Integrating JTOpen into WebSphere Studio Application Developer

Gregory S. Hurlebaus
PartnerWorld for Developers
IBM iSeries
Rochester, MN

Introduction

The intent of this document is to provide a "getting started" example using WebSphere® Studio Application Developer, WebSphere Application Server Advanced Edition 4.0, and the open source utility classes of JTOpen. This example describes the steps how to create and deploy a simple enterprise application using WebSphere Studio Application Developer and WebSphere Application Server 4.0 on the iSeries™ platform. The application is very simple and consists of a single JavaServer Page™ (JSP). The JSP will contain JavaBeans to specific classes within JTOpen to connect to an iSeries server and retrieve the current CPU percentage.

WebSphere Studio Application Developer

WebSphere Studio Application Developer is a new strategic tool set from IBM® aimed at developing and deploying web applications that are J2EE compliant. This example used the generally available Version 4.0.2. More information about WebSphere Studio Application Developer can be found at:
ibm.com/software/ad/studioappdev/

Let's get started. First, download and install the WebSphere Studio Application Developer. You can download a trial version and/or purchase WebSphere Studio Application Developer from the following Web site:
ibm.com/software/ad/studioappdev .
Then, read through the WebSphere Studio Application Developer on-line documentation to become familiar with the terminology and features of this new tool.

Overview of the steps that we will be performing

Here are the steps that we will be performing throughout this example:

1. Download JTOpen from the Internet.
2. Start WebSphere Studio Application Developer and create a new enterprise application.
3. Add JTOpen to the enterprise application.
4. Develop the enterprise application.
5. Run the enterprise application within WebSphere Studio Application Developer.
6. Deploy the enterprise application to the iSeries server and run the host enterprise application on the iSeries server.

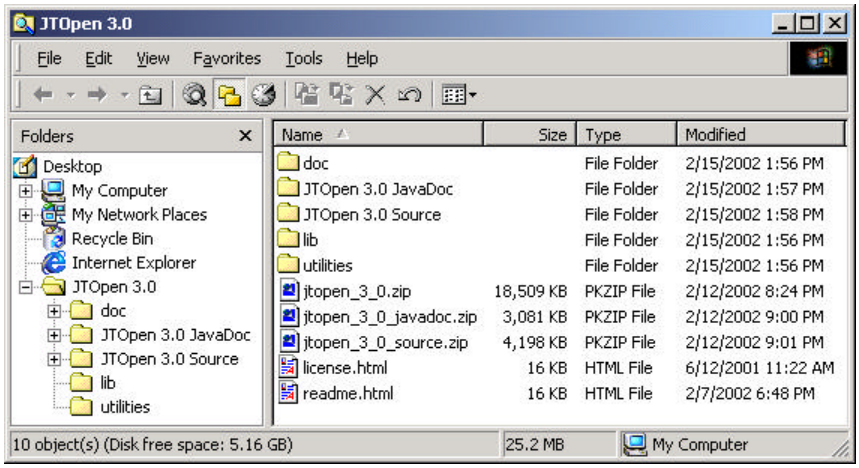
Download JTOpen from the Internet

The Open Source version of the IBM Toolbox for Java is called JTOpen. It can be downloaded from the Internet at:
http://oss.software.ibm.com/developerworks/opensource/jt400/

Select the **Toolbox downloads** link.
For an evaluation copy of the IBM Toolbox for Java, download JTOpen. This may require for you to register if you have not already done so. Then enter your password and validate the information. Press **Accept License**.

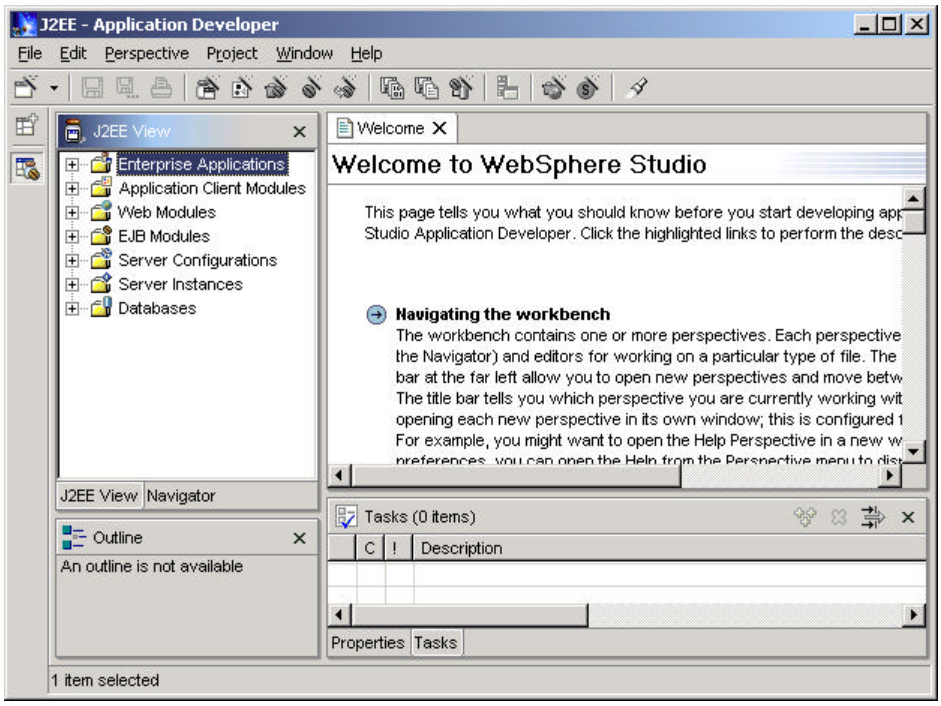
Then, select the **Single File Download** for JTOpen Version 3.0 to get the jtopen_3_0.zip (18.9MB) file. Save this file into a temporary folder called **JTOpen 3.0** on your desktop. Unzip the contents into this same folder. You will also need to download the latest javadoc jtopen_3_0_javadoc.zip (3.2MB) file if you would like to view the JavaDoc for the JTOpen classes. These are not included in the Single File Download package. If interested, you can also download the source java files. A couple of extra folders, **JTOpen 3.0 JavaDoc** and **JTOpen 3.0 Source**, have been created to hold these files.

You should see the following folder structure with a doc, lib, and utilities folders within the JTOpen 3.0 folder on your desktop (see below). The lib folder contains the **jt400.jar** file that we will be using later in this example.



Start WebSphere Studio Application Developer and create a new enterprise application

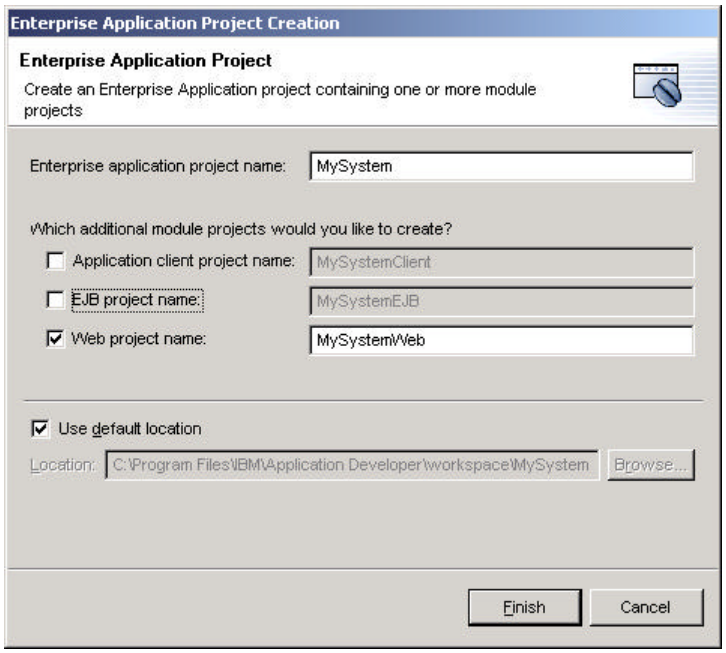
In this example, we will be using WebSphere Studio Application Developer version 4.0.2. When you initially start WebSphere Studio Application Developer, it should resemble the figure below.



