IBM WEBSPHERE 5.0 Skills Transfer - LAB EXERCISE

Developing Web Services - Part 2

What This Exercise is About

WebSphere Studio Application Developer has tools for working with each part of the Web Services programming model. For the Service Provider, there is a wizard for turning applications into Web Services. For the Service Requester, there is a wizard for creating client applications to access Web Services. Finally, for the Service Registry, there is a Web Services explorer for viewing UDDI public or private registries. This exercise will feature some of the Web Service tools available in WebSphere Studio Application Developer.

User Requirement

User must have IBM DB2 Universal Database (version 7.2) and IBM WebSphere Studio Application Developer Version 5.0.1 installed on a Windows 2000 workstation with Service Pack 3. Internet Explorer version 5.5 is also required. The WebSphere Studio Application Developer should be installed at C:\Program Files\IBM\WebSphere Studio. To complete all parts of the lab, the WebSphere Application Server Version 5.0.1 and WebSphere Network Development Version 5.0.1 packages are required. The lab source files (LabFiles50.zip) must be extracted to the root directory (e.g. C:\). Experience with previous versions of WebSphere Studio Application Developer and the J2EE programming model are also required. Throughout this lab, a userid profile named wsdemo with a password of wsdemo1 is assumed to be created on the system.

NOTE: This lab exercise makes the following assumptions: DB2 is installed at C:\SQLLIB; WebSphere Application Server Version 5.0.1 is installed at: C:\WebSphere\AppServer; WebSphere Network Deployment Version 5.0.1 is installed at: C:\WebSphere\DeploymentManager; WebSphere Studio Application Developer is installed at C:\Program Files\IBM\WebSphere Studio; and the LabFiles50 is extracted to C:\. Throughout this lab, batch files are used to simplify some selected operations. If the software is installed in different locations other than those specified above, you can modify the batch files, however, extensive testing has not been completed to verify all possible combinations. To accommodate different installed locations of the required products above, a batch file called setupVARs.bat has been created. It is located in \LabFiles50. You may update this file according to the instructions contained therein - specifying the drive letters and/or paths of the installed products. All subsequent batch files used throughout this lab calls the setupVARs batch file first. Effectively, you have a single location to make updates to product installation drives and directories.

What You Should Be Able to Do

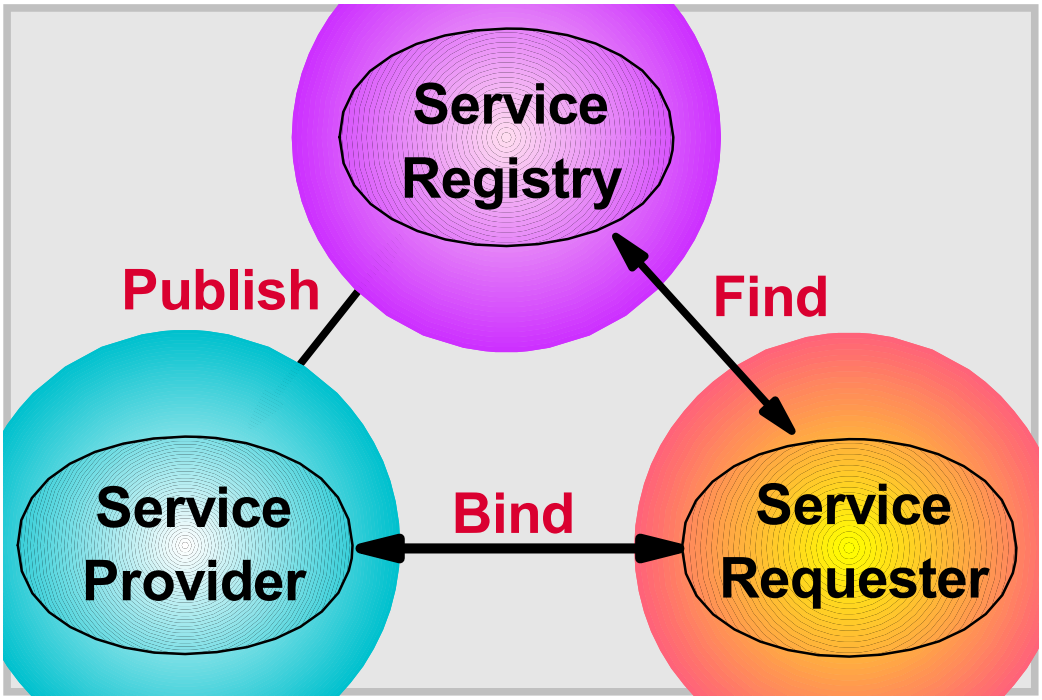
In this lab the focus is on working with the tools for creating and working with Web Services.

By acting as the Service Provider, you will turn part of an existing application into a Web Service. First you will add a Session EJB bean to your application which will communicate with an EJB in the MyBank application. After implementing the session EJB as a Web Service, you will publish the Web Service in the IBM Unit Test UDDI Registry. Using the Web Services explorer from within WebSphere Application Studio Developer, you will discover the service and write a sample application that will invoke the Web Service.

Background Information

There are three main roles and functions in the Web Services model. The Service Provider provides applications as Web Services and publishes them to the Service Registry. They accept requests, processing data, and return the result to Service Requesters. The Service Requester makes these requests (of Service Providers) and receives the result back. The Service Registry is used by both the Service Provider and the Service Requester. The Service Registry lists the different services available by Service Providers and the Service Requester queries the Service Registry for the services available by Service Providers. All of this is done in a loosely-coupled environment as the technical implementation between the different players is based solely on Open Standards (SOAP, XML, WSDL, UDDI), avoiding rigid proprietary implementations. Each role in the Web Services model focuses their efforts on their part of the model, shielded from the other's by the Open Standards.

The following graphic further illustrates the roles and functions in the Web Services model.



Introduction

During Part One, you will create a Web Service. You will be performing the role of the Service Provider. First you will create a stateless session enterprise java bean that will be turned into a Web Service. The stateless session bean contains a method called CreateAccount which uses the Account entity bean to create an Account in the MyBank Application. You also update the EJB deployment descriptor and complete the EJB to RDB mapping and generate the deployment code during this part. The Web Service wizard guides you through and creates all of the necessary artifacts to make this method available as a Web Service.

For Part Two, you will publish the Web Service to the Unit Test UDDI Registry all within the workbench environment.

Part Three illustrates how you can create a sample application that can serve as a Web Service consumer. The creation of the sample application can be created directly when running the Web Services wizard, however, you did not select this option in Part 4. The creation of the sample application is demonstrated within the workbench environment.

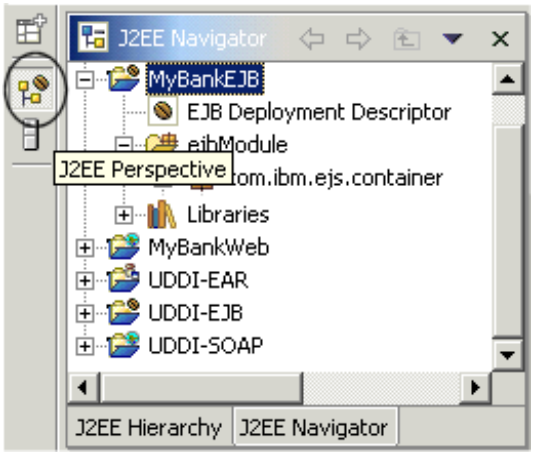
In Part Four, you run the sample application and verify proper execution of the Web Service by examining the contents of the DB2 database.

