J2EE Toolset in Action: Creating Web Applications

Applies To:

SAP NetWeaver Developer Studio 6.40, Service Pack 13.

Summary

This article describes how the SAP NetWeaver Developer Studio's J2EE toolset enables the development of Java applications.

By: Dobrinka Stefanova

Company: SAP AG

Date: January 23, 2006

Table of Contents

Introduction

"Action is eloquence"
William Shakespeare

Imagine your typical workday as developer. How many hours a day do you spend writing specifications, replying to e-mails, debugging, explaining to morons, sorry, misguided colleagues? And how many hours do you spend doing the real task you were hired for: programming? Haven't you asked yourself many times: "What am I: a developer, or secretary, a shrink or what?"! Let's say it straight: a developer's secret dream is to be left to code in peace on a desert island somewhere in the Pacific (well, with a beautiful girl living next island, needing help with her computer).

Now, imagine you have a devoted assistant who is ready to handle most boring tasks for you, so that you can program blissfully instead of wasting your precious mind on crap. Moreover, this assistant is even ready to handle the routine programming activities, and leave for you only the parts that call for imagination, ingenuity and brilliance of mind (yours). He also builds, deploys and runs while you do more important tasks (chat with friends on ICQ). How does it sound – brilliant, doesn't it? The only thing left is to find the right assistant among all candidates…

Well, you lucky boys and girls, it happens so that we have just the right one! It is called J2EE Toolset and is the part of the SAP NetWeaver Developer Studio enabling development of J2EE applications. Of course, we know you wouldn't hire someone if you hadn't tested his abilities and seen how he works. So, let's use this article as a demonstration of our friend's capabilities and devotion.

We'll create a sample Web application using the toolset, so that you can see this little buddy means business.

The Web Application: ITelO Project Database

Our Web application will be called ITelO Project Database. It is the user interface to the project database of a company named ITelO. It allows you to view detailed information about a selected project, such as participants, lead, and employee skills. You can also send employees to education courses from here.

Figure 1: General look-and-feel of the application

The underlying company database is provided by the Education Data Model (EDM) project. We won't go over the EDM in detail here. We'll only concentrate on our Web application, which acts as the uppermost layer of the EDM. Please, import the EDM projects from and configure the EDM from the **associated project file**, using the instructions in the *readme.html*.

The J2EE Development Perspective

The J2EE Toolset provides a set of J2EE-related views, assembled in the *J2EE Development* perspective. You can open it by using *Window → Open Perspective → Other*, and then *J2EE Development*. The views of this perspective show different information related to the development and deployment of J2EE applications.

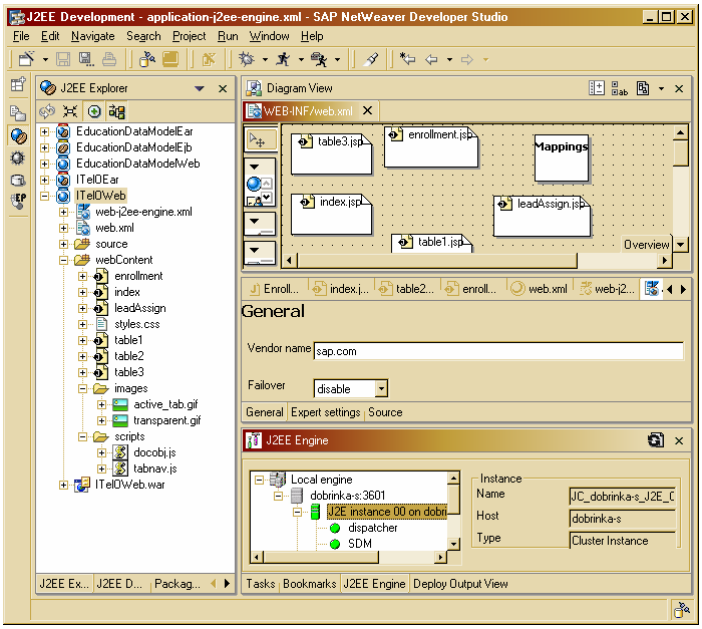


Figure 2: J2EE Development perspective

The J2EE Development perspective consists of the following views:

- *J2EE Explorer*/*J2EE DC Explorer* – Shows the ordinary and DC J2EE projects and their components
- *Diagram* – Shows the UML diagram representing the project, folder or package
- *J2EE Engine* – Shows the status of the J2EE Engine cluster
- *Deploy Output* – Shows the result of the deploy operations

As you can see, the *J2EE Development* perspective provides enough information for the whole process of development of J2EE applications, from the creating of the source code to the deploying on the J2EE Engine.

