SDN Contribution

SAP NetWeaver J2EE Preview: Developing J2EE 1.4 Applications

Applies to:

SAP NetWeaver J2EE Preview

Summary

Learn the basics of using the WTP-based toolset in SAP NetWeaver Developer Studio to develop J2EE 1.4 applications.

Author(s): SAP NetWeaver Product Management

Company: SAP AG

Created on: April 2006

Table of Contents

Developing J2EE 1.4 Applications

SAP NetWeaver Developer Studio Overview

The SAP NetWeaver Developer Studio (NWDS) is an environment that supports the development of applications for the J2EE Engine. The NWDS features tools for the full lifecycle of the development process: design, development, testing, debugging and deployment.
NWDS is based on the Eclipse platform, extending its numerous advantages and features.

The current version of NWDS is a preview of the integrated development environment support for the next major release of the J2EE Engine. It fully supports development of J2EE 1.4 compliant applications, and previews the future support for Java EE 5 compliant applications.

This document focuses on developing J2EE 1.4 compliant applications. Developing Java EE 5 compliant applications is described in the Java EE 5 Preview Project guide.

The toolset for developing J2EE 1.4 applications is based on the Eclipse Web Tools Platform (WTP) 1.0 project. It is an open-source initiative of the Eclipse community for creating quality tools for J2EE application development. For more information, see the Web Tools Platform Web site.

Development Approaches

In general, you can use two alternative approaches for creating J2EE 1.4 applications:

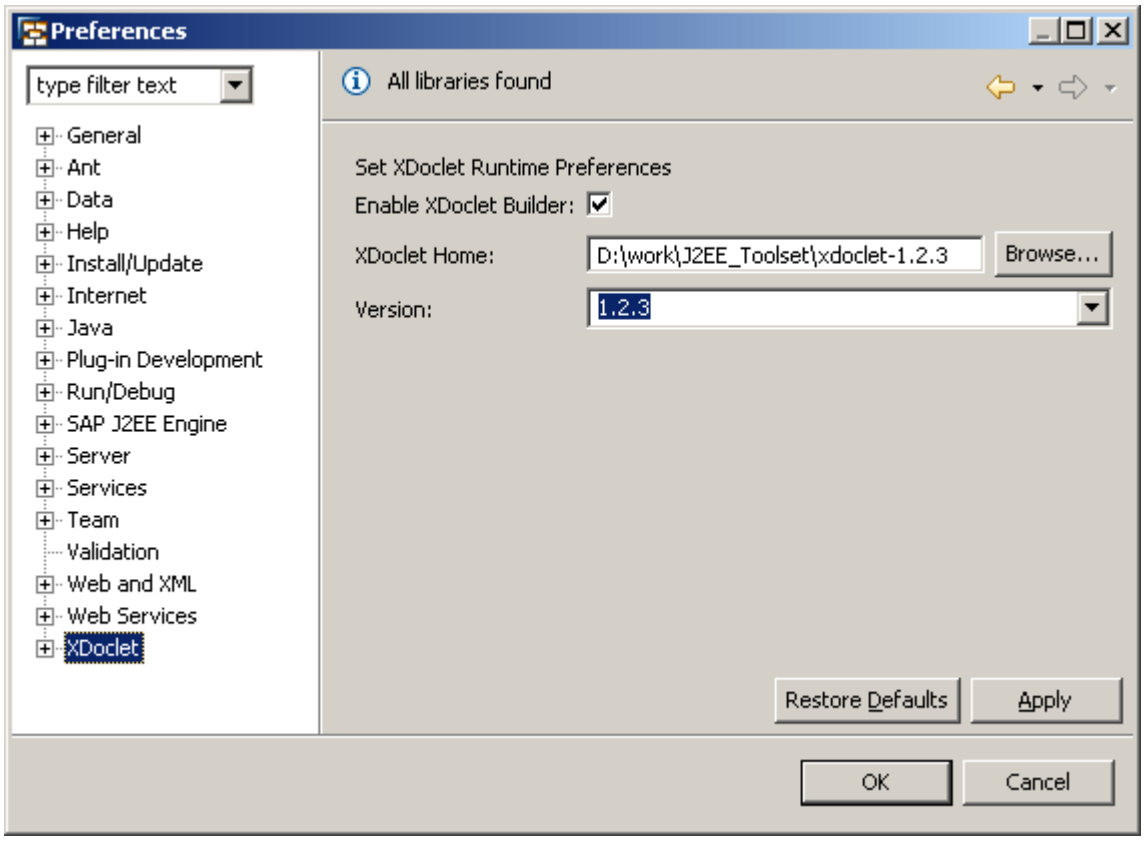
- UI approach - allows you to use the multipage editors for managing enterprise beans and deployment descriptors.
- XDoclet annotations approach - allows you to configure bean properties and deployment descriptors using XDoclet annotations.

The advantages and limitations of the two approaches are listed in the following table:

Approach	Advantages	Limitations
UI Approach	Easy and intuitive	No wizards for filters and listeners (you need to create them manually as ordinary Java files)
	Provides support for SAP-specific deployment descriptors	
XDoclet Approach	Support for filters and listeners	The XDoclet tool is not distributed with the SAP Preview installation. You must download it from the XDoclet Web site (xdoclet.sourceforge.org/).
	Represents the habitual development method for some developers	No specific XDoclet annotations for SAP-specific deployment descriptors. You have to use the UI approach for the latter.
		No wizard for entity beans

Setting up XDoclet in the SAP NetWeaver Developer Studio

1. Download the XDoclet 1.2.3 library from the XDoclet download page.
2. Extract the archive to a local directory.
3. In the SAP NetWeaver Developer Studio, choose *Window -> Preferences -> XDoclet*.
4. Specify *1.2.3* as the *Version*, and enter the directory where you extracted the XDoclet library.
5. Choose *OK*.



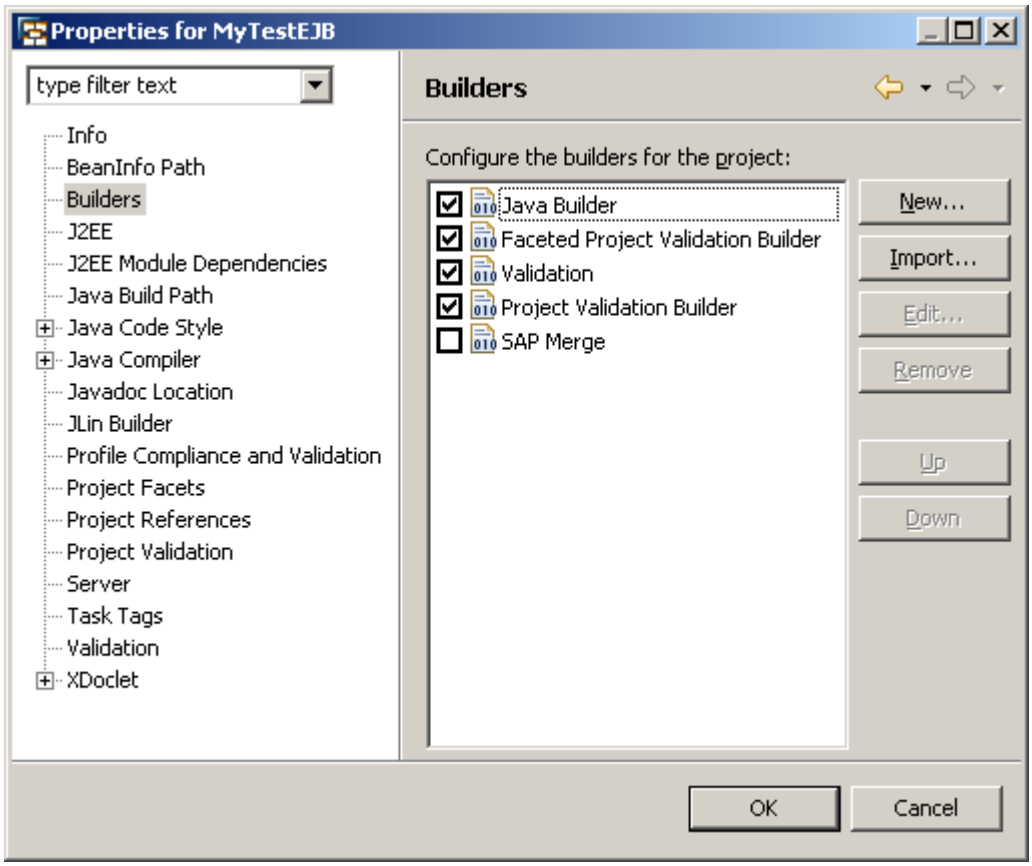
Merging the Changes between the UI and XDoclet Approaches

Using both approaches for simultaneous editing of one and the same component is **not** recommended. The reason is that the changes from one of the approaches may start taking precedence over the changes from the other approach, and the latter approach might never work for that component from then on.

However, in some cases it is possible to merge the changes you made using the different approaches. The merge mechanism is enabled only when there is an XDoclet builder enabled in the corresponding *Preferences* page.

The SAP NetWeaver Developer Studio creates a *sapMerge* directory in each corresponding project, and places its merge information there. The merge mechanism does not work in all cases. Furthermore, it has some limitations.

If you use only the UI approach, the merge mechanism is never started. You can also disable it explicitly by choosing in the context menu of the project *Properties -> Builders* and unselecting the *SAP Merge* option. In such case, the changes due to XDoclet annotations will take precedence over changes made using the UI.



Overview of Tasks

J2EE 1.4 applications are developed in the J2EE perspective. You can open it by choosing *Window -> Open Perspective -> Other -> J2EE*.

Tips:

Always use the same approach for **all** beans within the same EJB Project.

Do **not** combine the XDoclet and UI approach when developing a component.

Developing EJB Components

Developing EJB components involves the following steps:

1. Create an EJB Project

EJB Projects are the wrapper for all enterprise bean components. For detailed information, see Creating an EJB Project.

2. Create and edit the enterprise beans

   There are two separate wizards for creating beans: one for UI editing of beans, and one for XDoclet editing. If you choose the UI approach, the J2EE toolset will provide a multipage bean editor where you can view the bean properties and edit the bean methods, as well as multipage editors for all bean deployment descriptors. If you choose the XDoclet approach, you will have to use XDoclet annotations in your source code instead of multipage editors.

   For detailed information, see Creating Session Beans, Creating Entity Beans and Creating Message-Driven Beans.

3. Edit the deployment descriptors

   If you created the bean with the UI approach, you can use the multipage editors for editing purposes. If you created the bean using the XDoclet approach, you edit the standard J2EE multipage editors with XDoclet annotations in the bean class, and the SAP-specific deployment descriptors using the standard XML editors in the IDE.

   For detailed information, see Editing EJB Deployment Descriptors.

4. Add the EJB Project to an Enterprise Application Project

   Enterprise Application Projects are wrappers for complete enterprise applications with their EJB, Web, Connector and other components.

   For detailed information about Enterprise Application Projects, see Creating an Enterprise Application Project.

Developing Web Components

Developing Web components involves the following steps:

1. Create a Dynamic Web Project

   Dynamic Web Projects are the wrapper for all Web components (Servlets, JSP files, filters, listeners and so on). For detailed information, see Creating a Web Project.

2. Create and edit the Web components
3. Edit the deployment descriptors
4. Add the Web Project to an Enterprise Application Project

Developing Connectors

Developing J2EE connectors involves the following steps:

1. Create a Connector Project
2. Edit the *ra.xml* and *connector-j2ee-engine.xml* descriptors
3. Add the Connector Project to an Enterprise Application Project

For detailed information about creating connectors, see Creating Connectors.

Creating Data Sources

The concepts and principles of working with data sources are explained in detail in the Working with Data Sources tutorial.

Adding data sources to enterprise applications is described in Working with DataSources: Case: J2EE 1.4 Application.

Assembling Applications

You assemble applications in Enterprise Application Projects. An Enterprise Application Project is the wrapper for EJB Projects, Web Projects and Connector Projects. When an Enterprise Application Project is built, a corresponding EAR archive is created, wrapping the JAR and WAR files containing the EJB and Web components.

For detailed information about creating Enterprise Application Projects, see Creating an Enterprise Application Project.

Deploying Applications

You deploy J2EE 1.4 applications developed **in** the SAP NetWeaver Developer Studio from the *Servers* view. This procedure is different from the procedure for deploying J2EE 1.4 applications developed **outside** the NWDS. The latter procedure is described in Deployment Guide.

For detailed information about deploying, see Deploying Applications.

Remote Debugging of Applications

From the SAP NetWeaver Developer Studio, you can debug remotely the J2EE applications directly on the SAP NetWeaver Application Server. Debugging supports JSR 45 for JSP pages in Web applications.

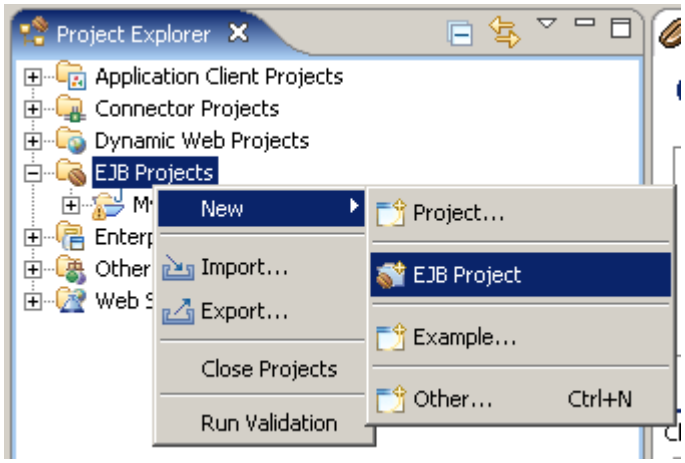
To do so, you must restart the J2EE Engine in Debug mode, and configure a set of properties. Then you set the necessary breakpoints in the Java source code or JSP source file from the NWDS, and establish a Debug connection.

For more information, refer to the JavaOne section of the SAP Developer Network (sdn.sap.com).

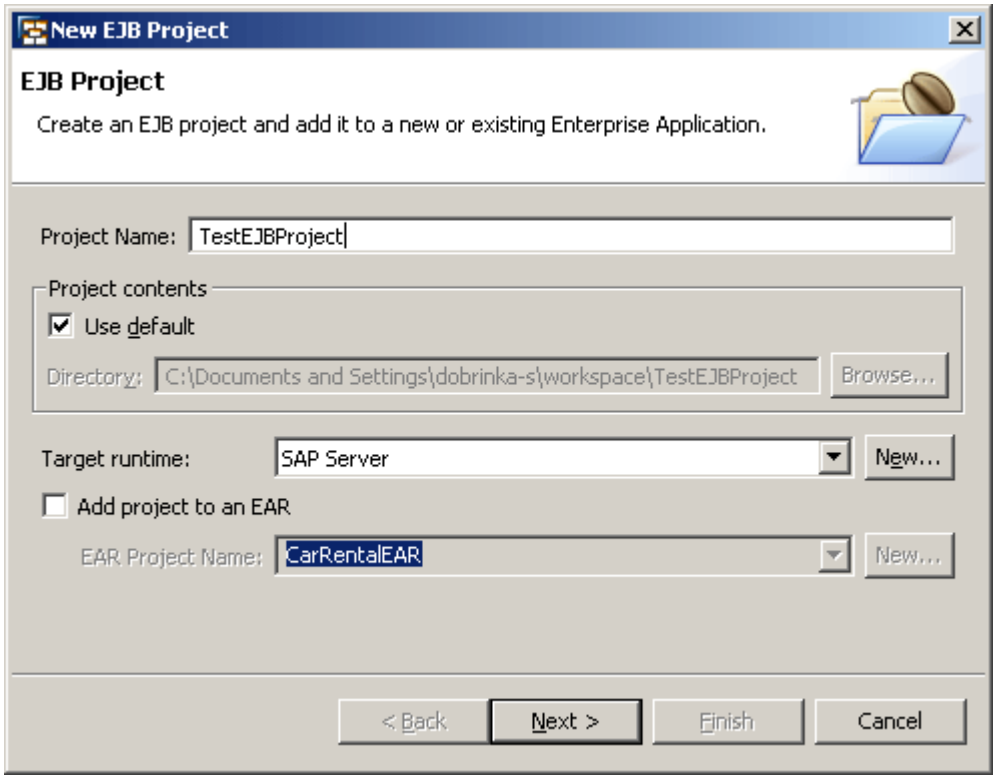
Detailed Procedures

Creating an EJB Project

1. In the *Project Explorer*, select the *EJB Projects* folder and choose *New -> EJB Project* from the context menu.



2. On the first page of the *New EJB Project* wizard, specify the basic project information, and choose *Next*.



Note: Always choose *SAP Server* as the *Target runtime*. If you leave this field blank, the SAP NetWeaver Developer Studio cannot find the necessary EJB libraries for building the project.