Exercise Instructions

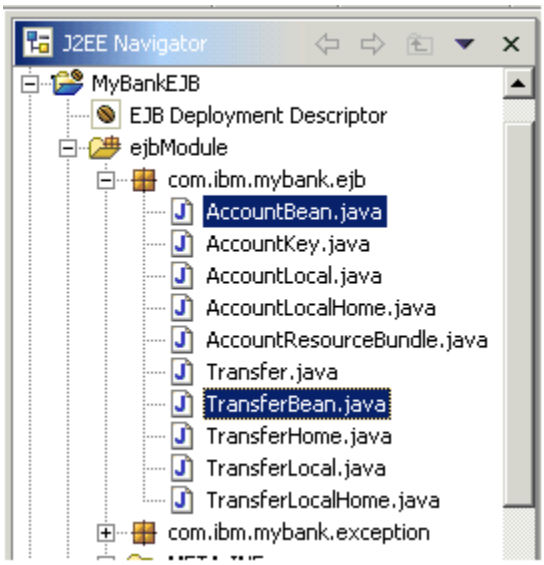
Part One: Create a Web Service

__1. Familiarize yourself with the J2EE artifacts that are part of the MyBank application.

__ a. Switch perspectives to the J2EE Perspective. You can select the J2EE Perspective by selecting the correct icon on the toolbar on the left side of the WebSphere Studio Application Developer window. Alternatively, select **Window > Open Perspective > J2EE**. Ensure you are on the **J2EE Navigator** tab of the J2EE perspective.



__ b. There is an Account entity bean and a Transfer stateless session bean. To familiarize yourself with these artifacts, expand **MyBankEJB > ejbModule > com.ibm.mybank.ejb** package. You will see the Account and Transfer beans.



__ c. After you are through viewing the artifacts, collapse **MyBankEJB** by clicking the '-' to the left of the MyBankEJB module.

__ d. Next you will examine the WSDL file for the transferFunds method on the Transfer stateless session bean. This has already been implemented as a Web Service.

__ e. Expand **MyBankWeb > Web Content > wsdl > com > ibm > mybank > ejb**.

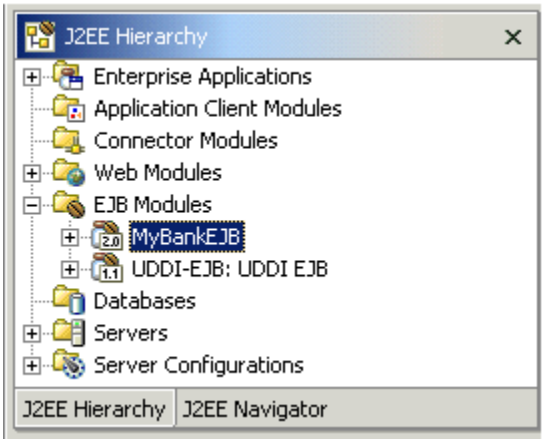
__ f. There are four files that describe this Web Service. The TransferService.wsdl file defines the mechanics and interface of the Web Service. This information will be programmatically published in the UDDI Registry later on in this lab. The TransferBinding.wsdl file contains additional details about the service like the request and response formats. The Transfer.wsdl file describes the mappings of the service request flowing across the wire. NOTE: While the concepts of UDDI and WSDL are very closely related, the UDDI specification says nothing about WSDL. WSDL (an XML-based Web Services Description Language) plays a key role in describing Web Services.

> **NOTE:** The EJB binding is proprietary and is not consumed or used with WebSphere Studio Application Developer. This EJB binding file, however, is emitted and used in the WebSphere Studio Application Developer, Integration Edition. This is generated for high reusability. Also, if you are creating a Web Service from a Java bean (e.g. the Java bean exists), a Java binding would be emitted.

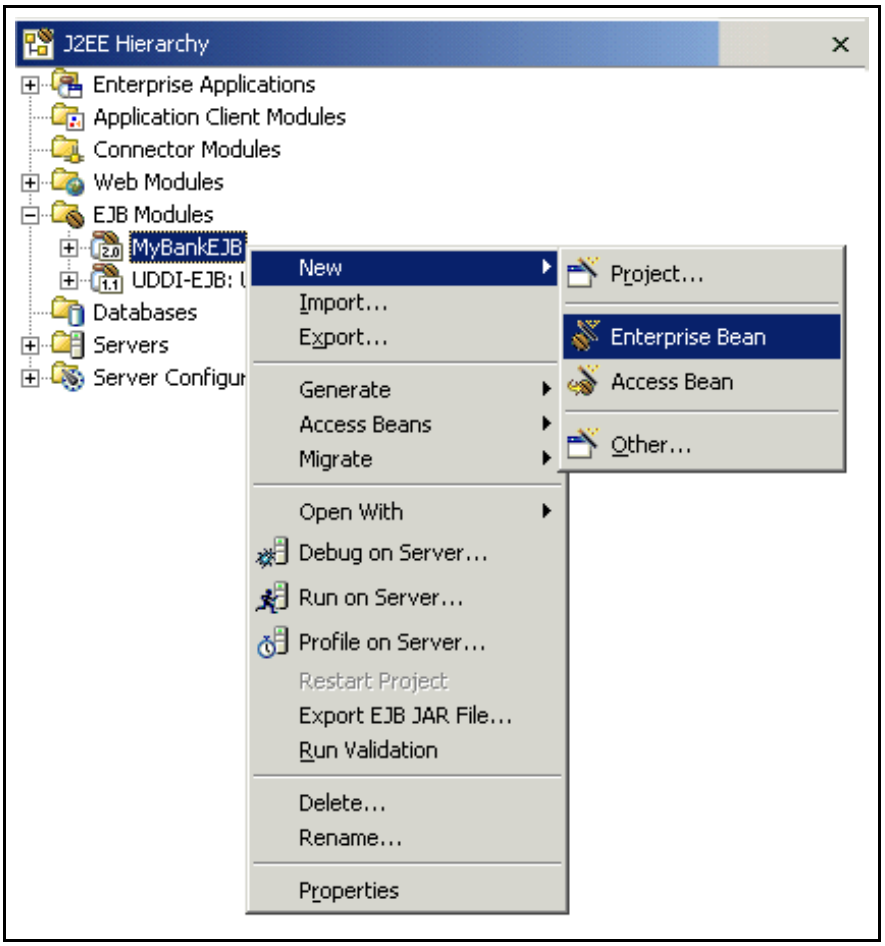
__2. Create a stateless session EJB that will be implemented as a Web Service.

__ a. A session bean is used in this example, however, Web Services can be generated from Javabeans and other artifacts as well. In this lab example, you will deploy a CreateAccount method as a Web Service. You cannot create Web Services directly from entity beans, therefore it is necessary to create a session bean that communicates with the Account entity bean. Both stateless and stateful session beans are able to be implemented as Web Services.

