

## IBM® Feature Pack for Web Services– LAB EXERCISE

## Feature Pack for Web Services

### Bottom-up development of a JAX-WS service

What this exercise is about .....	1
Lab requirements .....	1
What you should be able to do .....	2
Introduction .....	2
Development models .....	2
Exercise instructions .....	3
Part 1: Create the dynamic Web project.....	4
Part 2: Import the implementation code .....	11
Part 3: Creating a Web service from the implementation code .....	18
Part 4: Install and deploy the application .....	27
Part 5: Creating the client code.....	42
Summary.....	57

## What this exercise is about

The objective of this lab is to show you how to develop a complete Web Services application using the Feature Pack for Web Services for WebSphere Application Server V6.1.

You will take a bottom up development approach beginning with a single Java file containing code that you will use as the basis for your Web Service implementation. It will then show you how to develop a Web Service, package your code and artifacts into an EAR file, install the EAR file as an application for WebSphere Application Server version 6.1, start the application, perform a simple test to confirm the application is deployed and started correctly, create a Web services client, and then do an end-to-end test using the client.

## Lab requirements

The system on which you are installing WebSphere Application Server must be running a supported distributed operating system. For more information on supported operating systems, see the WebSphere Application Server V6.1 information center. In addition, the system must have the following resources available:

- An installation of IBM Web Services Feature Pack for WebSphere Application Server V6.1
- An installation of IBM Application Server Toolkit V6.1

## What you should be able to do

At the end of this lab you should be able to:

### Use a bottom up approach to develop a Web service

- Take an existing Java file containing the code that you want to serve as the implementation of a Web Service and, using the IBM WebSphere Application Server Toolkit Version 6.1, create a Web Service for the IBM Feature Pack for Web Services for WebSphere Application Server V6.1
- Package the Web Service into an EAR file
- Deploy the EAR file onto WebSphere Application Server V6.1
- Start the Web Service application
- Perform a simple test to confirm that the application is installed and listening at the expected URL
- Develop a client for the Web Service application
- Perform an end-to-end test from the client to the Web Service and back.

---

## Introduction

The Feature Pack for Web Services builds on Web Services technologies available in WebSphere Application Server V6.1. A few of the major enhancements introduced in this Feature Pack are:

- A new JAX-WS programming model ( with support for JAX-B, annotations, SOAP 1.2 and more )
- Support for WS-Reliable Messaging and WS-Secure Conversation
- Improved administration of Web Services configuration through the use of Policy Sets

## Development models

Web services can be created using one of two approaches: top-down development and bottom-up development. Top-down Web services development involves creating a Web service from a WSDL file. Bottom-up Web services development involves creating a Web service from a Java file that contains code that will be the basis for implementing the Web service.

When creating a Web service using a top-down approach, first you design the implementation of the Web service by creating a WSDL file. You can do this using the WSDL Editor. You can then use the Web services wizard to create the Web service and skeleton Java™ classes to which you can add the required code.

Although bottom-up Web service development may be faster and easier, especially if you are new to Web services, the top-down approach is the recommended way of creating a Web service – especially if interoperability is a concern. By creating the WSDL file first you will ultimately have more control over the Web service, and can eliminate interoperability issues that may arise when creating a Web service using the bottom-up method.

### Lab Exercise # 1 – Bottom up development

**Part 1: Create project and import the implementation code**

In this portion of the lab you will create a Dynamic Web Project in AST and import the implementation code for the Web Service that you will create.

**Part 2: Creating a Web service from the implementation code**

In the part you will actually create the Web Service from the implementation code and export the EAR file.

**Part 3: Install and deploy the application**






This section will show how to install the EAR file from Part 2, how to start the Web Service application from the administrative console, and how to do some simple verification.

**Part 4: Creating the client code**

This section will show you how to develop a Web Services client for the Web Service that has been created. It also shows how to run the client and interact with the Web Service that was created and installed.

**Exercise instructions**

Instructions and subsequent documentation use symbolic references to directories which are listed as follows:

Reference Variable	 Location	  Location
<WAS_HOME>	C:\WebSphere\AppServer	 /opt/WebSphere/AppServer  /usr/WebSphere/AppServer
<TEMP>	C:\temp	/tmp
<hostname>	Host name or host address for the machine where the profiles are being created	Host name or host address for the machine where the profiles are being created

**Note:**

- This lab exercise assumes that the following software is already installed on your system. Use the documentation associated with the software to identify prerequisite software and hardware and for installation and verification instructions: IBM WebSphere Application Server Version 6.1.0.9
- IBM WebSphere Application Server Version 6.1 Feature Pack for Web Services
- IBM WebSphere Application Server Toolkit Version 6.1.1.3 and the add-on for the Feature Pack for Web Services.

---

## Part 1: Create the dynamic Web project

This section will describe the process of creating a dynamic Web project – an environment that will be used to build a Web service application using the IBM WebSphere Application Server toolkit version 6.1 (AST). The AST comes with IBM WebSphere Application Server V6.1 and this release of AST has specific support for the IBM WebSphere Application Server V6.1 Feature Pack for Web Services (WSFP).

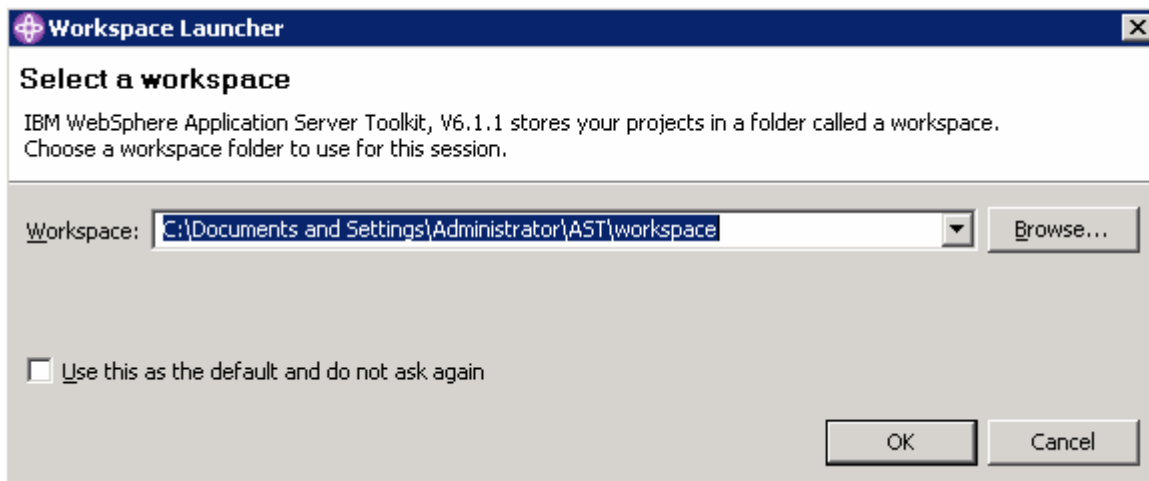
- \_\_\_\_ 1. Start the AST workspace.

To start the AST

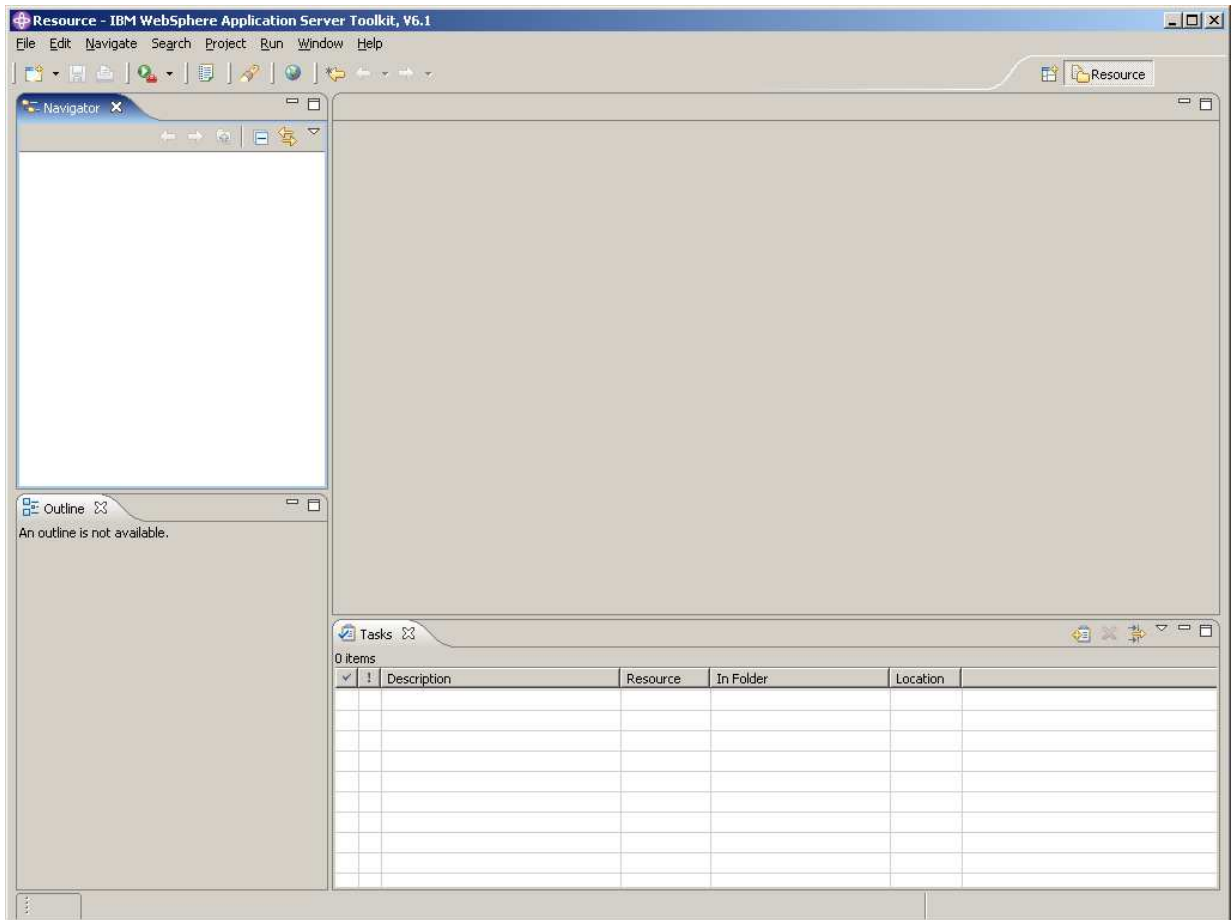
### Using the Start Menu:

**Start-> All Programs -> IBM WebSphere -> Application Server Toolkit V6.1-> Application Server Toolkit**

- \_\_\_\_ 2. When prompted to select a workspace, keep the default workspace of **C:\Documents and Settings\Administrator\AST\workspace** and click **OK**.

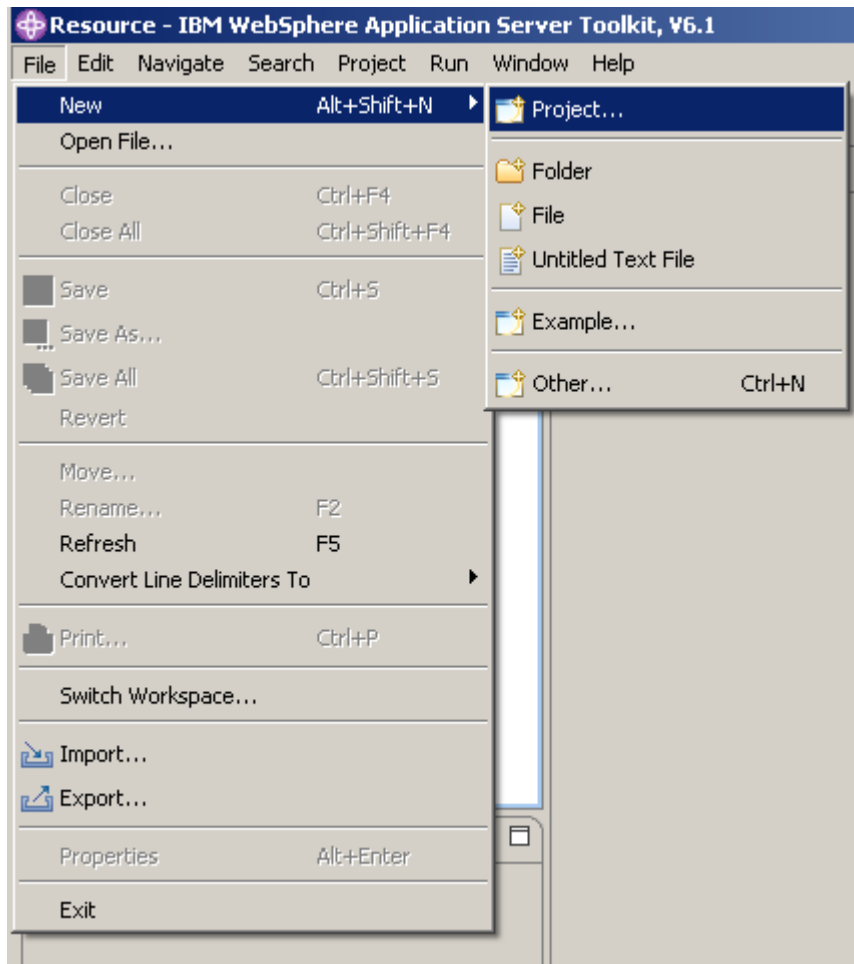


3. This will open the AST. Close the welcome screen by clicking the **X** on the welcome tab and you will see the screen shown below.

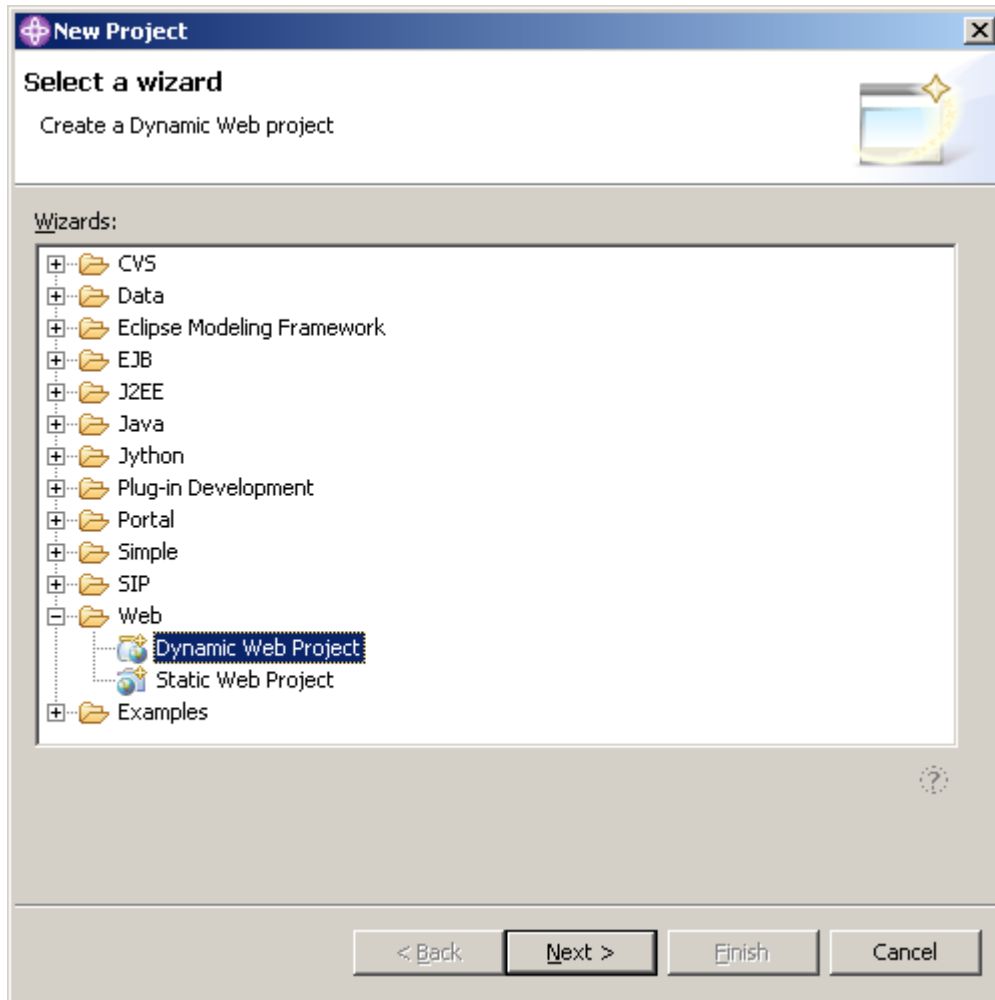


\_\_\_ 4. Creating a new dynamic Web project.

\_\_ a. Select **File** from the navigation bar. Select **New -> Project** to create a new project.



\_\_ b. Expand the Web folder and select **Dynamic Web Project**. Click **Next**.



- c. For the Project Name enter **WSFPLab2**. Check the **Add project to an EAR** check box. Enter **WSFPLab2EAR** for the EAR Project Name. Click **Next**.

**New Dynamic Web Project**

**Dynamic Web Project**  
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project contents:  
 Use default  
Directory:

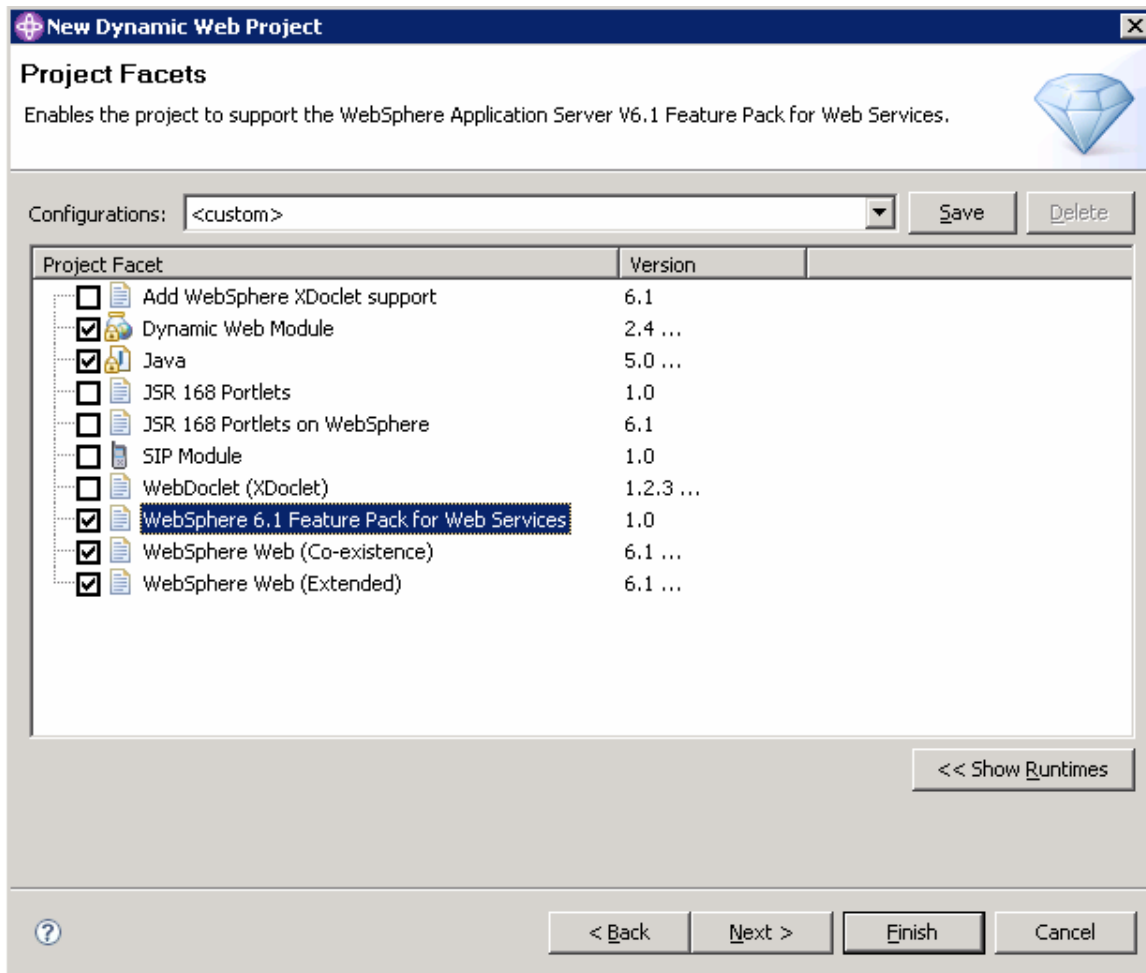
Target Runtime

Configurations  
   
Hint: Get started quickly by selecting one of the pre-defined project configurations.