Creating a Wrapper: The Web Project

In the SAP NetWeaver Developer Studio, Web projects are the "wrapper" for all Web components that eventually will be packed in the same WAR. A Web project provides:

- Wizards for generating JSP, servlet, listener and other Web components – These handle the boring tasks of creating standard components according to the J2EE specification. The wizards generate the stubs in the source code, depending on the inherited classes and interfaces, and other specified settings.
- Editors for manipulation of Web deployment editors – The availability of editors saves you most work over the complex deployment descriptors. You can model the descriptors visually. We'll go over these editors in greater detail later on in this article.

You can easily build and maintain the Web archive (WAR) using a Web project.

Here is the procedure for creating a Web project for the ITelO Project Database:

1. In the SAP NetWeaver Developer Studio, select the *File → New → Project* menu path.

   The *New Project Wizard* appears.

2. Select *J2EE → Web Module Project* and press *Next*.
3. In the next wizard page, enter `EDMTestWeb` as project name, select the desired location, and press *Finish*.

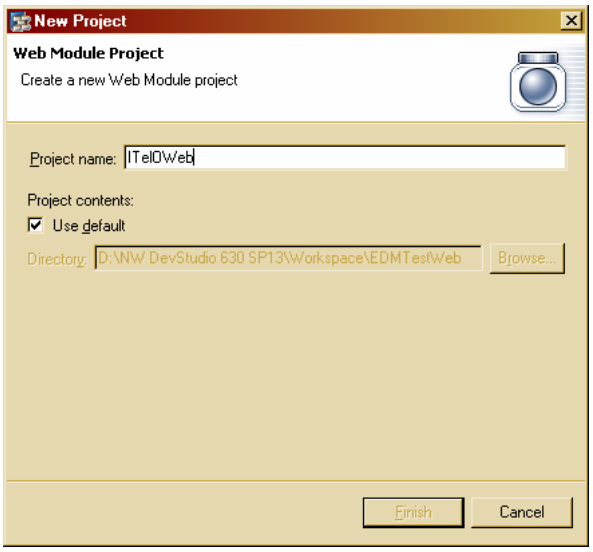


Figure 3: Web project wizard

The created project appears in the *J2EE Explorer* view. It has the deployment descriptors (*web.xml* and *web-j2ee-engine.xml*) and two folders:

- *source* – It will contain the Java source files of the project.
- *webContent* – It will contain the Web components (JSPs and HTMLs).

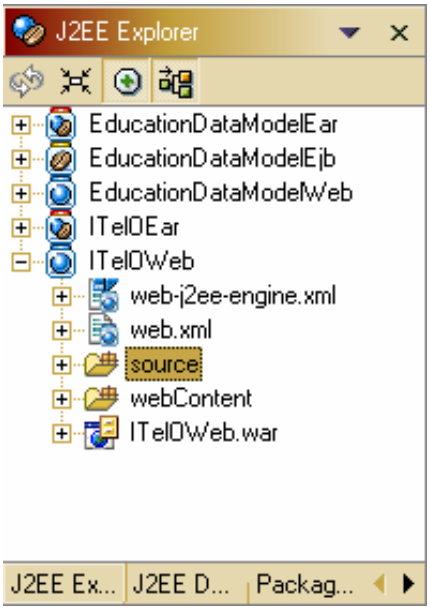


Figure 4: The EDMTestWeb project in the J2EE Explorer

Creating a JSP File

For the JSP files, the J2EE Toolset provides the JSP editor with the following features:

- Code assistance – You can get code assistance for the JSP and HTML tags, and for the request and response attributes of the doPost(HttpServletRequest request, HttpServletResponse response) method of the HTTP servlet corresponding to the compiled JSP.
- Web preview – You can preview the developed JSP as it would look in the Web browser.