Create a new enterprise application

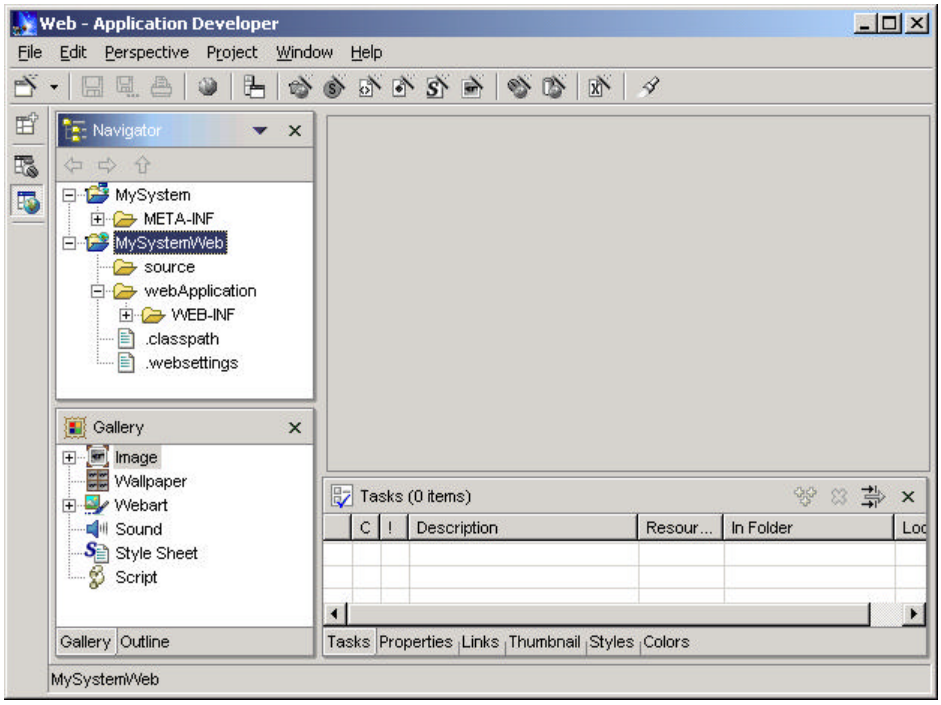
From the main menu, select **File** —>**New** —> **Enterprise Application Project**.

Fill in **MySystem** for the enterprise application project name. Since this example will not involve client side code or Enterprise JavaBeans™ (EJB), uncheck Application client project name and EJB project name as the figure below shows.



Press **Finish**.

For web-type development, switch to the Web Perspective by selecting **Perspective** —> **Open** —> **Web**. A new enterprise application called **MySystem** and a Web Project called **MySystemWeb** were both created and are displayed in the Navigator window (shown below).



If you do not see the option for the Web perspective, select **Perspective** —> **Open** —> **Other ...** Then, select Web and press OK.

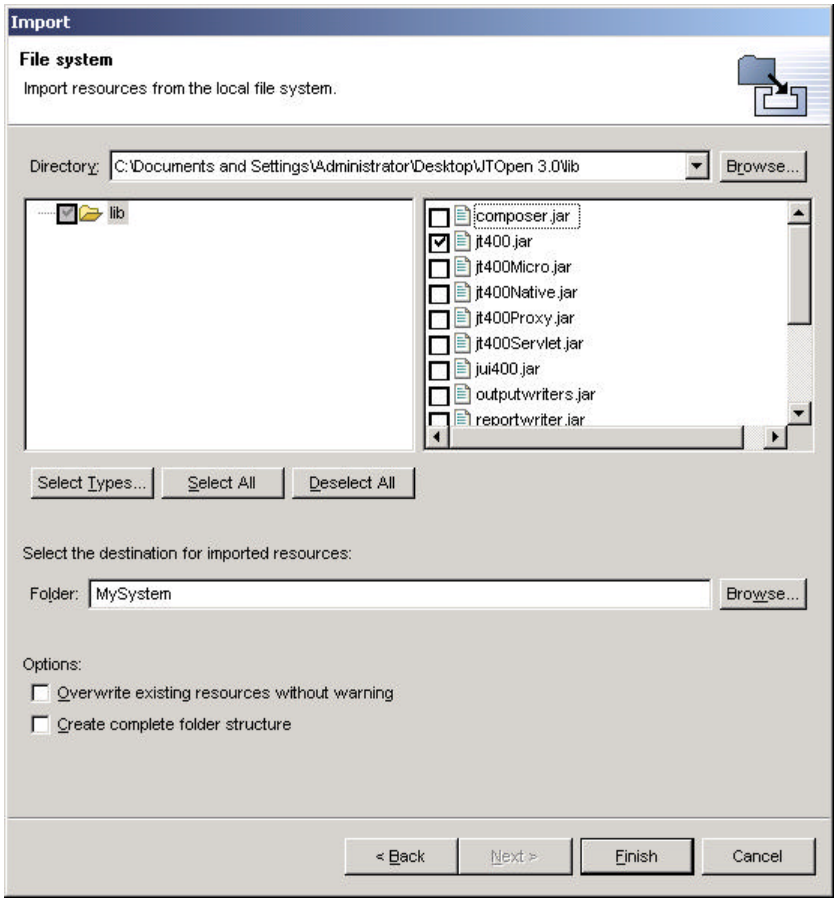
Add JTOpen to the enterprise application

Because we want to associate the JTOpen set of utility classes as part of our enterprise application, we will add it to the **MySystem** folder and have the Web application **MySystemWeb** reference it.

Scroll down the Navigator window and select **MySystem**. Now select **File** —> **Import...** to display the Import dialog. Since we need to import a jar file, select **File system** and press **Next >**. Press the **Browse...** button and locate the directory which contains the jt400.jar file. In this example, it is in your Desktop\JTOpen 3.0\lib folder.

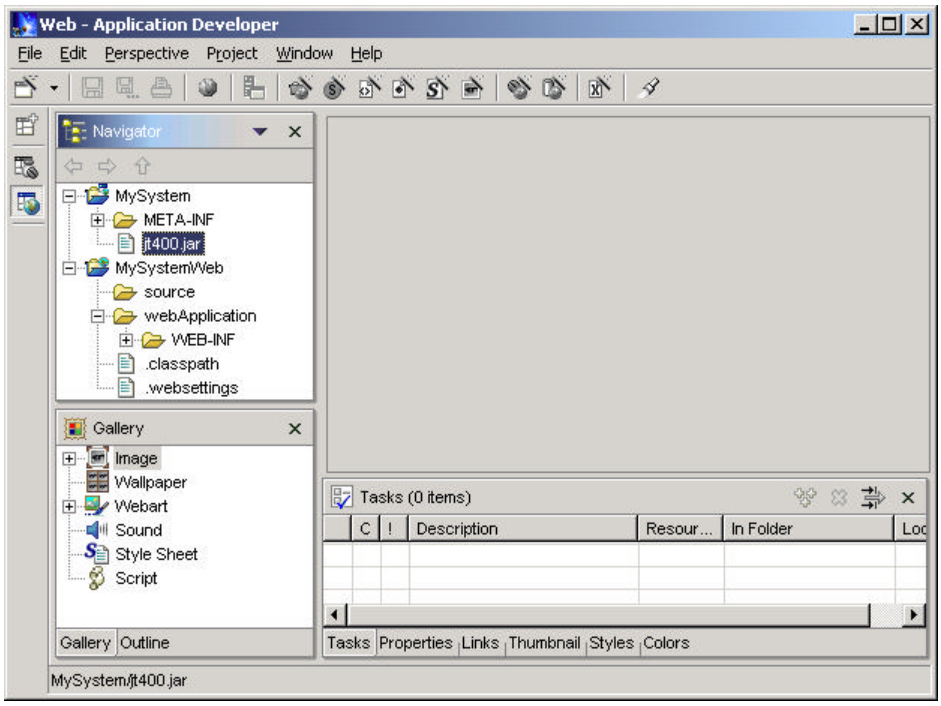
Expand the **lib** folder by clicking just to the left of the checkbox. Then select the checkbox next to jt400.jar — this will place a checkmark in the box.