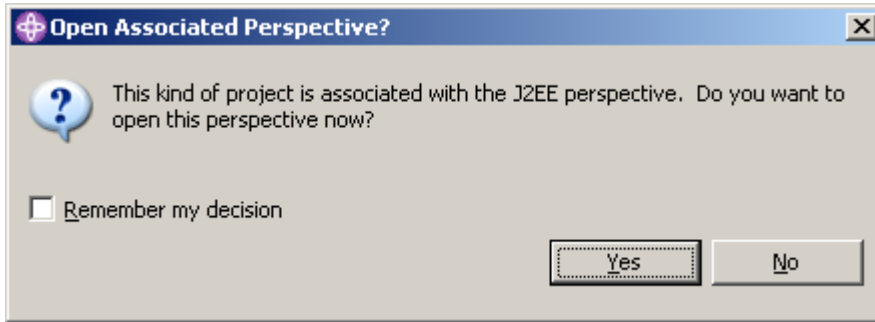
EAR Membership  
 Add project to an EAR  
EAR Project Name:



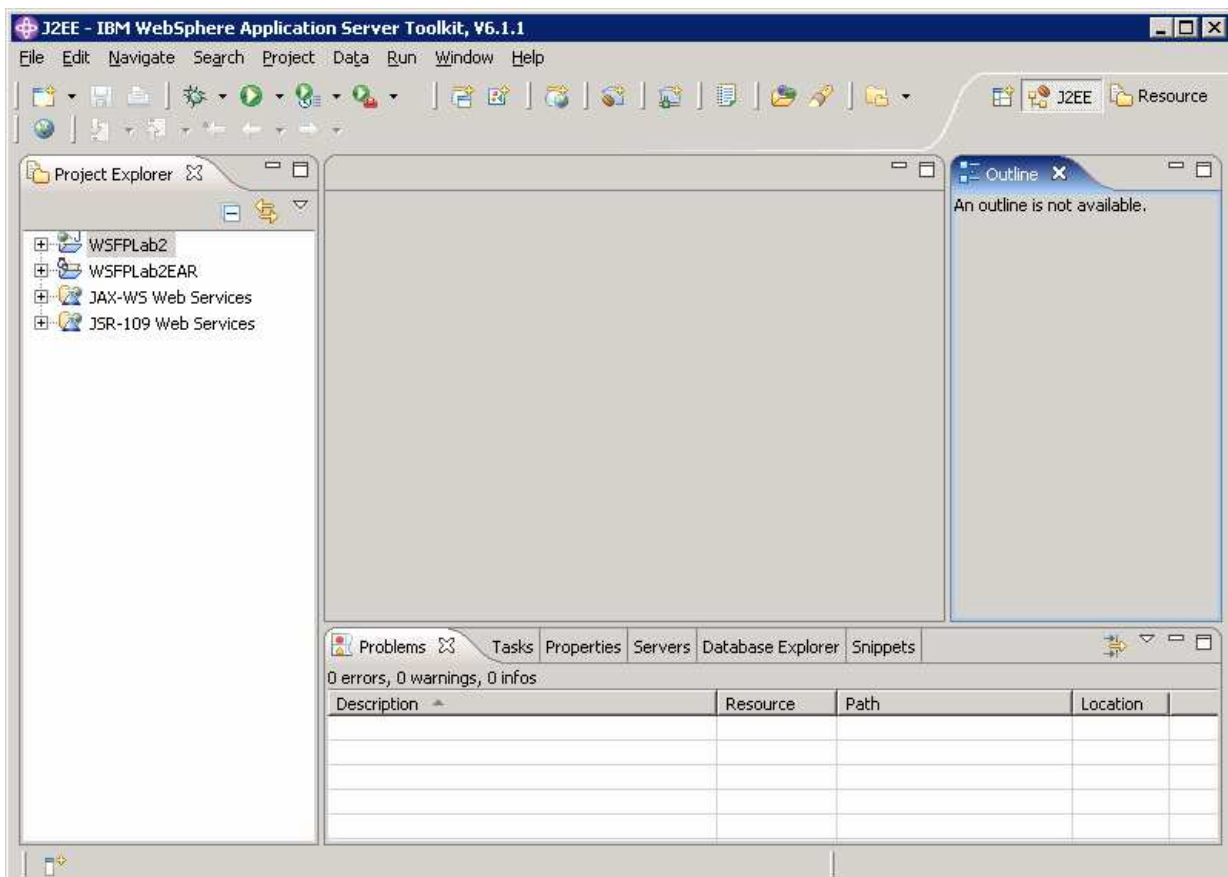
\_\_\_ d. This will open the **Project Facets** panel. Click on the check box for **WebSphere 6.1 Feature Pack for Web Services** and then click **Finish**.



\_\_ e. If you see a prompt asking to open the J2EE perspective, click Yes.



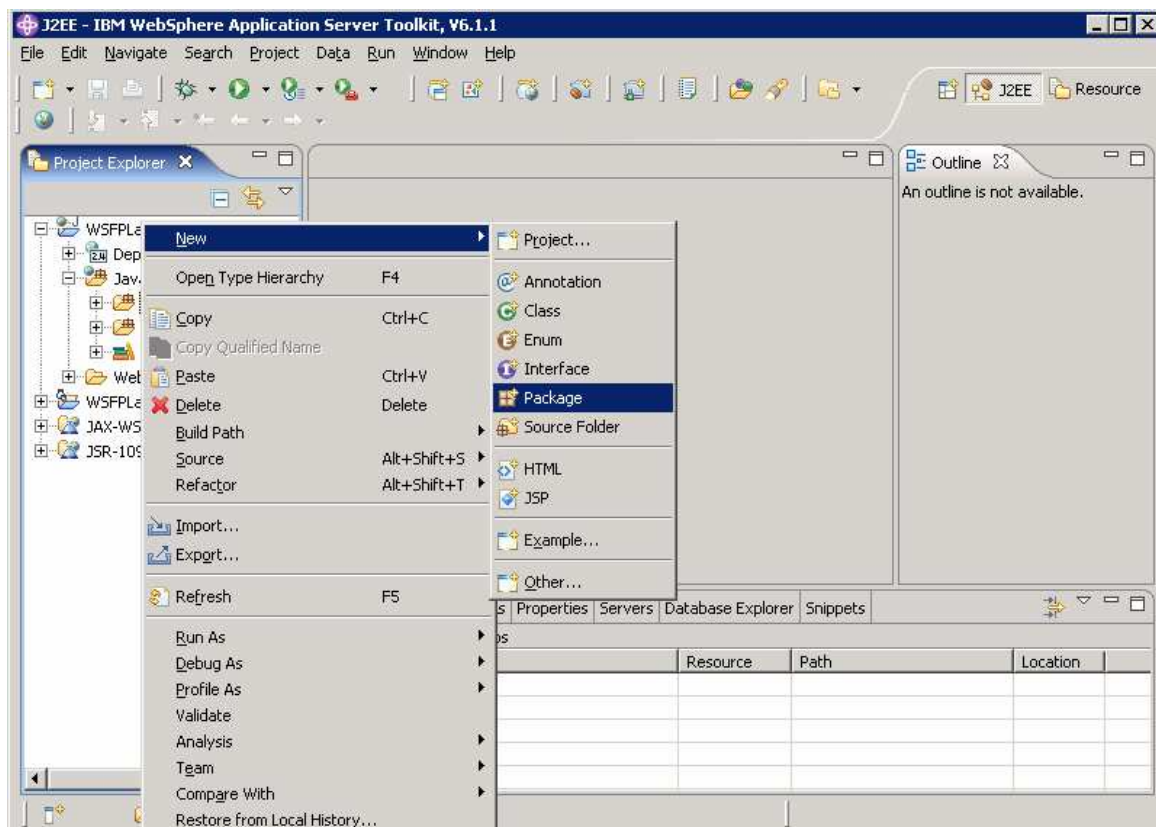
\_\_ f. The Dynamic Web Project should now be visible in the Project Explorer panel of the AST.



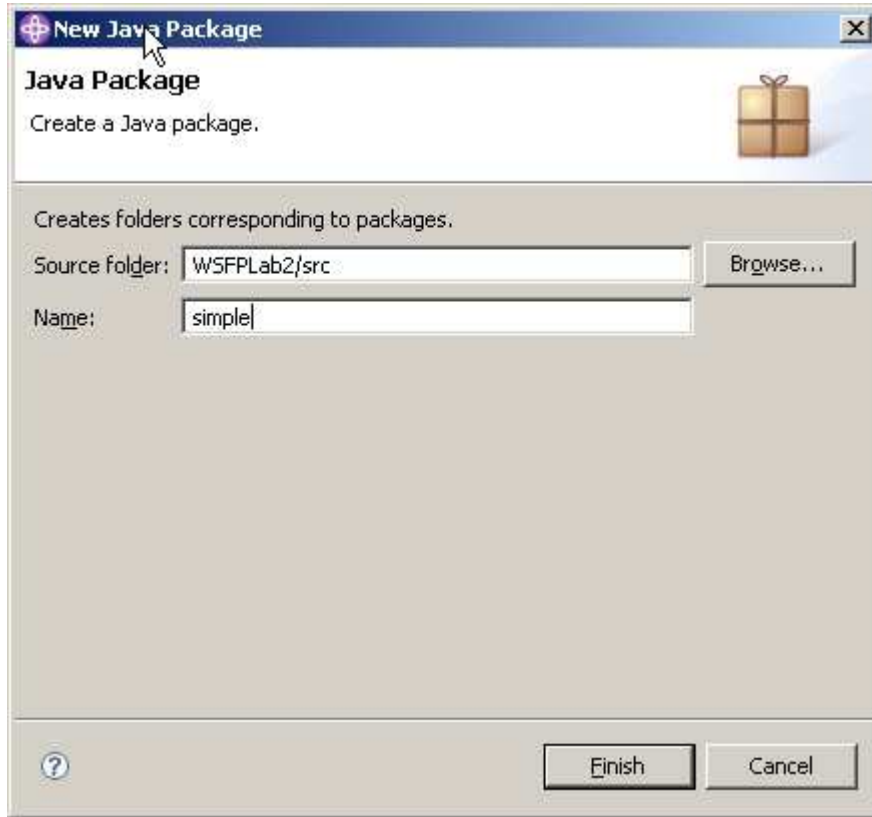
## Part 2: Import the implementation code

This section will describe the process of importing the implementation code – the existing Java code that will be the basis for your Web Service implementation.

- \_\_\_ 1. Importing the implementation code.
  - \_\_\_ a. Expand the **WSFPLab2** dynamic project in the Project Explorer panel of the AST.
  - \_\_\_ b. Expand the **Java Resources** folder.
  - \_\_\_ c. Right click the **src** folder.
  - \_\_\_ d. Select **New** then **Package**.

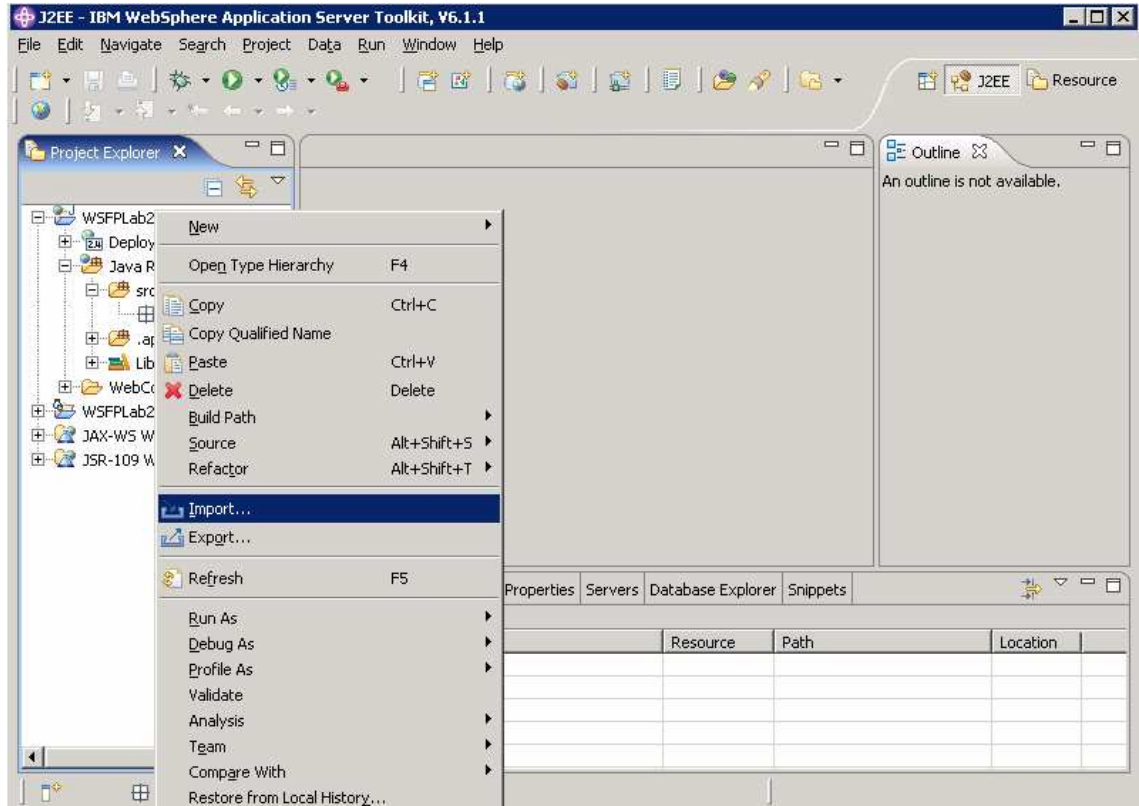


\_\_ e. Enter “simple” for **Package Name**.



\_\_ f. Click **Finish**.

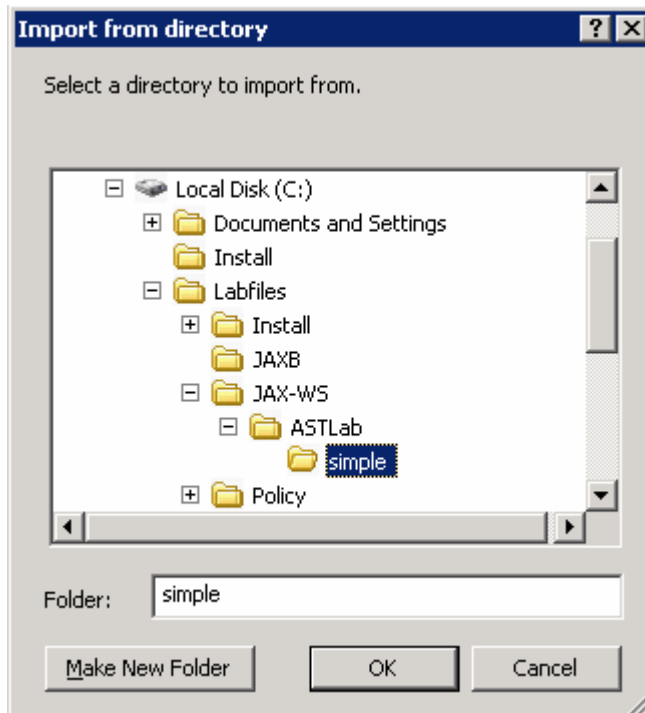
\_\_ g. Right click **simple** then select **Import**.



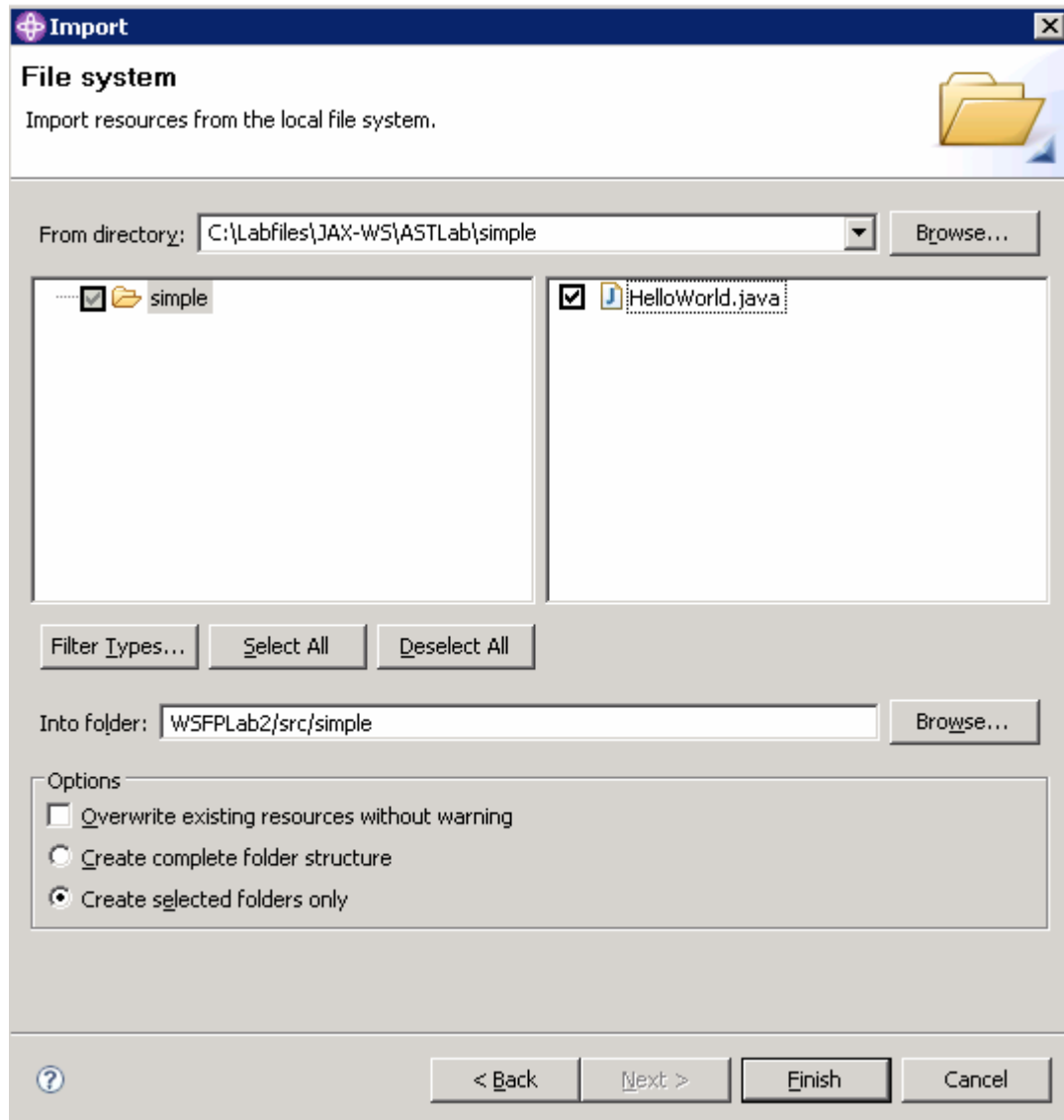
\_\_ h. In the **Import** Panel, expand **General**, select **File System** and click **Next**.



\_\_\_ i. Browse to C:\Labfiles\JAX-WS\ASTLab click the **simple** folder and click OK.