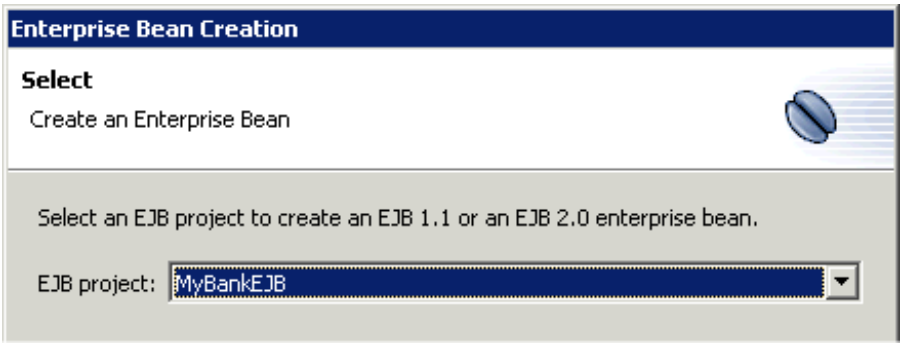
__ b. In the J2EE Perspective, select the **J2EE Hierarchy** view. **Expand EJB Modules.**



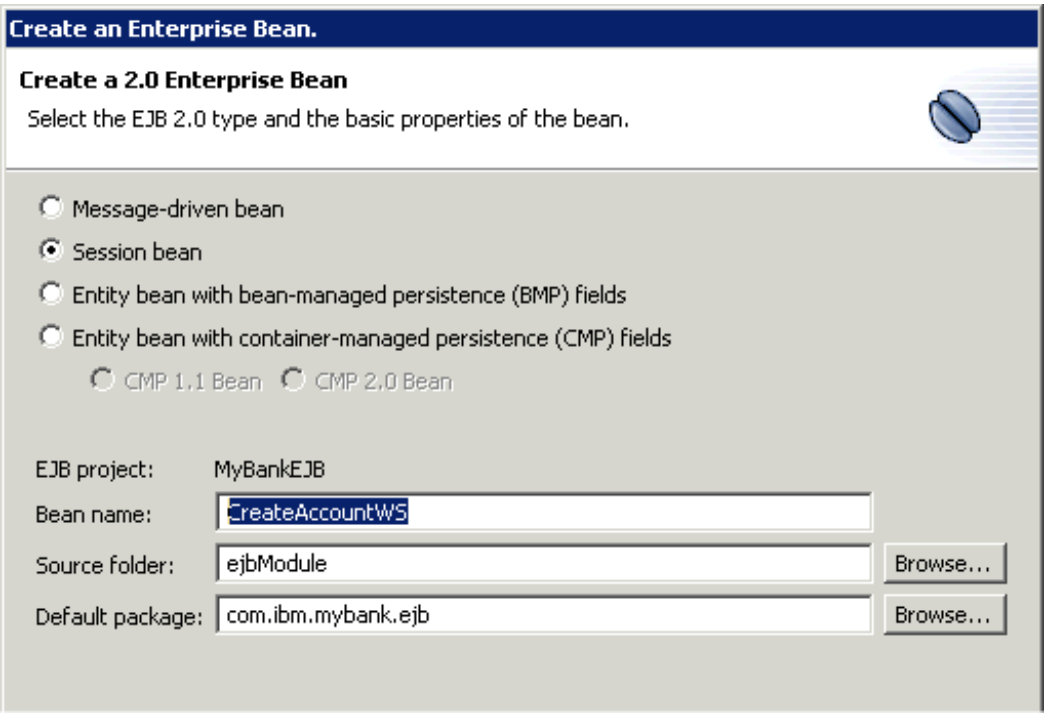
__ c. Select the **MyBankEJB** module. Right-click on **MyBankEJB** and select **New** > **Enterprise Bean**.



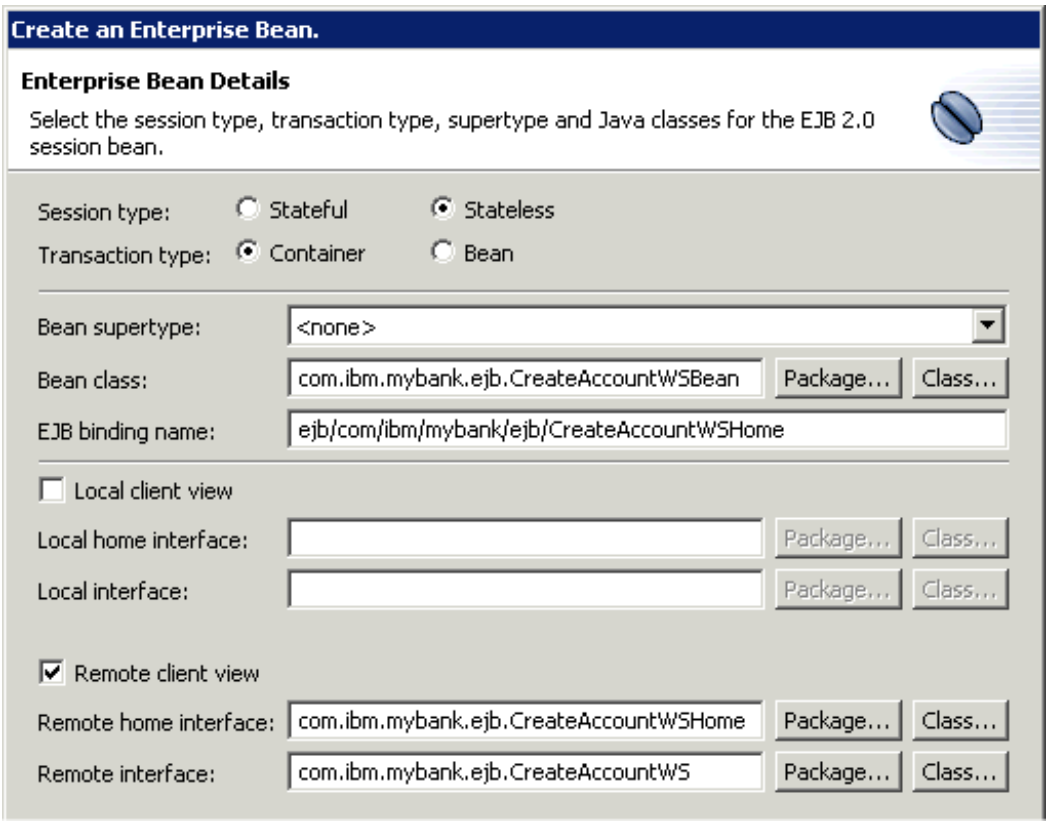
__ d. On the Enterprise Bean Creation panel, ensure the **MyBankEJB** EJB Project is selected. Click **Next**.



__ e. On the second panel, Create a 2.0 Enterprise Bean, type **CreateAccountWS** for the Bean name. Accept the remaining defaults of Session bean. Click **Next**.