Verify that your figure matches the figure below.



Press **Finish** to import the jt400.jar file to the enterprise application.

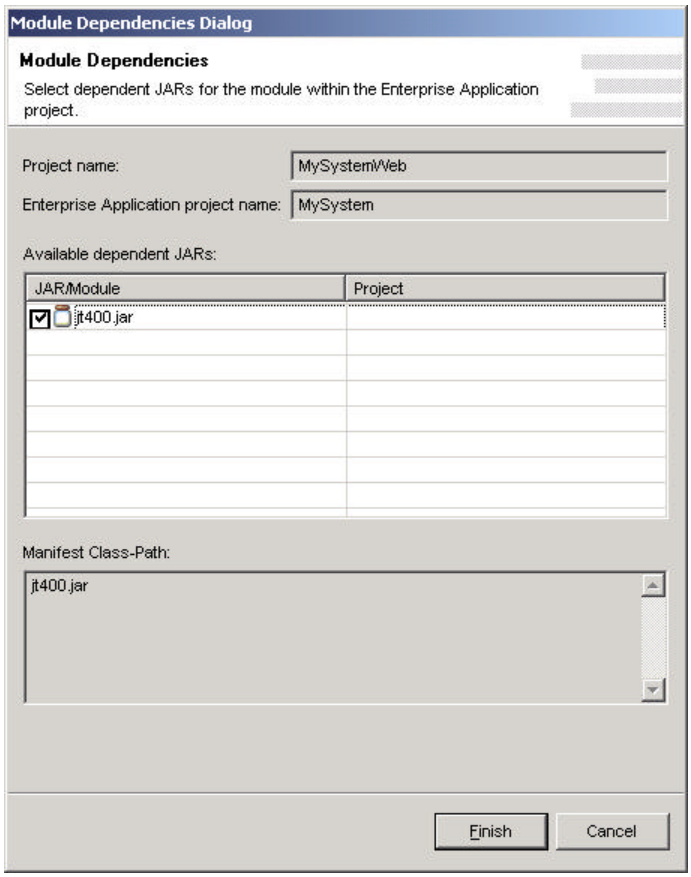
You can see the jt400.jar file in your project when you expand the **MySystem** folder.



Now we need to have the web application be able to use this new jar file.

Select **MySystemWeb**, then right click and select **Edit Module Dependencies...**

Select the box next to jt400.jar to place a checkmark in the box (see below).



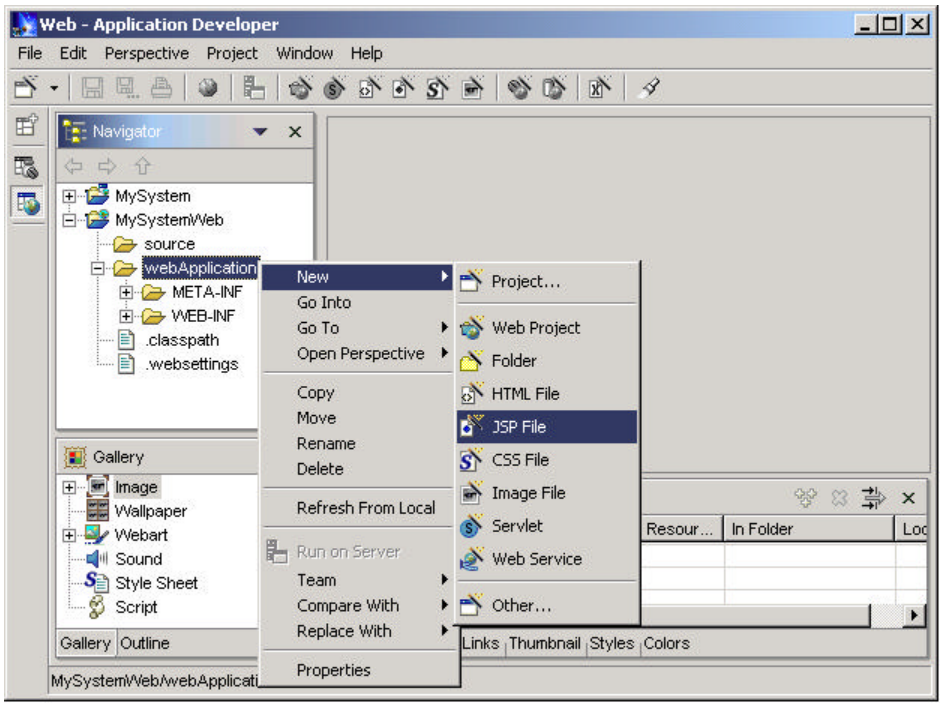
Press **Finish**.

Now you can start using in the web application any of the classes that are contained within the jt400.jar file.

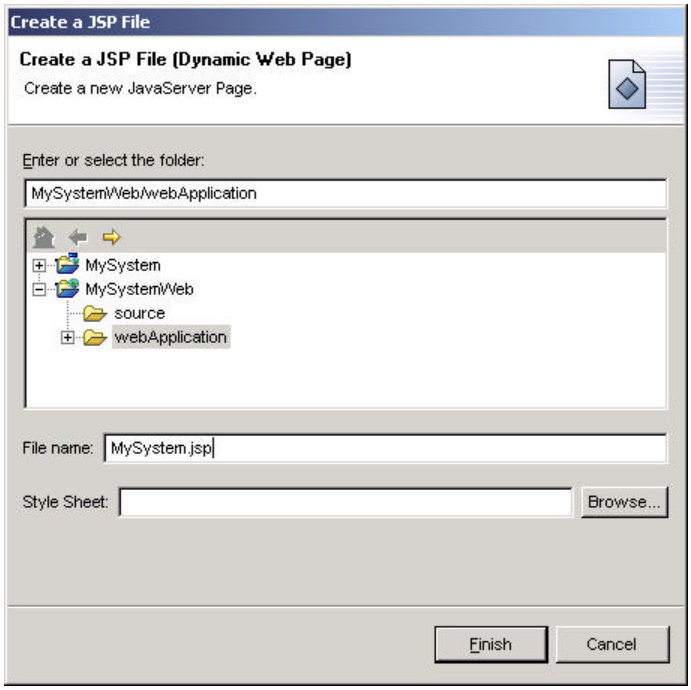
Develop the enterprise application

Create MySystem.jsp file

To create the MySystem.jsp file used in this example, right click on the **WebApplication** folder of the **MySystemWeb** web project in the Navigator window. Select **New** —> **JSP File**.