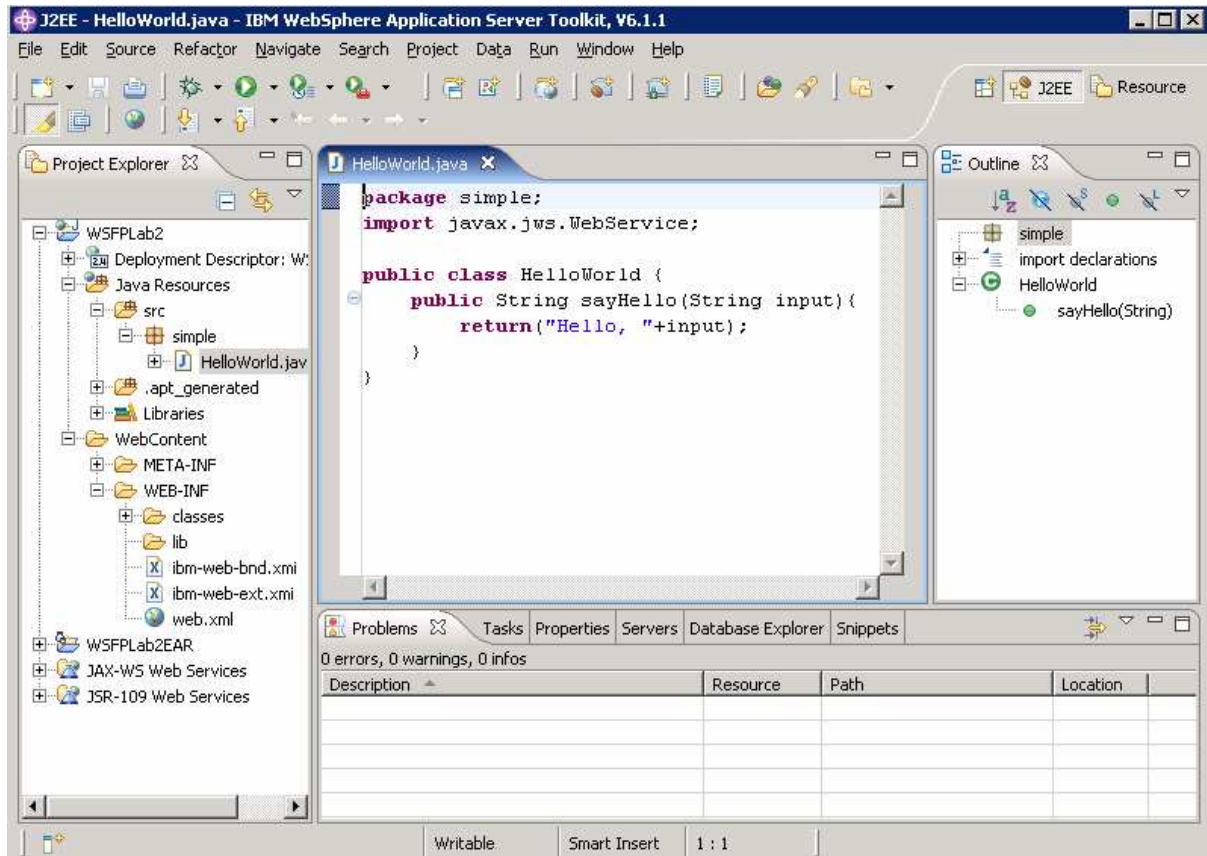


\_\_ j. Place a check mark beside **HelloWorld.java** and click **Finish**.





- \_\_\_ k. In the **Project Explorer**, expanding **Java Resources** and **WEB-INF**, reveals something like the content below. Note: there is only the one **Java** file and no **wsdl** file.

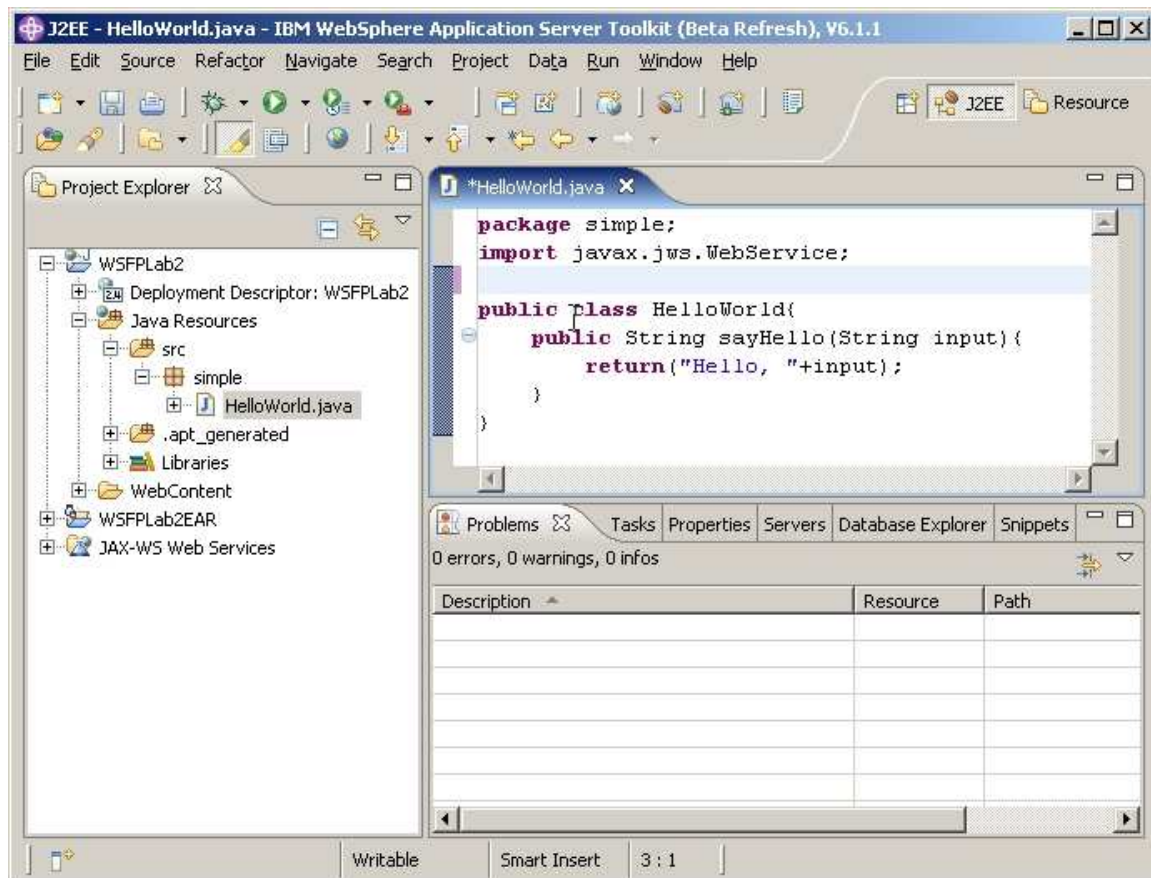


## Part 3: Creating a Web service from the implementation code

This section will describe what must be done to use the implementation code to create a Web Service.

1. You must now annotate the implementation code. From the **J2EE Perspective**, use the **Project Explorer** panel.

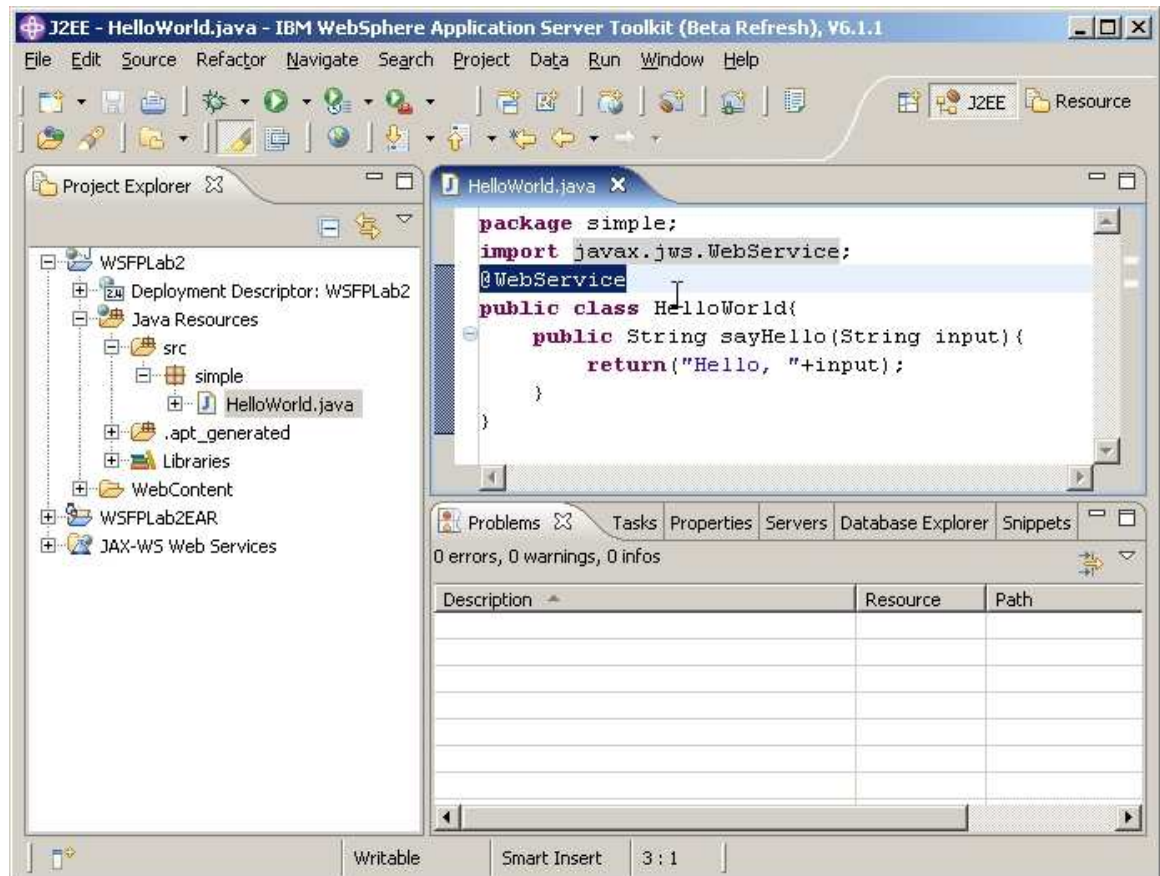
a. Double left click on **HelloWorld.java** to bring up the source.



- \_\_ b. This class contains the implementation that is to be made into a Web Service. To do that, add a single annotation (**@WebService**) just before the line containing:

**"public class HelloWorld{".**

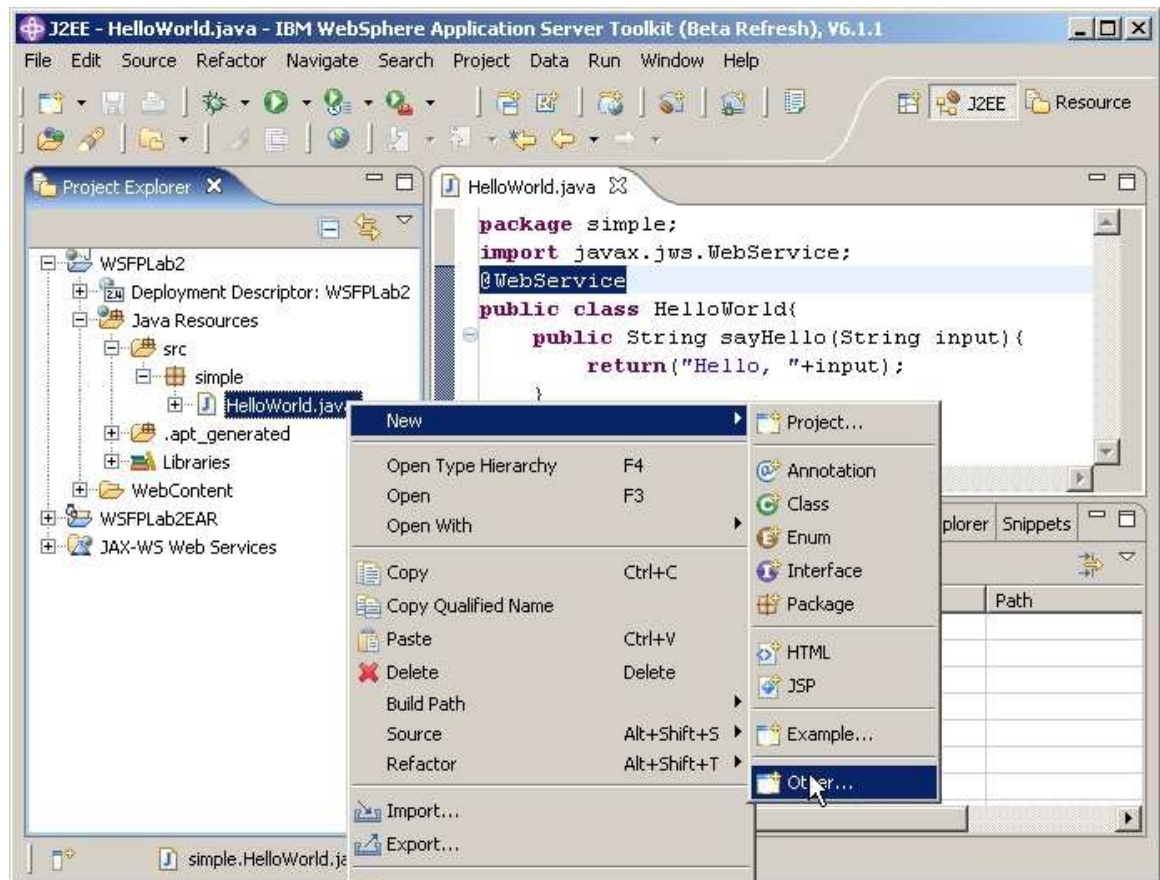
The content will now look like:



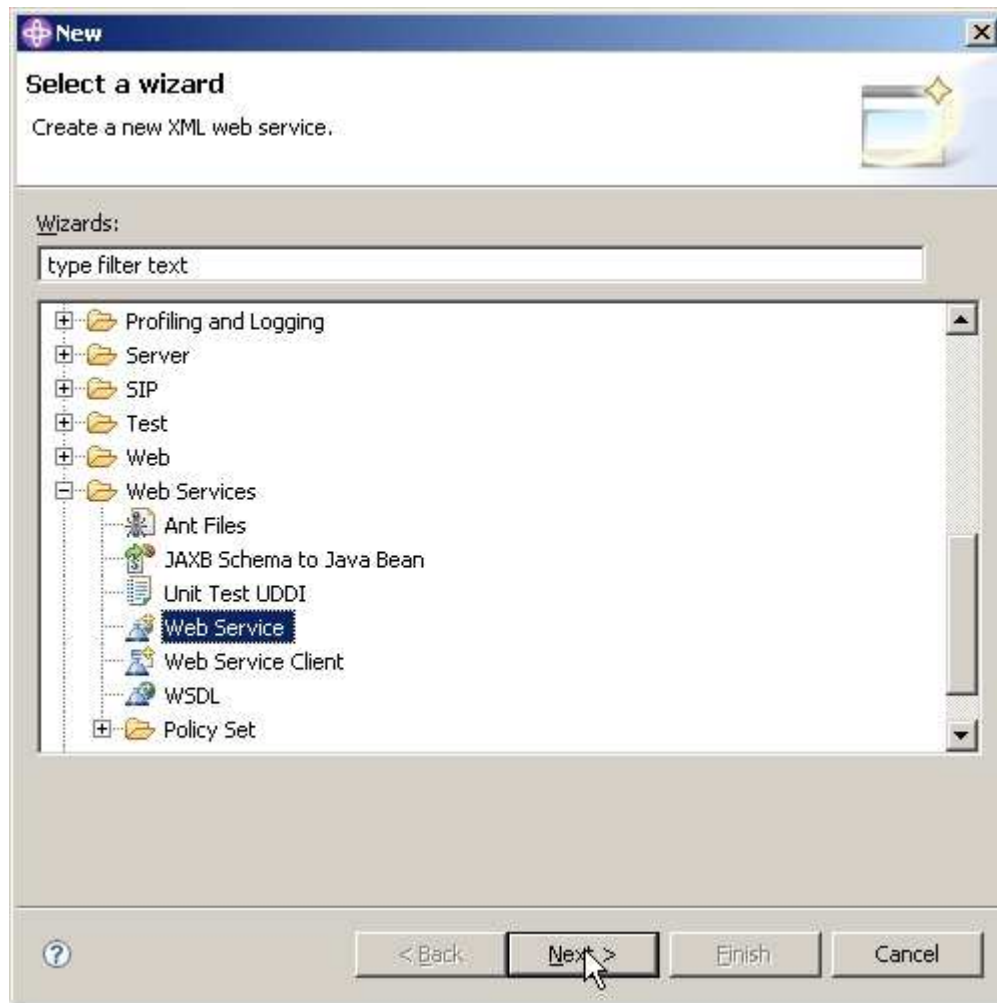
- \_\_ c. Save your changes with **CTRL + S**.

\_\_\_ 2. Create the **Web Service**.

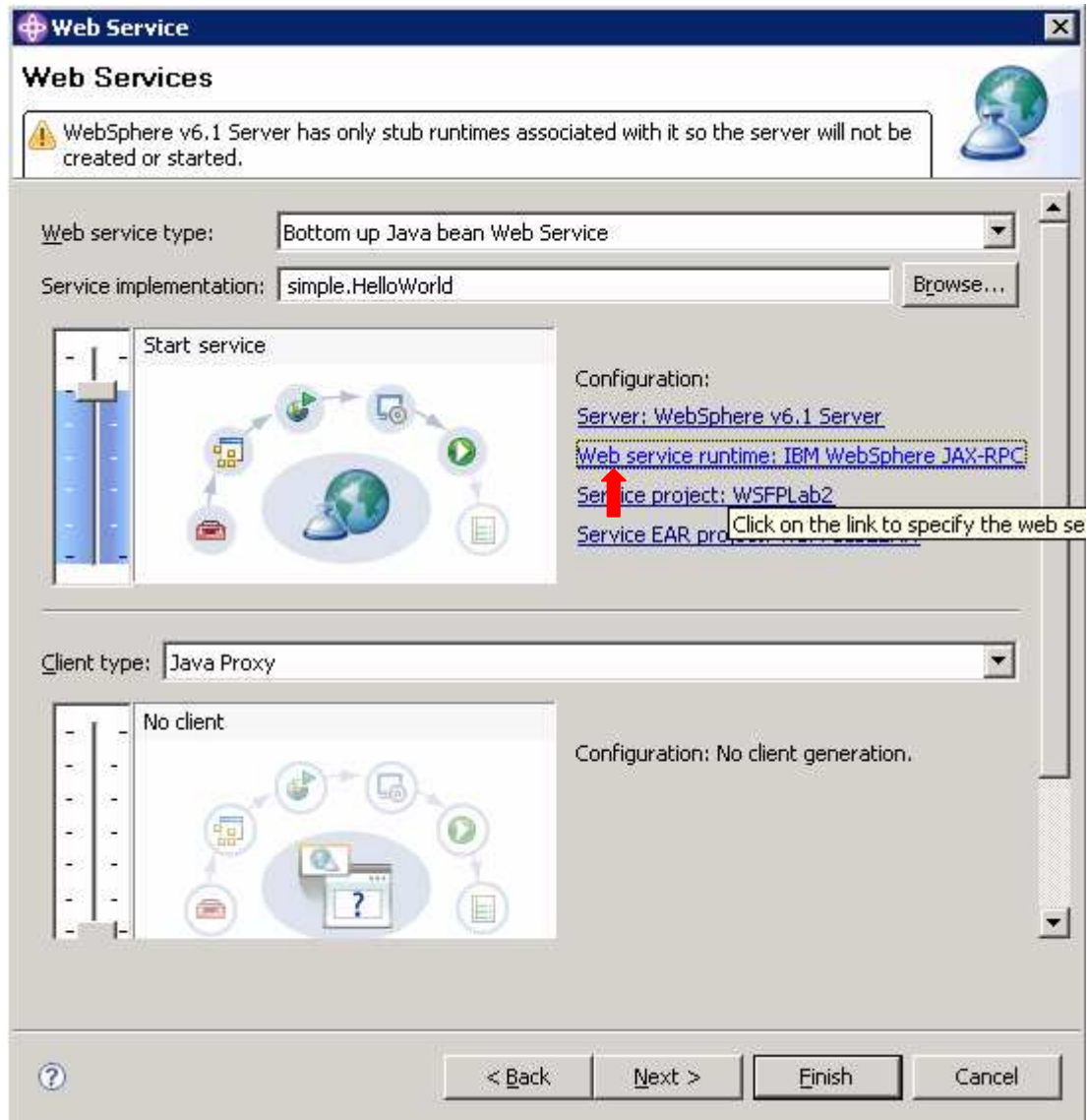
\_\_\_ a. Right click on HelloWorld.java in the Project Explorer window and choose **New** then **Other**.