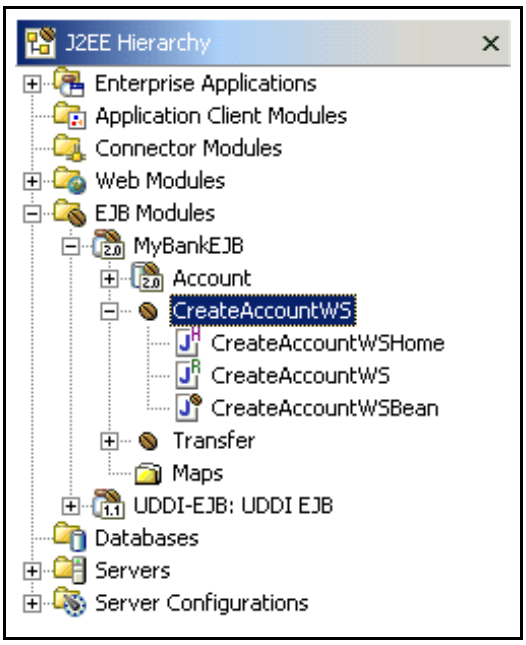
__ f. On the Enterprise Bean Details panel, inspect and accept the defaults. You are creating a stateless session bean. Click **Next**.



__ g. On the EJB Java Class Details panel, click **Finish**. You will add some imports by typing them in the actual class file momentarily.

__3. Update the CreateAccountWS Enterprise Bean.

__ a. Expand **EJB Modules > MyBankEJB** > **CreateAccountWS**. The following class files (home interface, remote interface, and the bean) have been created for you under MyBankEJB and CreateAccountWS.



__ b. Double-click the **CreateAccountWSBean.java** file in the J2EE Hierarchy view. This will open the CreateAccountWSBean.java file in the editor. Type the following import statements below the package statement (which is the first line in the file). To save time, you can copy the import statements from \LabFiles50\WebServices\WebServices_snippet1.txt.

```
import com.ibm.mybank.ejb.*;
import java.text.NumberFormat;
import javax.ejb.*;
import javax.naming.*;
```

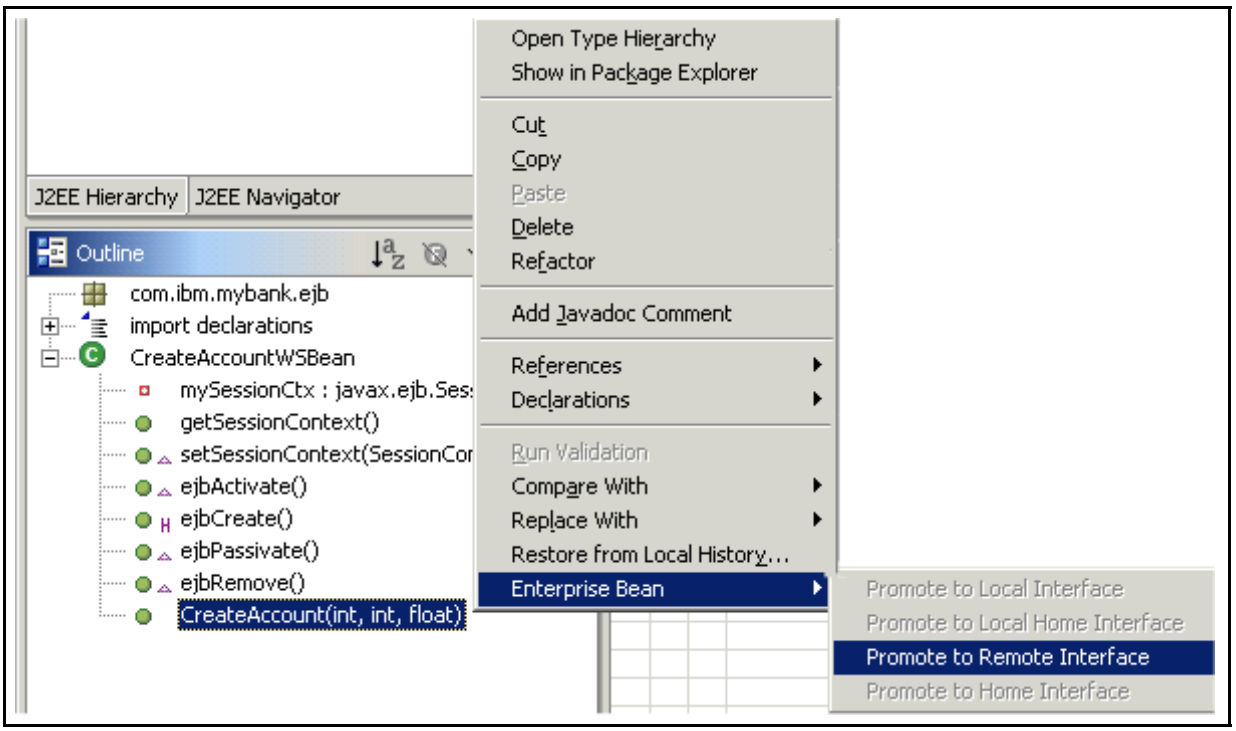
__ c. Add the **CreateAccount** method to the CreateAccountWSBean.java inside the class definition. To save time, you copy the code from \LabFiles50\WebServices\WebServices_snippet2.txt. **Copy this code in after the public void ejbRemove() method and before the last closing bracket**. This is the method that you will implement and make available as a Web Service.

```
public String CreateAccount(
int acctNumber,
int type,
float balance) {
AccountLocalHome accountHome;
String message = null;

try {
```

```
                NumberFormat dollarFormat =
                     NumberFormat.getCurrencyInstance();
                InitialContext initCtx = new
                     InitialContext();
                Object objref =
                initCtx.lookup("java:comp/env/bank/Account");

                accountHome = (AccountLocalHome) objref;

                AccountKey accountNumber = new
                     AccountKey(acctNumber);
                // Create the EJB.
                AccountLocal account =
                     accountHome.create(accountNumber, type,
                     balance);

                message =
                     "Account "
                          + acctNumber
                          + " with a balance of "
                          + dollarFormat.format(balance)
                          + " successfully created.";
                }
           // Determine cause of failure.
           catch (NamingException e) {
                System.out.println(e.getMessage());
                e.printStackTrace();
           } catch (DuplicateKeyException e) {
                System.out.println(e.getMessage());
                e.printStackTrace();
           } catch (Exception e) {
                System.out.println(e.getMessage());
                e.printStackTrace();
           } finally {
                return message;
           }
      }
```

__ d. Next, **Save** CreateAccountWSBean.java (type **Ctrl+S**).

__ e. The Outline view should be visible in the lower left corner of the WebSphere Studio Application Developer workspace. Select the **CreateAccount** method, right-click and select **Enterprise Bean** > **Promote to Remote Interface**. This step is necessary to make this method available.