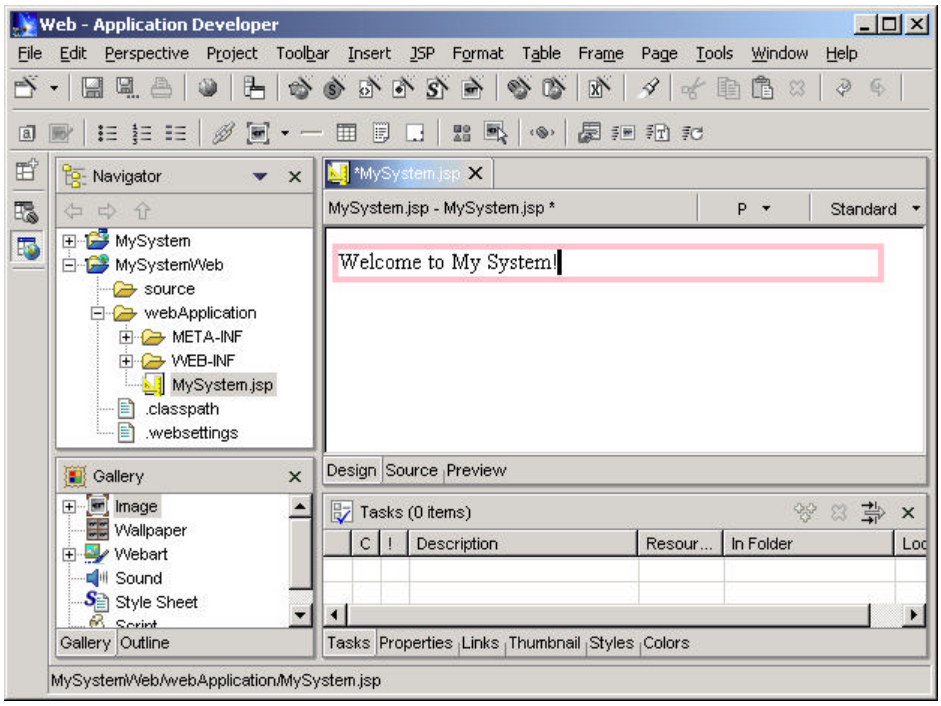
Enter **MySystem.jsp** in the file name text box.



Press **Finish**.

The steps that we just discussed will create a JavaServer Page file with some text in it that says "Place MySystem.jsp's content here".

Highlight the text "Place MySystem.jsp's content here" with the cursor and press the delete key. Next, type **Welcome to My System!** on the blank page.

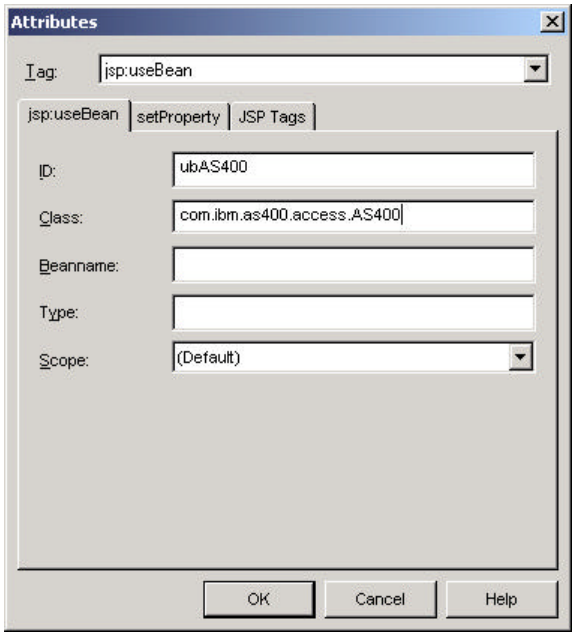


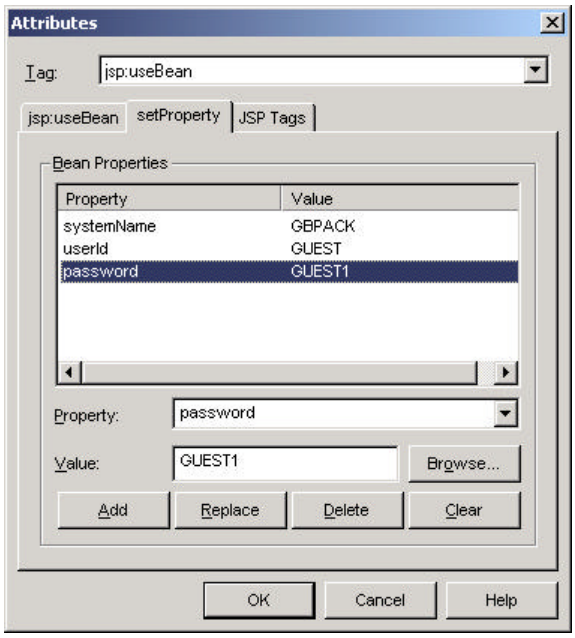
Select **File** —> **Save MySystem.jsp** to save the work you have done so far.

Retrieve System CPU

Now ,we will add a bean for the **AS400** class, a bean for the **SystemStatus** class, and call a method on SystemStatus called **getPercentProcessingUnitUsed**.

To add a bean for the **AS400** class, place the cursor on the line following your Welcome statement. Next, select **JSP** —> **Insert Bean...**. The ID can be anything you would like. In this example, we used **ubAS400** to represent user bean AS400. Enter **com.ibm.as400.access.AS400** for the Class field. Then, select the **setPropery** page tab. Using the Property dropdown arrow, select **systemName**, enter your IBM iSeries host name, and press **Add**. Again, using the Property dropdown arrow select **userId**, enter a valid userid of your iSeries server, and press **Add**. For the final property, using the Property dropdown arrow, select **password** and enter the valid password for the user profile. Then, press **Add**.

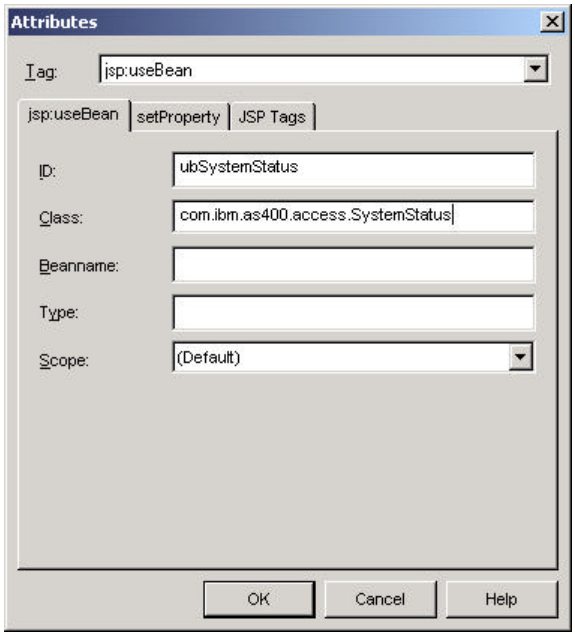


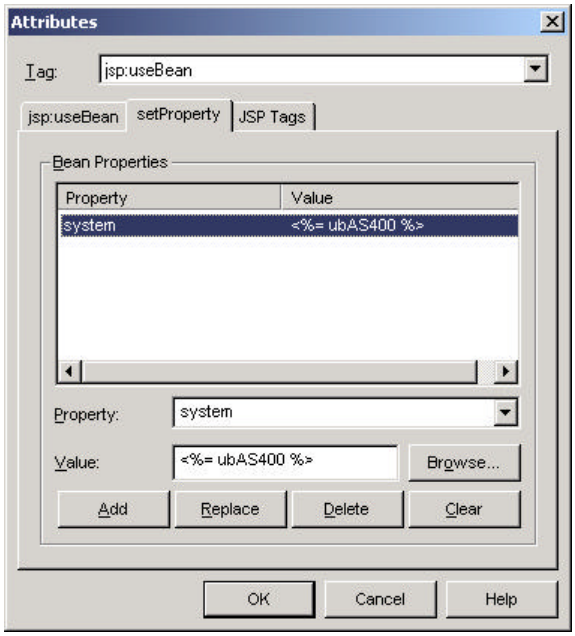


When finished, press **OK**.