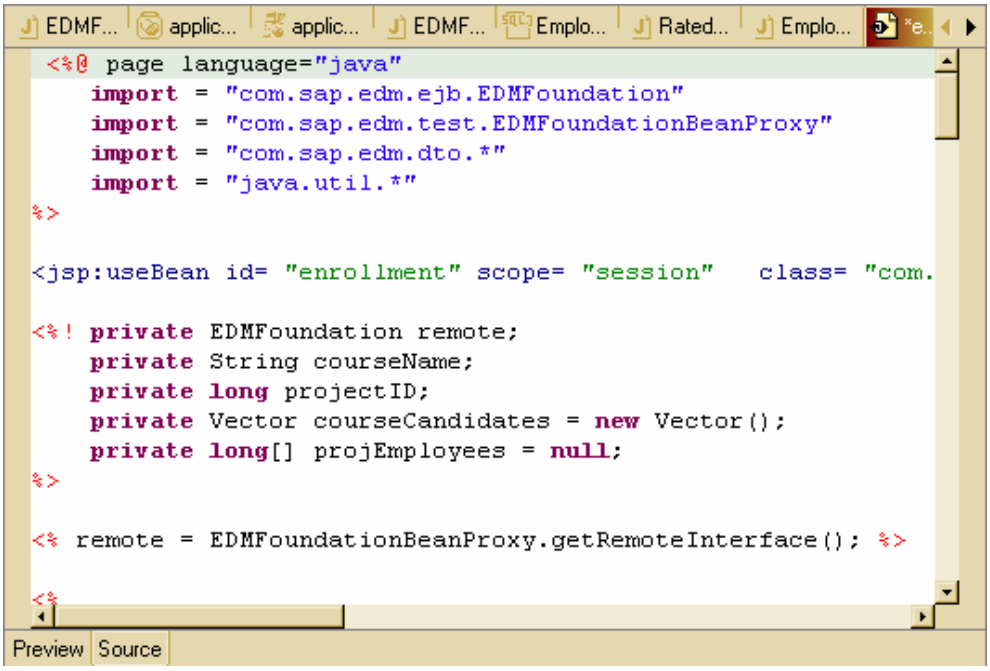


Figure 5: JSP Editor

For illustration, here is the procedure for creating the *intex.jsp* of the ITelO Project Database:

1. In the *J2EE Explorer* view, select the *EDMTestWeb* project node or the *webComponent* node of the project.
2. From the context menu, select *New → JSP*.
3. Enter `index.jsp` as JSP name and select *Finish*.

The JSP file appears in the project in the *webContent* folder. Automatically, the JSP is also described in the *web.xml* deployment descriptor. The following XML source is added:

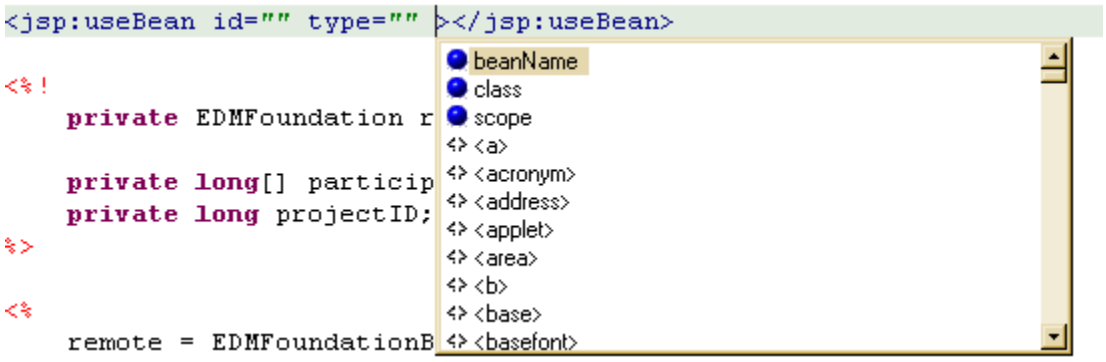
```
<servlet>
  <servlet-name>index.jsp</servlet-name>
  <jsp-file>/index.jsp</jsp-file>
</servlet>
```

To understand better the features of the JSP editor, try playing around with the JSP source code.