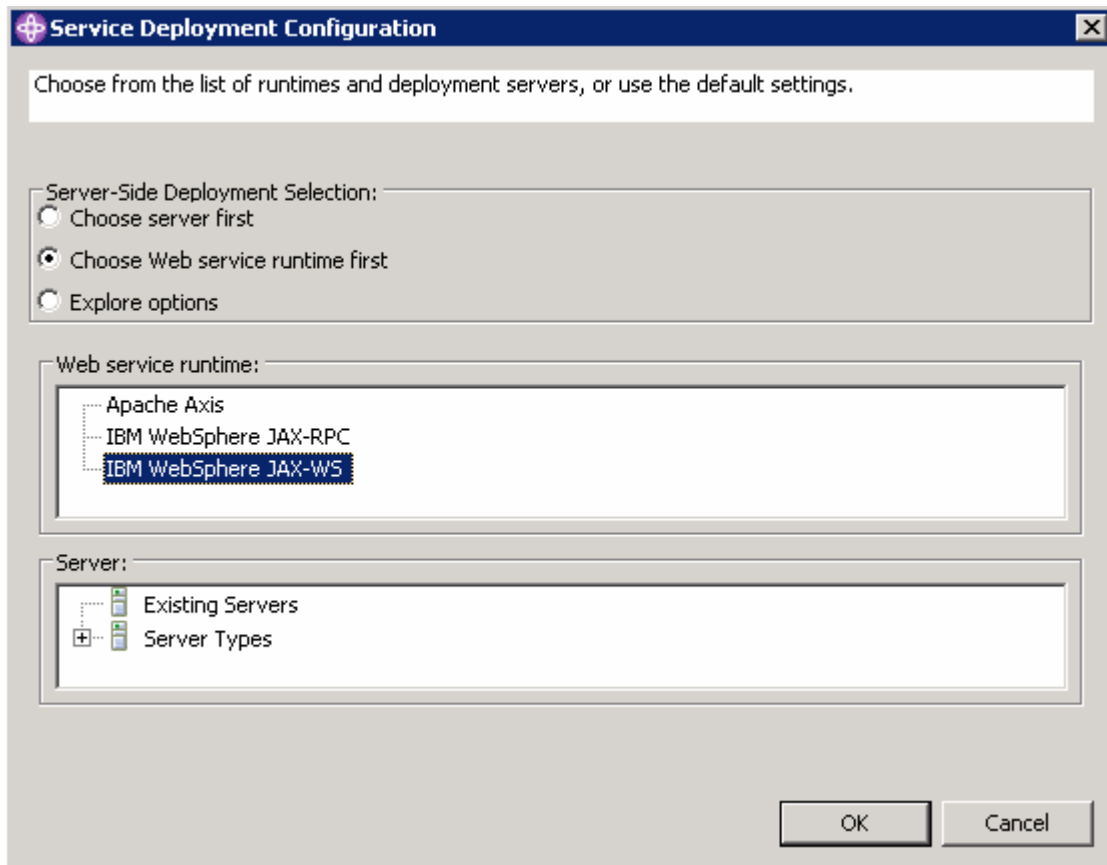
\_\_\_ b. From the “**Select a Wizard**” panel, expand **Web Services**, choose **Web Service** and click **Next**.



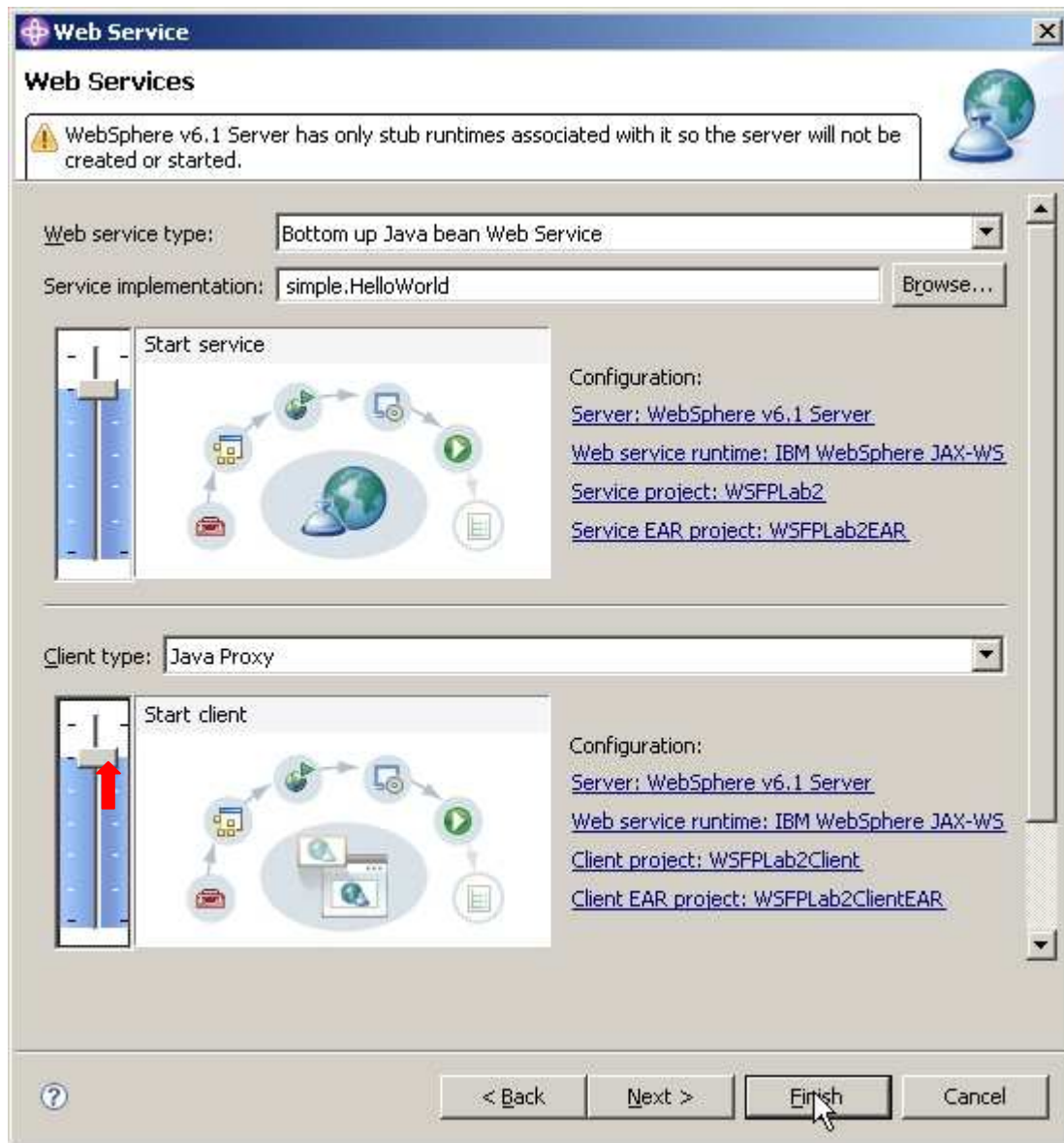
- c. Change the **Web service runtime**. Click on **Web service runtime: IBM WebSphere JAX-RPC** to change the choice of runtimes.



- d. Select the **IBM WebSphere JAX-WS** runtime option, and click **OK**. This will insure that the Web Service created will comply to the JAX-WS specification.



\_\_ e. Raise the **Client Configuration** settings so that they are like what is shown below.



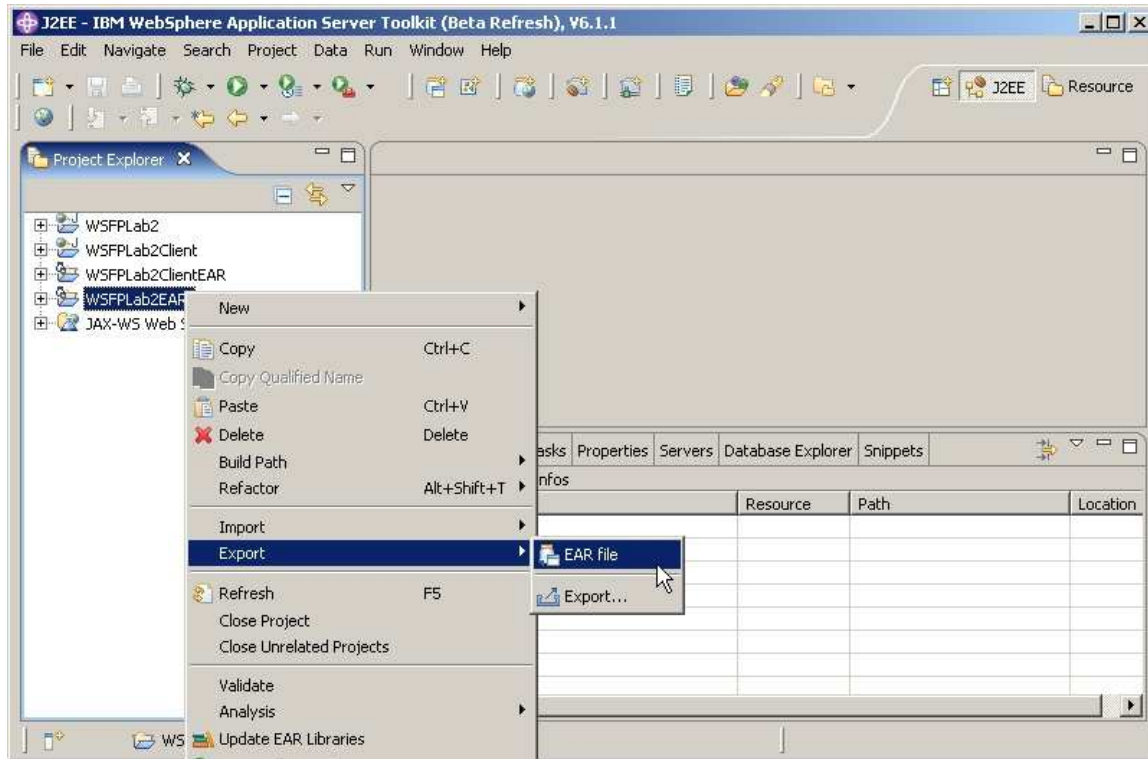
\_\_ f. Select **Finish**. The **AST** will prepare the **EAR** file associated with the **Web Service** and create the client side artifacts needed to support client development. For the client, no actual implementation code (or even a skeleton for the actual implementation code) is created – this must be created by the user!



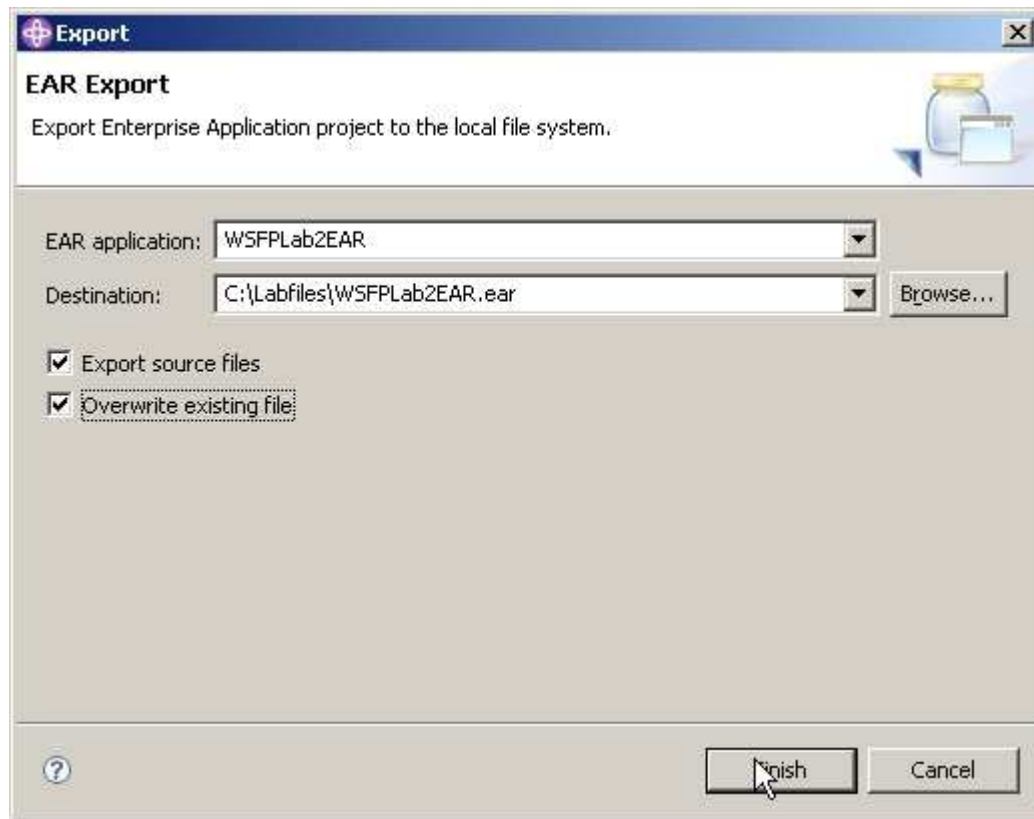
\_\_\_ 3. Exporting the **EAR** file.

\_\_\_ a. In the **Project Explorer** window, **right-click** on the **WSFPLab2EAR** folder to open the menu.

\_\_\_ b. Select **Export** and the **EAR** file.



- \_\_\_ c. Select **WSFPLab2EAR** for the **EAR** application. For the **Destination** select **C:\LabFiles**. Check the boxes to **Export source files** the **source code** and **Overwrite existing file**. Click **Finish**.



- \_\_\_ d. The **EAR** file has been exported and can now be installed to the **WebSphere Application Server**.

## Part 4: Install and deploy the application

This section describes how to install and deploy the EAR file that has been created to an installation of WebSphere Application Server Version 6.1 with the Feature Pack for Web Services installed. This will use the profile that was created in Part 1 of this lab.

- \_\_\_ 1. **Start the application server.**
  - \_\_\_ a. Open a command prompt and change to **C:\WebSphere\AppServer\profiles\AppSrv01\bin**.
  - \_\_\_ b. Run the command **startserver server1**.

```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings>cd ..
C:\>cd WebSphere/Appserver/profiles/appsrv01
C:\WebSphere\AppServer\profiles\AppSrv01>cd bin
C:\WebSphere\AppServer\profiles\AppSrv01\bin>stopserver server1
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\profiles\AppSrv01\logs\server1\stopServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.

C:\WebSphere\AppServer\profiles\AppSrv01\bin>startserver server1
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\profiles\AppSrv01\logs\server1\startServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1264
C:\WebSphere\AppServer\profiles\AppSrv01\bin>

```

- \_\_\_ c. Wait for the server to start.
- \_\_\_ 2. **Start the Integrated Solution Console (ISC).** There are two ways to start the ISC.

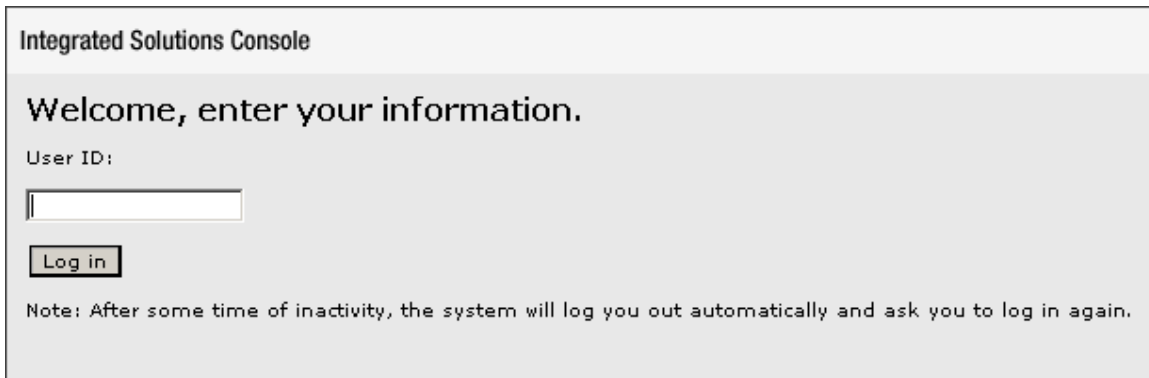
- \_\_\_ a. Using the Start Menu **Windows** :

Click on Start->Programs->IBM WebSphere->Application Server v6.1->Profiles->wslab->Administrative Console

- \_\_\_ b. Using a browser. Start the browser and enter the following in the address window::

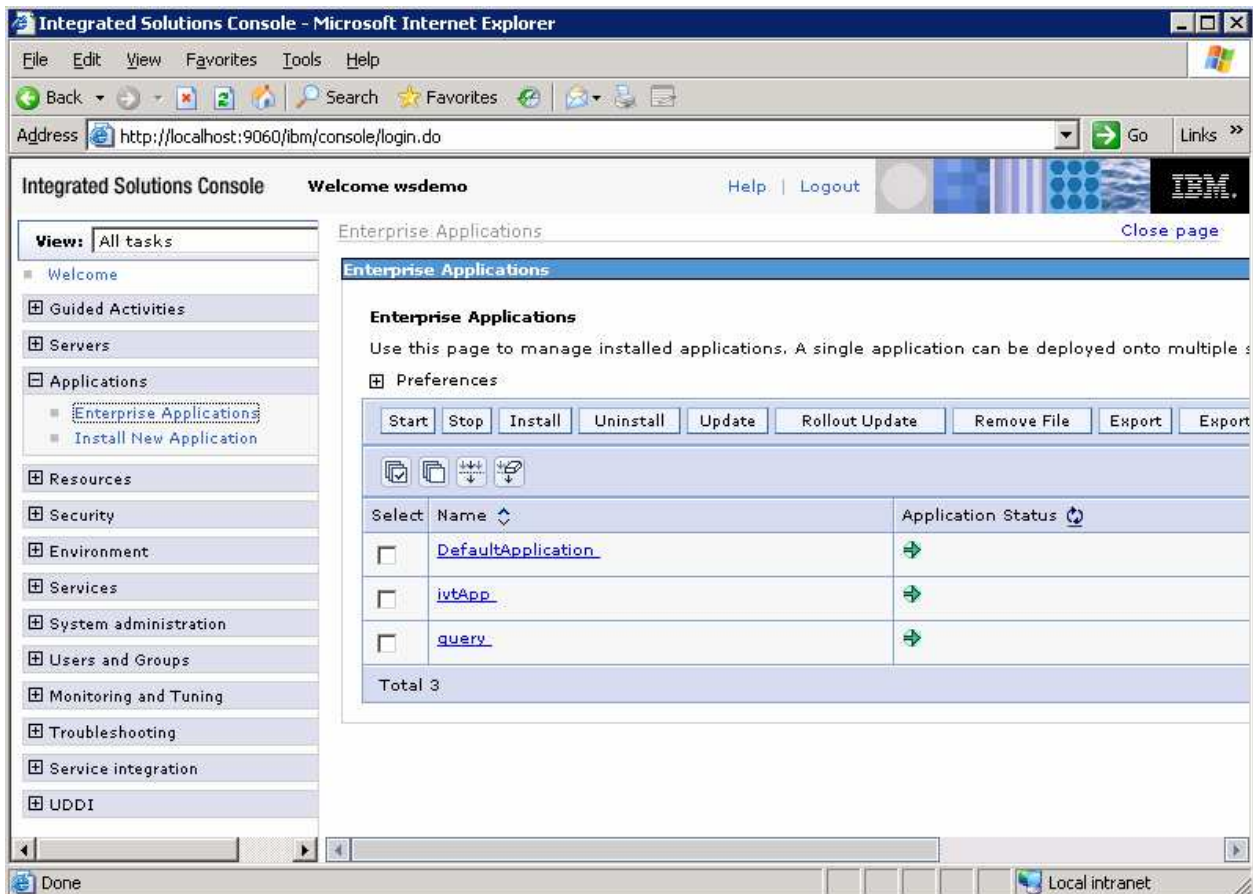
<http://localhost:9060/ibm/console>

- 3. Log in to the **ISC** as **wsdemo**.