Next, we need to add the SystemStatus bean.

Select **JSP** —> **Insert Bean...**. Use **ubSystemStatus** for the ID and **com.ibm.as400.access.SystemStatus** for the class. Next, select the **setProperty** page tab. Using the Property dropdown arrow, select **system**. Press the **Browse...** button and select **ubAS400**. This tells the SystemStatus bean to use the ubAS400 bean that we previously added for its system property. Press **OK**. You will see that the browser dialog was generated and added <%= ubAS400 %> for the Value field. Press **Add** to set the property.



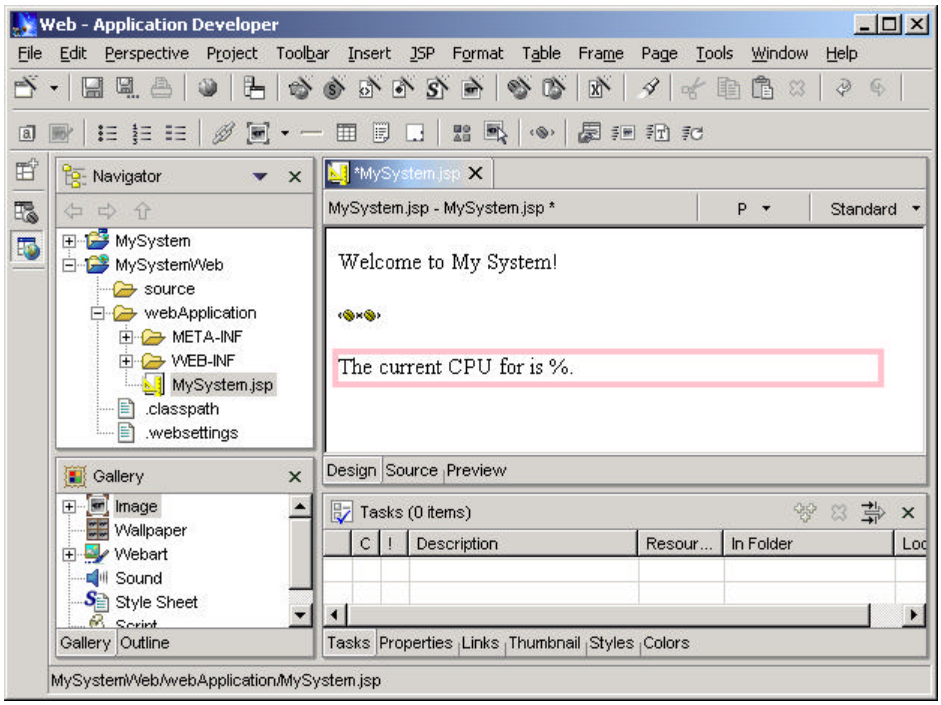


When finished, press **OK**.

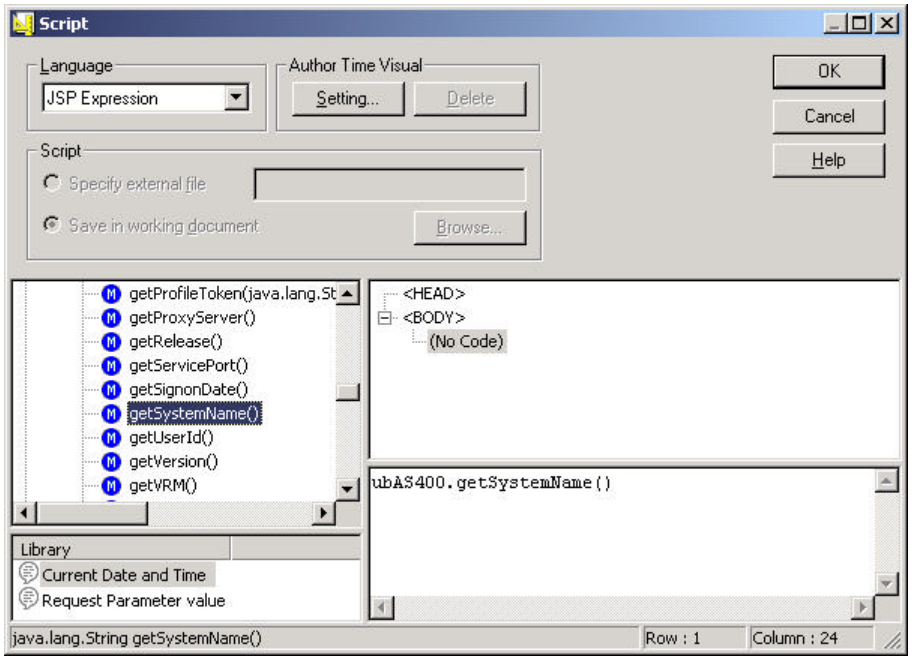
Add a call to the getPercentProcessingUnitUsed method

Let's edit the JSP and add a couple method calls from the two beans we have included.

Add the following text "**The current CPU for is %**" to your JSP on the next line.

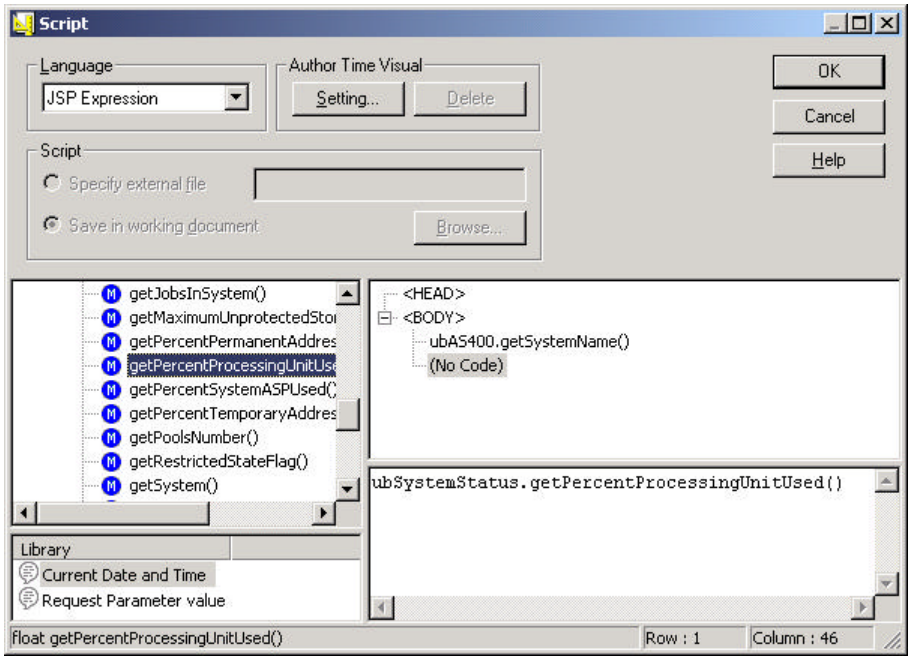


Place the cursor after the word "for" and select **JSP** —> **Insert Expression**. Select the plus(+) next to **ubAS400** to display the Properties and Method folders. Select the plus (+) next to **Method** to see all the public methods. Double click the **getSystemName()** method to add it to the script.