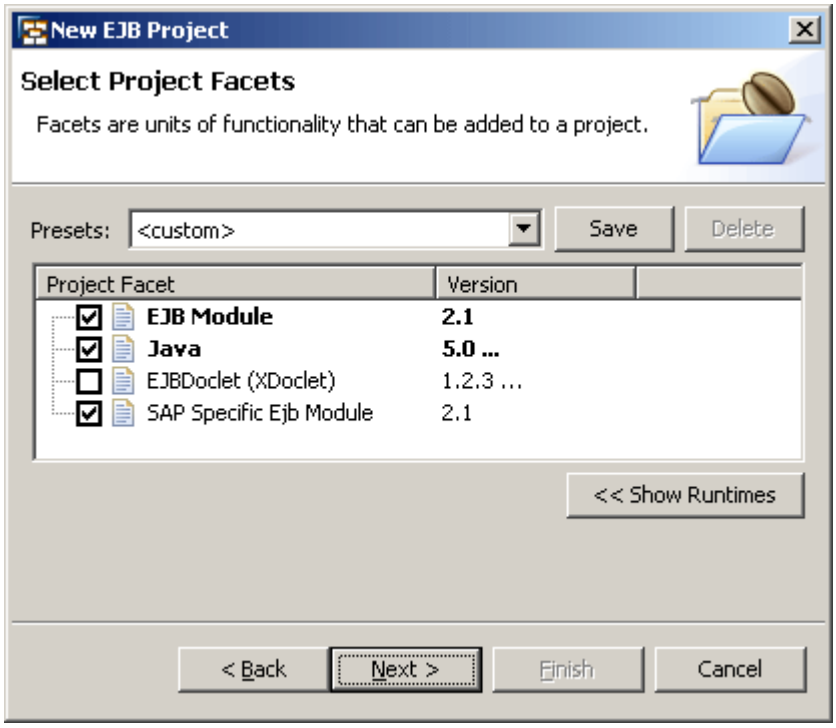
3. In the next wizard page, you can define the project facets.

   If you want to use the XDoclet approach for developing beans, check the *EJBDoclet* option.

   If you do not want to use SAP-specific deployment descriptors (*ejb-j2ee-engine.xml*), uncheck the *SAP Specific Ejb Module* option.

   Hint: If you use the same facet settings for all EJB Projects, you can save the currently selected facets as a preset. To do so, select the relevant facets and choose *Save*.

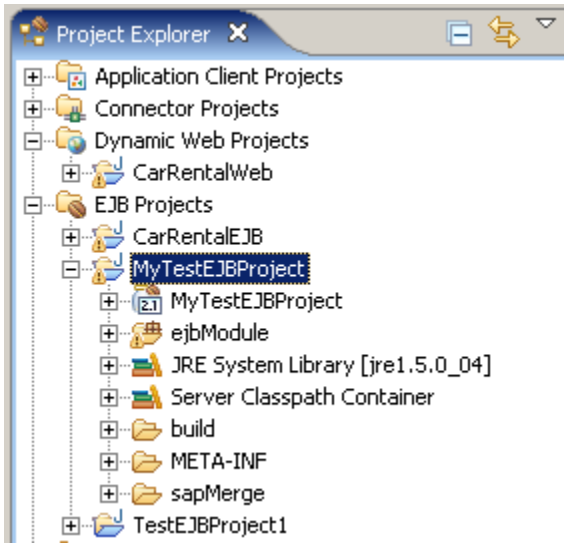
   Note: You can change the project facets at any time by choosing *Properties -> Project Facets* in the context menu for the project.



4. Choose *Next*.
5. If you need to generate a client JAR for the project, check the corresponding option and specify the JAR file name.

   Note: The SAP NetWeaver Developer Studio works only with the default source directory for the project. Even if you specify a different directory, the default one will be used. This limitation is described in Limitations.
6. Choose *Finish*.

   The EJB Project appears in the *Project Explorer*.



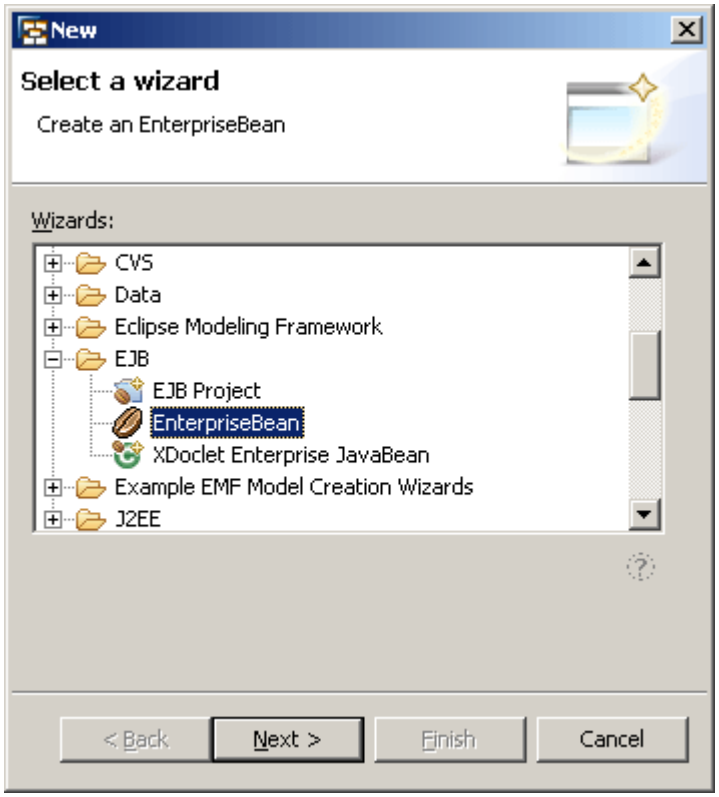
7. If you have not already done so, add the EJB Project to an Enterprise Application Project.

Creating Session Beans

There are two alternative approaches for developing session beans: the UI approach and the XDoclet approach.

UI Approach

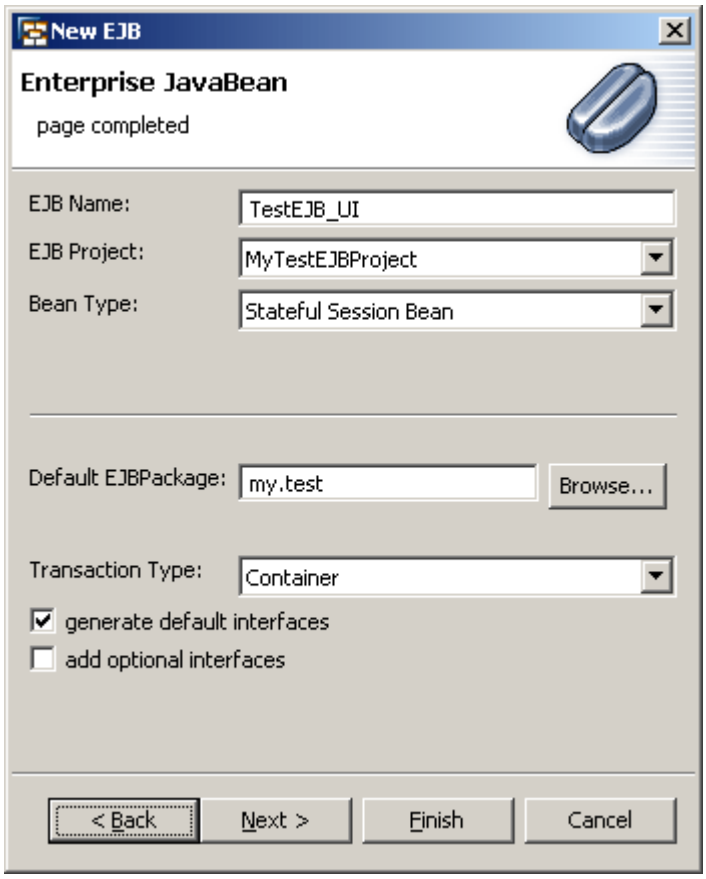
1. In the *Project Explorer*, select the EJB Project to wrap the bean.
2. From the context menu, choose *New -> Other -> EJB -> Enterprise Bean* and choose *Next*.

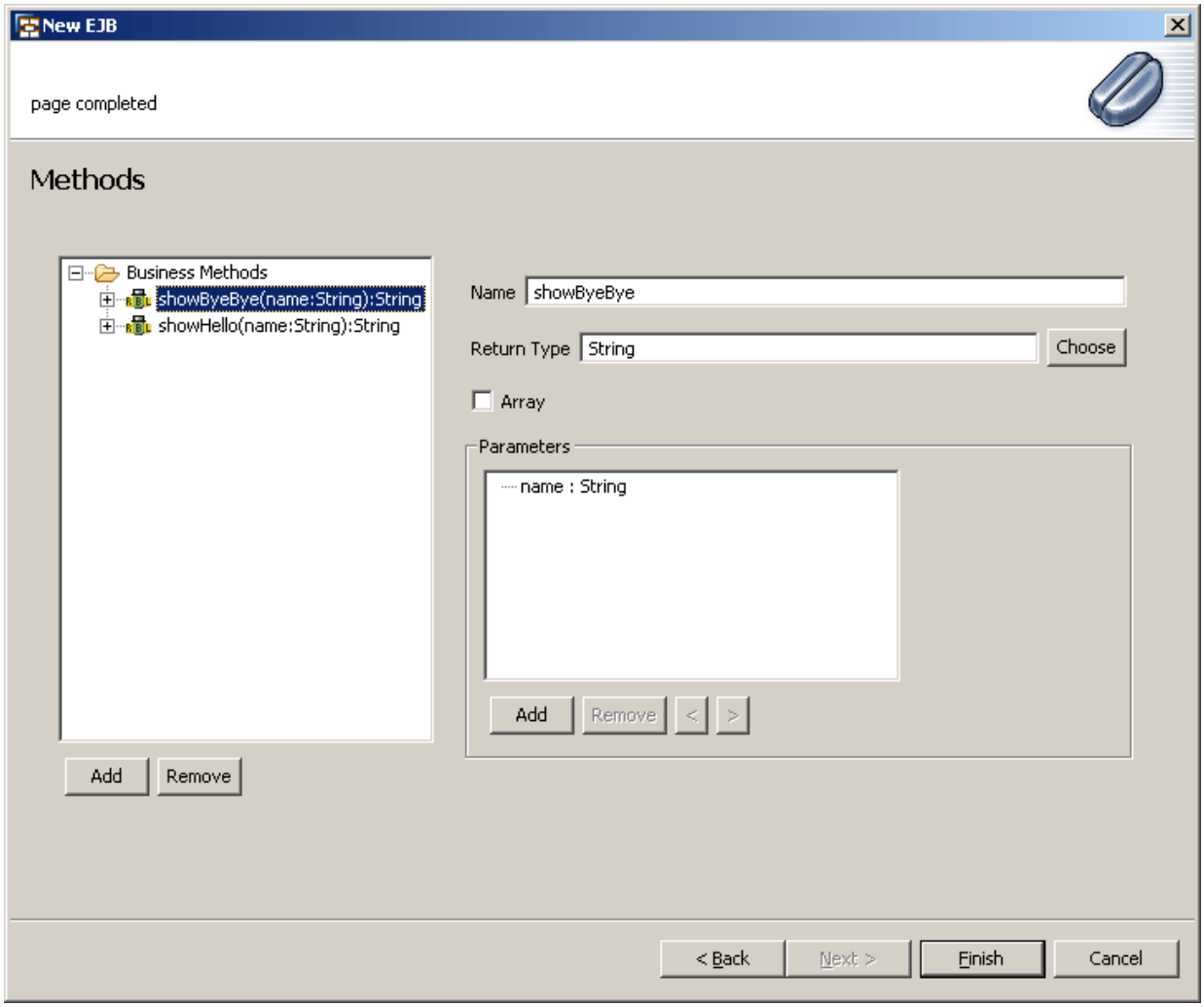


3. Throughout the wizard pages, specify the bean settings as necessary.

   Select the *generate default interfaces* option if you want to generate the bean with all default interfaces (remote, home, local and local home).

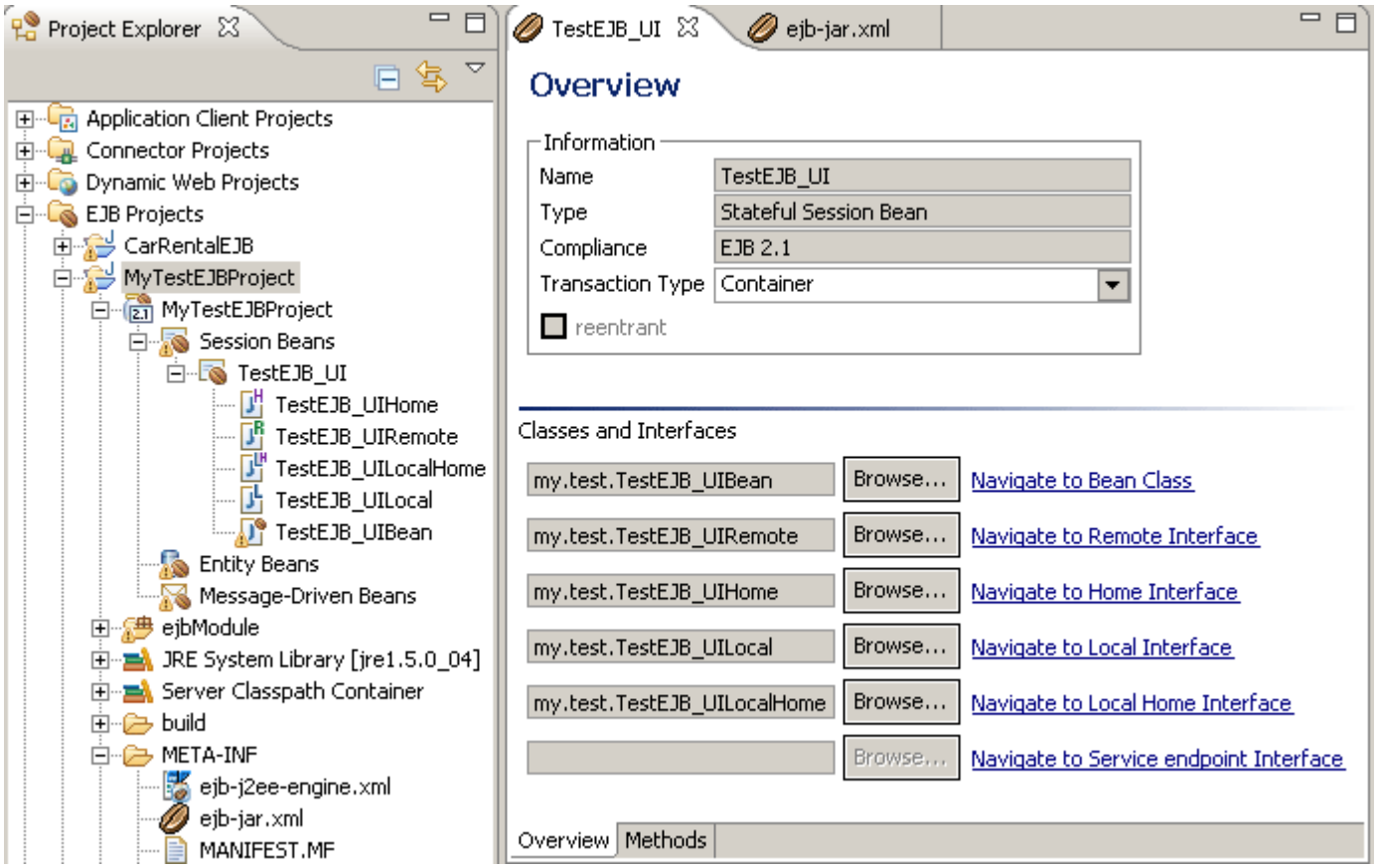
   Select the *add optional interfaces* option if you want to generate a service end point interface.


New EJB
Enterprise JavaBean
page completed
EJB Name:
TestEJB_UI
EJB Project:
MyTestEJBProject
Bean Type:
Stateful Session Bean
Default EJBPackage:
my.test
Browse...
Transaction Type:
Container
generate default interfaces
add optional interfaces
< Back
Next >
Finish
Cancel



4. Choose *Finish*.

The new bean appears in the *Project Explorer* under the corresponding EJB Project node. The bean editor automatically opens on the right.

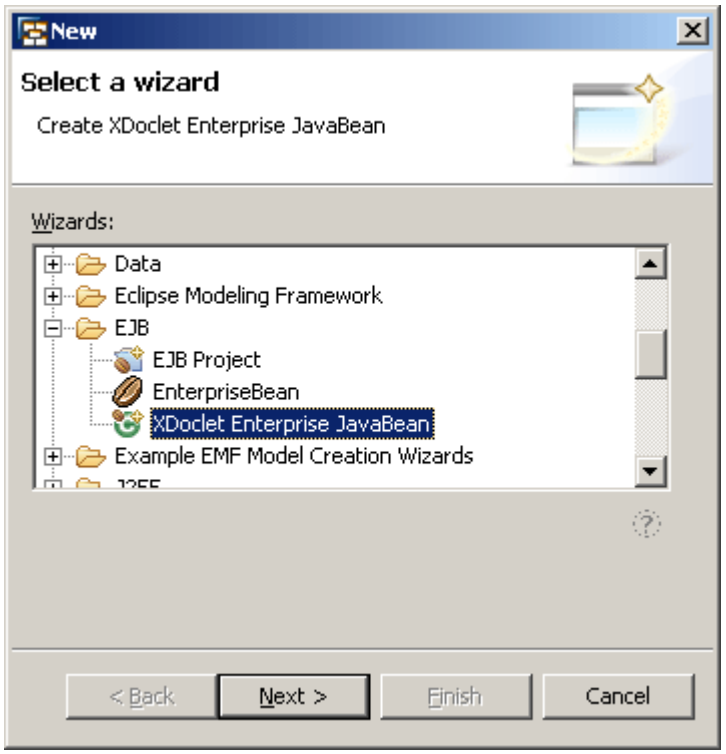


From the bean editor, you can navigate through the bean classes, and manage the bean's business methods.