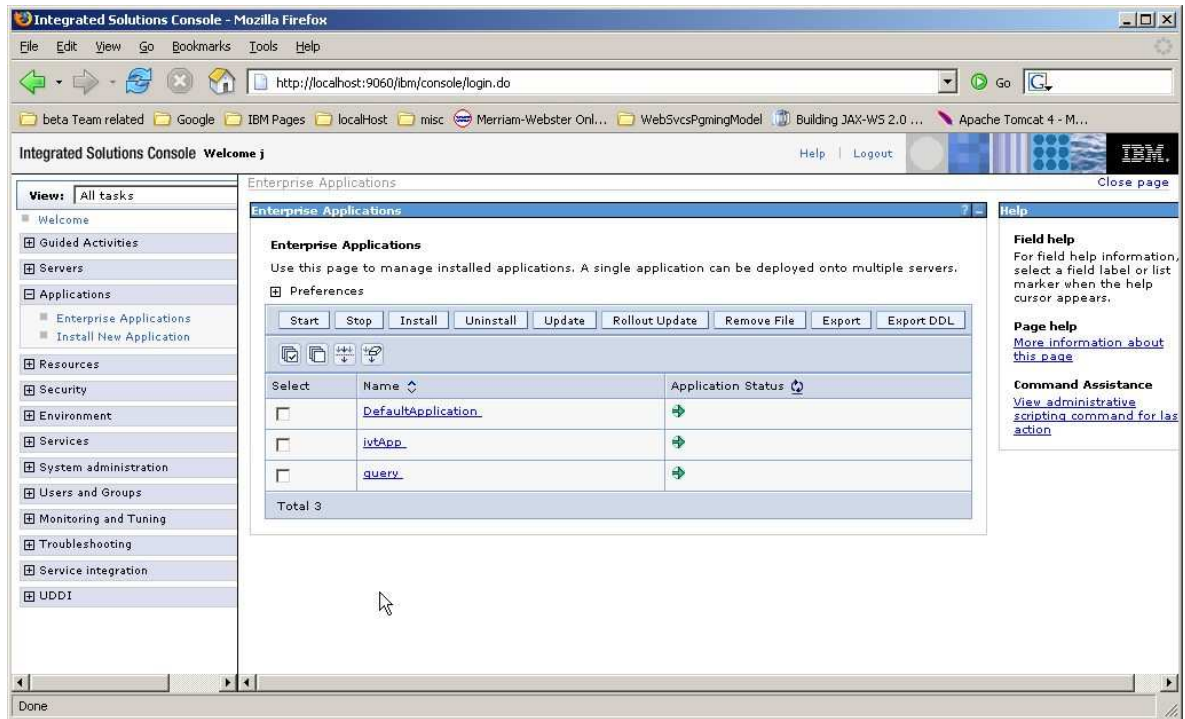


- 4. Install the **WSFPLab2EAR** file.

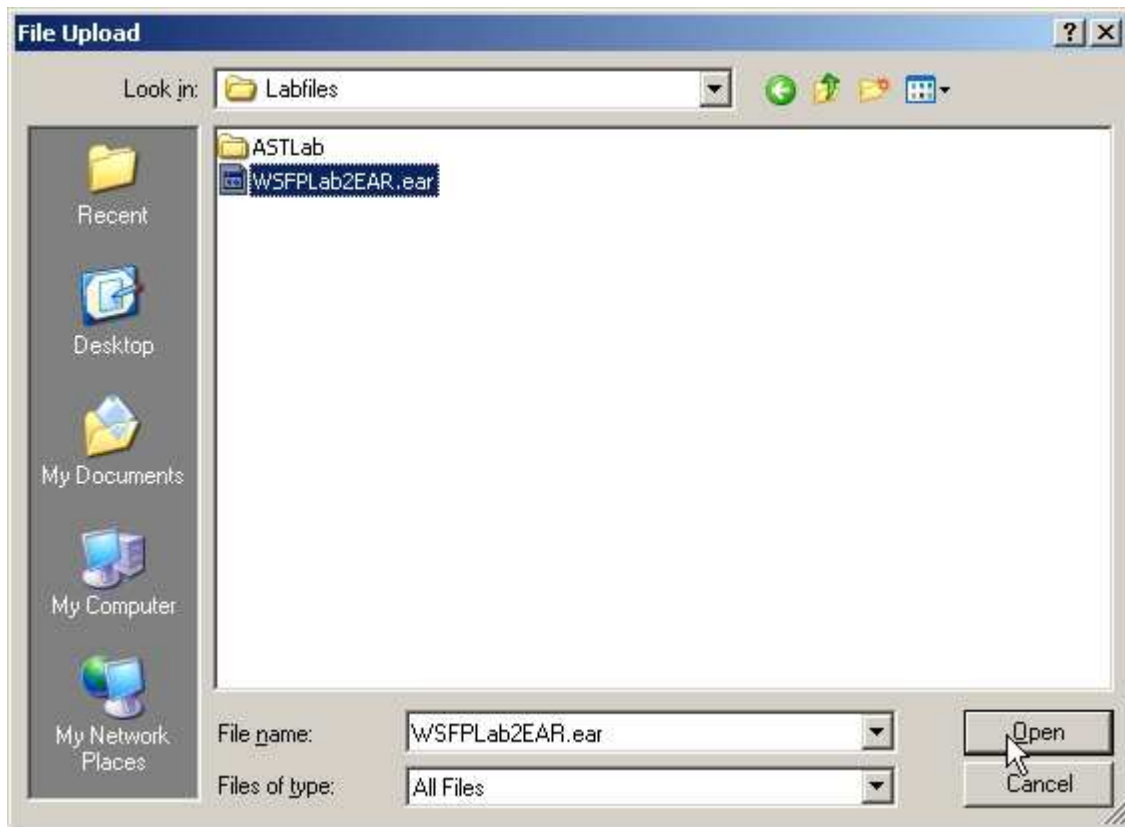
a. Expand **Applications** and click on **Enterprise Applications**. The **ISC** will display a screen similar to the one shown below.



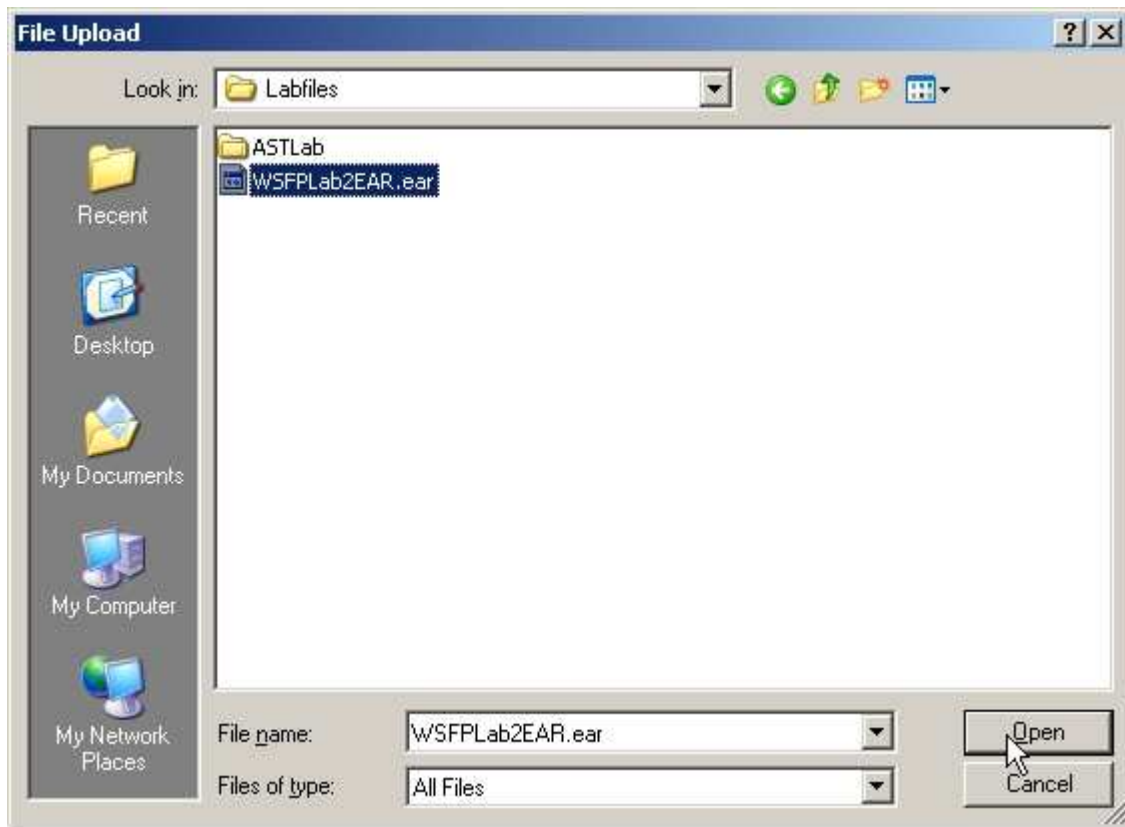
\_\_ b. In the **Enterprise Applications** panel, select the **Install** button.



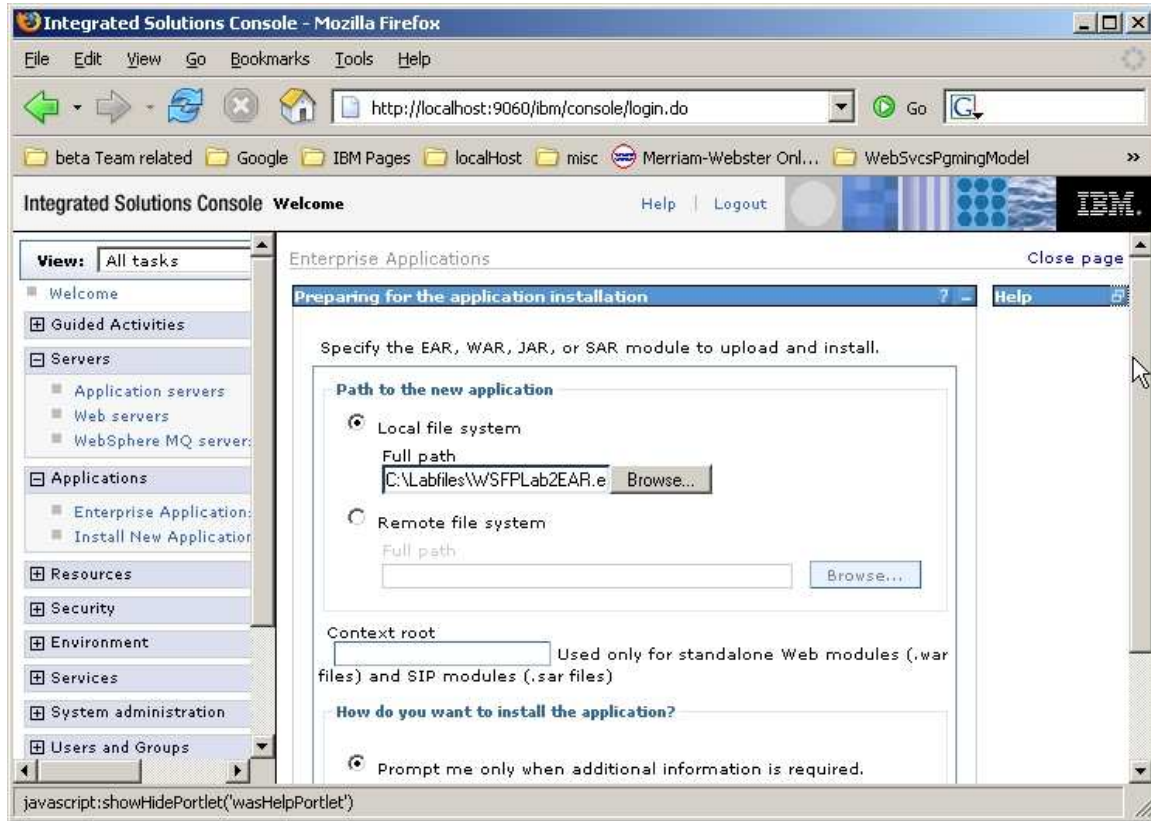
\_\_ c. Click **Browse** and browse to **C:\LabFiles** where you exported **WSFPLab2EAR.ear**.



\_\_ d. Select **WSFPLab2EAR.ear** and Click **Open**.



\_\_ e. In the ISC click **Next**.





\_\_ f. On the **Select installation options** panel click **Next**.

\_\_ g. On the **Map modules to servers** panel click **Next**.

\_\_ h. On the **Summary** panel click **Finish**.

\_\_ i. In the **ISC**, you should see the following:

Application WSFPLab2EAR installed successfully.

To start the application, first save changes to the master configuration.

Changes have been made to your local configuration. You can:

- [Save](#) directly to the master configuration.
- [Review](#) changes before saving or discarding.

To work with installed applications, click the "Manage Applications" button.

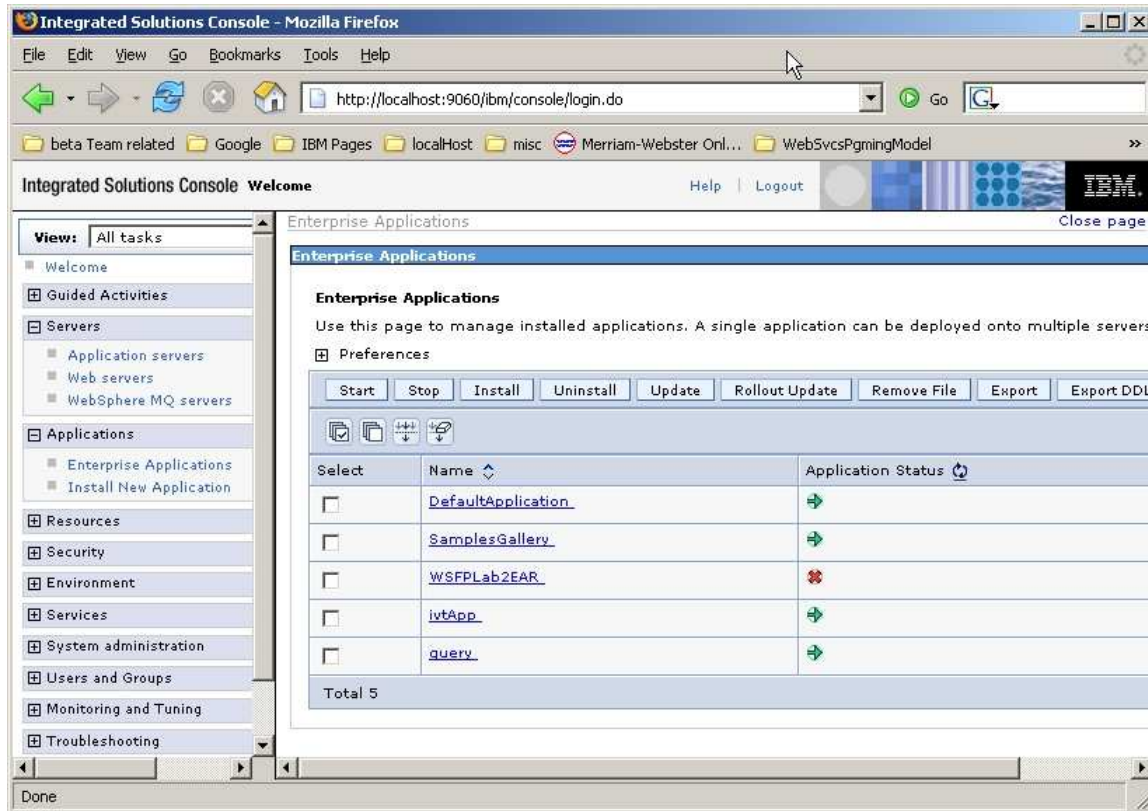
[Manage Applications](#)

\_\_ j. Wait to see that the application installs successfully. The messages shown above will be displayed. Click **Save** to save the changes to the master configuration.

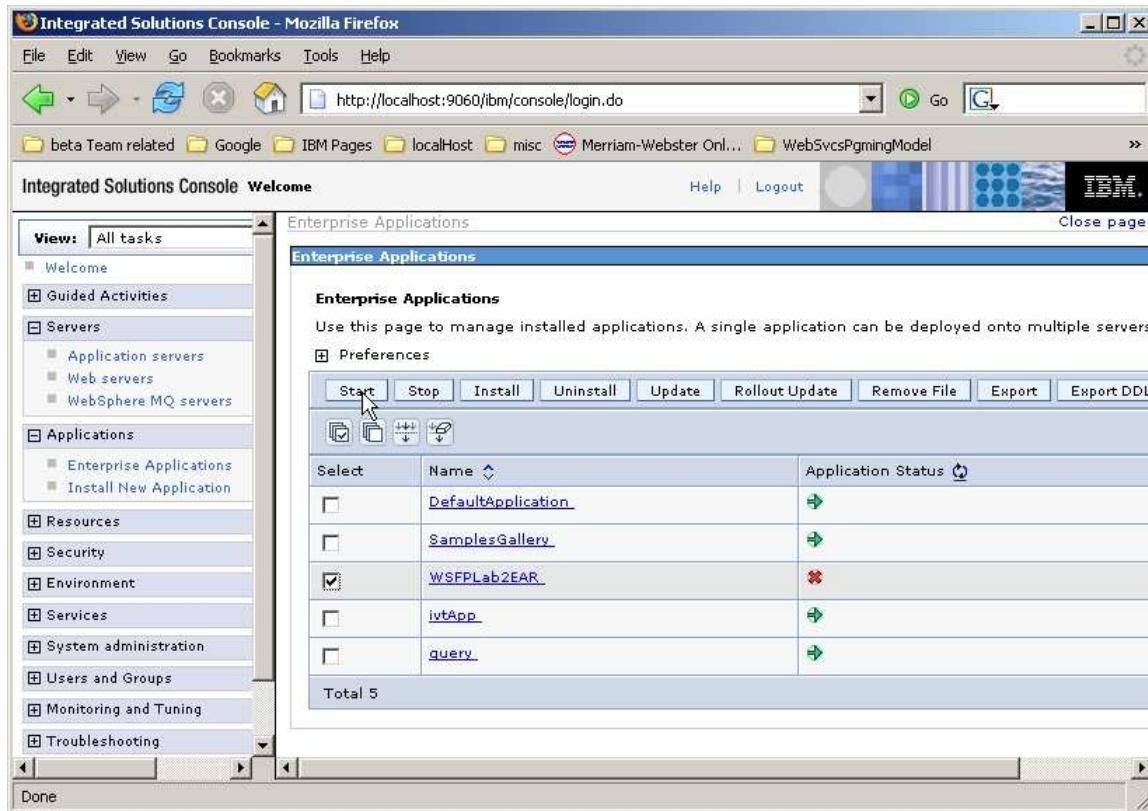
\_\_ k. The **WSFPLab2EAR.ear** has now been deployed.

5. Starting the application.

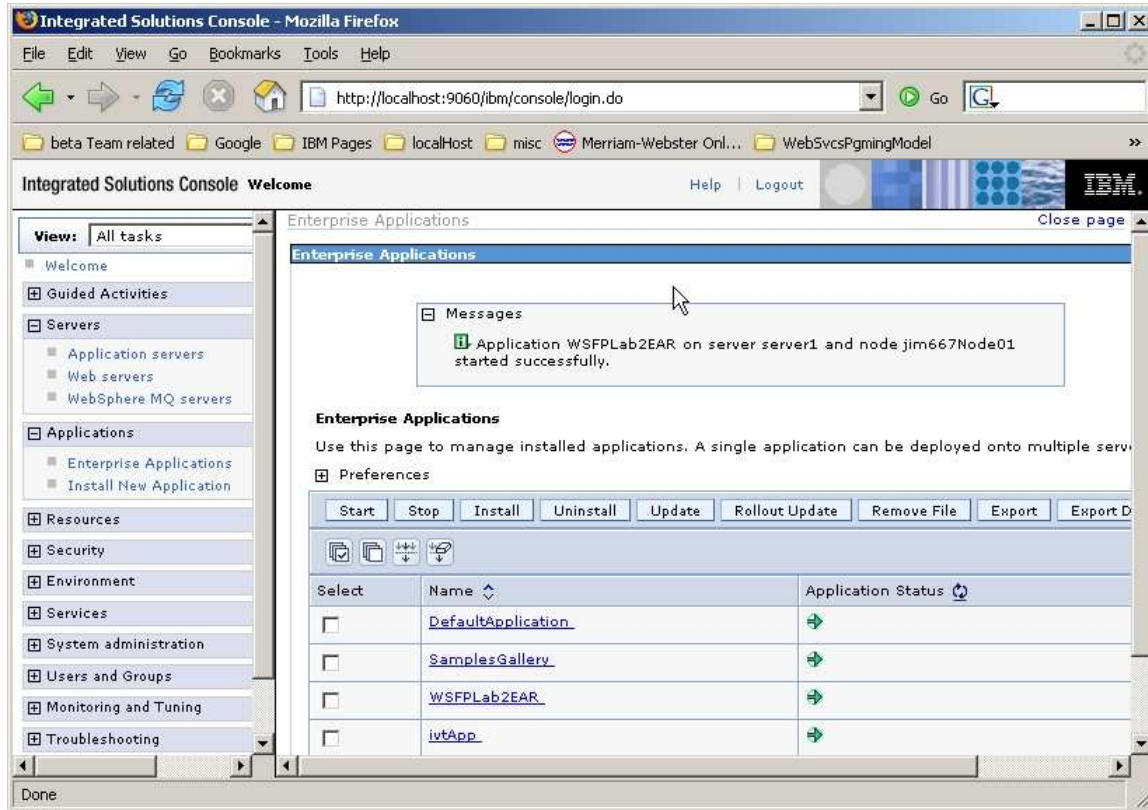
- a. Return to the **Enterprise Applications** panel. The **ISC** will display a screen similar to the one shown below. The **WSFPLab2EAR** application will show a red "x" in the **Application Status** column to indicate that the application is not yet started.