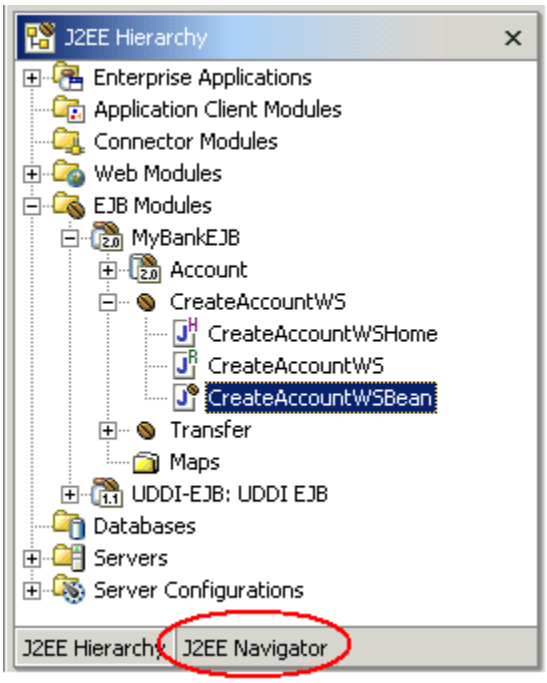


__ f. After promoting the method to the remote interface, **close** the file by clicking on the X on the right.

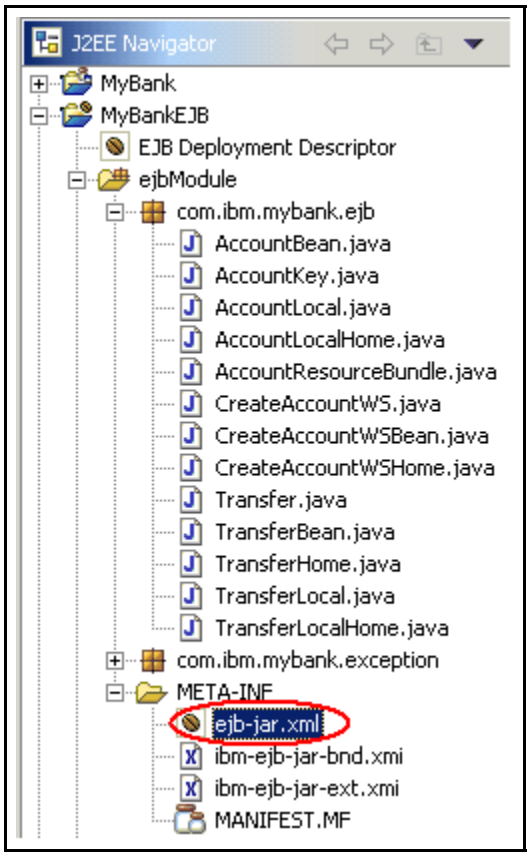
__4. Update the EJB Deployment Descriptor.

__ a. In preparation for testing the Web Service we have created, you need to edit and modify the EJB deployment descriptor file called ejb-jar.xml. This is necessary so EJBs can communicate with our DB2 database.

__ b. Select the **J2EE Navigator view** in the J2EE Perspective.

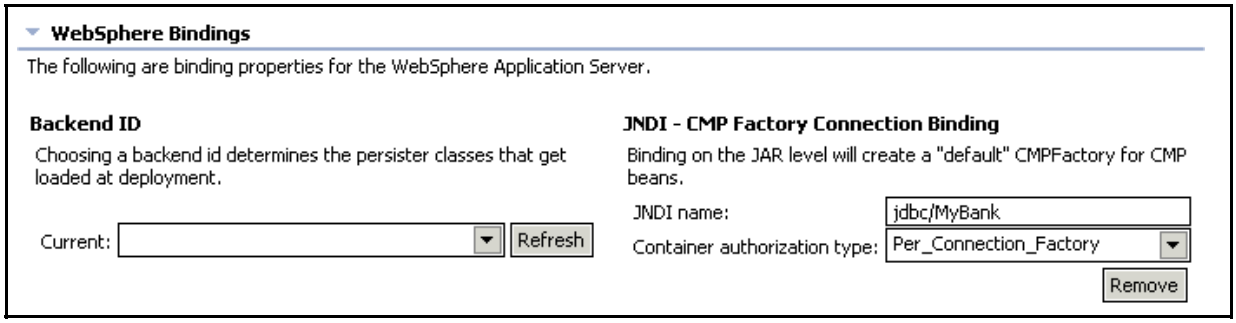


__ c. Expand **MyBankEJB > ejbModule > META-INF**.



__ d. Open the **ejb-jar.xml** file in an editor. Simply double click the file name.

__ e. In the Overview tab, scroll all the way to the bottom and update the WebSphere Bindings section. For the JNDI - CMP Factory Connection Binding, type **jdbc/MyBank** for the JNDI name. Select **Per_Connection_Factory** from the dropdown for the Container authorization type.



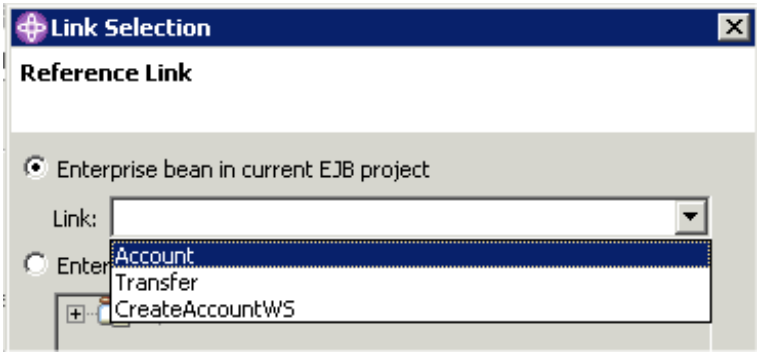
__ f. Select the **References** tab. Select the **CreateAccountWS** EJB and click **Add**.

__ g. Click **EJB local reference** when the wizard pops up.

__ h. Click **Next**.

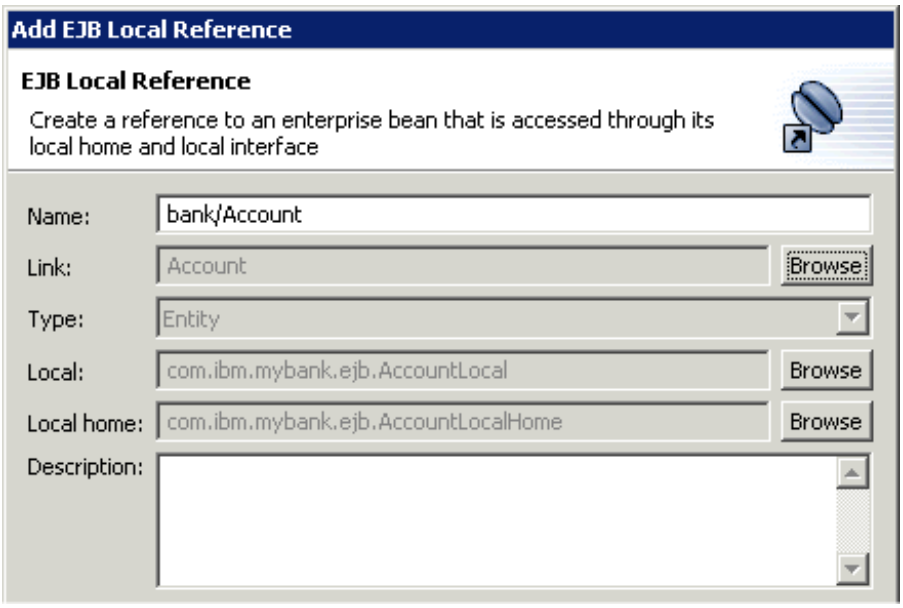
__ i. For the Name, type **bank/Account**. Click **Browse...** to the right of the **Link** field.