Next steps: Implementing the bean class and editing the deployment descriptor using the corresponding multipage editors.

XDoclet Approach

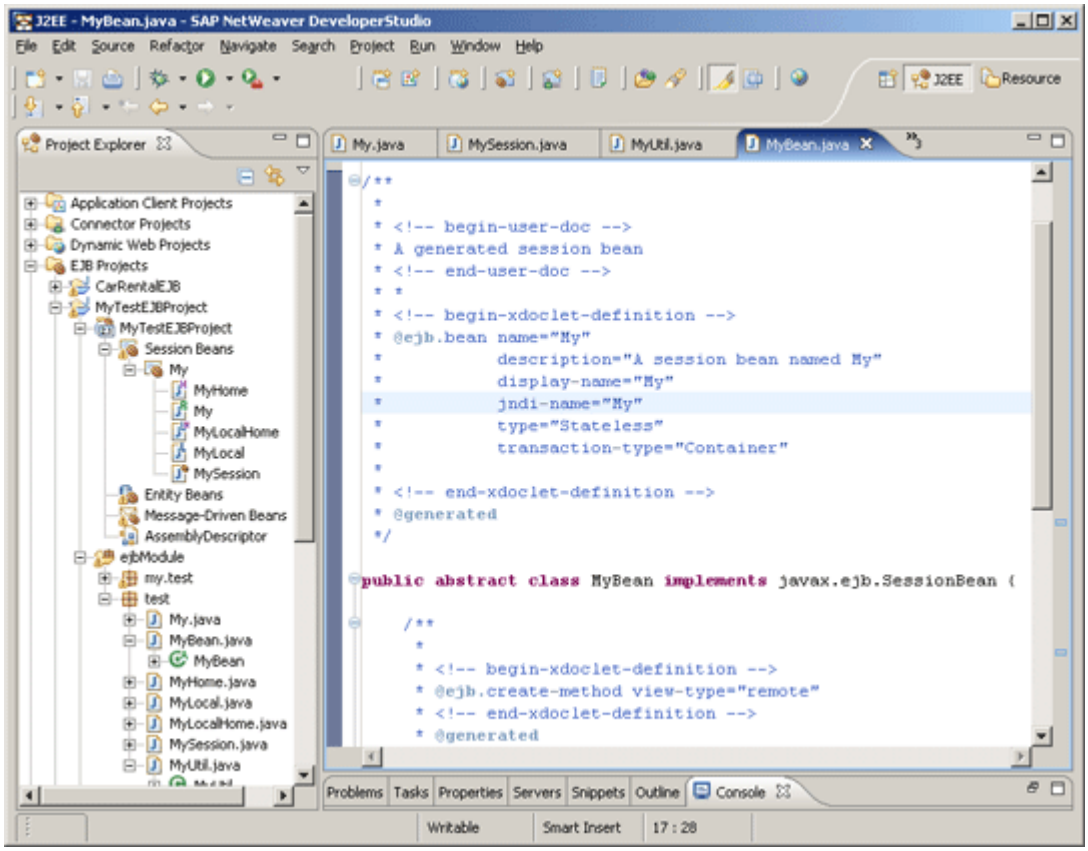
1. In the *Project Explorer*, select the EJB Project to wrap the bean.
2. From the context menu, choose *New -> Other -> EJB -> XDoclet Enterprise Bean* and choose *Next*.



3. Throughout the wizard pages, specify the bean settings as necessary.

4. Choose *Finish*.

The new XDoclet-enabled session bean appears in the *Project Explorer*. Its classes are generated with default XDoclet annotations according to the EJB XDoclet preferences (*Window -> Preferences -> XDoclet -> ejbdoclet*). For example, if the *RemoteInterface* option is checked, the bean is generated with a remote interface.

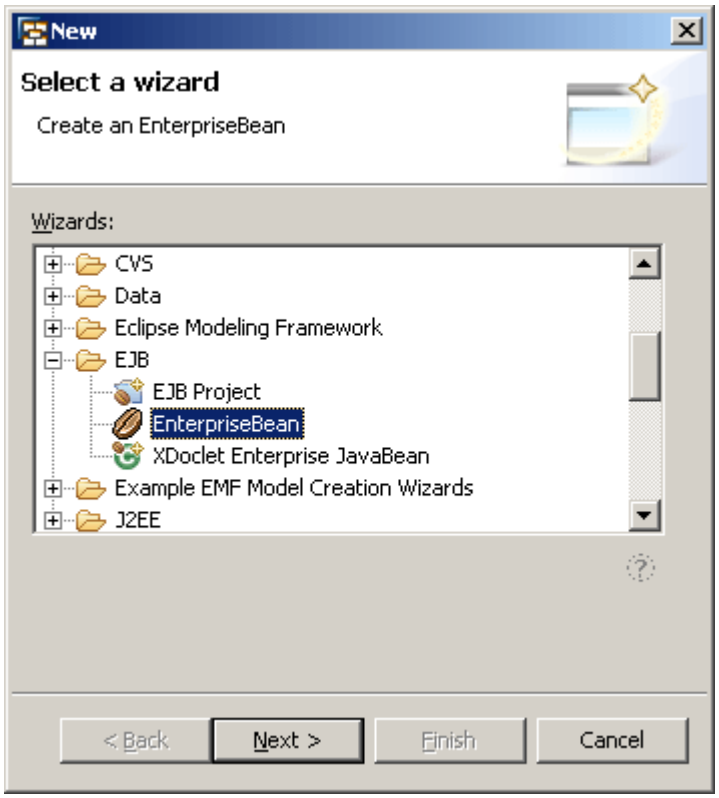


Creating Entity Beans

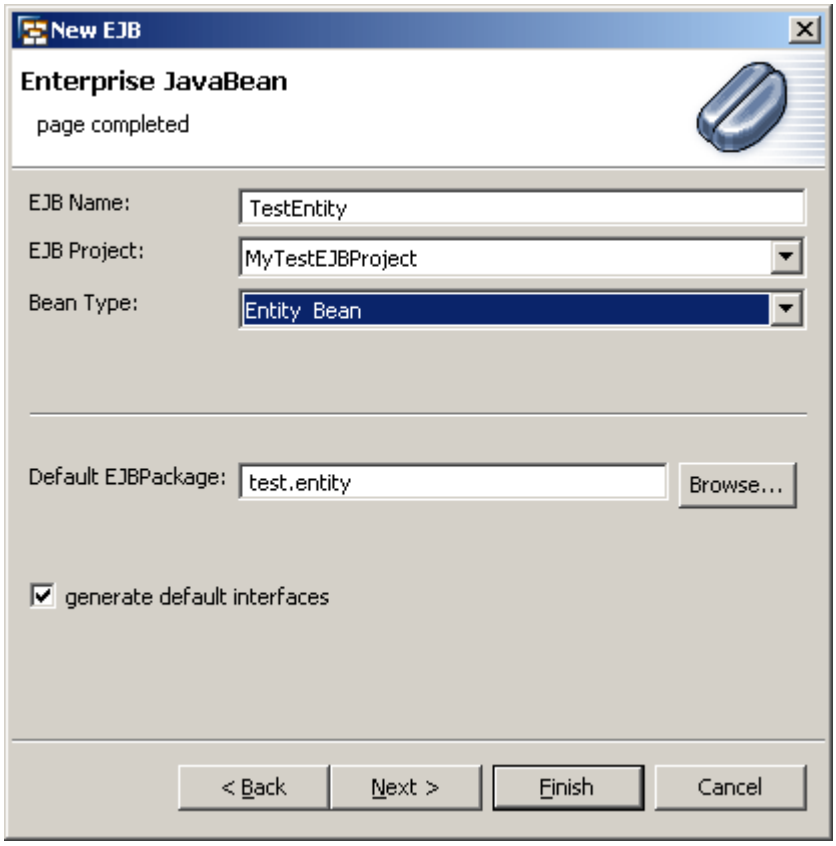
There are two alternative approaches for developing entity beans: the UI approach and the XDoclet approach.

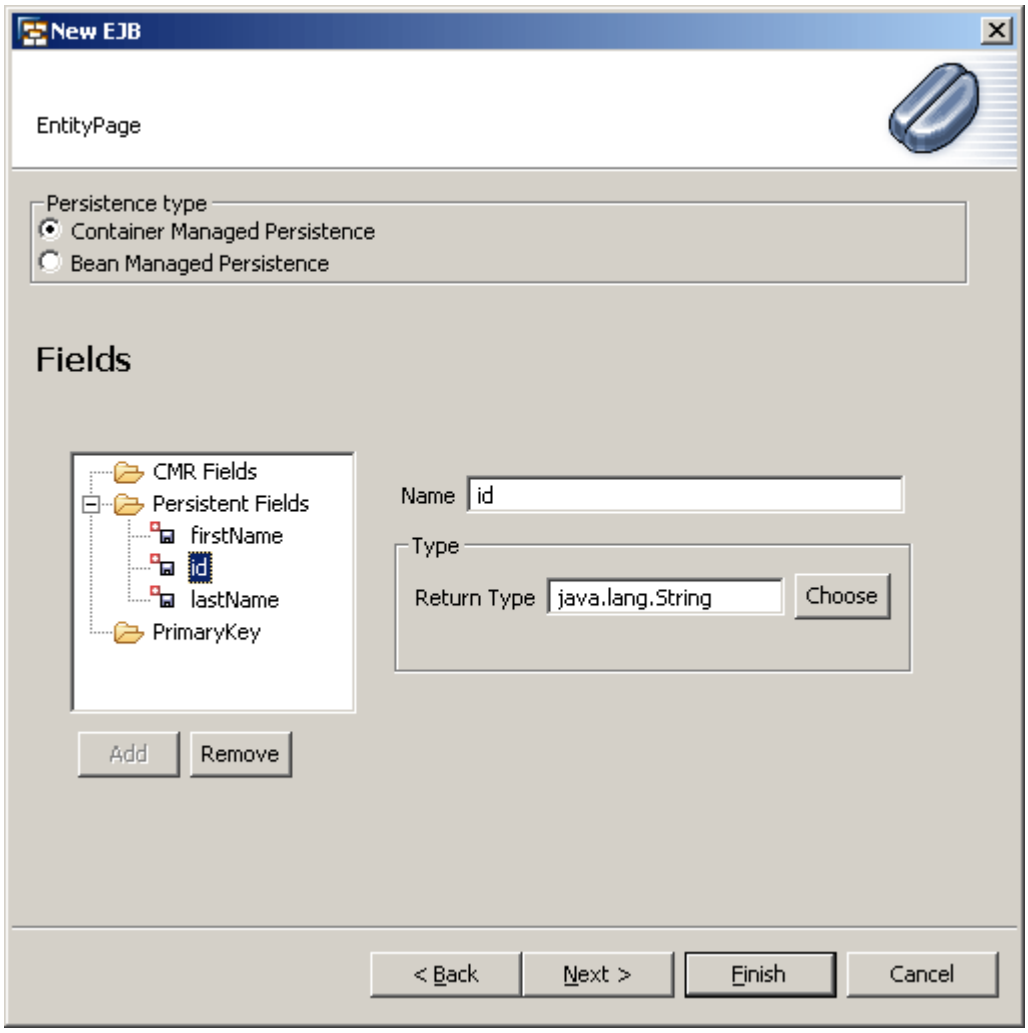
UI Approach

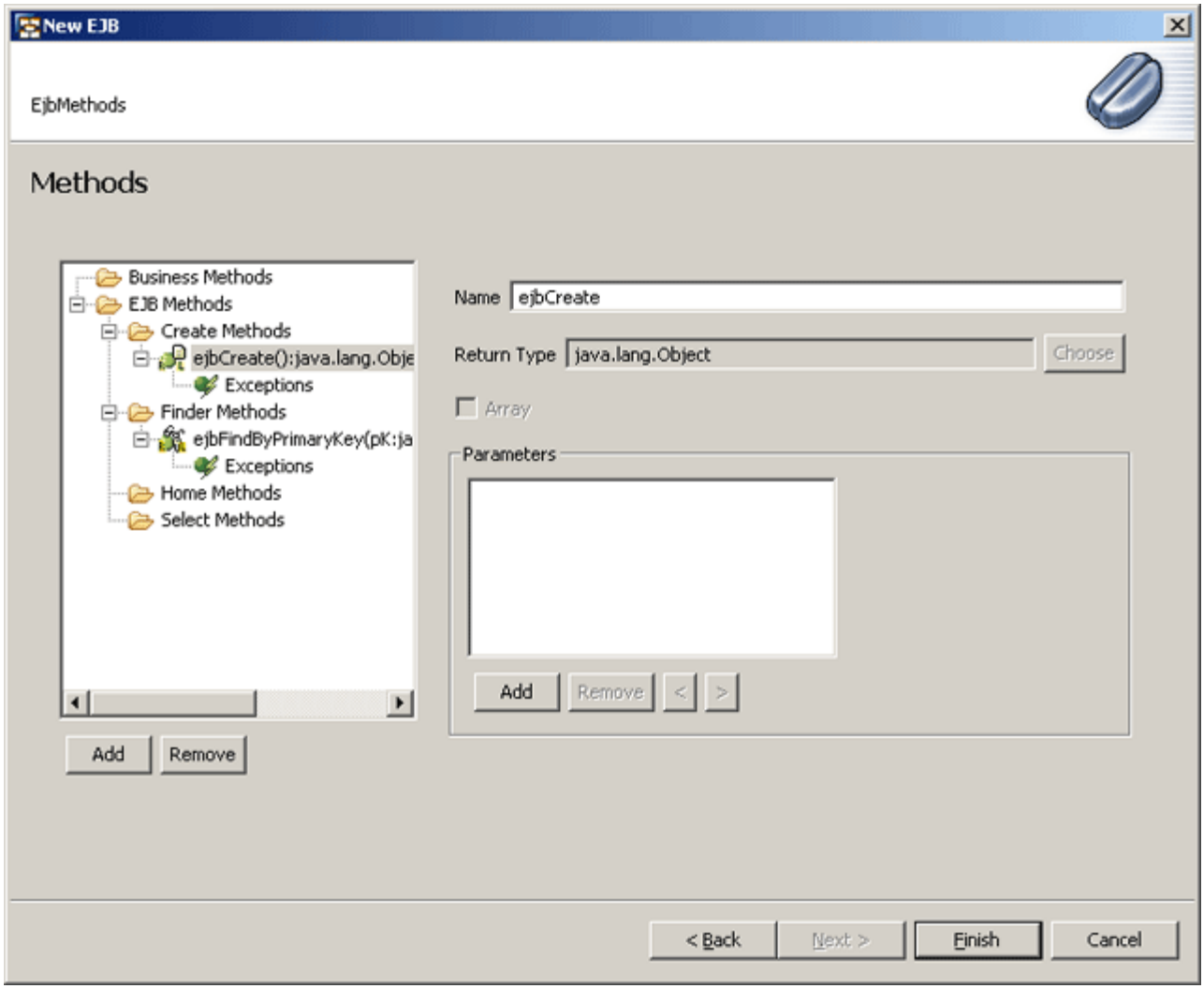
1. In the *Project Explorer*, select the EJB Project to wrap the bean.
2. From the context menu, choose *New -> Other -> EJB -> Enterprise Bean* and choose *Next*.



3. Throughout the wizard pages, specify the bean settings as necessary.




New EJB
EntityPage
Persistence type
Container Managed Persistence
Bean Managed Persistence
Fields
CMR Fields
Persistent Fields
firstName
id
lastName
PrimaryKey
Name id
Type
Return Type java.lang.String
Choose
Add
Remove
< Back
Next >
Finish
Cancel



4. Choose *Finish*.

The new bean appears in the *Project Explorer* under the corresponding EJB Project node.

