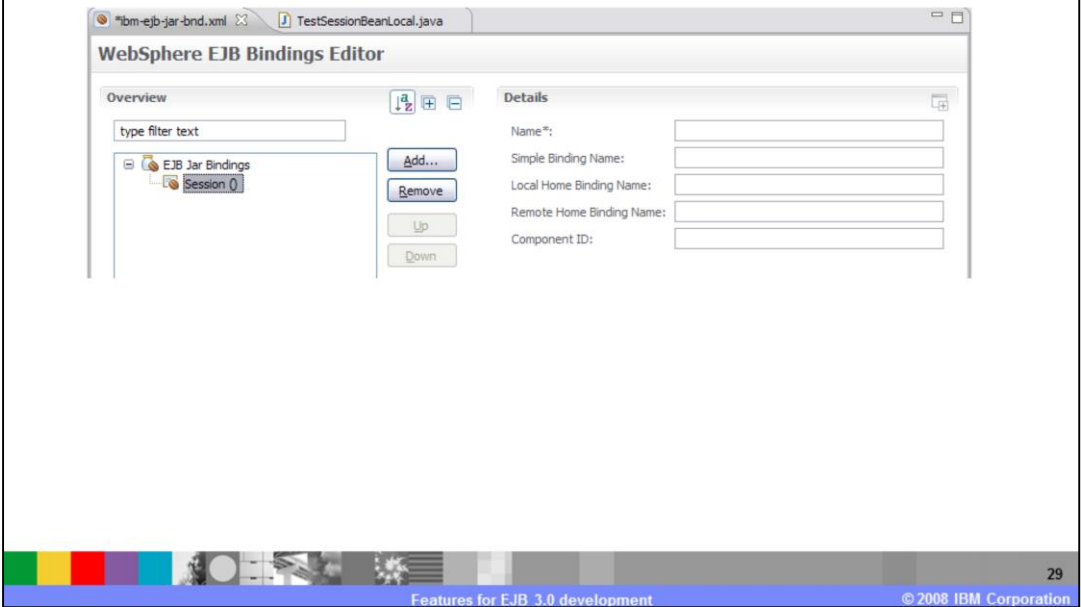
- Clicking the add button displays a menu of what bindings you may add
- Each item in the binding editor will have different options
  - ▶ Depending on the binding selected the editor displays entry fields related to that binding under the details section



When using the EJB bindings file editor you can click the available add button to add bindings for your EJBs.

When a binding is selected to be added, the editor will display the appropriate entry fields to help edit that binding.

## Example: Bindings for a session bean



Shown here is a screen capture of a session bean binding. You can see on the right hand side of the bindings editor that entry fields related to the session bean binding have been made available, including entry fields for the name, simple binding name, local and remote home binding names, and the component ID.

## Summary

- Rational Application Developer provides new tools for EJB 3.0 development
  - ▶ New wizards for session and message driven beans
  - ▶ Annotation support to simplify working with annotations in an editor and in the annotation view
  - ▶ Optional deployment descriptor editor and stub generation
  - ▶ New EJB binding XML file and editor



In summary, Rational Application Developer version 7.5 provides excellent support for EJB 3.0. This presentation has shown the new wizards for creating EJB 3.0 projects, creating EJB 3.0 enterprise Java beans, and the annotation support to work with these beans. The optional deployment descriptor was shown to be generated either at the time the project is created, or created later using the project Java EE pop up menu. The new EJB binding XML file editor allows you to specify bindings for your EJBs.

## References

- The specification for EJB 3.0  
<http://jcp.org/en/jsr/detail?id=220>
- Helpful documentation on the EJB 3.0 feature pack for WebSphere

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.ejbfep.multiplatform.doc/info/ae/ae/cejb\\_bindingsefp.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.ejbfep.multiplatform.doc/info/ae/ae/cejb_bindingsefp.html)

# Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

Rational      WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>

EJB, Java, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.