- \_\_ b. Click the box next to **WSFPLab2EAR**, a check mark, “☑”, will appear in the box to the left of **WSFPLab2EAR**. Click **Start**.



\_\_ c. Wait for the application to start. The following page will be displayed.



\_\_\_ 6. Test the Web Service.

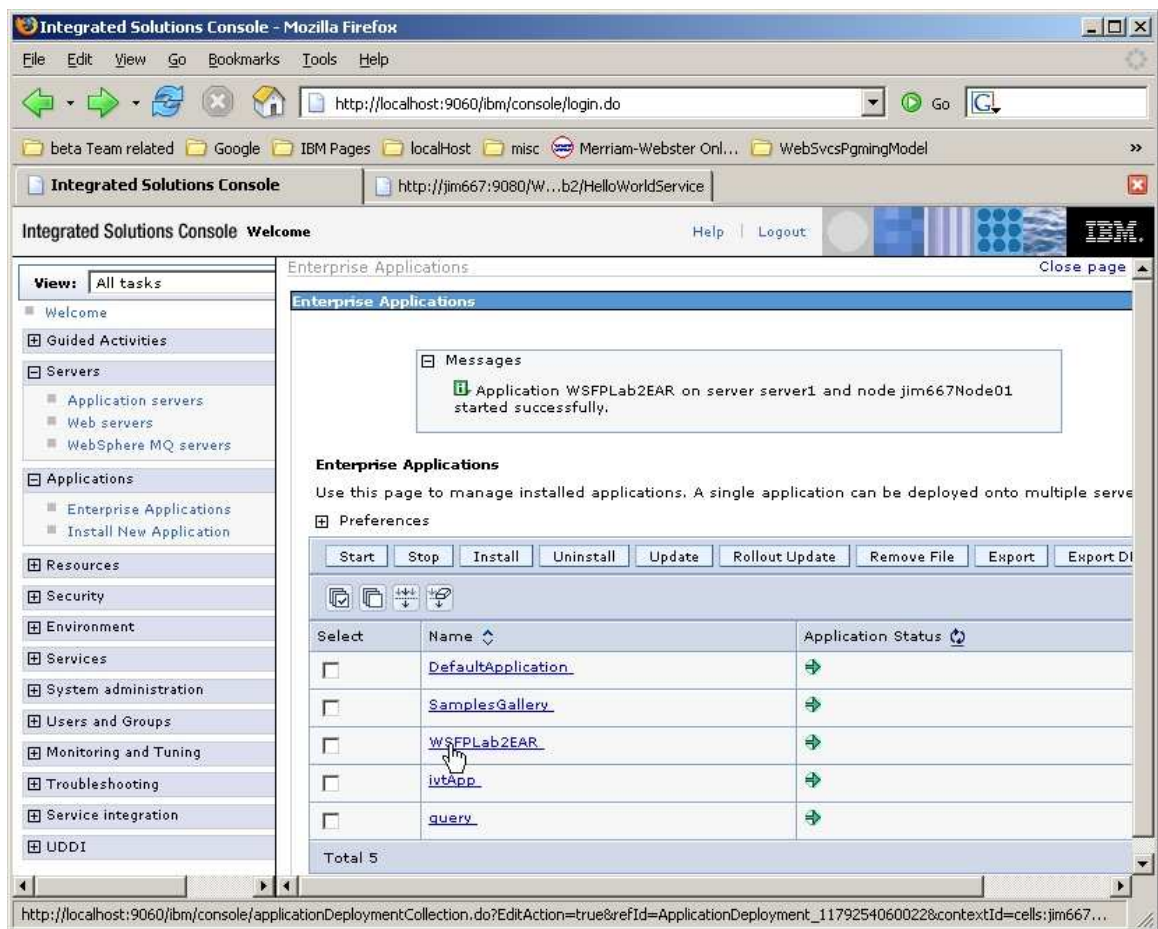
- \_\_\_ a. To verify that the Web Service was deployed and started properly, a browser will be given the **URL** for the Web Service and it should display a message from the application server indicating that this is a Web Service.

The URL for the Web service can be built by concatenating the following:

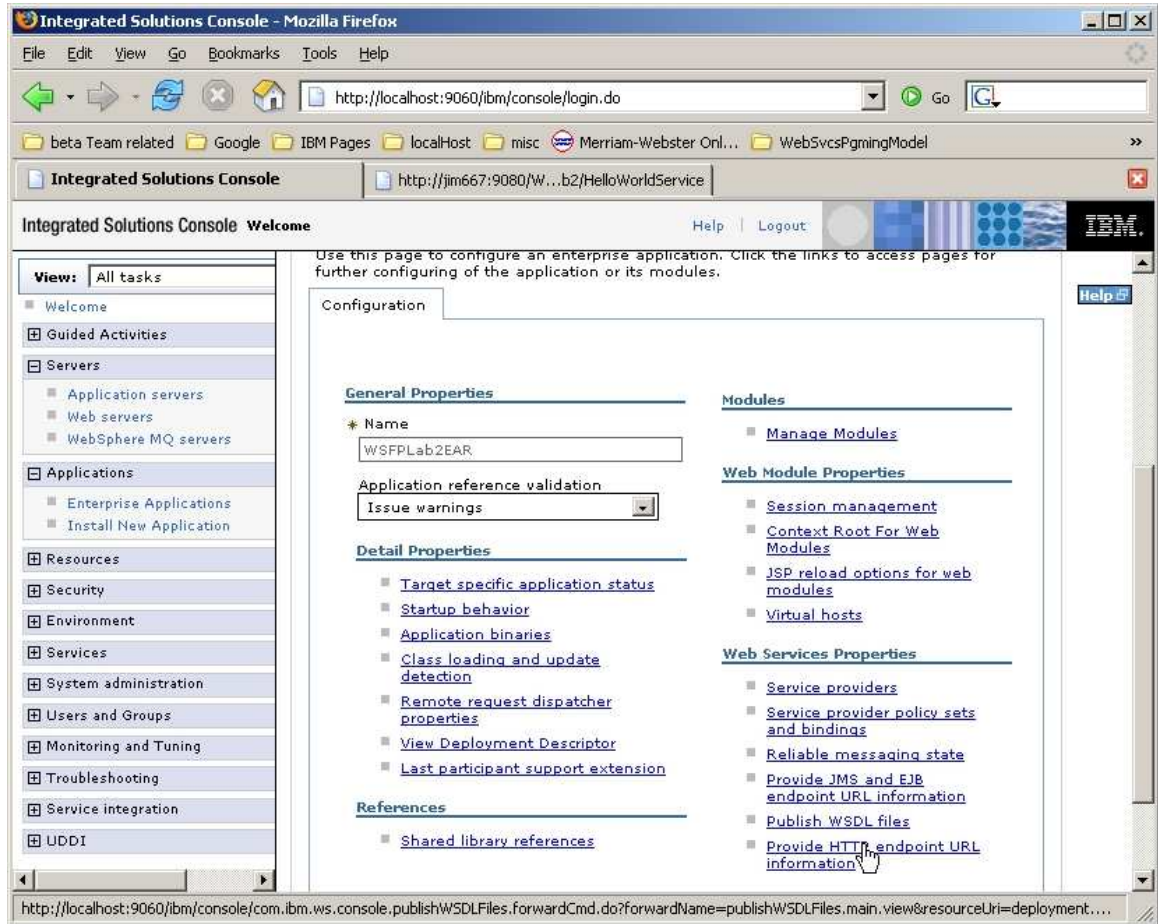
**<HTTP URL Prefix>/WSFPLab2/HelloWorldService**

The next steps show how to find this information and create this URL.

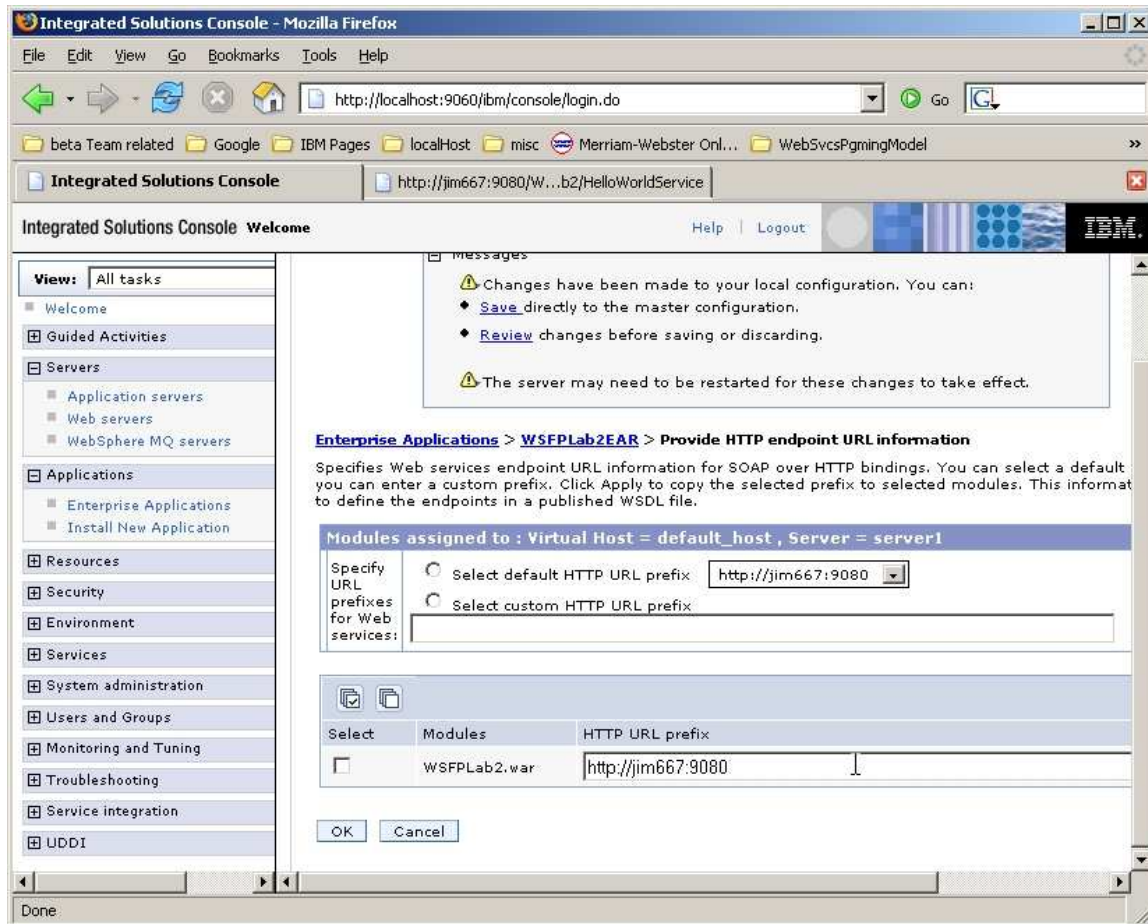
- \_\_\_ b. To find the **<HTTP URL Prefix>**, in the **ISC** navigate to the **Enterprise Applications** panel. Select **WSFPLab2EAR**.



\_\_ c. Select **Provide HTTP endpoint URL information**.



- d. The value displayed in the window labeled **HTTP URL prefix** is what is needed for **<HTTP URL Prefix>**. In this case it is `http://jim667:9080`. It will be different on other systems.



- \_\_\_ e. Putting the URL prefix together with /WSFPLab/LabService gives the URL for the Web Service. So with the above prefix the test URL would be:

`http://jim667:9080/WSFPLab2/HelloWorldService`

- \_\_\_ f. Open a new browser window and paste the URL for your system in the address field and press Enter. You should see a screen like that below.



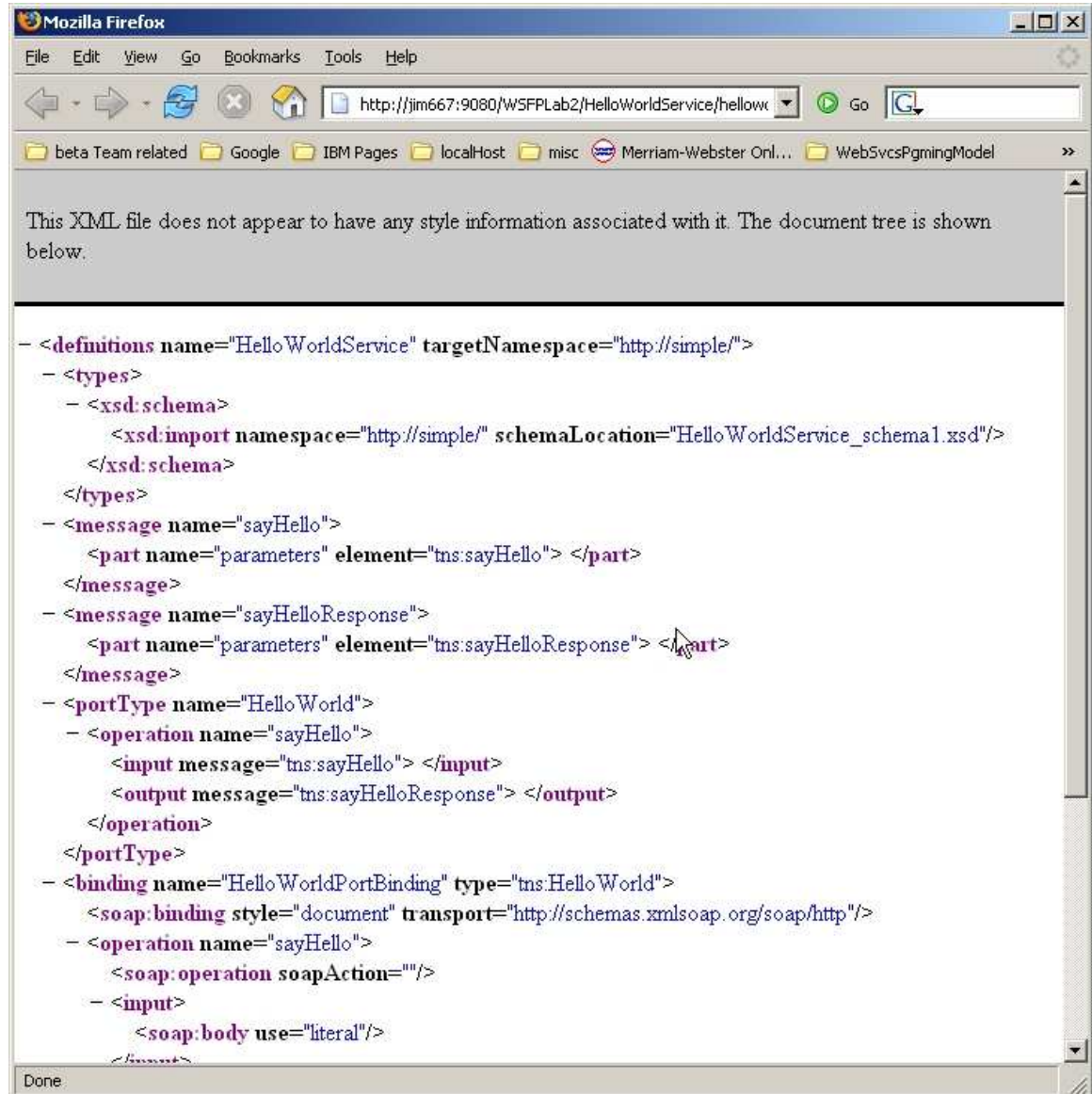
- \_\_\_ g. This shows that the Web Service application is installed, recognized by WebSphere Application Server, is running, and is listening on the expected port.



\_\_ h. To see the WSDL for this Web Service, you may append “/wsdl” to the URL. This would yield:

<http://jim667:9080/WSFPLab2/HelloWorldService/wsdl>

and submit from your browser. You should see something like:

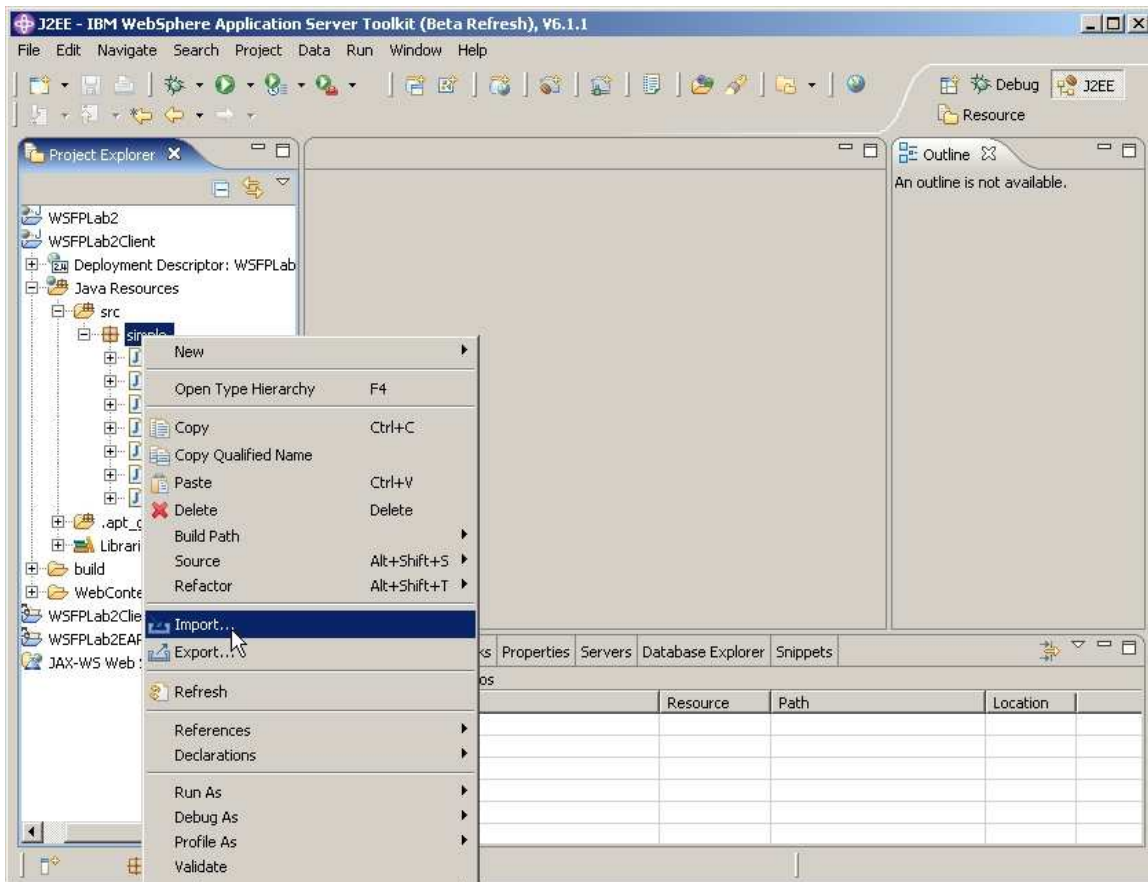


\_\_ i. Leave your browser window open so you can reference this wsdl as you develop the client code.