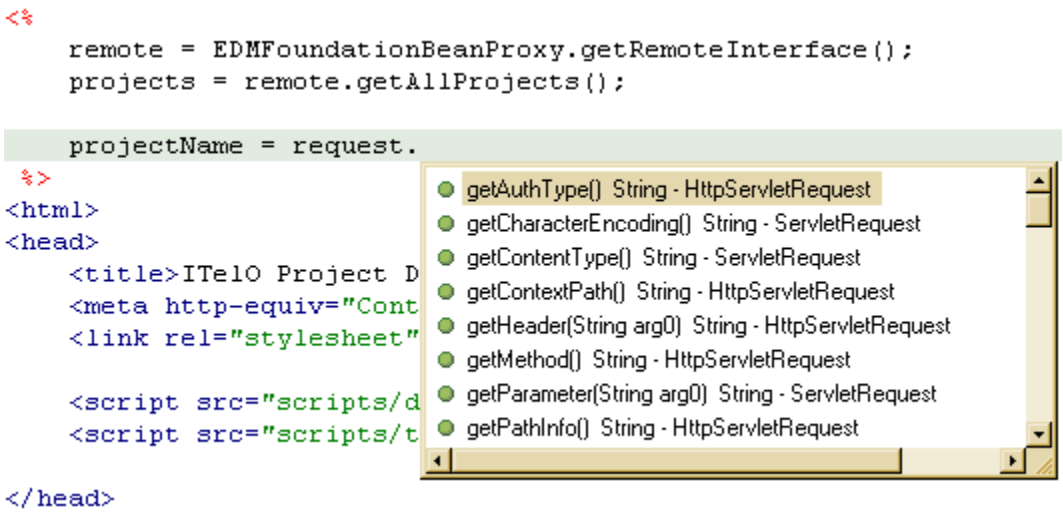
Place a left angle bracket (“<”) somewhere in the code. The JSP editor displays as code assistance a dropdown menu with the possible JSP and HTML tags. If you continue typing characters, the editor will show only the entries that match the characters. Choose the *<jsp:useBean></jsp:useBean>* option. The

```
<jsp:useBean id="" type="" ></jsp:useBean>
```

Will appear in the source code, and the dropdown menu will show the rest of the tag attributes, so you can add them.



You can also use code assistance with the bean object, and the *request* and *response* attributes of the *doPost(HttpServletRequest request, HttpServletResponse response)* method of the HTTP servlet corresponding to the compiled JSP. For example:



Creating a Servlet

OK, now, we all know that the development of servlets is stereotyped: you implement a base interface (*GenericServlet*), or extend a class (*HttpServlet*) and override the methods you will need. Naturally, the J2EE Toolset allows you to leave this task to the corresponding wizard. See the example with the ITelO Project Database for illustration.

1. In the *J2EE Explorer* view, select the *ITelOWeb* project node of the project.
2. From the context menu, select *New → Servlet*.

   The **New Servlet** wizard appears.
3. Enter the following information for the servlet:

Field/Section	**Value**
Web Project	`EDMTestWeb`
Servlet Name	`EnrollmentResultServlet`
Servlet Type	`HTTP Servlet`
Servlet Package	`com.sap.edm.test`
Servlet Methods	`init()` `doPost()`

4. Select **Finish**.

   The servlet appears in the `ITelOWeb` project structure (*J2EE Explorer* view).
5. In the *J2EE Explorer*, unfold the *source* folder and browse to *com/sap/edm/test/EnrollmentResultServlet.java*, which is the main servlet class.
6. Double-click the *EnrollmentResultServlet.java* to open it in the multipage editor.

   You can see that the servlet body along with the skeleton of the *init* and *doPost* methods is generated.

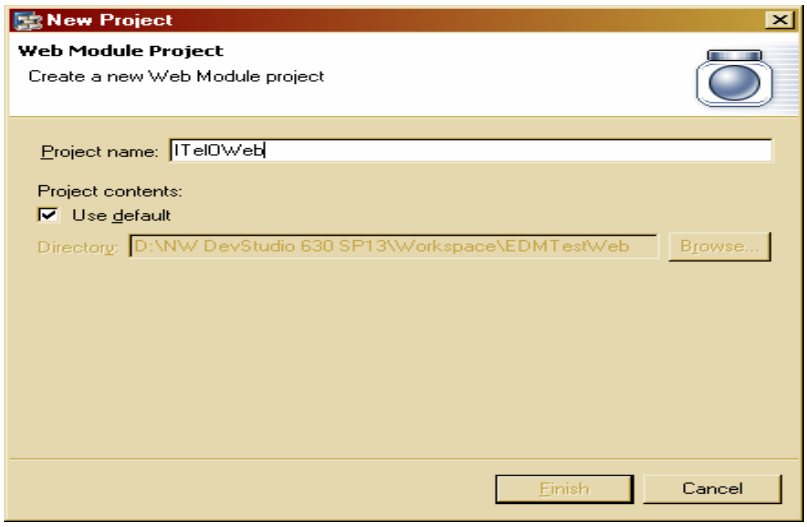


Figure 3: The generated skeleton servlet code

7. In the *EnrollmentResultServlet.java* source code, get an instance of the *EDMFoundationBeanProxy* class in the *init()* method:

```
private EDMFoundation remote;


public void init(ServletConfig cfg) throws ServletException {
      super.init(cfg);


      try {
            remote =
EDMFoundationBeanProxy.getRemoteInterface();


      } catch (Exception e) {


      }
}
```

8. Fill the *doPost* method with the appropriate actions for generating the result from assigning employees to the course:
9. Use *Source → Organize Imports* from the context menu to add all required Java classes to the import clauses.
10. Save the changes.