Press **OK** when finished.

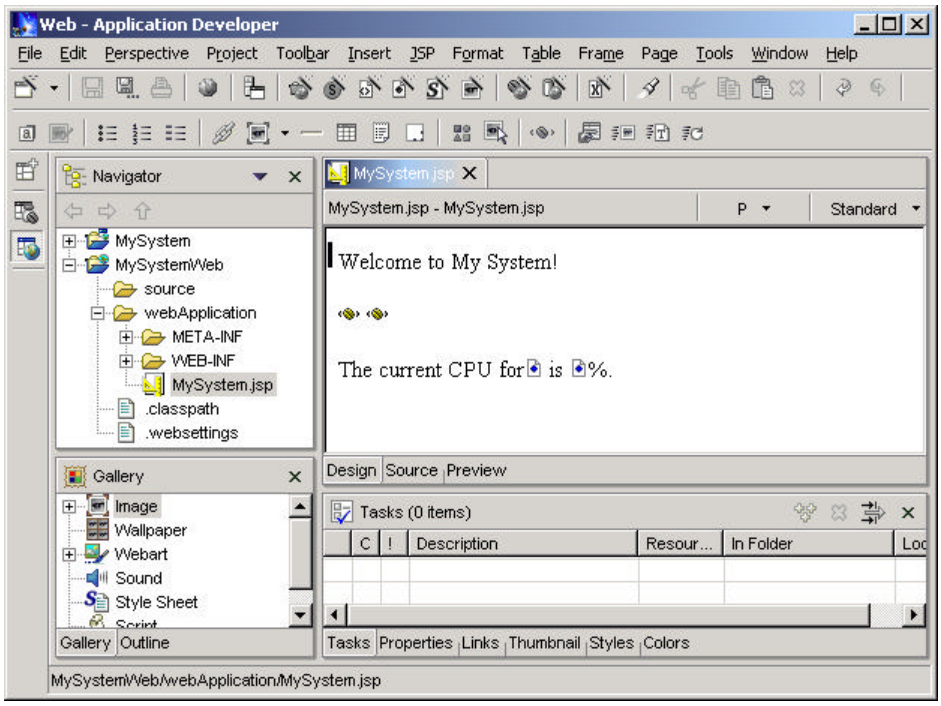
Place the cursor in front of the "%" and select **JSP** —> **Insert Expression**. Select the plus (+) next to **ubSystemStatus** to display the Properties and Method folders. Select the plus (+) next to **Method** to see all the public methods. Double click the **getPercentProcessingUnitUsed()** method to add it to the script.



Press **OK** when finished.

Again, let's save the project with **File** —> **Save All**.

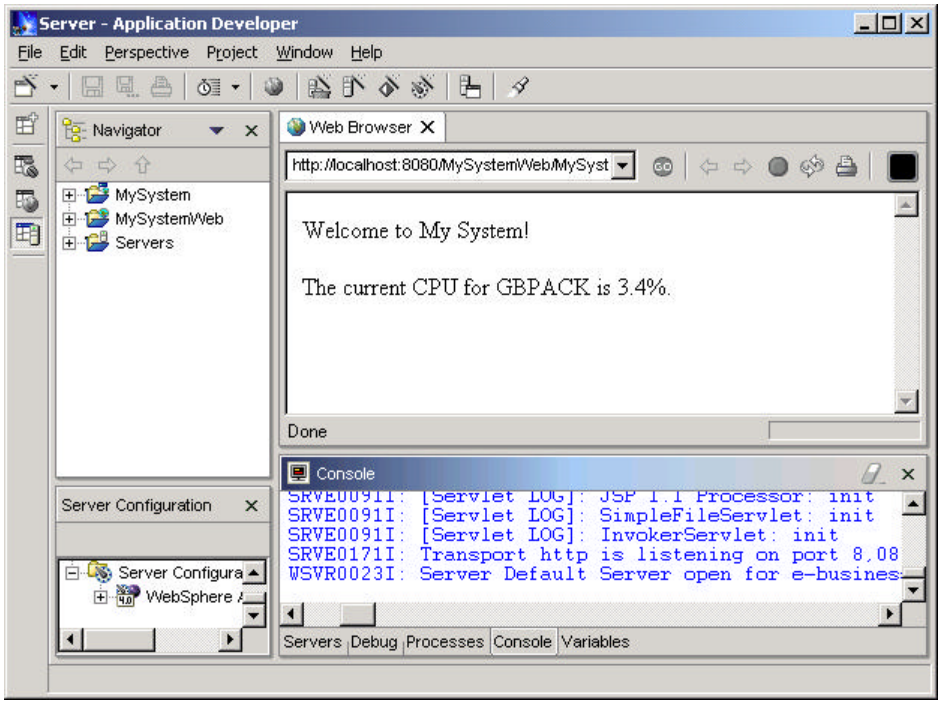
Your **MySystem.jsp** file should look like the following graphic.



Run the Enterprise Application within WebSphere Studio Application Developer

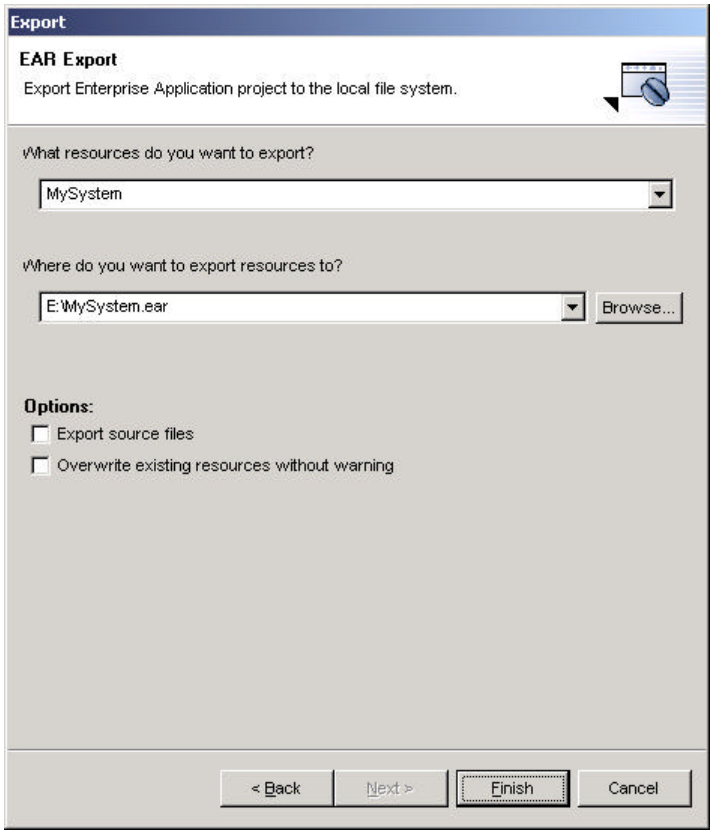
With the **MySystem.jsp** file selected, right click and select **Run on Server**. You will see a couple of message boxes appear showing that the IBM WebSphere Application Server, Release 4.0.2 Advanced Single Server Edition for Multiplatforms is starting.

The console will display status messages from WebSphere. After the WebSphere Application Server has started, it will then compile and execute the MySystem JavaServer Page and display it within the browser of WebSphere Studio Application Developer.