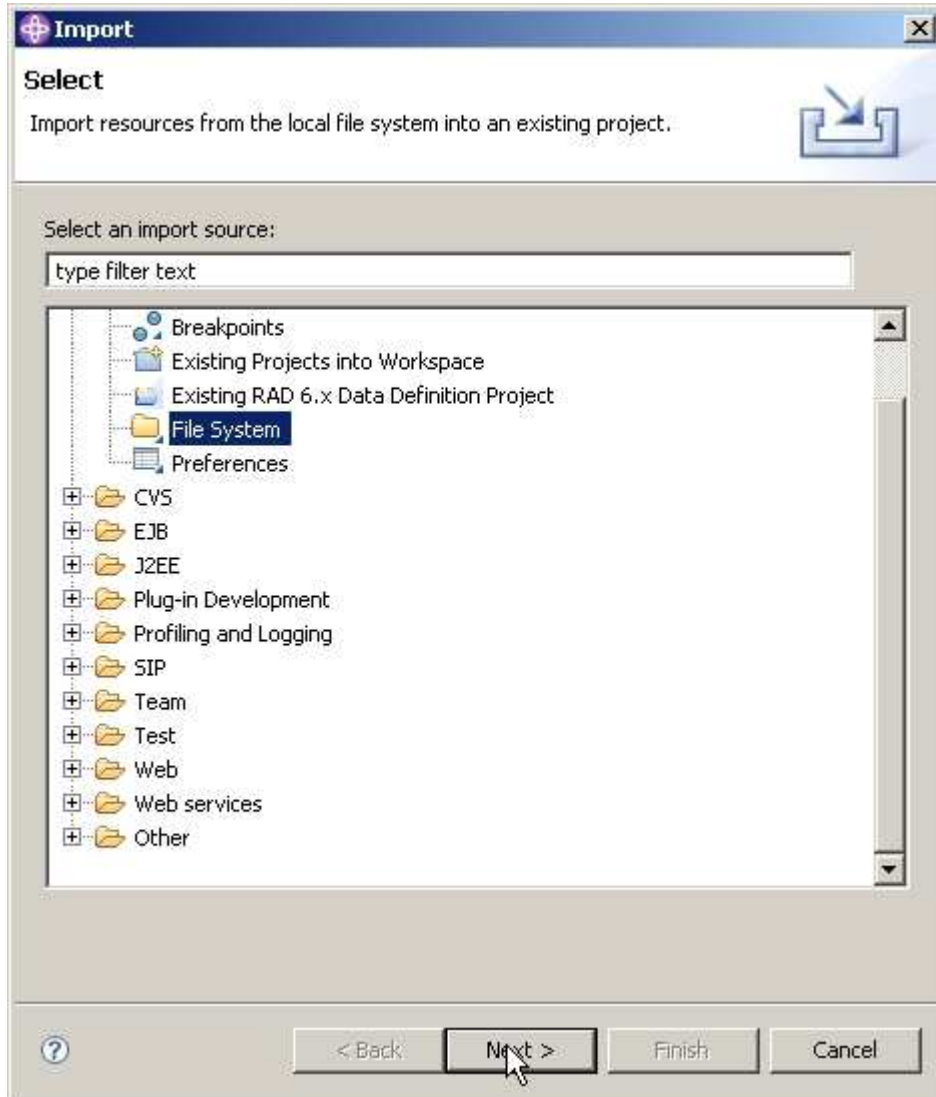
## Part 5: Creating the client code

In this section of the lab, a Java Web services client will be created to interact with the Web Service that has been created, deployed, and started.

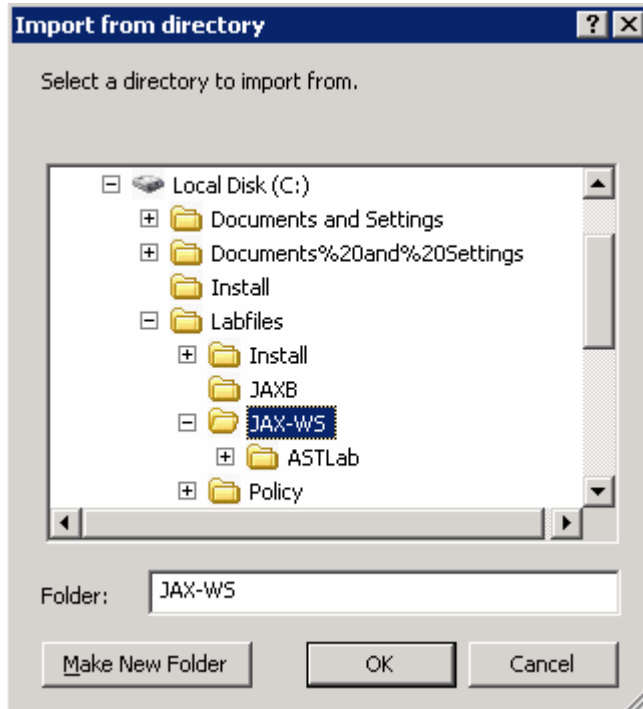
7. Import the client implementation code.
  - a. In the **AST Project Explorer** panel, expand **WSFPLab2Client**. This is the client project that was created by **AST** at the same time the Web Service was created. It contains the client side artifacts and files but does not implementation code. In the following steps you will import a template for the implementation code for a simple client and will make the changes needed for it to run against the Web Service you have installed.
  - b. Select **Java Resources** and then expand **src**. **Right click** on the package named **simple** and then click on **Import**.



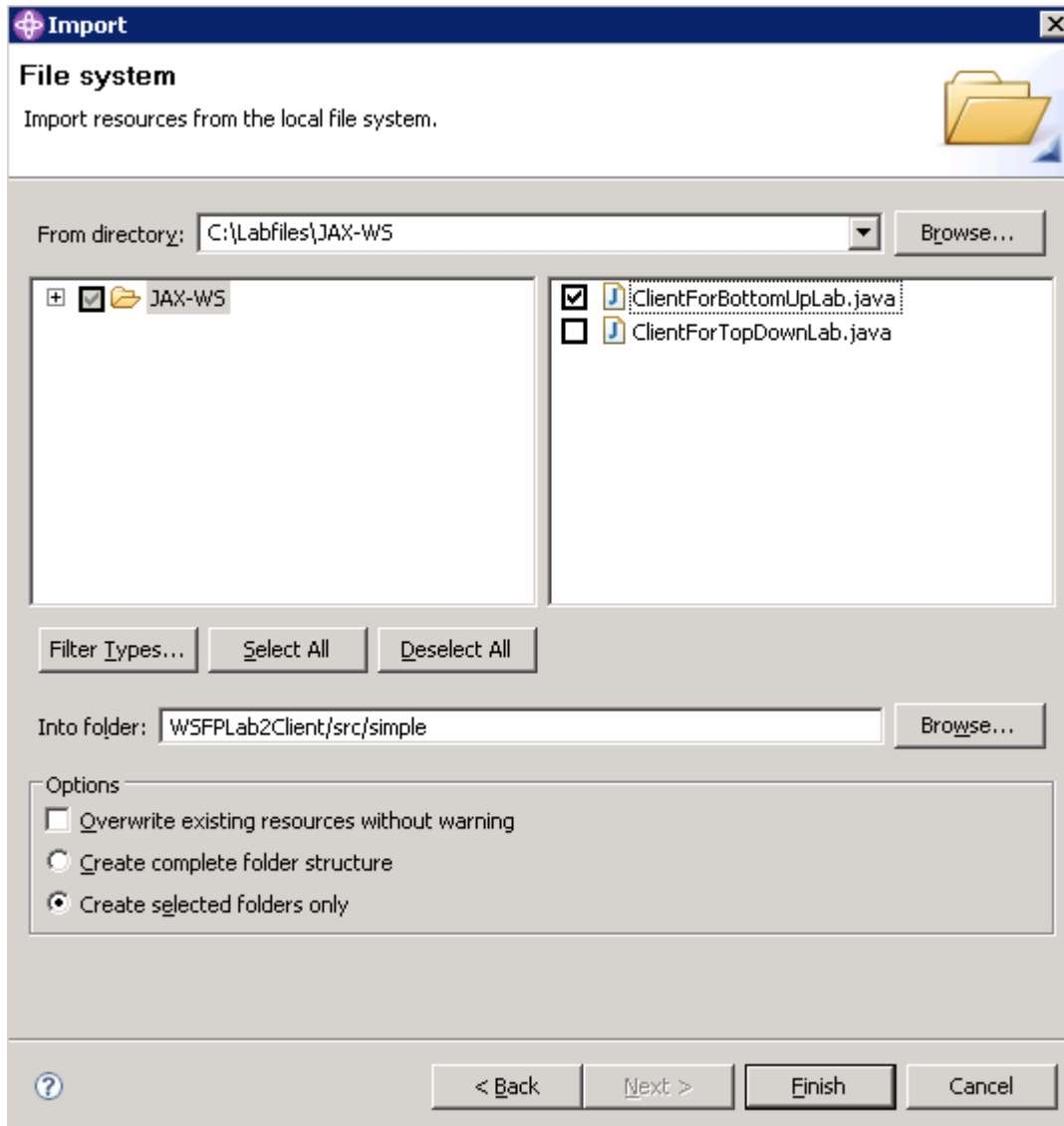
\_\_ c. On the **Import** panel select File System and click **Next**.



\_\_\_ d. Navigate to the **C:\Labfiles\JAX-WS** directory and click **OK**.



- \_\_\_ e. Select the check box on the left next to **simple**. Then on the right make sure to select only the check box for the **ClientForBottomUpLab.java**. Click **Finish**.



- \_\_\_ 8. Modify the client code - **ClientForBottomUpLab.java** is built from a template for a JAX-WS stand-alone client. It must be modified so that it will work for this lab.

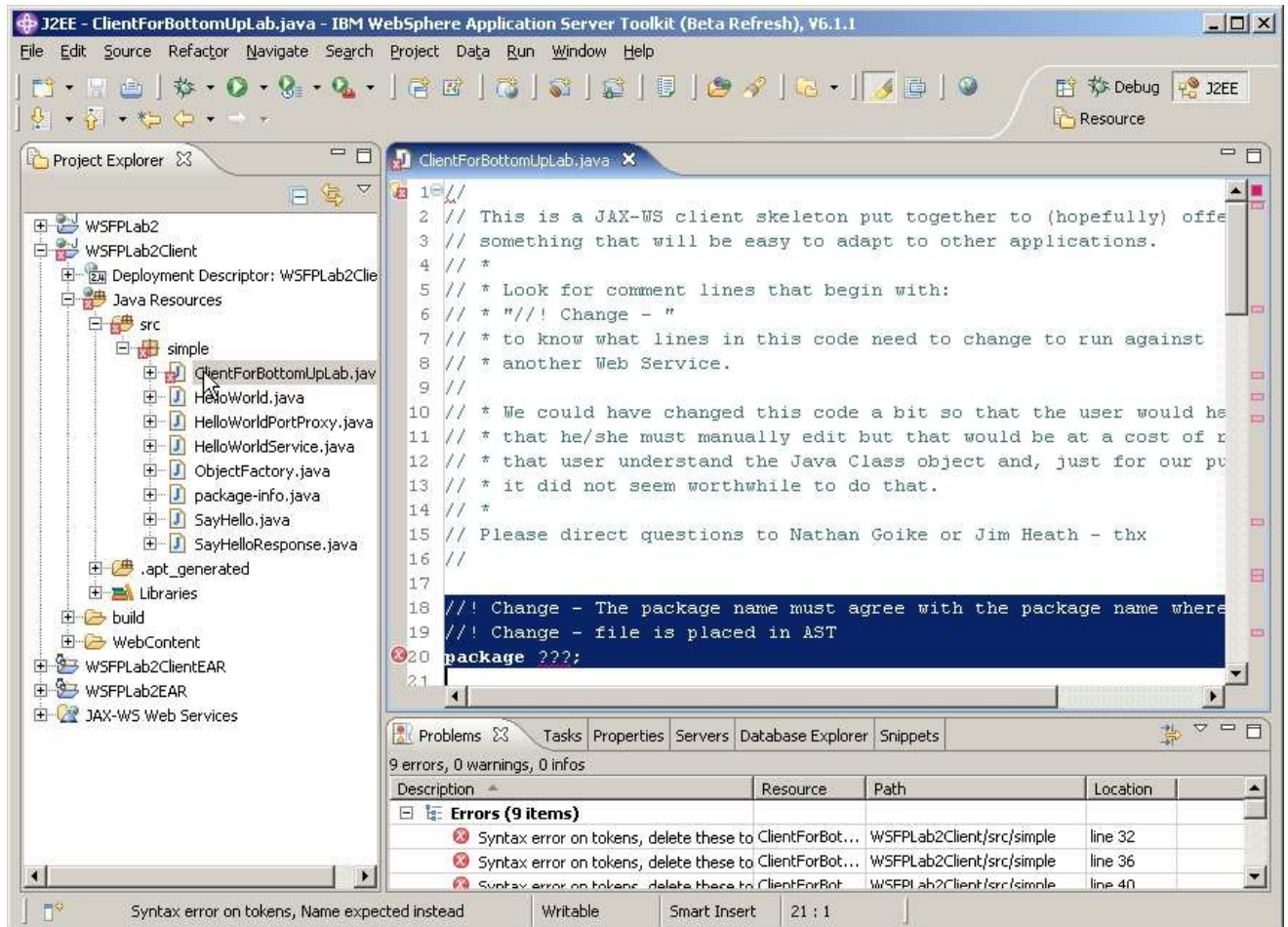
The lines that will require changes are preceded by one or more lines of the form:

```
///! Change - <instructions>
```

You will need to read the instructions and make the appropriate modifications.

Until you make the changes needed, the file will fail compilation – this is deliberate. Reference the wsdl that is in your browser window from the earlier step to identify the values to use for some of the changes you will need to make.

- \_\_\_ a. Some of the statements in **ClientForBottomUpLab.java** must be changed based on your environment. In the **AST**, expand the package **simple**. Double click on the **ClientForBottomUpLab.java** file contained within, to edit it.



**\_\_\_ 9. Change #1**

Change from:

```

!!! Change - The package name must agree with the package name where this
!!! Change - file is placed in AST
package ???;

```

To:

```

!!! Change - The package name must agree with the package name where this
!!! Change - file is placed in AST
package simple;

```

**\_\_\_ 10. Change #2**

Change from:

```

!!! Change - The package name must agree with the package name where this
!!! Change - file is placed in AST but use "/" as delimiter instead of "."
    private static final String PACKAGE_NAME = ???;

!!! Change - The value of the name attribute from
!!! Change - the <wsdl:service ... > element
    private static final String SERVICE_NAME = ???;

!!! Change - The value of the "name" attribute from
!!! Change - the <wsdl:port ... > element
    private static final String PORT_NAME = ???;

```

To:

```

!!! Change - The package name must agree with the package name where this
!!! Change - file is placed in AST but use "/" as delimiter instead of "."
    private static final String PACKAGE_NAME = "simple";

!!! Change - The value of the name attribute from
!!! Change - the <wsdl:service ... > element
    private static final String SERVICE_NAME = "HelloWorldService";

!!! Change - The value of the "name" attribute from
!!! Change - the <wsdl:port ... > element
    private static final String PORT_NAME = "HelloWorldPort";

```

**\_\_\_ 11. Change #3**

Change from:

```

//! Change - the URL to use to contact the Web Service
    private static final String
        URL_ENDPOINT = ???;

```

To ( Use the appropriate URL for your system ):

```

//! Change - the URL to use to contact the Web Service
    private static final String
        URL_ENDPOINT = "http://jim667:9080/WSFPLab2/HelloWorldService";

```

**\_\_\_ 12. Change #4**

Change from:

```

//! Change - Next real line is really like:
//! Change - <wsdl:portType.name> portTypeName =
//! Change -         service.getPort(QNAME_PORT, <wsdl:portType.name>.class);
    ??? portTypeName = service.getPort(QNAME_PORT, ???_CLASS);

```

To:

```

//! Change - Next real line is really like:
//! Change - <wsdl:portType.name> portTypeName =
//! Change -         service.getPort(QNAME_PORT, <wsdl:portType.name>.class);
    HelloWorld portTypeName = service.getPort(QNAME_PORT, HelloWorld.class);

```

**\_\_\_ 13. Change #5**

Change from:

```

//! Change - this line so that it is:
//! Change - String replyString = portTypeName.<operationName>(args);
//! Change - where you need to change <operationName> to the operation to invoke
//! Change - and "args" will be arguments to that operation
    String replyString = portTypeName.???;

```

To:

```

//! Change - this line so that it is:
//! Change - String replyString = portTypeName.<operationName>(args);
//! Change - where you need to change <operationName> to the operation to invoke
//! Change - and "args" will be arguments to that operation
    String replyString = portTypeName.sayHello("Welcome to Web Services");

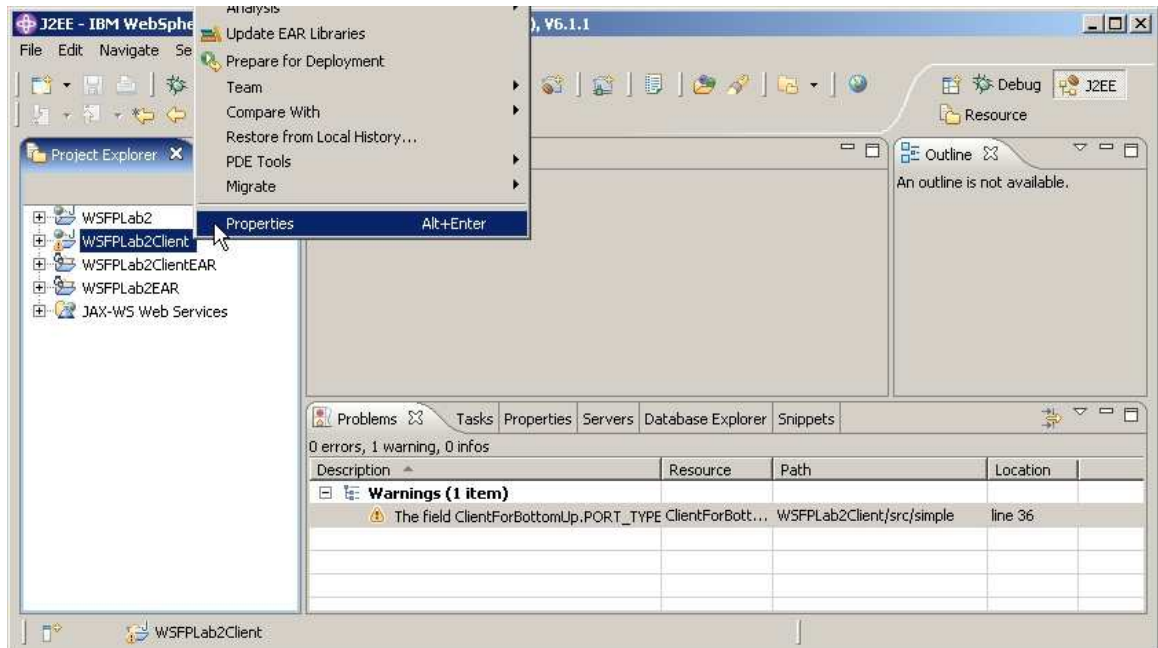
```



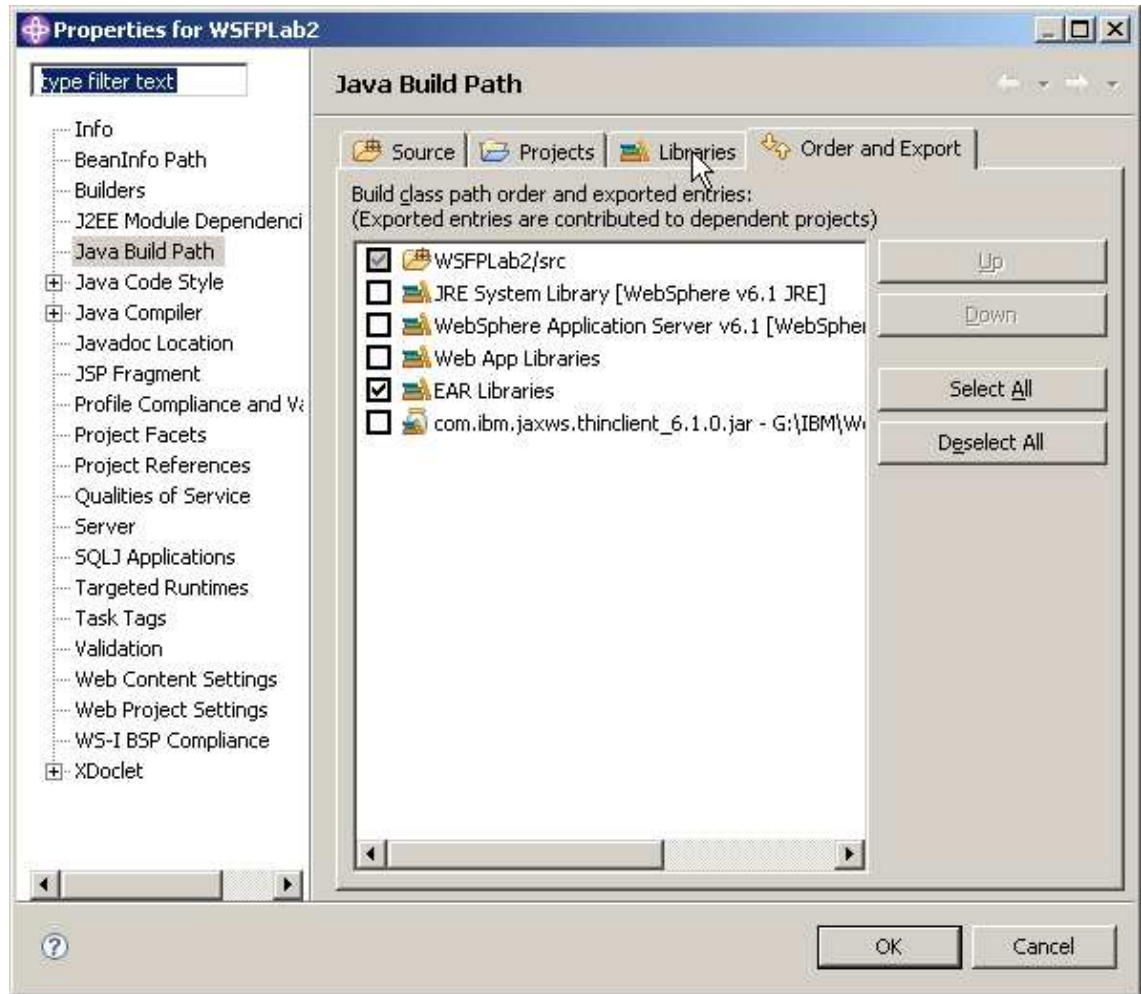
\_\_\_ 14. Save your changes with **Ctrl + S**. All the errors should be removed from the problems tab.