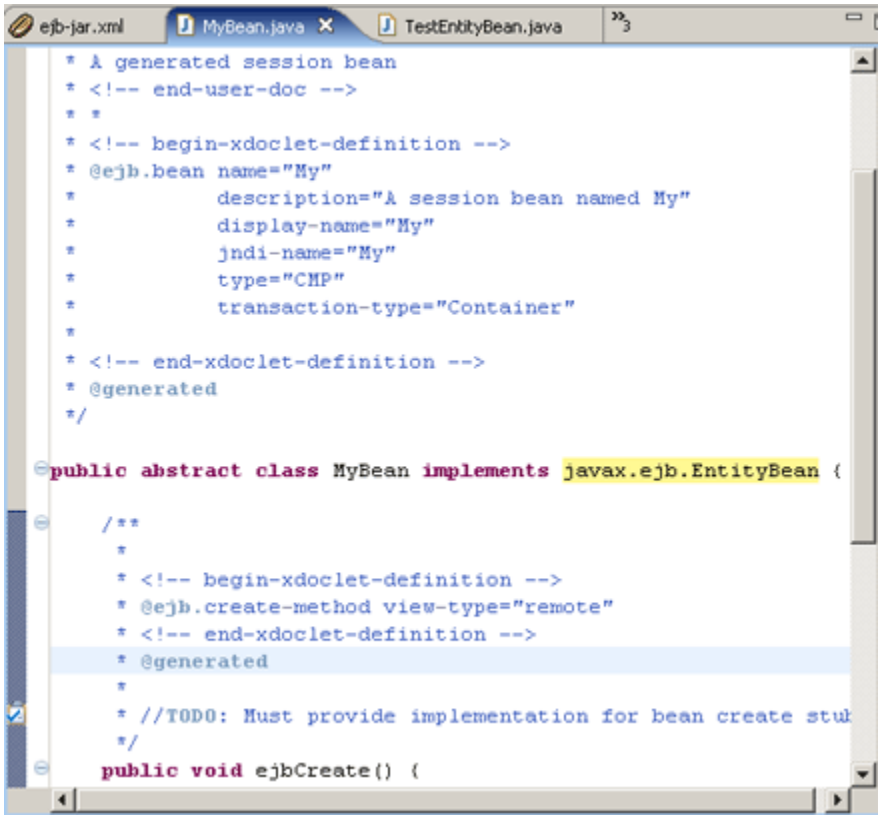
Next steps: Implementing the bean class and editing the deployment descriptor using the corresponding multipage editors.

XDoclet Approach

The SAP NetWeaver Developer Studio does not provide a wizard for XDoclet-enabled entity beans due to limitations in the WTP implementation. The following procedure represents a workaround for creating an entity bean: You generate a session bean using the wizard, and then modify the generated XDoclet annotations to define an entity bean.

1. Follow the procedure for creating XDoclet session beans.

2. Open the generated bean class. Change the value of the ***@ejb.bean type*** annotation to "CMP" or "BMP", change the implemented interface to ***javax.ejb.EntityBean***, and enter all other necessary annotations for your entity bean. For more information about using entity bean annotations, refer to the @ejb Tag Reference on the XDoclet Web site.



3. Choose *Save* from the toolbar.

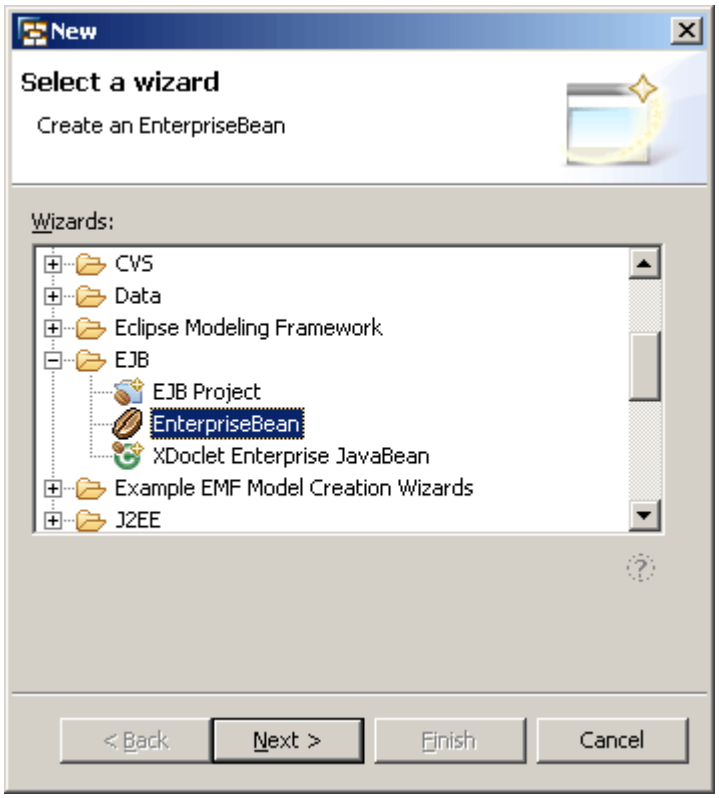
Before saving, the SAP NetWeaver Developer Studio runs the XDoclet tool. All changes resulting from the new annotations are taken into account, and the tool creates the corresponding bean interfaces.

Creating Message-Driven Beans

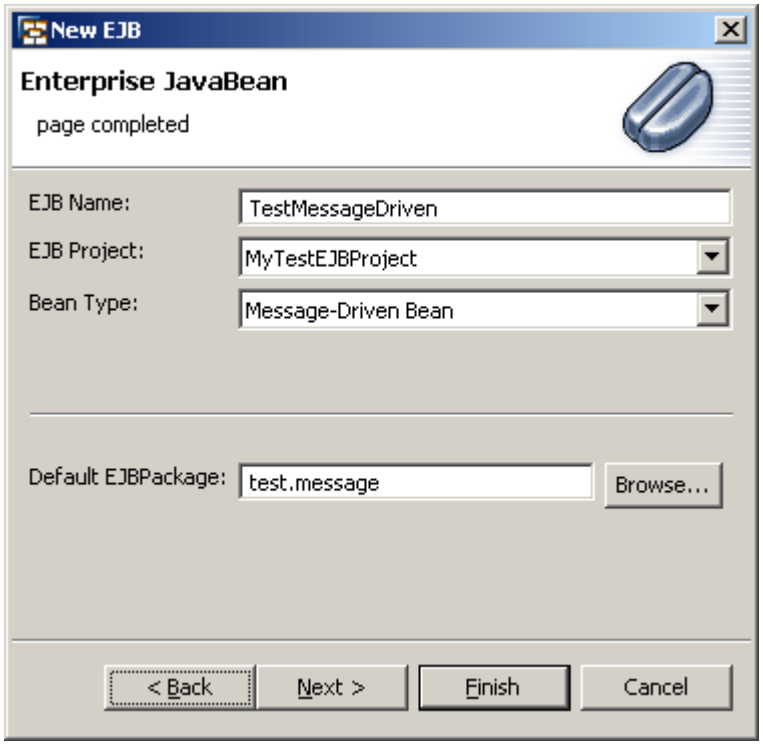
There are two alternative approaches for developing message-driven beans: the UI approach and the XDoclet approach.

UI Approach

1. In the *Project Explorer*, select the EJB Project to wrap the bean.
2. From the context menu, choose *New -> Other -> EJB -> Enterprise Bean* and choose *Next*.

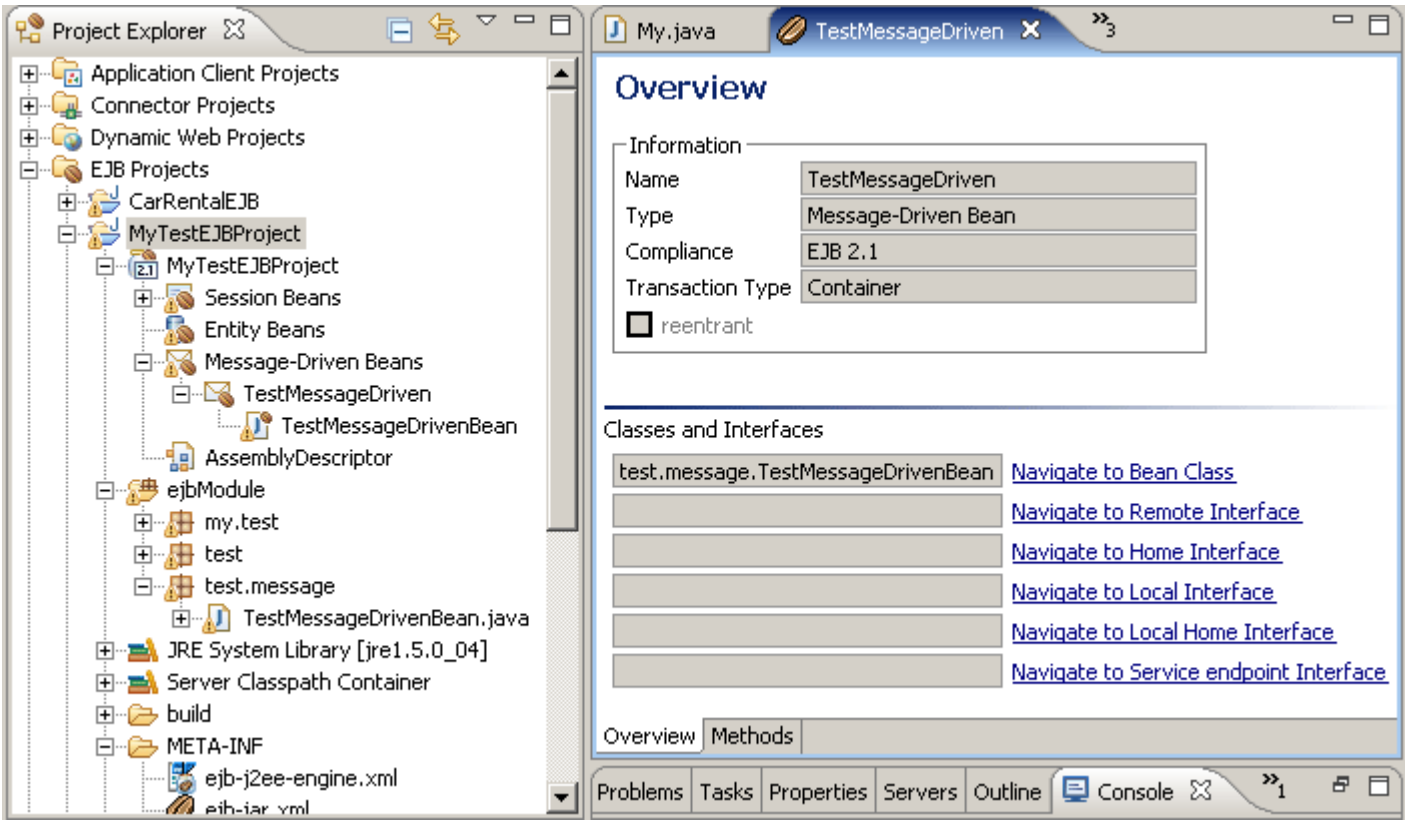


3. Throughout the wizard pages, specify the bean settings as necessary.



4. Choose *Finish*.

The new bean appears in the *Project Explorer* under the corresponding EJB Project node. The bean editor automatically opens on the right.

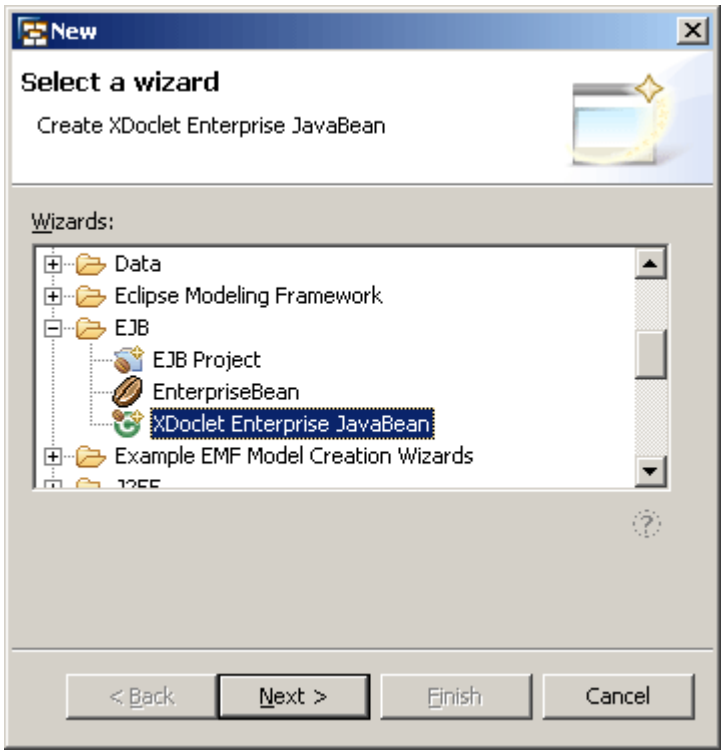


From the bean editor, you can navigate through the bean classes, and manage the bean's business methods.

Next steps: Implementing the bean class and editing the deployment descriptor using the corresponding multipage editors.

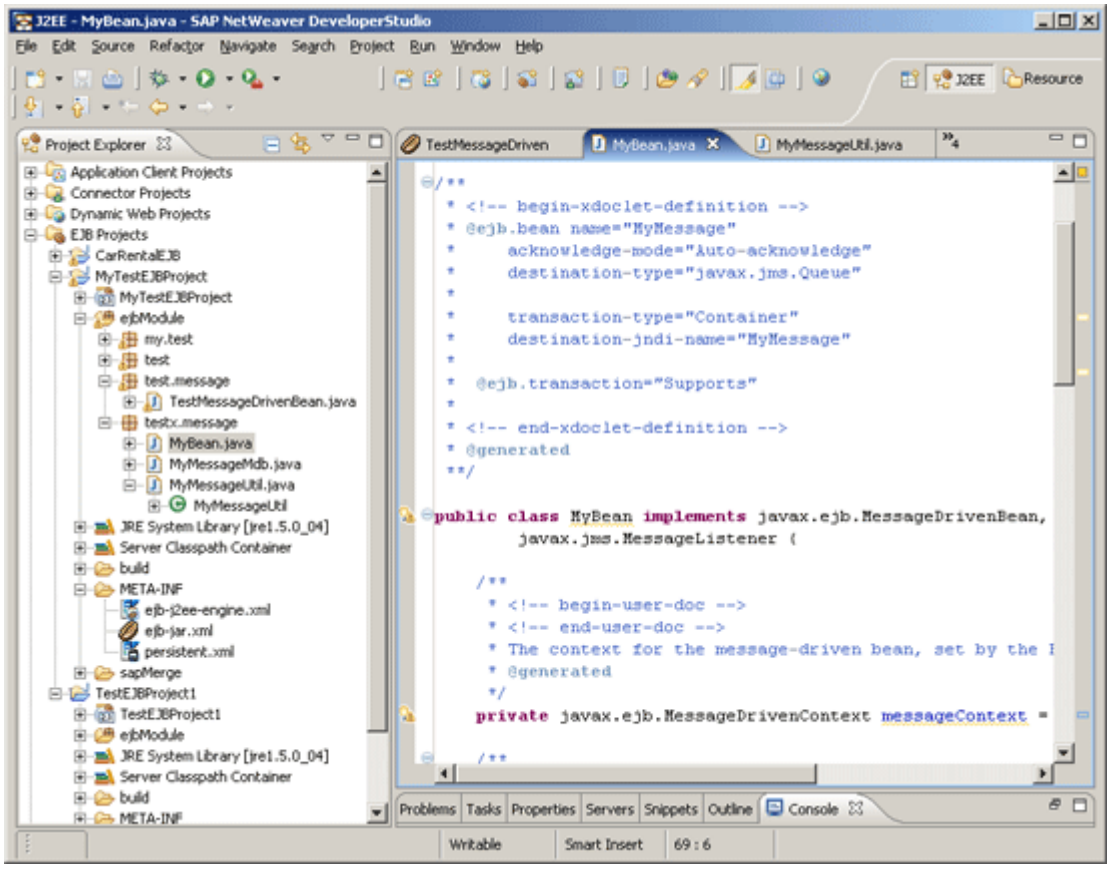
XDoclet Approach

1. In the *Project Explorer*, select the EJB Project to wrap the bean.
2. From the context menu, choose *New -> Other -> EJB -> XDoclet Enterprise Bean* and choose *Next*.



3. Throughout the wizard pages, specify the bean settings as necessary.
4. Choose *Finish*.

The new XDoclet-enabled session bean appears in the *Project Explorer*. Its classes are generated with default XDoclet annotations according to the EJB XDoclet preferences (*Window -> Preferences -> XDoclet -> ejbdoclet*). For example, if the *RemoteInterface* option is selected, the bean is generated with a remote interface.