Deploy the enterprise application to the iSeries server and run the enterprise application on the iSeries server

To deploy this enterprise application, you need to save everything into an Enterprise Application Resource (EAR) file. Select **File** —> **Export...** to display the Export dialog. In the Export dialog, select **EAR File** and press **Next**. Using the dropdown arrow, select **MySystem** for the *What resources do you want to export?* question. For the *Where do you want to export resources to?* question, select the root directory of the file share for your iSeries server. In this example, **E:\MySystem.ear** was used.

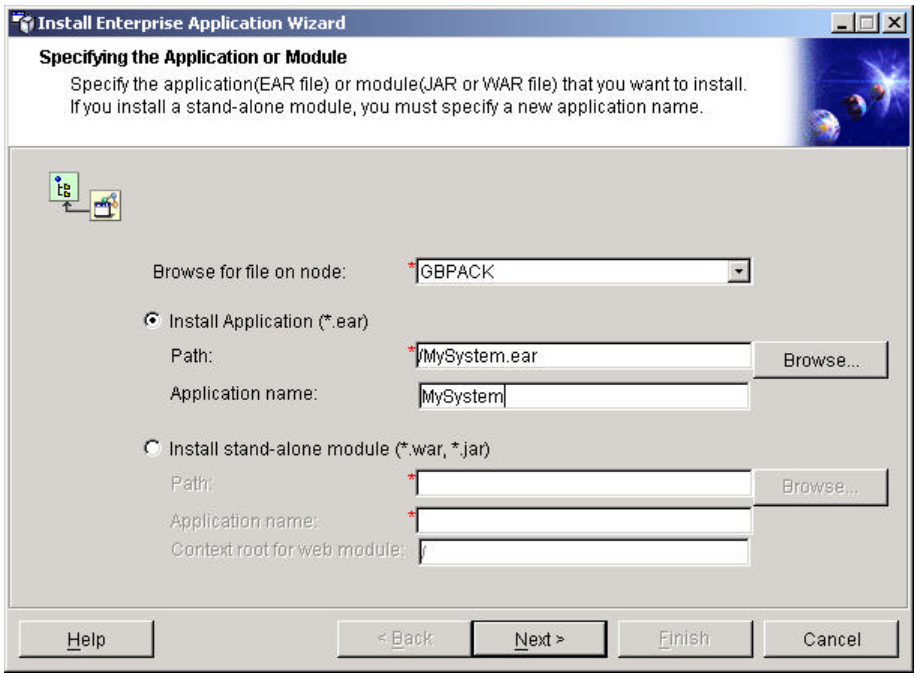


Press **Finish**.

Adding the enterprise application to WebSphere Application Server on your iSeries server

In the WebSphere Advanced Administrative Console, right click on **Enterprise Application** and select **Install Enterprise Application**.

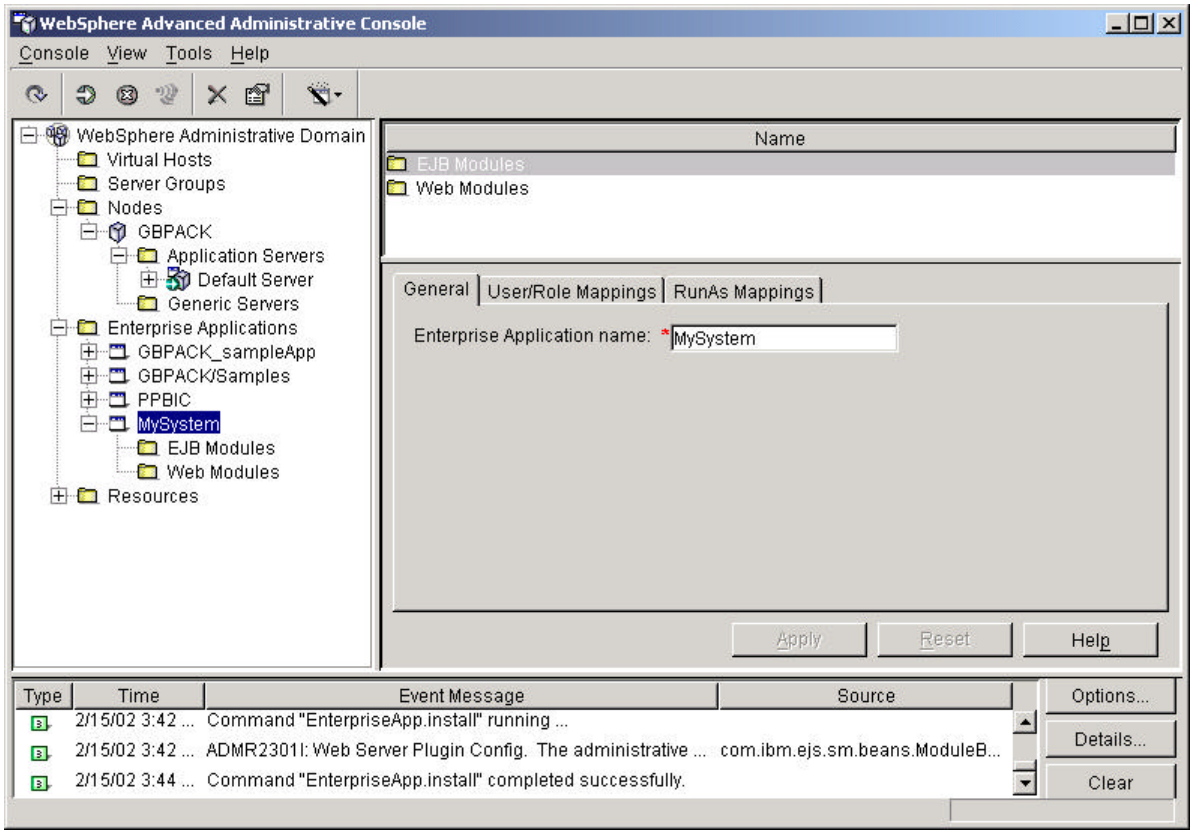
Use the **Browse...** button to select the **MySystem.ear** file. Enter **MySystem** for the Application name.



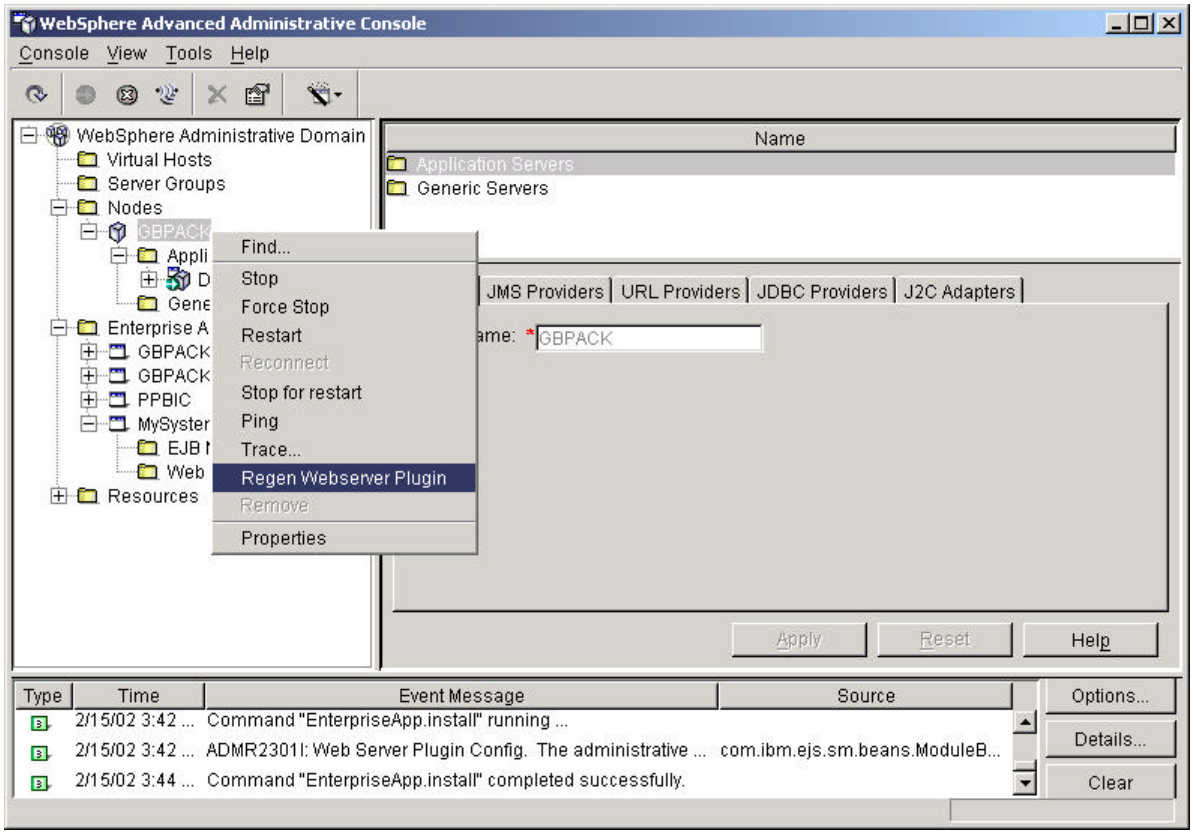
Press **Next** to continue. Keep pressing **Next** until the Finish button is enabled. Press **Finish**.