\_\_\_ 15. Changing the path.

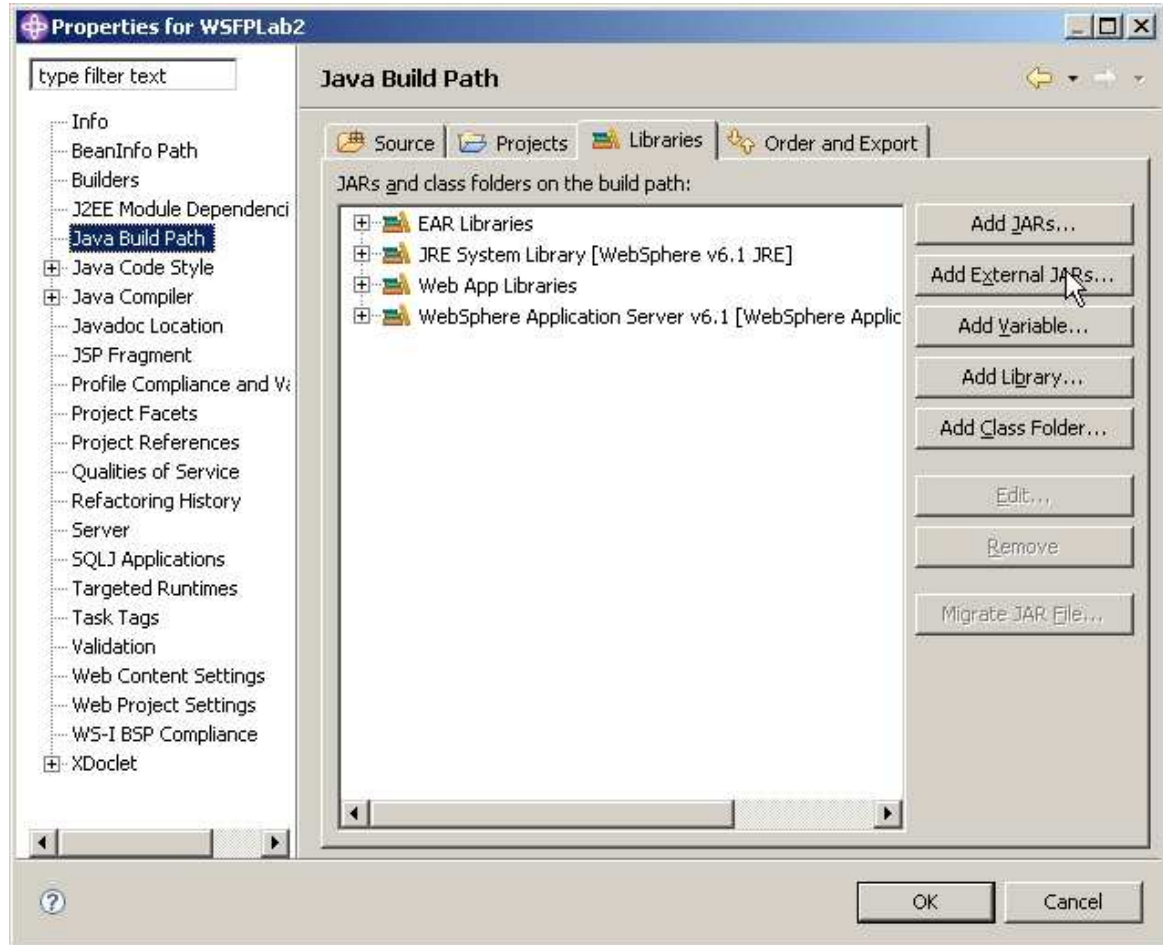
\_\_\_ a. Before you can run the client code, you must change the build path being used. In the Project Explorer **right-click** on WSFPLab2Client and select **Properties**.



\_\_ b. Select **Java Build Path**. Then select the **Libraries** tab.



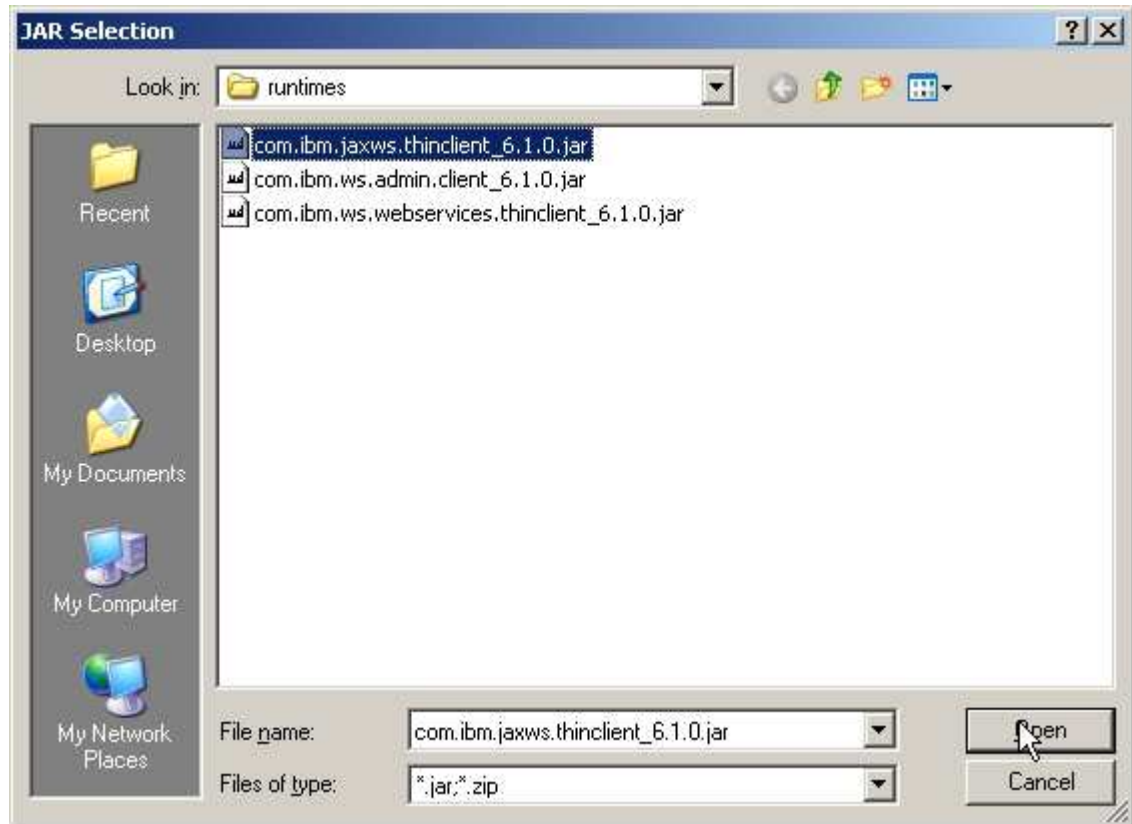
\_\_ c. Select the button for **Add External JARs**.



\_\_ d. Select the IBM WebSphere Thin Client for Web Services jar file by navigating to:

**C:\WebSphere\Appserver\Runtimes**

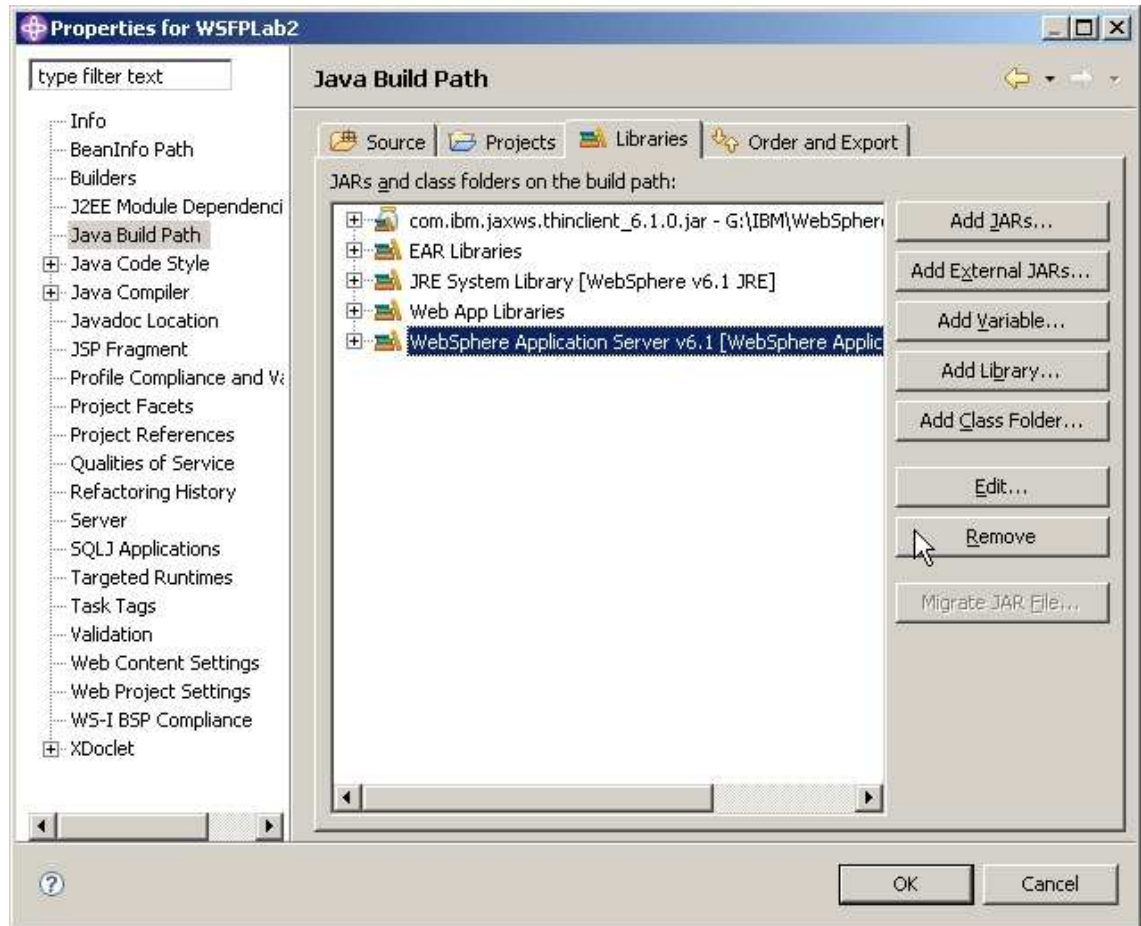
and selecting the file named **com.ibm.jaxws.thinclient\_6.1.0.jar** and clicking **Open**.



\_\_\_ e. In the Libraries list, select the entry named:

WebSphere Application Server v6.1[WebSphere Application Server v6.1stub]

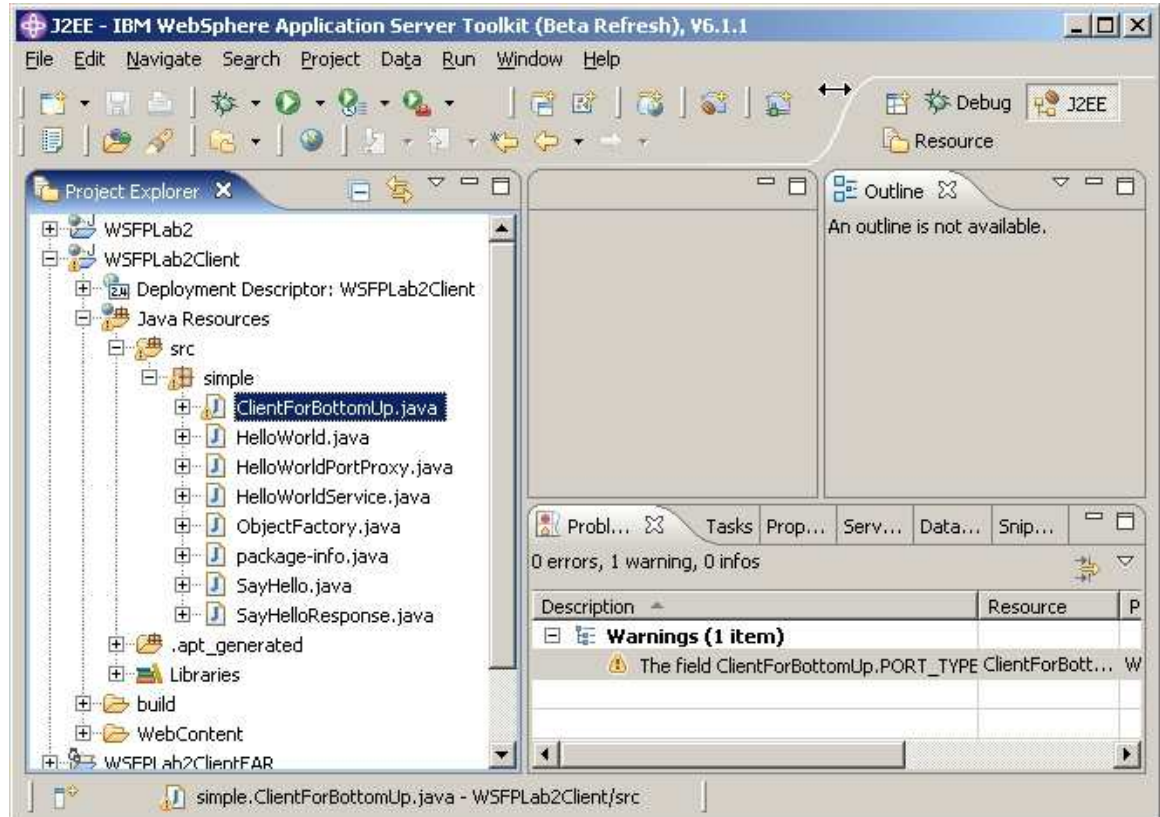
and click **Remove**.



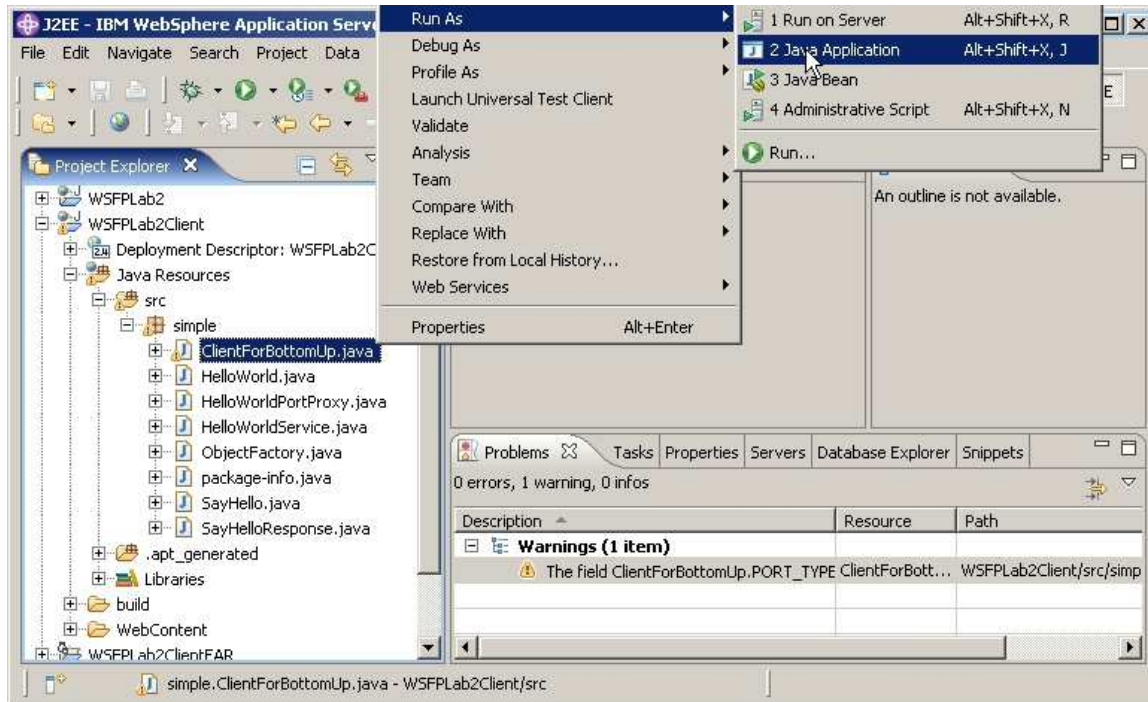
\_\_\_ f. Click **OK**.

\_\_\_ 16. Running the client code.

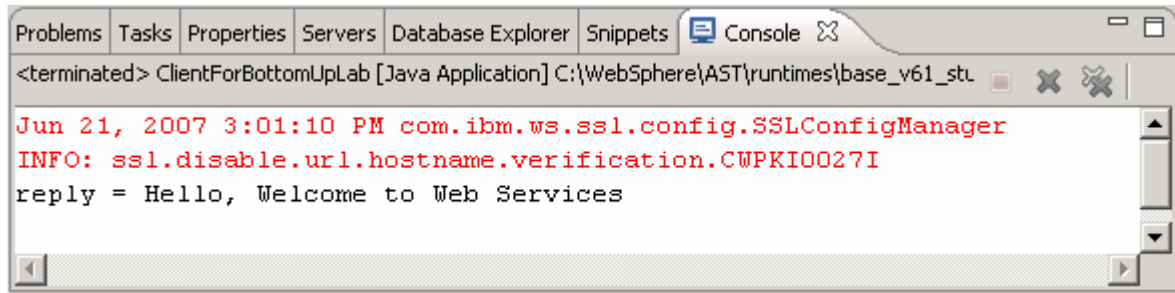
- \_\_\_ a. Expand **WSFPLab2Client** from the **Project Explorer** in the **AST**, expand **Java Resources**, expand **src**, expand **simple** and select **ClientForBottomUp.java**



\_\_ b. Expand right-click **ClientForBottomUp.java** and select **Run As** and click **Java Application**.



\_\_ c. The following output should appear in the console of the AST.



The screenshot shows an Eclipse IDE console window. The title bar includes tabs for Problems, Tasks, Properties, Servers, Database Explorer, Snippets, and Console. The console content is as follows:

```
<terminated> ClientForBottomUpLab [Java Application] C:\WebSphere\AST\runtimes\base_v61_stl  
Jun 21, 2007 3:01:10 PM com.ibm.ws.ssl.config.SSLConfigManager  
INFO: ssl.disable.url.hostname.verification.CWPKI0027I  
reply = Hello, Welcome to Web Services
```



## Summary

In this lab, you took a bottom-up development approach to build a Web Service starting with a Java file containing code that will be the basis for implementing the Web service. You performed the following steps to do this:

1. Created a Dynamic Web Project in the AST and imported the implementation code for the Web Service.
2. Created a Web Service From the implementation code and exported the EAR file containing that Web Service
3. Installed the EAR file, started the Web Service application, and performed simple verification
4. Developed a Web Services client for the Web Service, ran the client code and interacted with the Web Service you created

This page is left intentionally blank.