Editing the EJB Deployment Descriptors

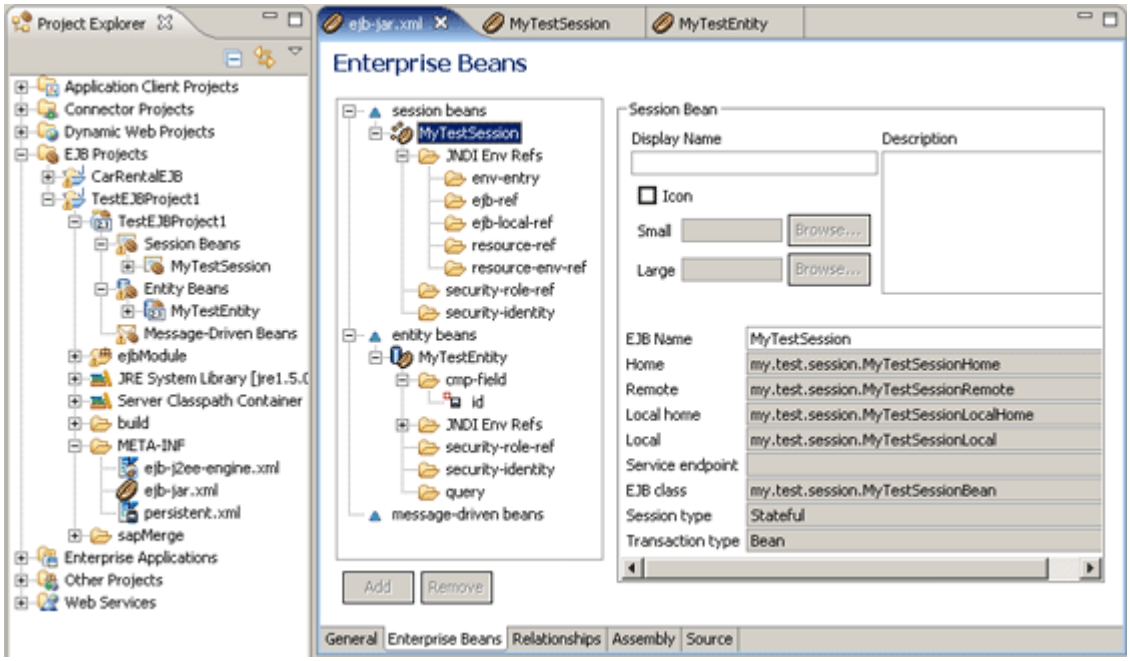
UI Approach

With the UI approach, you can edit all available EJB deployment descriptors: *ejb-jar.xml*, *ejb-j2ee-engine.xml* and *persistent.xml*. The SAP NetWeaver Developer Studio provides a different editor for each of the three descriptors. All editors have a *Source* tab, which shows the generated XML source code. You can also edit the XML directly in that tab, and the changes will be reflected in the other pages of the editor.

To edit a deployment descriptor:

1. In the *Project Explorer*, expand the relevant EJB Project and open its *META-INF* folder. All EJB deployment descriptors are placed there.

2. Doubleclick the descriptor you want to edit. The corresponding multipage editor opens on the right.



3. In the editor pages, specify the EJB settings as necessary.

4. Choose *Save* in the toolbar.

Next steps: Creating an Enterprise Application project to wrap the EJB Project.

XDoclet Approach

You can only use the XDoclet approach to edit the *ejb-jar.xml* deployment descriptor. For the SAP-specific *ejb-j2ee-engine.xml*, you can use the available multipage editor - that is, the UI approach.

Note: Do **not** edit deployment descriptors using a combination of the XDoclet and UI approach. The only exception is when you want to edit the SAP-specific deployment descriptors, which does not have XDoclet support and can be edited using the UI when you edit the rest of the deployment descriptors using XDoclet. However, if you edit a descriptor using XDoclet and subsequently make changes using the UI, you cannot edit the descriptor using XDoclet any more. The SAP NetWeaver Developer Studio configures the deployment descriptor taking only the UI into acount.

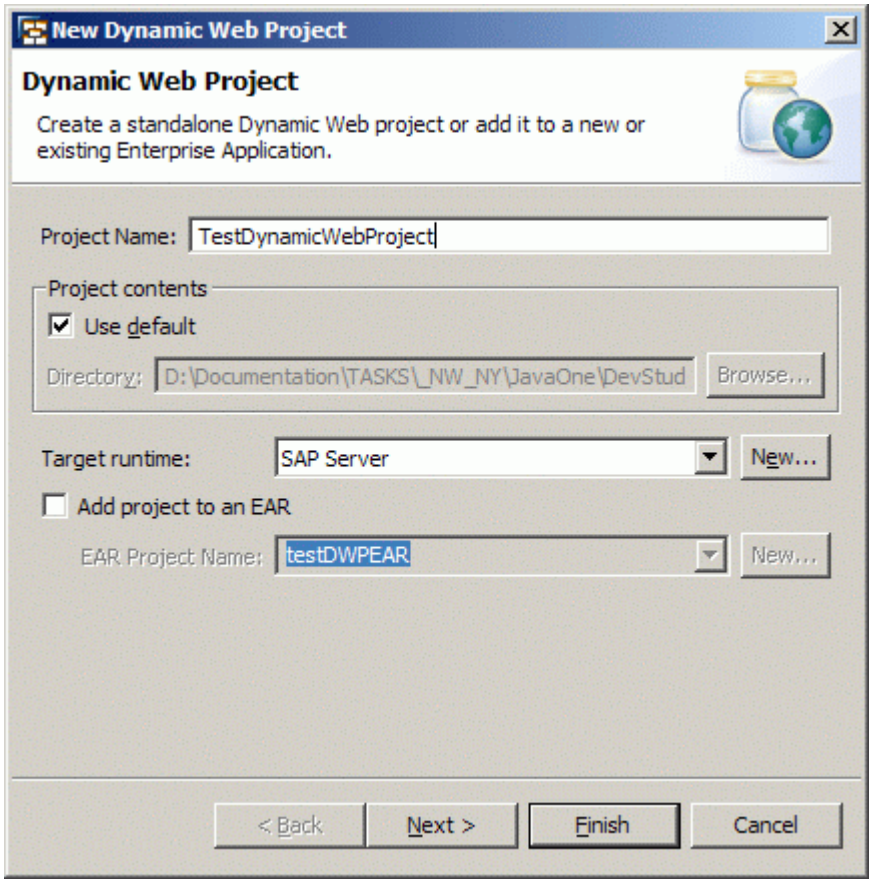
To configure the deployment descriptors by means of the XDoclet approach, you use the corresponding annotations in the bean class source code. When you save the changes in the class, the XDoclet tool automatically creates the corresponding changes in the deployment descriptors.

Creating a Web Project

1. In the *Project Explorer* select *Dynamic Web Projects* and choose from the context menu *New -> Dynamic Web Project*.

2. Click *Next*.

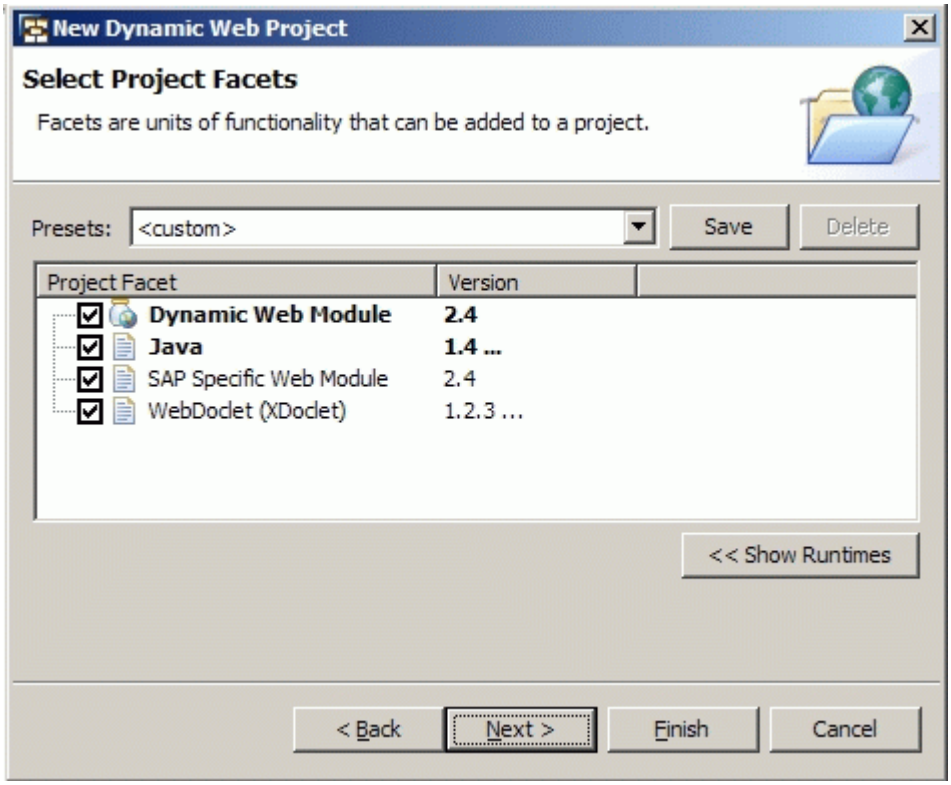
3. In the first page of the *New Dynamic Web Project* wizard, submit the basic project information.



Note: Always choose SAP Server as *Target runtime*! If you leave this field blank, the SAP NetWeaver Developer Studio cannot find the necessary WEB libraries for building the project.

4. Choose *Next*.

5. In the next wizard page, you can define the project facets.



If you use the same facet settings for all Web Projects, you can save the currently selected facets as a ready preset option. To do so, write a name for the facet and select all necessary facets and choose *Save*.

Note: You can change the project facets later at any time from *Properties -> Project Facets* in the context menu of the project.

If you want to use the XDoclet approach for developing Web components, check the *WebDoclet* option. Note that you can not use the multi page editors after you use the XDoclet options.

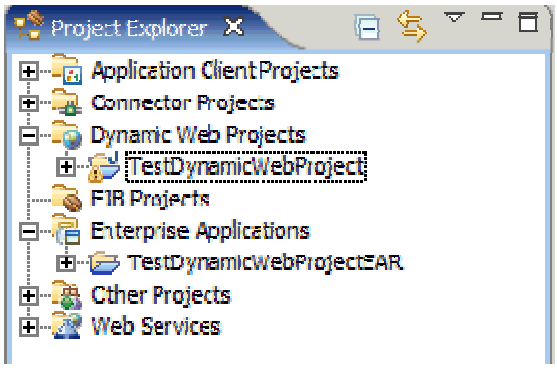
If you don't want to use SAP-specific deployment descriptors (*web-j2ee-engine.xml*), uncheck the *SAP Specific Web Module* option.

6. Choose *Next*.

7. If necessary, configure the context root.

8. Choose *Finish*.

The Dynamic Web Project appears in the *Project Explorer* under Dynamic Web Projects and Enterprise Applications (if you have selected to add the web project to an EAR).



9. You can now proceed with the creation and modification of the web project component classes (such as servlet, JSPs and deployment descriptors).

Creating Servlets

For information go to Creating Servlets .

Creating JSPs

For information go to Creating JavaServer Pages (JSP) files .

Editing Deployment Descriptors

You can use the XDoclet approach to edit only the *web.xml* deployment descriptors. For the SAP-specific *web-j2ee-engine.xml*, you can use the available multipage editor, that is the UI approach.

You can do so using two approaches:

UI Approach

Use the multipage editors of the SAP NetWeaver Developer Studio:

1. Open the deployment descriptor (web.xml or web-j2ee-engine.xml)
2. Select the necessary tab and enter your changes.
3. Save the file.

Warning: You should not use the UI approach after you have used the XDoclet approach. For more information, see Merging the Changes.

XDoclet Approach

The *WebDoclet* facet must be installed on the project. To do so:

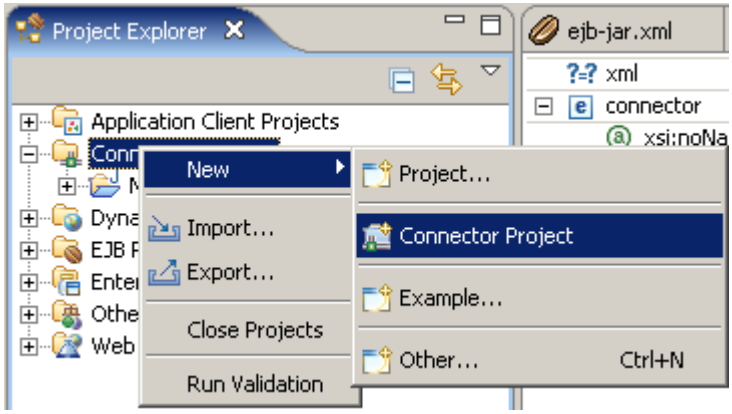
1. Select the project
2. Open from the context menu *Properties -> Project Facets*
3. Install it on the project with the *Add/Remove Project Facets* button.

To modify the tags added by XDoclet in your source code:

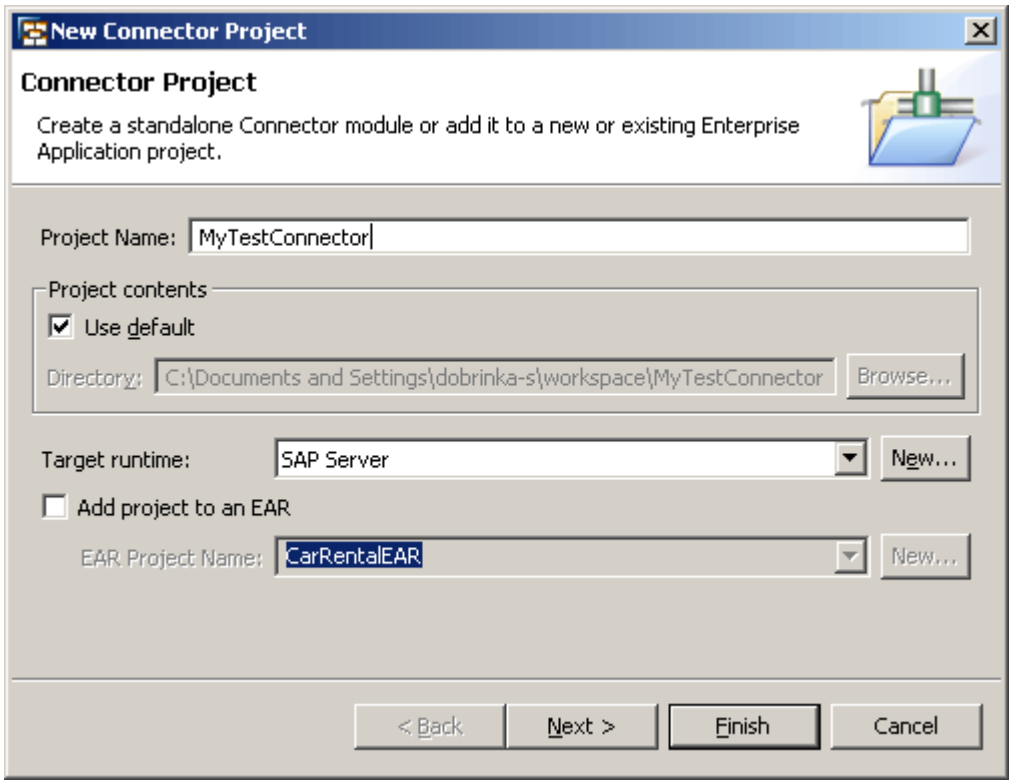
1. Open a source file.
2. Modify the XDoclet tags at the beginning of the file.
3. Save the file. The changes will be automatically reflected in the deployment descriptor files by XDoclet.

Creating Connectors

1. In the *Project Explorer*, select the *Connector Projects* folder and choose *New -> Connector Project* from the context menu.



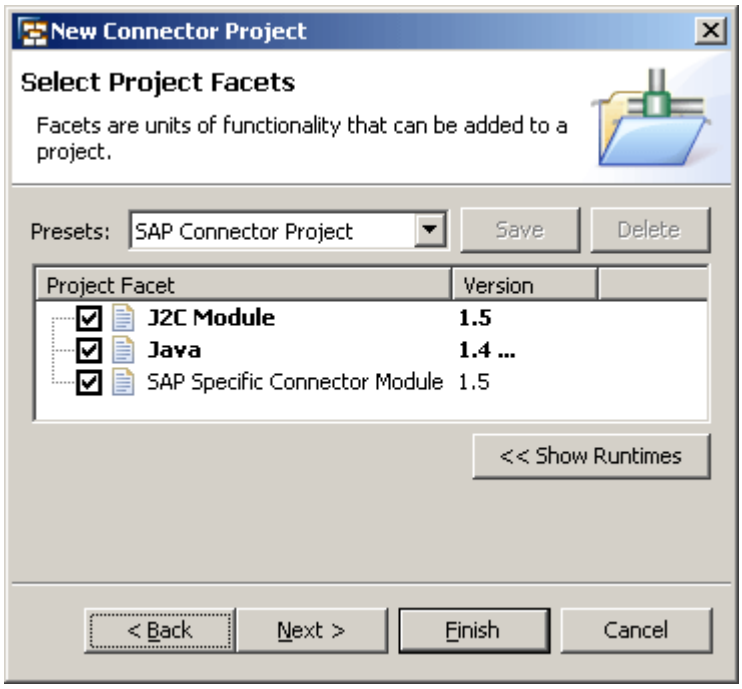
2. On the first page of the New Connector Project wizard, specify the basic project information, and choose *Next*.



Note: Always choose SAP Server as the *Target runtime*. If you leave this field blank, the SAP NetWeaver Developer Studio cannot find the necessary J2EE libraries for building the project.