Summing up the Whole Application: The Enterprise Project

The role of the Enterprise project is to gather together all components of an application. In terms of the J2EE Toolset, the components are other J2EE projects (Web or EJB). You can easily add or remove components from the Enterprise project.

The Enterprise project is also the access point for building the application EAR and deploying it on the target J2EE Engine from the SAP NetWeaver Developer Studio.

To create the Enterprise project for the ITeoO application, do the following:

1. Select *File* → *New* → *Project*.
2. Select *J2EE* → Enterprise *Application Project* and then *Next*.
3. In *Project name*, enter `EDMTestEAR`.

   Keep the *Use default* option checked.
4. Click *Next*.
5. In the *Referenced Project* pane, select the *EDMTestWeb* project.

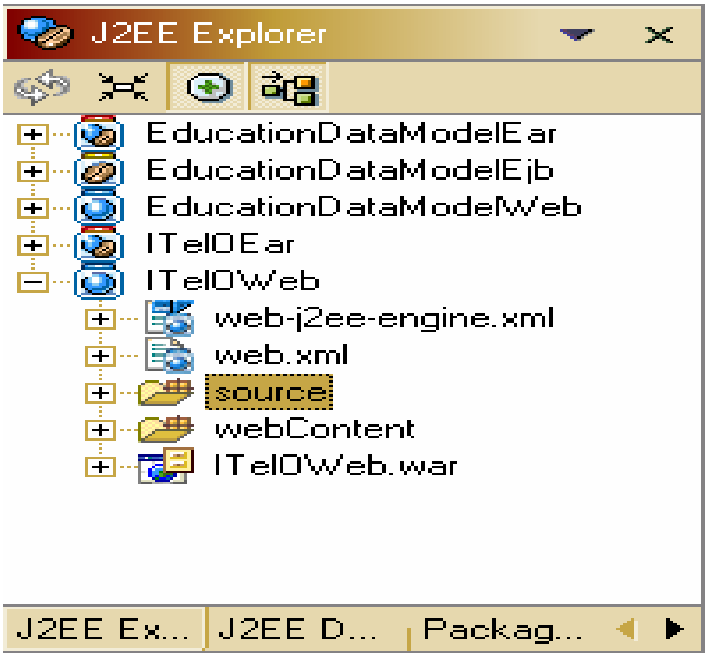


Figure 4: Referencing projects in the Enterprise project

6. Click *Finish*.

The Enterprise project now contains reference to the *EDMTestWeb* project, and it will be included in the application EAR.

The Multipage Editors: Your Tool for Easy Handling of Deployment Descriptors

The manipulation of the deployment descriptors is one of the most cumbersome tasks for the J2EE application developer. We know that studying complex DTDs and writing XML files with multiple tags, attributes and values can be boring and annoying, and at the same time difficult, requiring a lot of concentration. The J2EE Toolset, however, acts here as your faithful assistant, ready to build up all necessary XML code, as long as you describe exactly what you want. If you are not an SAP employee, you may find it especially useful for the generation of SAP-specific descriptors.

You model the deployment descriptors visually, using the multipage editors. Each deployment descriptor has a different multipage editor that allows you to define what the descriptor will do. However, in many cases it is not even necessary to tell him explicitly what to do – he makes intelligent guesses! For example, when you add a component using the J2EE Toolset, the toolset automatically adds it to all related deployment descriptors – no need to remind him to do so. This applies not only for Web components (JSP, HTML or servlets) but also for EJB components (beans, classes, filters, and so on). As shown through the descriptors of the ITelO Project Database application, you can easily tune the Web application using the multipage editor of its deployment descriptors.

For illustration, let's model some of the deployment descriptors of our ITelO application.

Editing the web.xml

Initially, the *web.xml* is filled with base according to the components you have added. However, editing the *web.xml* further gives you broader opportunities for tuning the Web logic of your application. Here, we'll demonstrate adding some useful features via the deployment descriptor.