__ j. Select the **Account** in the Enterprise bean in current EJB project dropdown.



__ k. Click **OK**.

__ l. The remaining values are now filled in as shown below. Click **Finish**.

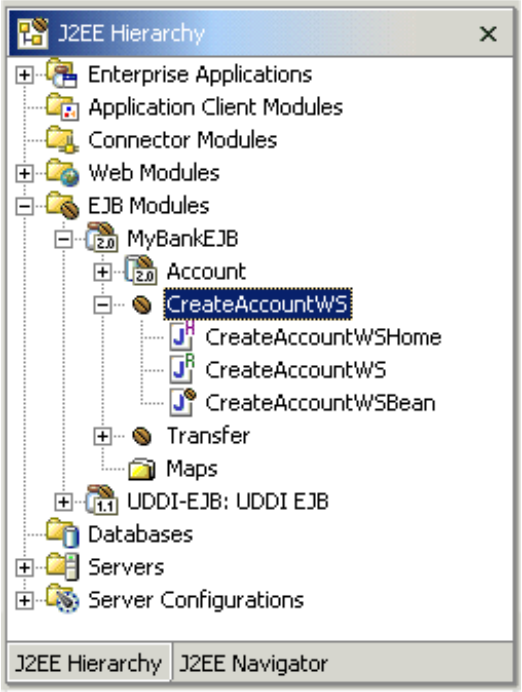


__ m. Save the EJB Deployment Descriptor file by typing **Ctrl+S**.

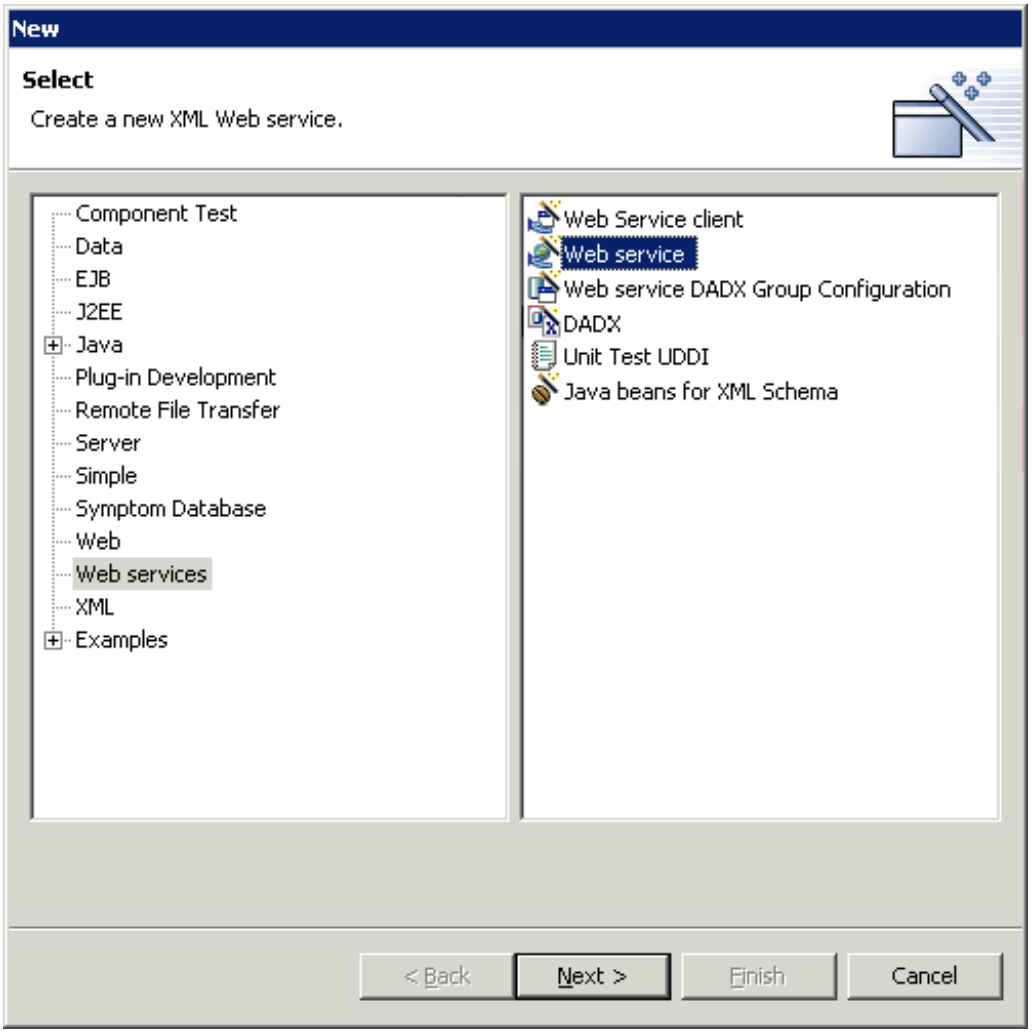
__ n. Next, **Close** the EJB Deployment Descriptor file by clicking on the X on the right.

__5. Next, you will create the Web Service.

__ a. In the J2EE Perspective, select the **J2EE Hierarchy** tab. Expand the **EJB Modules > MyBankEJB** and select **CreateAccountWS** (as shown below). Note: the EJB artifacts may already be expanded.