3. On the next wizard page, you can define the project facets.

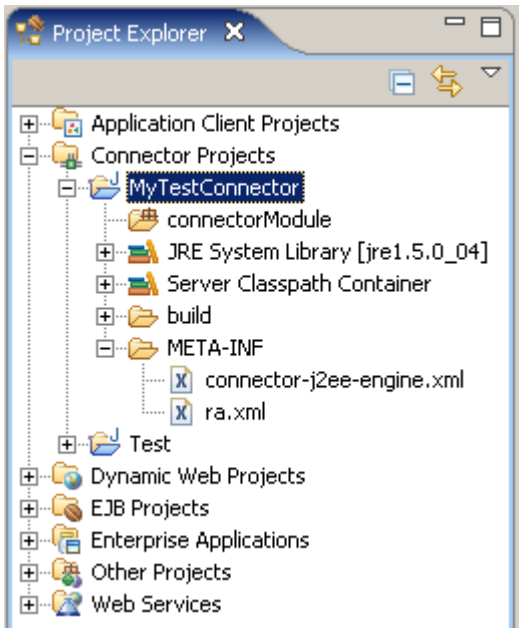
   It is recommended that you use the default facets.



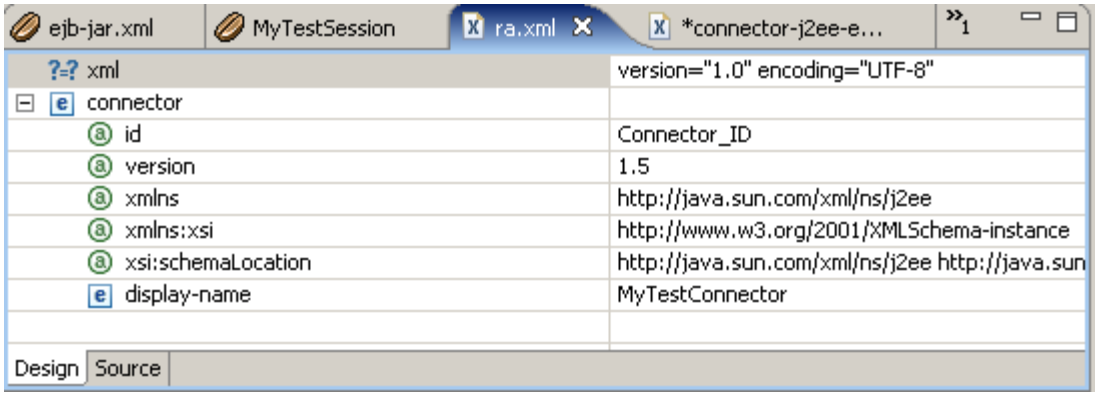
4. Choose *Finish*.

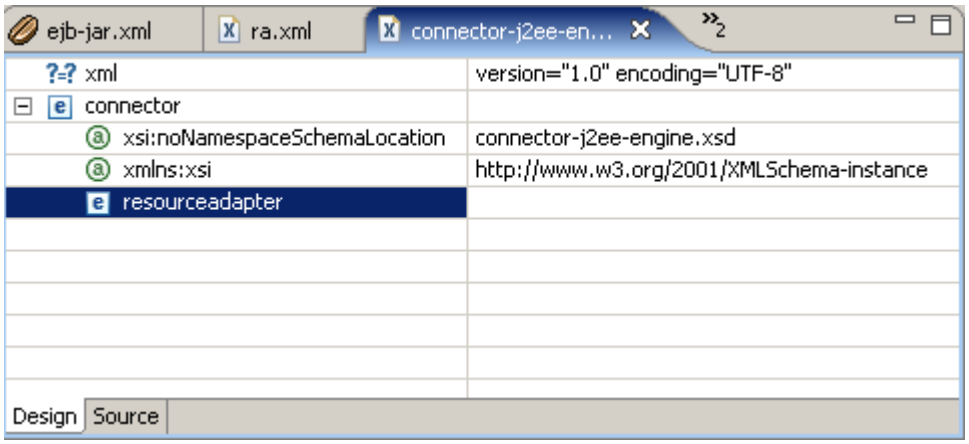
   Note: The SAP NetWeaver Developer Studio works only with the default source directory for the project. Even if you specify a different directory, the default one will be used. This limitation is described in Limitations.

   The Connector Project appears in the *Project Explorer*.



5. Edit and save the *ra.xml* and *connector-j2ee-engine.xml* using the default XML editor.

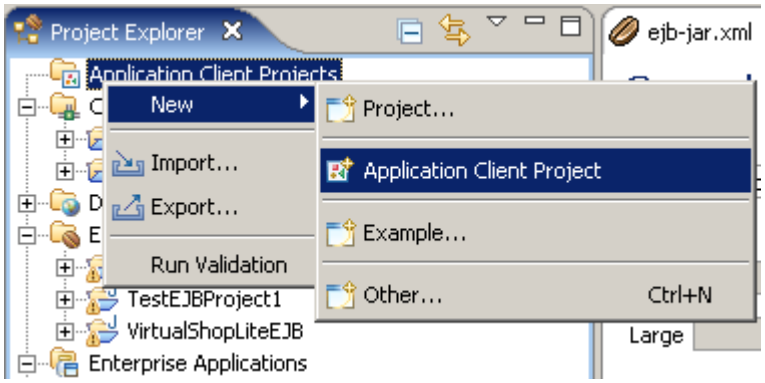




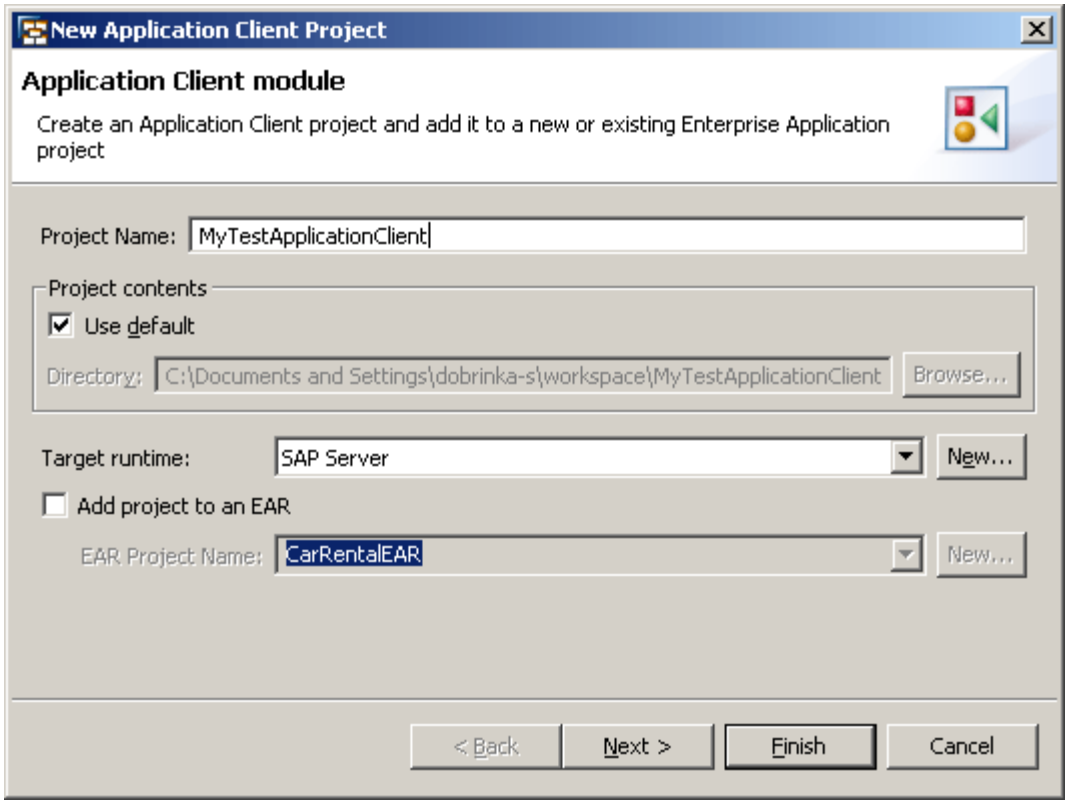
6. Add the Connector Project to an Enterprise Application Project.

Creating Application Clients

1. In the Project Explorer, select the Application Client Projects folder and choose New -> Application Client Project from the context menu.



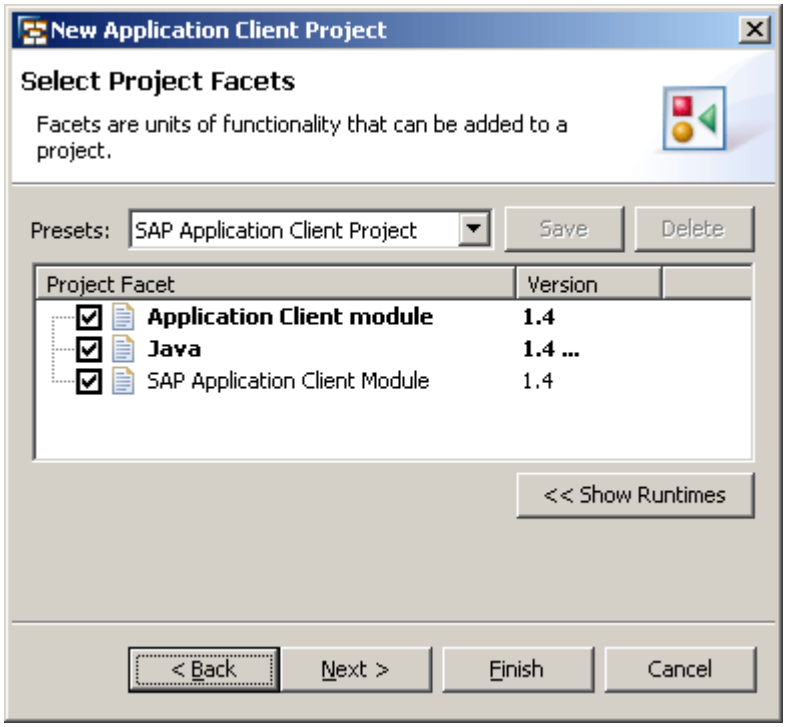
2. On the first page of the New Application Client Project wizard, specify the basic project information, and choose *Next*.



Note: Always choose SAP Server as the *Target runtime*. If you leave this field blank, the SAP NetWeaver Developer Studio cannot find the necessary J2EE libraries for building the project.

3. On the next wizard page, you can define the project facets.

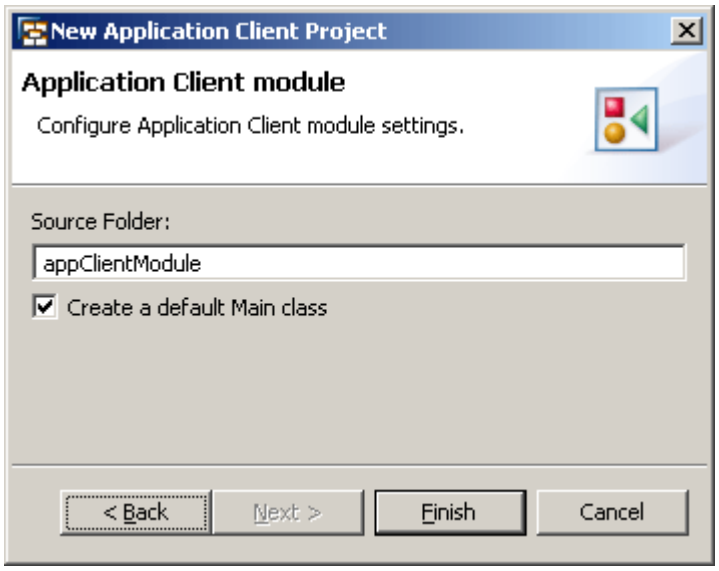
   It is recommended that you use the default facets.



4. Choose *Next*.

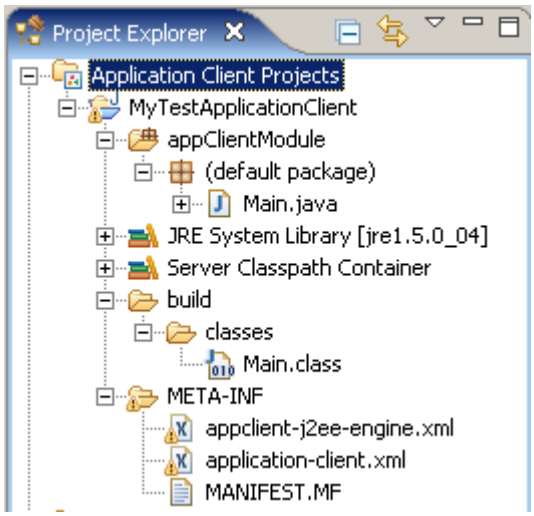
5. If necessary, create a default main class by choosing the corresponding option.

   Note: The SAP NetWeaver Developer Studio works only with the default source directory for the project. Even if you specify a different directory, the default one will be used. This limitation is described in Limitations.

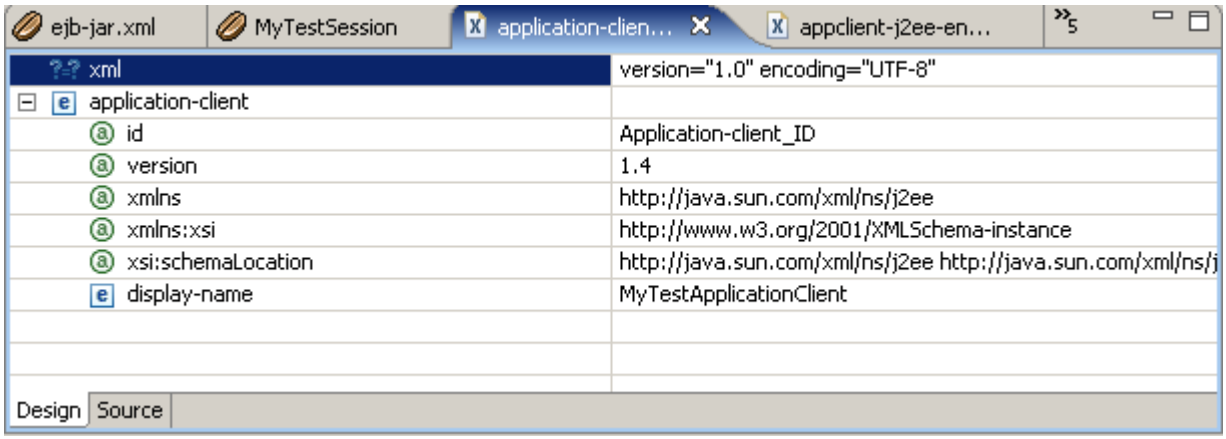


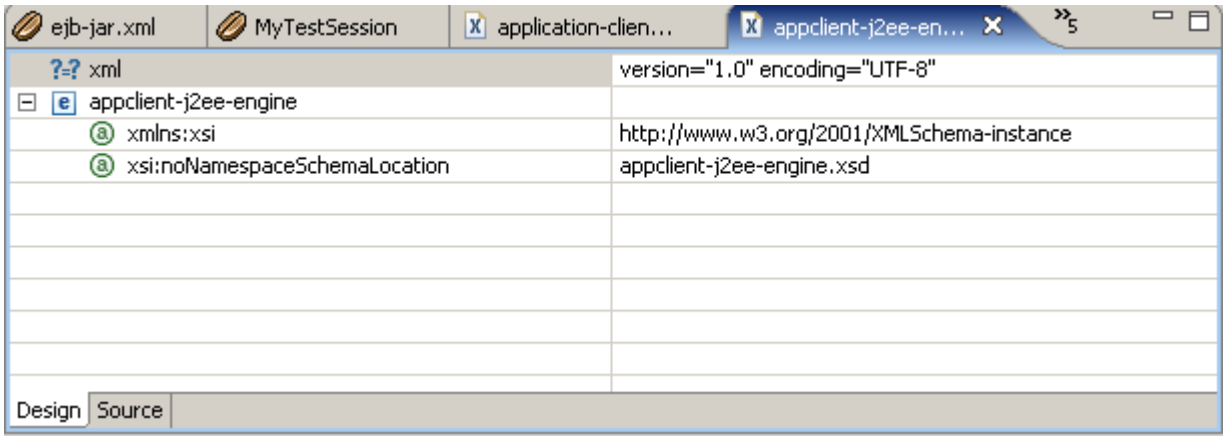
6. Choose *Finish*.

   The Application Client Project appears in the *Project Explorer*.