Adding EJB References

1. Enter the *EJBs* tab.
2. Select the *EJB References* node, and choose *Add*.

   The *Choose EJBs* dialog appears. It shows the EJB components available in the EJB projects.
3. Unfold the *EducationDataModelEjb* node, and select *EDMFoundationBean*.
4. Click *OK*.

   The link to the *EDMFoundationBean* appears in the *EJBs* tab.

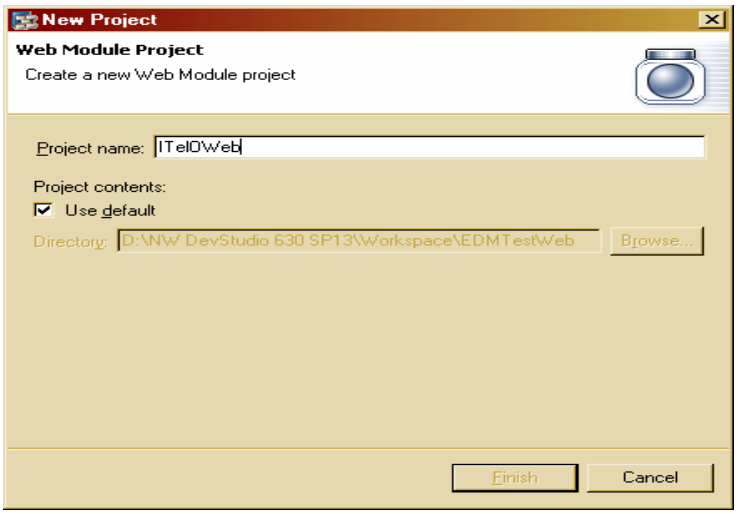


Figure 7: EJBs tab of the web.xml multipage editor

The SAP NetWeaver Developer Studio adds the following source code to the

web.xml:

```
<ejb-ref>
  <ejb-ref-name>ejb/EDMFoundationBean</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.sap.edm.ejb.EDMFoundationHome</home>
  <remote>com.sap.edm.ejb.EDMFoundation</remote>
</ejb-ref>
```

(Optional) Adding a Welcome Page

Using welcome pages you can shorten the URL for access to your Web applications. For example, if you have to access the application index page via http://<host>:<port>/<contextroot>/<page>, by using a welcome page you can access the application via http://<host>:<port>/<contextroot>/.

1. Enter the *Others* tab.
2. Select the *Welcome Pages* node, and click *Add*.
3. Click the ellipsis button next to the *Welcome File* field.

   The *Choose resource* dialog appears. It shows all Web pages available in the project.
4. Select *index.jsp* as a welcome page and click *OK*.

   The *index.jsp* appears as a subnode of the *Welcome Pages* node.