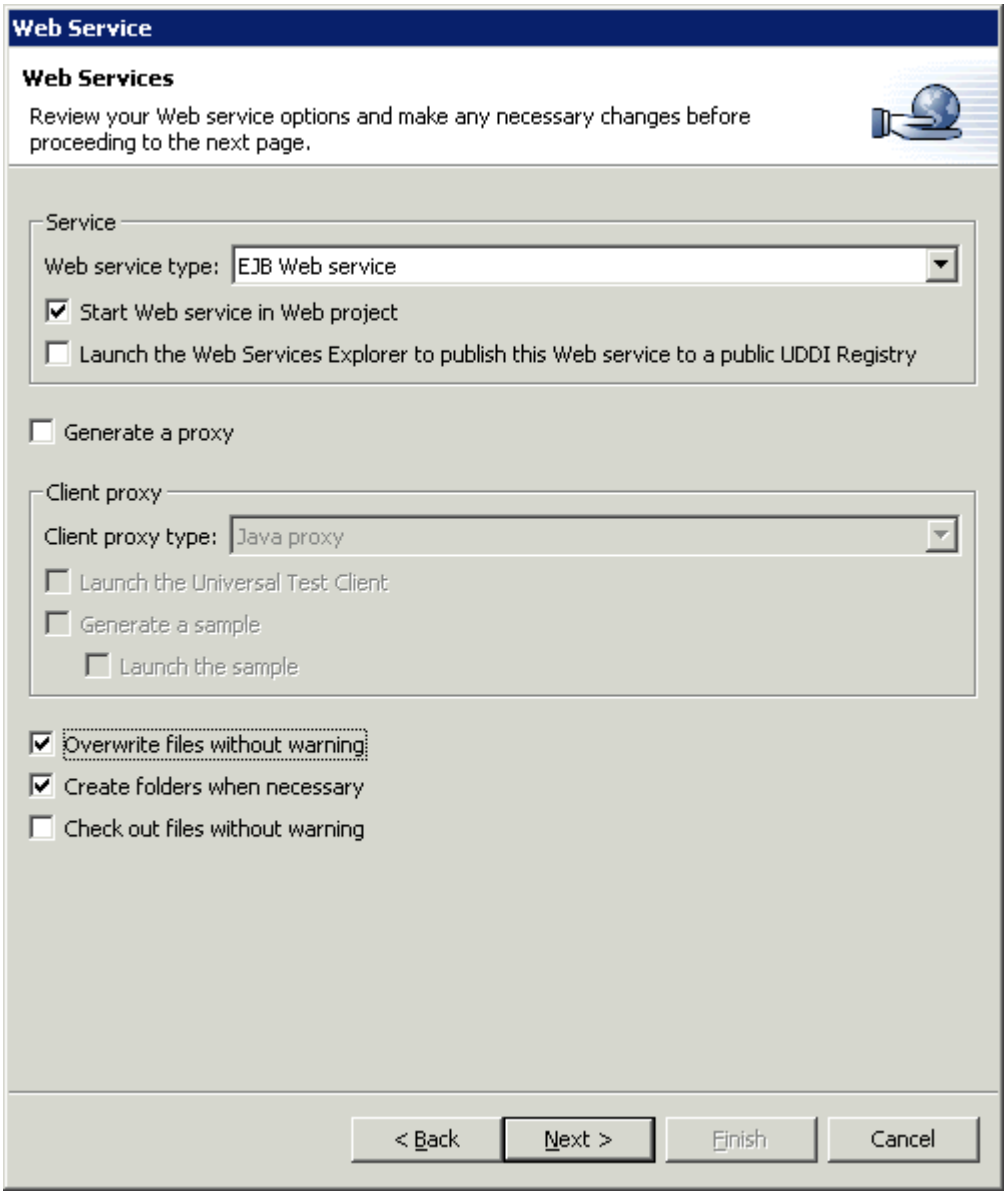


__ b. Select **File** > **New** > **Other...**. Then select the **Web Services** wizard in the left panel and **Web Service** in the right Panel.



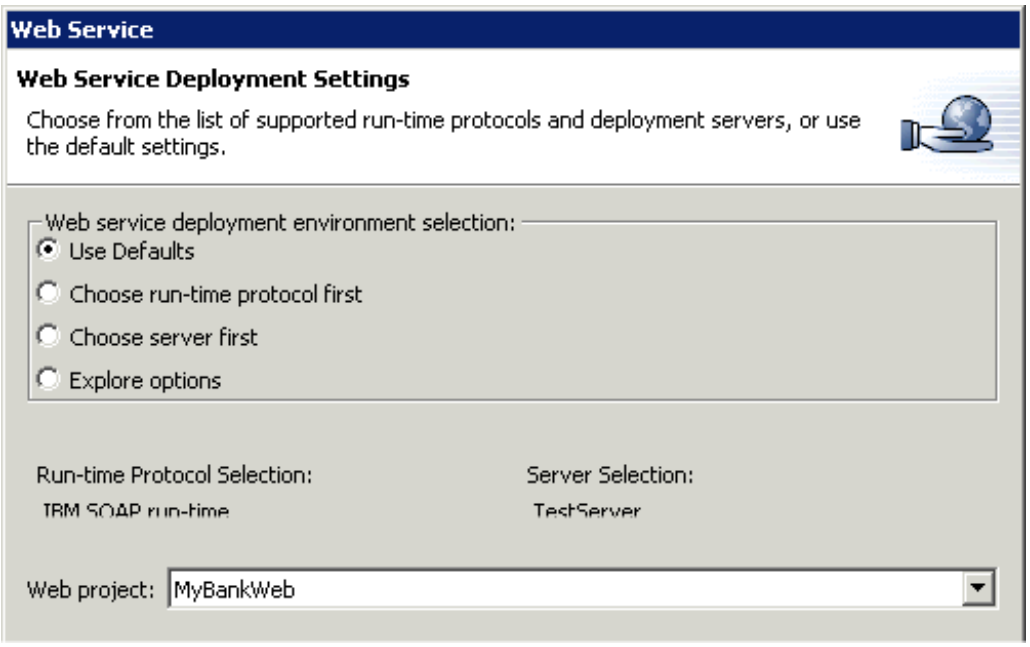
__ c. Click **Next**.

__ d. From the Web Service Type drop-down, select **EJB Web Service**. Ensure the **Start Web Service in Web project**, **Overwrite files without warning,** and **Create Folders when necessary** checkboxes are selected. The resulting panel should look like the following.



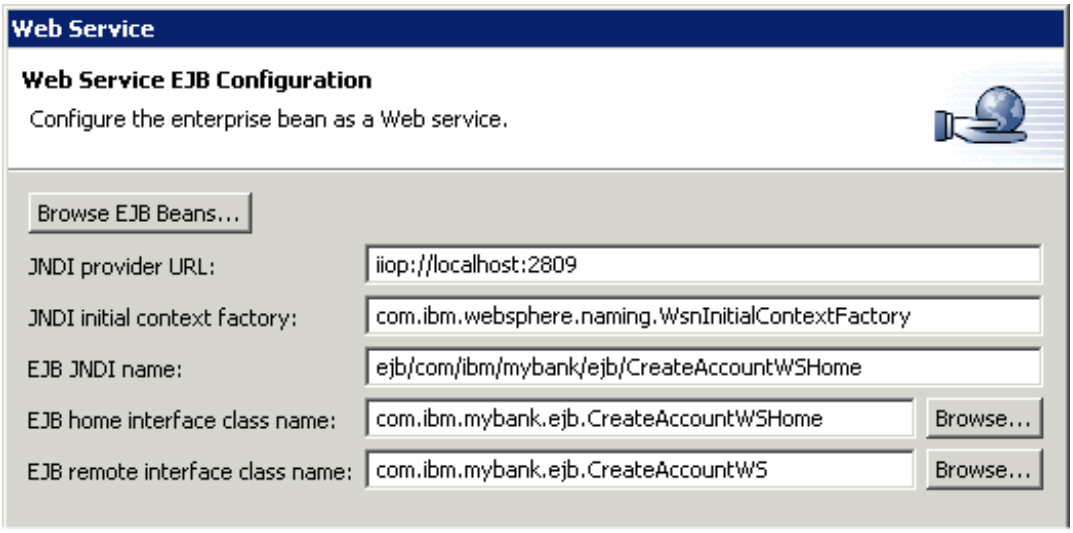
__ e. Click **Next**.

__ f. Accept the defaults on the Web Service Deployment Settings panel. This allows you to select a runtime protocol and the server to run on. The Web project is where the Web Services artifacts are created.