7. Edit and save the *application-client.xml* and *appclient-j2ee-engine.xml* using the default XML editor.

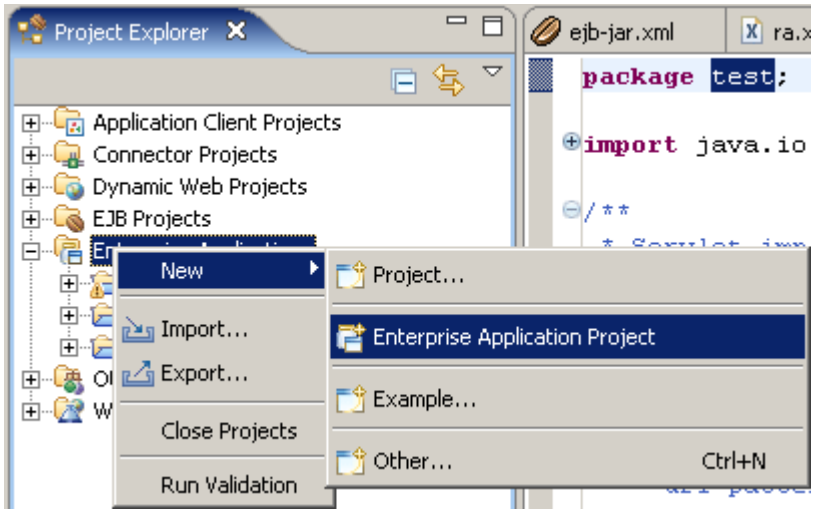




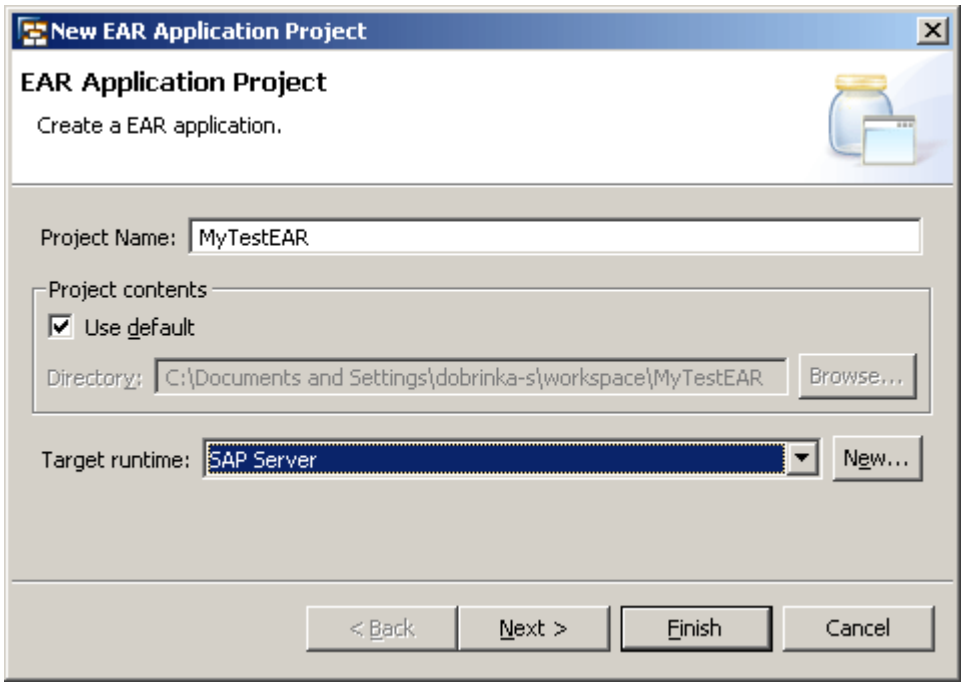
8. If you created a main class, edit and save it.
9. If you haven't already done so, add the Application Client Project to an Enterprise Application Project.

Creating an Enterprise Application Project

1. In the Project Explorer, select the Enterprise Applications folder and choose New -> Enterprise Application Project from the context menu.



2. In the first page of the New Application EAR Project wizard, submit the basic project information, and choose *Next*.



Note: Always choose SAP Runtime as *Target runtime*! If you leave this field blank, the SAP NetWeaver Developer Studio cannot find the necessary J2EE libraries for building the project.

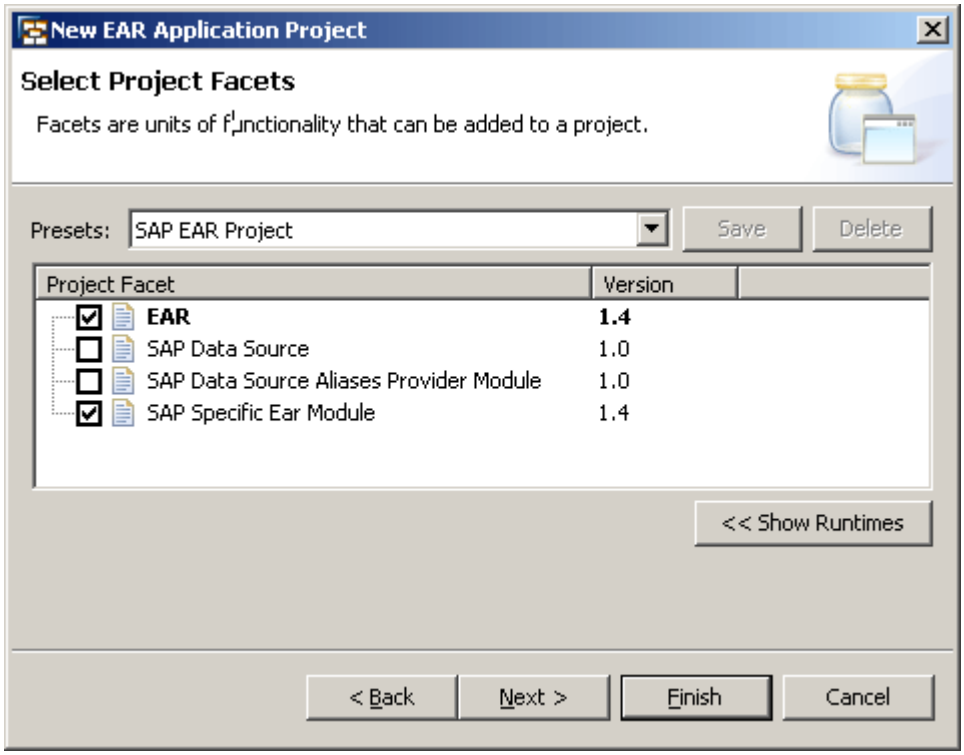
3. In the next wizard page, you can define the project facets.

   If you want to use data-sources.xml or data-source-aliases.xml, check the SAP Data Source or SAP Data Source Aliases Provider option respectively.

   If you want to use SAP-specific deployment descriptor (application-j2ee-engine.xml), check the SAP Specific Ear Module option.

   Hint: If you use the same facet settings for all EJB Projects, you can save the currently selected facets as a ready preset option. To do so, select all necessary facets and choose Save.

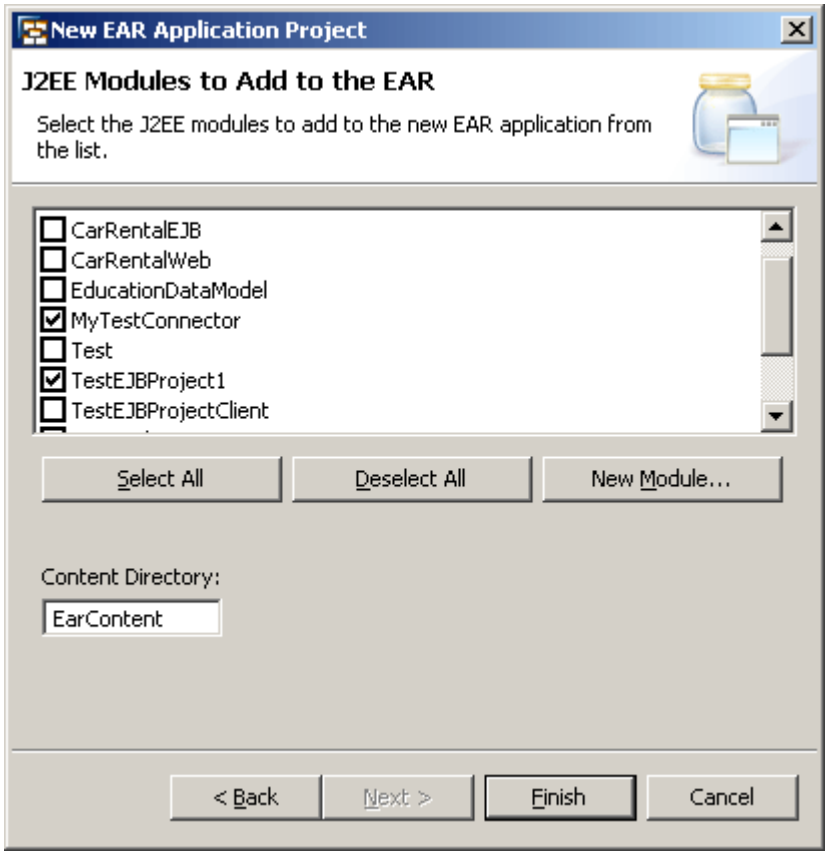
   Note: You can change the project facets later at any time from Properties -> Project Facets in the context menu over the project.



4. Choose *Next*.

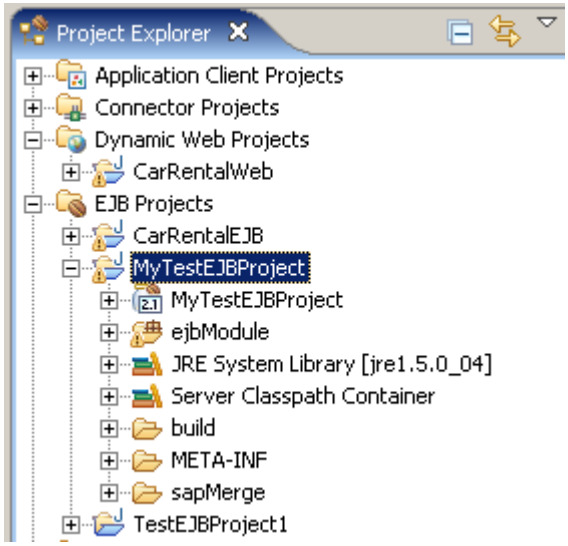
5. Choose the projects that will be added to the enterprise application.

   Note: The SAP NetWeaver Developer Studio works only with the default source directory for the project. Even if you specify a different directory, the default one will be used. This limitation is described in Limitations.



6. Choose *Finish*.

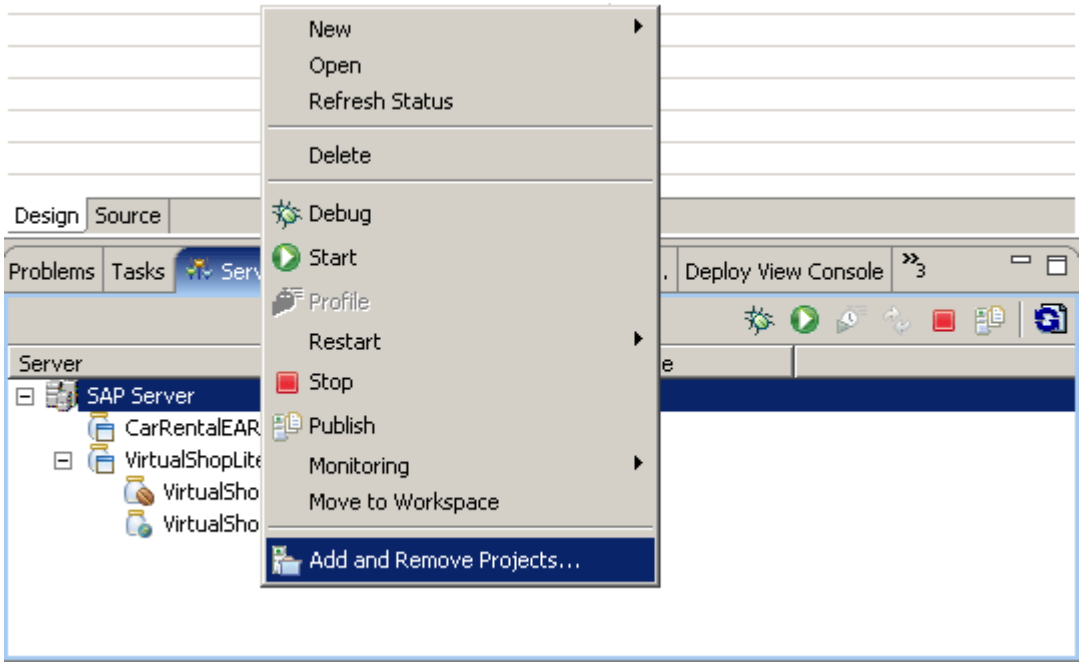
The Enterprise Application Project appears in the *Project Explorer*.

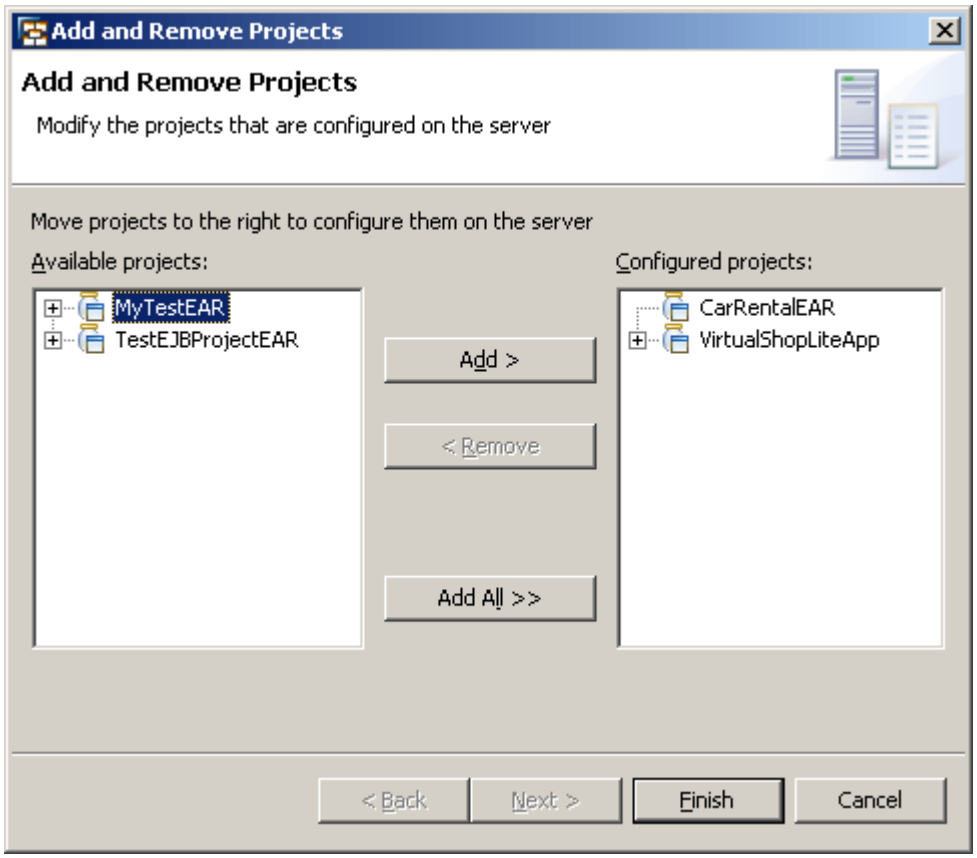


Deploying Applications

Deploying Enterprise Application Projects

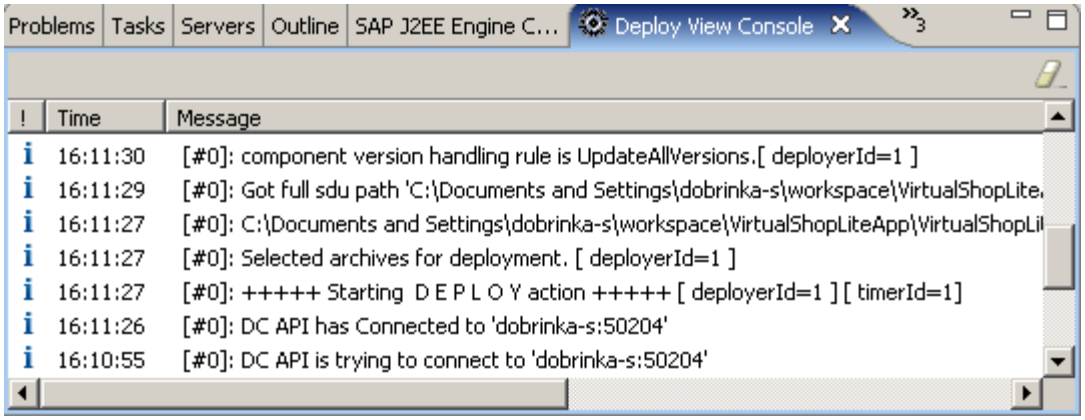
1. Use the *Add and Remove Projects* option from the context menu in the *Servers* view, and select the new projects to add there.