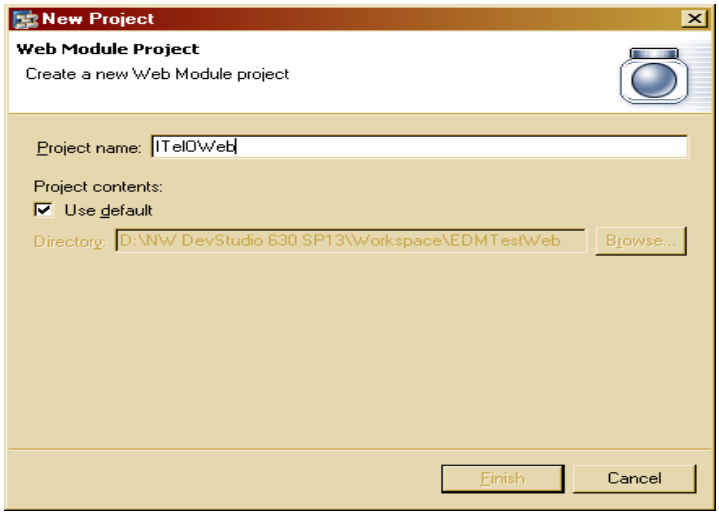


Figure 8: Others tab of the web.xml multipage editor

The SAP NetWeaver Developer Studio adds the following content to the *web.xml*:

```
<welcome-file-list>

  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

Editing the web-j2ee-engine.xml

Adding EJB References

1. Enter the *References* tab.
2. Select the ejb-ref node and click *Add*.
3. In the dialog that is shown, select the EDM Foundation Bean and click *OK*.

Figure 9: References tab of the web-j2ee-engine.xml multipage editor

Editing the application.xml

Setting the Context Root

1. Enter the *Modules* tab.

   The modules included in the application are displayed here. In this case, it contains only the EDMTestWeb.war module.

2. Select the *EDMTestWeb.war* node.
3. In the *Context Root* field, enter `ITelO`.

4. Save the changes.

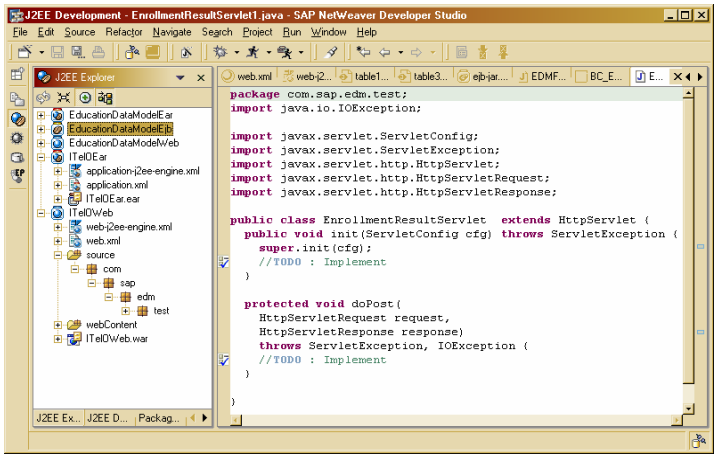


Figure 10: Modules tab of the application.xml multipage editor

The First Moment of Truth: Building and Deploying

This is really the moment of truth, as it shows whether the development was successful or not. You can deploy the created J2EE application on the J2EE Engine directly from the SAP NetWeaver Developer Studio. You only have to define the correct host and port for the J2EE Engine (from *Window → Preferences → SAP J2EE Engine*, as shown in Figure). If the SAP NetWeaver Developer Studio is installed along with the J2EE Engine on the local host, it automatically detects the local J2EE Engine and defines the correct settings for it in the *Preferences* dialog.

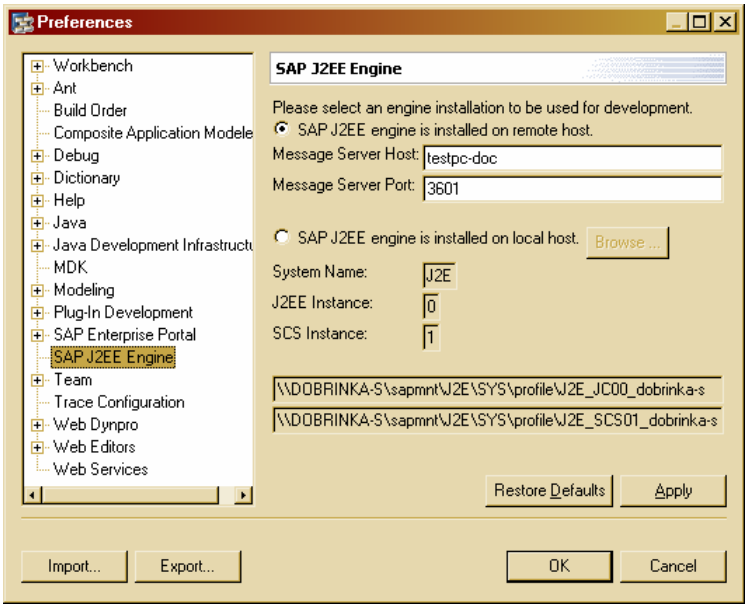


Figure 11: Setting the host and port of the J2EE Engine

You can check if the J2EE Engine settings you supplied are correct from the *J2EE Engine* view of the *J2EE Development* perspective. If the settings are correct and the target J2EE Engine is running, you can see the status of its processes in the *J2EE Engine* view. You can also perform basic management operations over the J2EE Engine from that view.

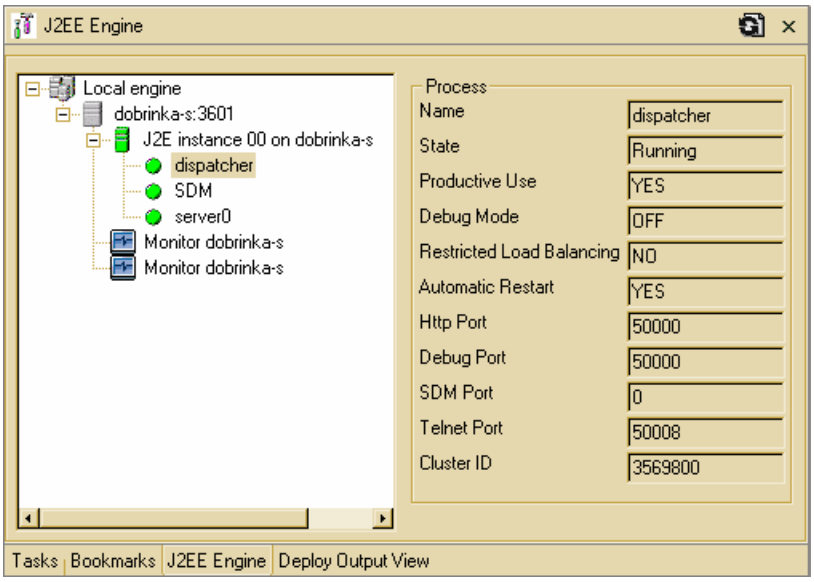


Figure 125: J2EE Engine view

The result from the deployment operation is shown in the *Deploy Output* view.

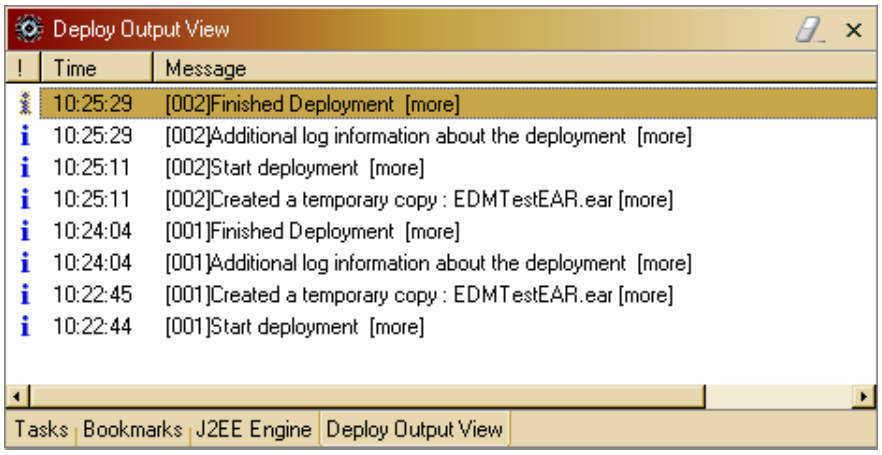


Figure 13: Deploy Output view