When the application has been installed, press **OK** on the Information dialog.

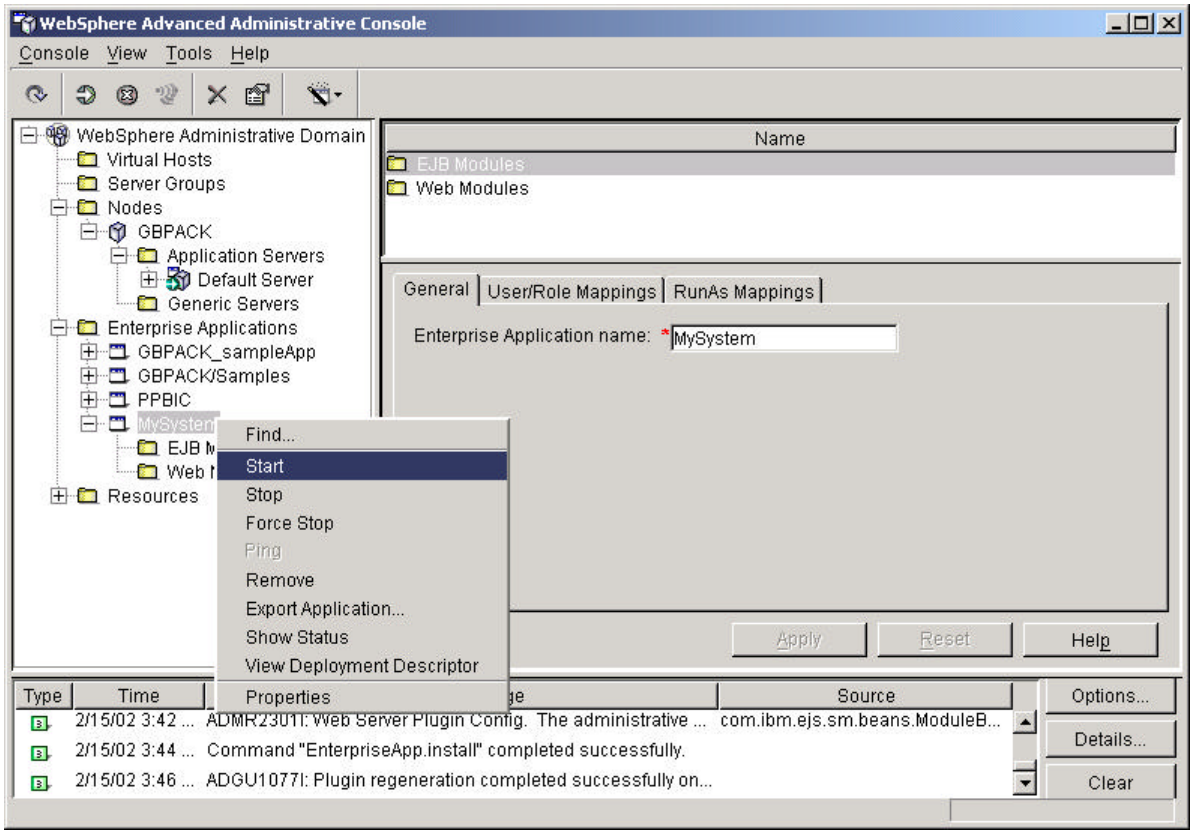
The **MySystem** enterprise application now exists under enterprise applications.



Right click on the node name and then select **Regen Webserver Plugin**.



Right click on the **MySystem** enterprise application and select **Start**.

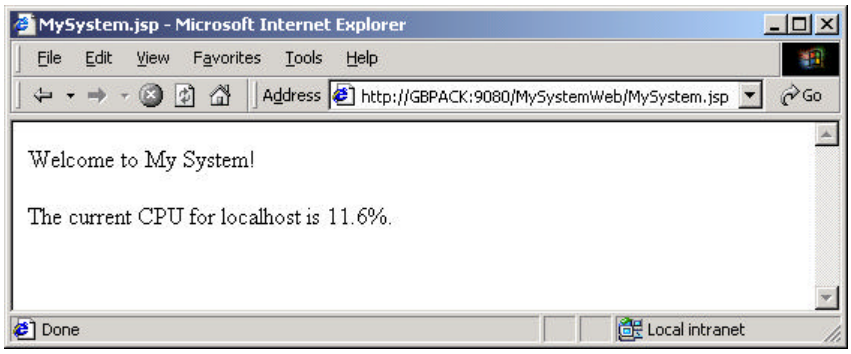


Select **OK** when the Information dialog indicates that the enterprise application has started.

Run the enterprise application

WebSphere Application Server 4.0 has a built-in HTTP Server that listens on port 9080 by default. This port can be used for testing the application or a separate HTTP server instance can be created to forward requests to WebSphere. For this example, the internal HTTP Server is used.

The following screen capture shows the **MySystem.jsp** page using the WebSphere Application Server 4.0 HTTP Server. With this example, the URL is **http://gbpack:9080/MySystemWeb/MySystem.jsp**.



Note: Since we are using JTOpen to a local iSeries server, you may see **localhost** for the system name.

Conclusion

Hopefully after reviewing this example, you have a greater understanding of how to create and deploy an application using WebSphere Studio Application Developer and WebSphere Application Server 4.0. While this example is very simple, it provided some basic principles that will apply to more complicated examples. You are encouraged to enhance this example to gain the necessary skills to create and deploy more complex applications using WebSphere Studio Application Developer and WebSphere Application Server 4.0.

For additional information about using utility JARs in WebSphere Studio Application Developer, review the technical article *Developing J2EE utility JARs in WebSphere Studio Application Developer* at: www7b.boulder.ibm.com/wsdd/library/techarticles/0112_deboer/deboer2.html?n1102 .

Trademarks

IBM, WebSphere, and iSeries are registered trademarks or trademarks of the IBM Corporation in the United States or other countries or both.

Java, JavaServer Pages, Enterprise JavaBeans, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

This publication may have referred to products that are not available in your country.

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