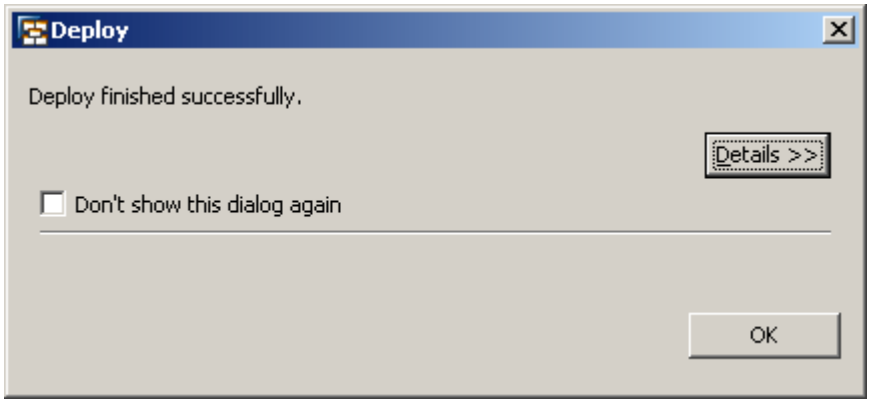


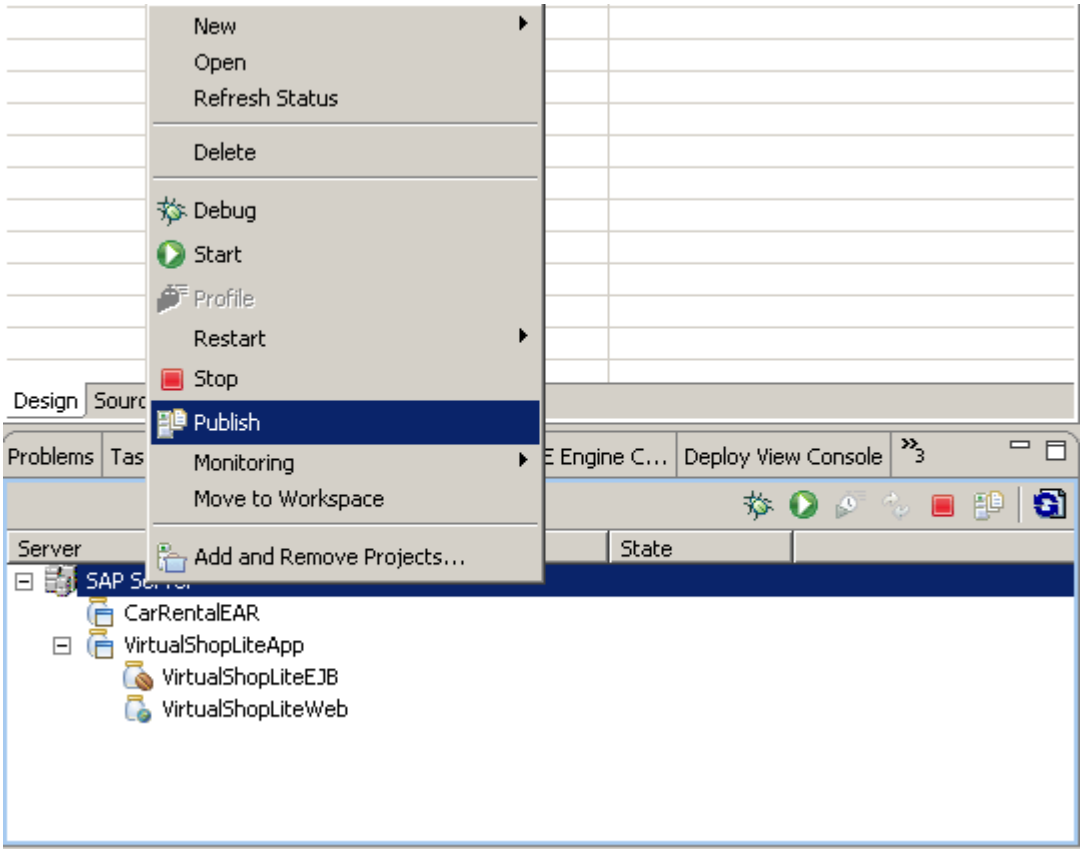
2. To deploy the applications, select the *SAP Server* node and choose *Publish* from the context menu.

   The SAP NetWeaver Developer Studio switches to the *Deploy View Console* view. The current results from the deploy operations are displayed there.



If the deployment finishes successfully, you will see a success dialog:





Note: The *Publish* option deploys **all** projects added to the *Servers* view.

Re-deploying Enterprise Application Projects

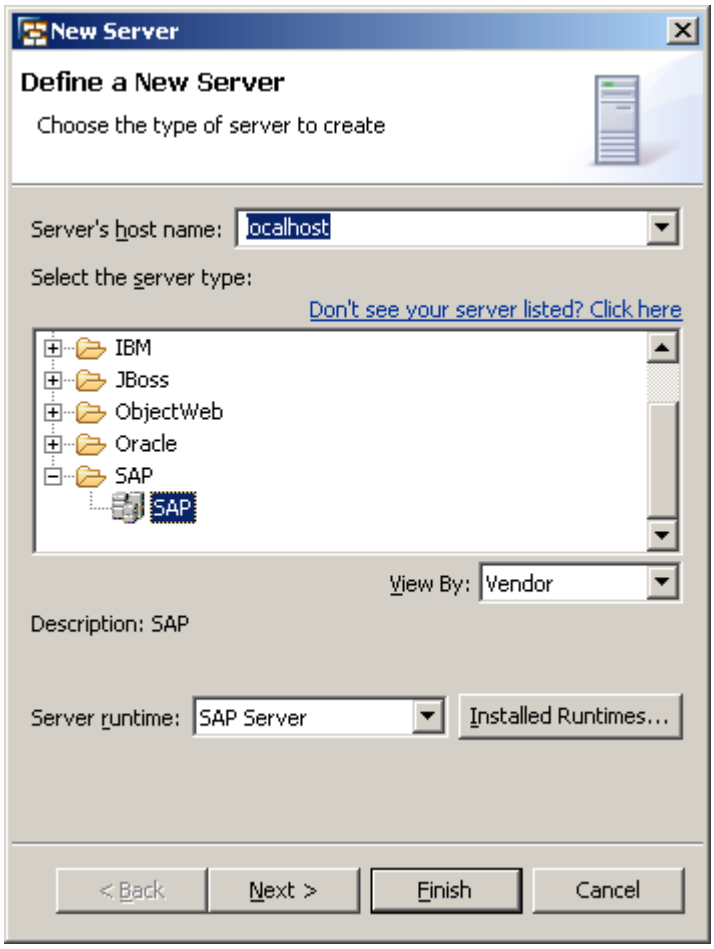
Once you have added an Enterprise Application Project in the *Servers* view, you can re-deploy it many times from there. You only choose *Publish* from the context menu of the *SAP Server* node in the *Servers* view, and the deployment of the projects starts (see also step 8 of the Adding a SAP Server Instance section).

Adding a SAP Server Instance

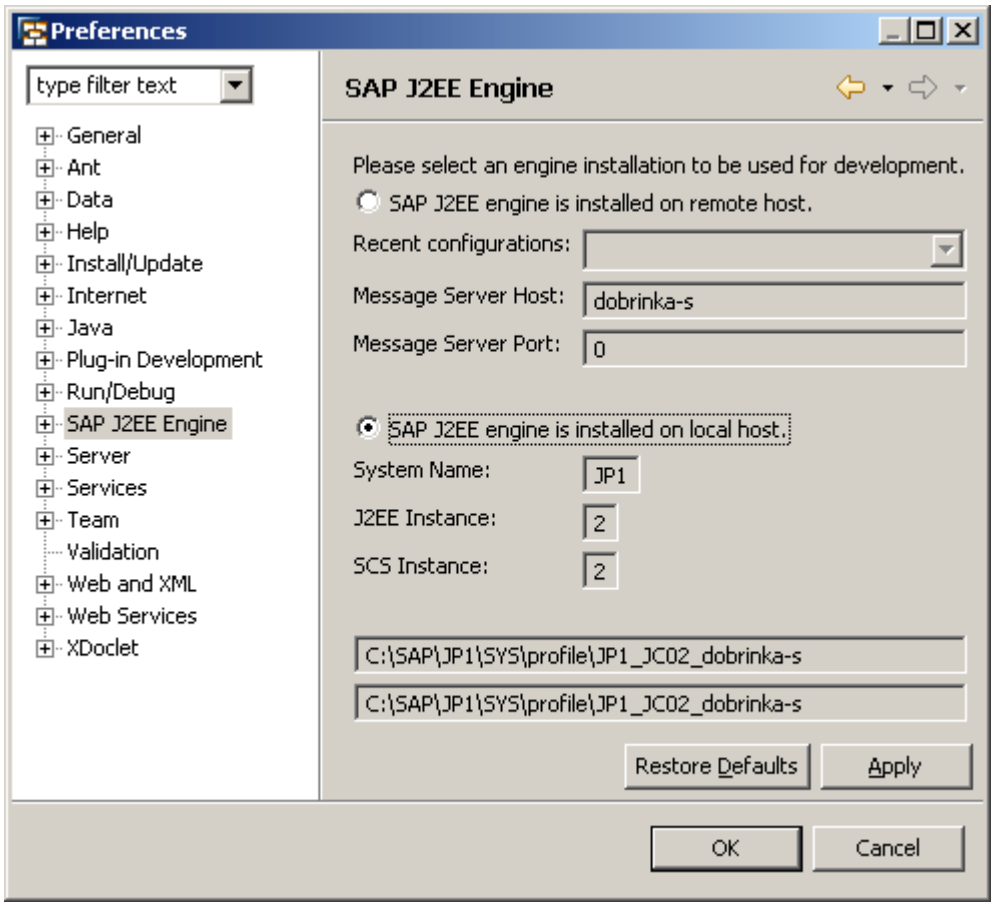
By default, you have a pre-defined SAP Server instance in the *Servers* view, and you do not have to add one manually. You have to do so, if you want to connect to a different instance of the J2EE Engine from the one installed initially with the SAP NetWeaver Developer Studio (of example, this could be a test PC with a running J2EE Engine).

To add a new SAP Server instance:

1. If the *Servers* view is not already open, open it by choosing *Window -> Show View -> Other -> Server -> Servers*.
2. From the context menu in the *Servers* view, choose *New -> Server*.
3. In the New Server wizard, choose *SAP -> SAP* and then *Next*.

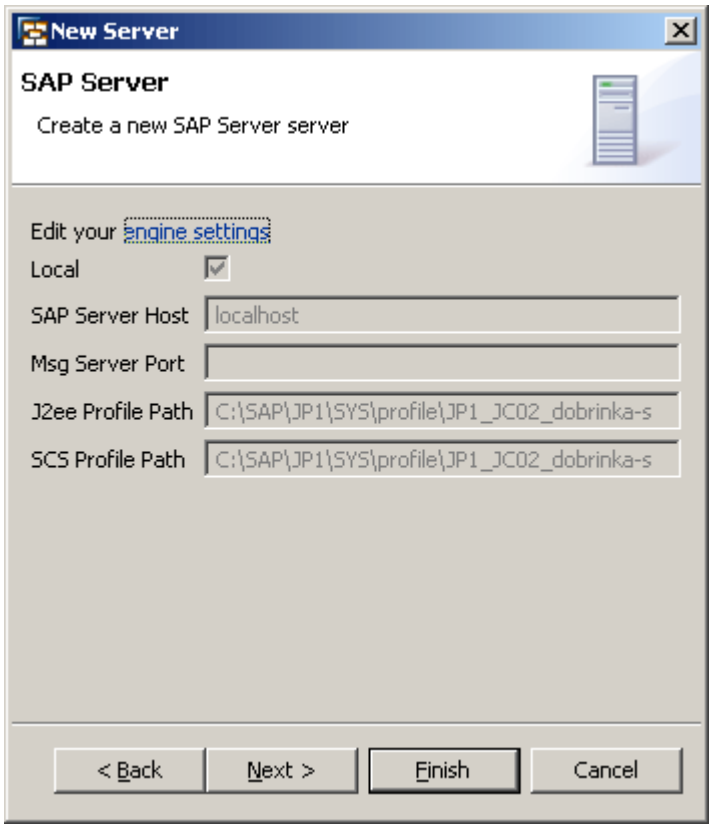


4. Check if the *SAP Server Host* and *Msg Server Port* settings have correct values. If they do not, specify correct values by choosing the *engine settings* link (it opens the *SAP J2EE Engine* section of the *Preferences* dialog).



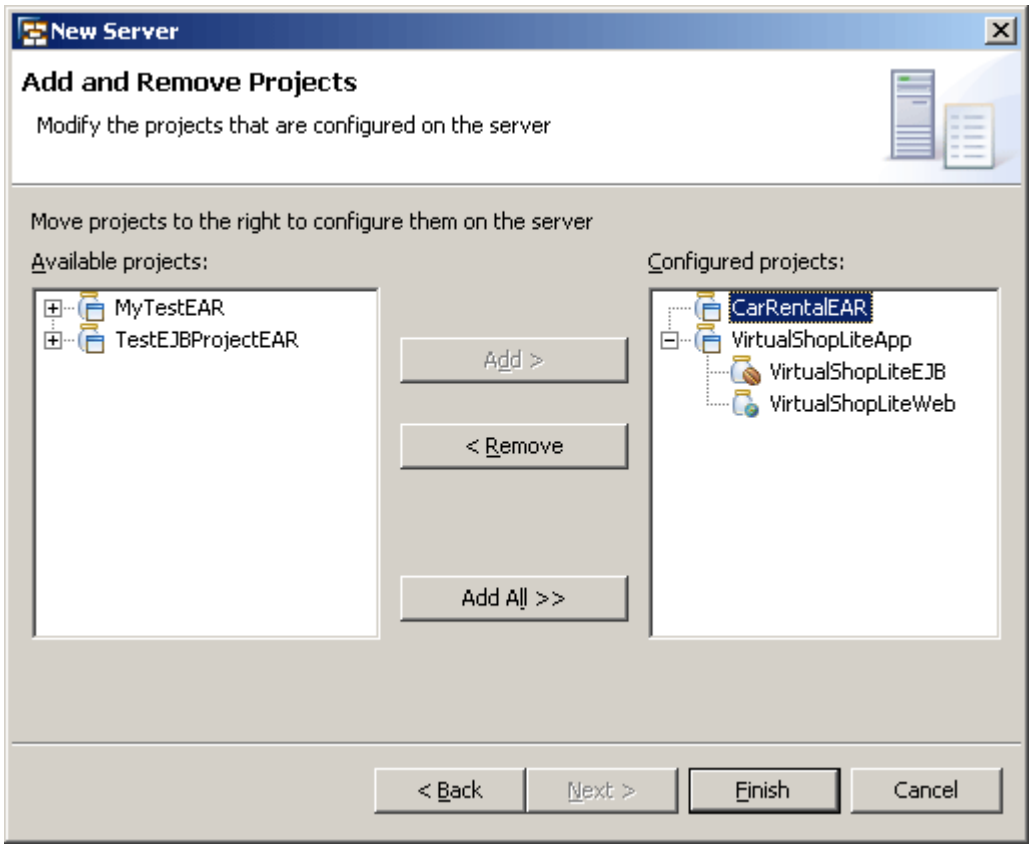
Note: These settings (message server host and port) are **required** for connecting to the J2EE Engine.

Note: By default, the *Message Server Host* and the *Message Server Port* is are set to the host and message port of the J2EE Engine installed along with the SAP NetWeaver Developer Studio by the JavaOne Installer.



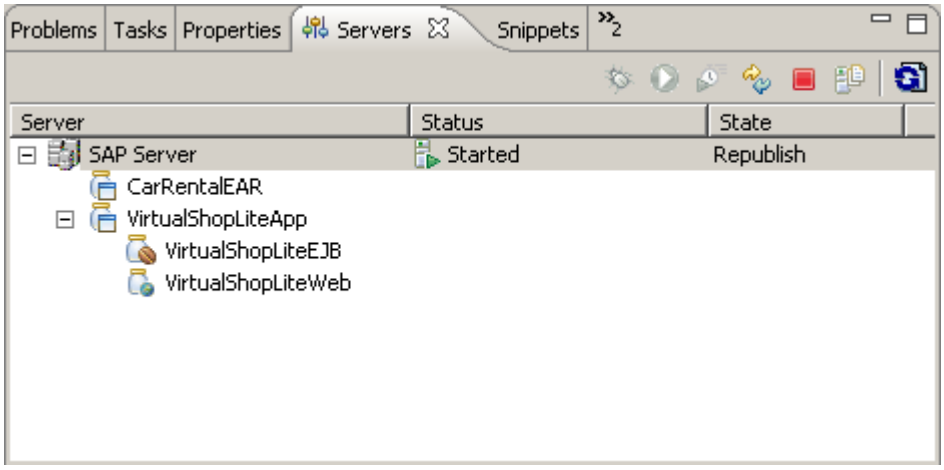
5. Choose *Next*.

6. Choose the Enterprise Application Projects that you want to deploy on the J2EE Engine at this stage (you can always add new projects from the *Servers* view).



7. Choose *Finish*.

   The *SAP Server* entry appears in the *Servers* view.



8. To deploy the applications you selected, select the *SAP Server* node and choose *Publish* from the context menu.

Copyright