Here is the procedure for building and deploying the ITelO application:

1. Select the *Window → Preferences → SAP J2EE Engine* menu path and check if the settings there are correct for the J2EE Engine you want to connect.
2. In the *J2EE Explorer*, select the root node of the *EDMTestEAR* project and select *Build Application Archive* from the context menu.

   The application archive appears in the *EDMTestEAR* project node.
3. Select the EDMTestEAR archive (*EDMTestEAR.ear*) and choose *Deploy to J2EE engine*.
4. If you are connecting this J2EE Engine for the first time (for the current session of the SAP NetWeaver Developer Studio), you will be prompted to submit SDM password. Enter the password and click *OK*.

   If the password is correct, the SAP NetWeaver Developer Studio will initiate the deployment and you can see the progress in the *Deploy Output* view.
5. You can view the resulting application in your Web browser.

   There are two variants for accessing the application:

   - If you defined a welcome page, you can access the ITelO Project Database application with the following URL in your Web browser:

     http://<j2ee_engine_host>:<j2ee_engine_http_port>/ITelO/

     For example:

     http://localhost:50000/ITelO/
   - If you did not define a welcome page, you can access the application with the following URL:

     http://<j2ee_engine_host>:<j2ee_engine_http_port>/ITelO/index.jsp

     For example:

     http://localhost:50000/ITelO/index.jsp

When you rebuild an Enterprise project in the SAP NetWeaver Developer Studio, all other (EJB, Web and so on) projects are automatically rebuilt too. Hence, when you make changes to components in those projects, you don't have to rebuild each project manually but you can rebuild the Enterprise project instead.

Coming Next

OK, now, what do you think of employing the J2EE Toolset as your full-time indispensable personal assistant? If you haven't made up your mind yet, wait for our next article, demonstrating how the toolset

combines brains and muscles to support you in creating EJB components.

Author Bio

Dobrinka Stefanova is an information developer at SAP. She is responsible for the user documentation of the J2EE Toolset, integrated in the SAP NetWeaver Developer Studio.