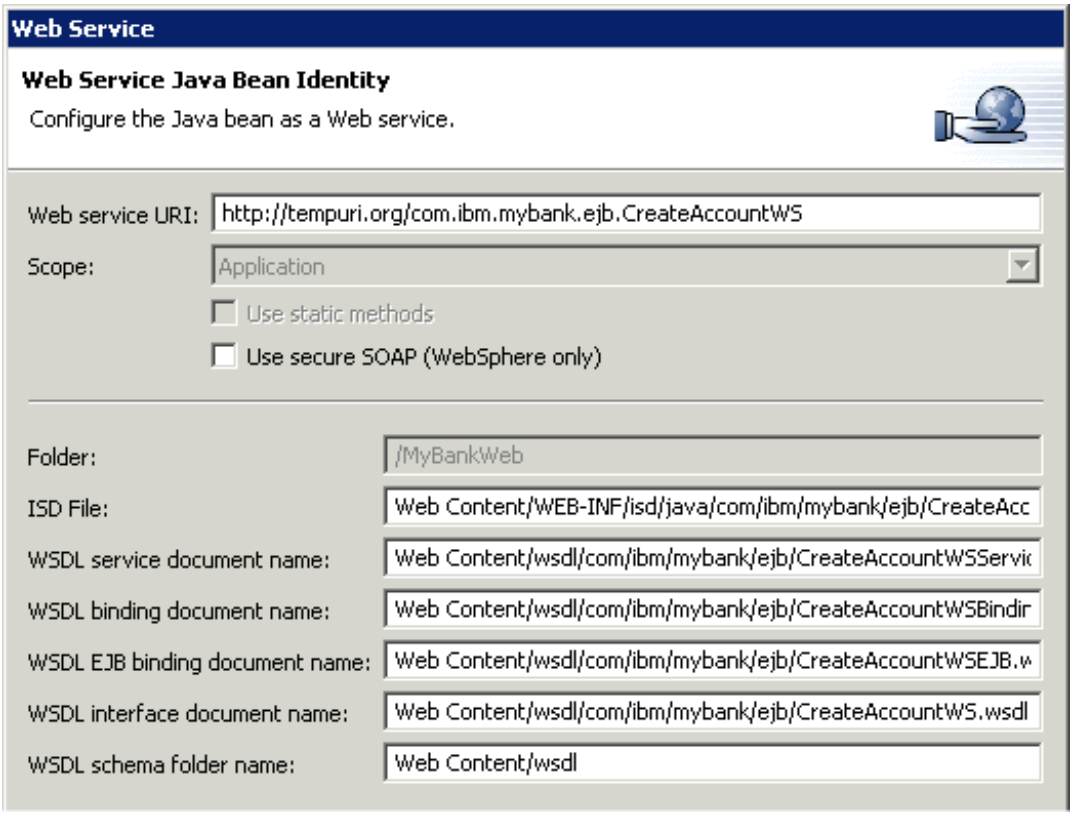
__ g. Click **Next**.

__ h. On the Web Service EJB Configuration panel, accept the defaults.



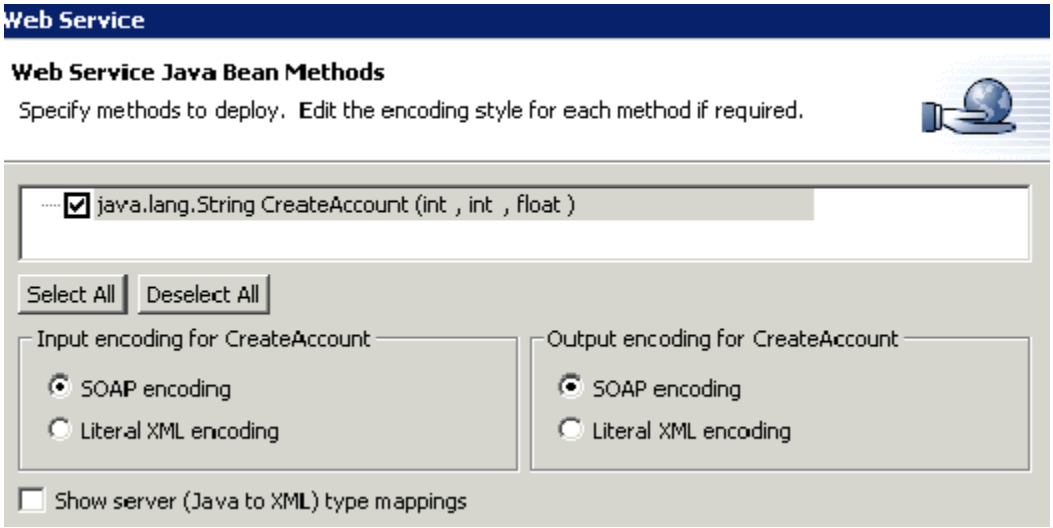
__ i. Click **Next**.

__ j. On the Web Service Java Bean Identity panel, accept the defaults.



__ k. Click **Next**.

__ l. On the Web Service Java Bean Methods, accept the defaults. Here you can see the CreateAccount method that is being deployed as the Web Service.

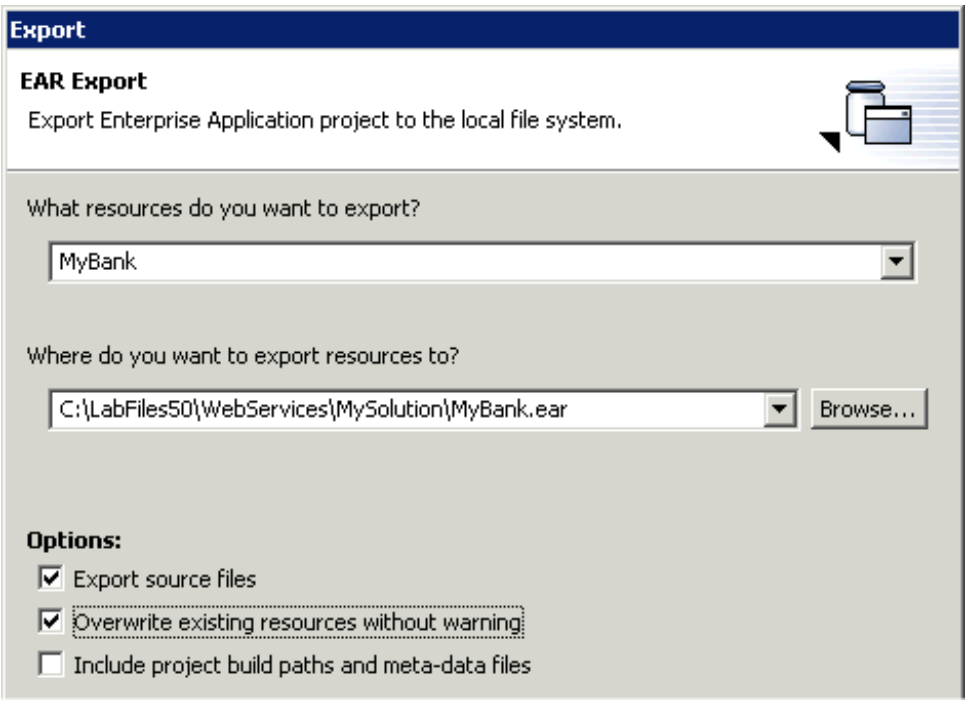


__ m. Click **Finish**.

__ n. As part of the Web Service creation, the TestServer is started. The server should be stopped. Switch to the **Server Perspective**. Select **TestServer** from the **Server tab**. Right click on **TestServer** and select **Stop**. This will stop the WebSphere Test Environment server.

__6. Select **File** > **Export...** > **EAR file** and click **Next**.

__7. For the resource, select **MyBank**. For the "Where do you want to export resources to?", Click **Browse...**. Navigate to **\LabFiles50\WebServices\MySolution** (using the drive letter where you expanded LabFiles50 to) and type **MyBank** for the file name. The Files of Type should default to ".ear". Click **Open**. Select **Export source files** and **Overwrite existing resources without warning**.



__8. Click **Finish**.

__9. Next, you will Generate the EJB to RDB Mappings.

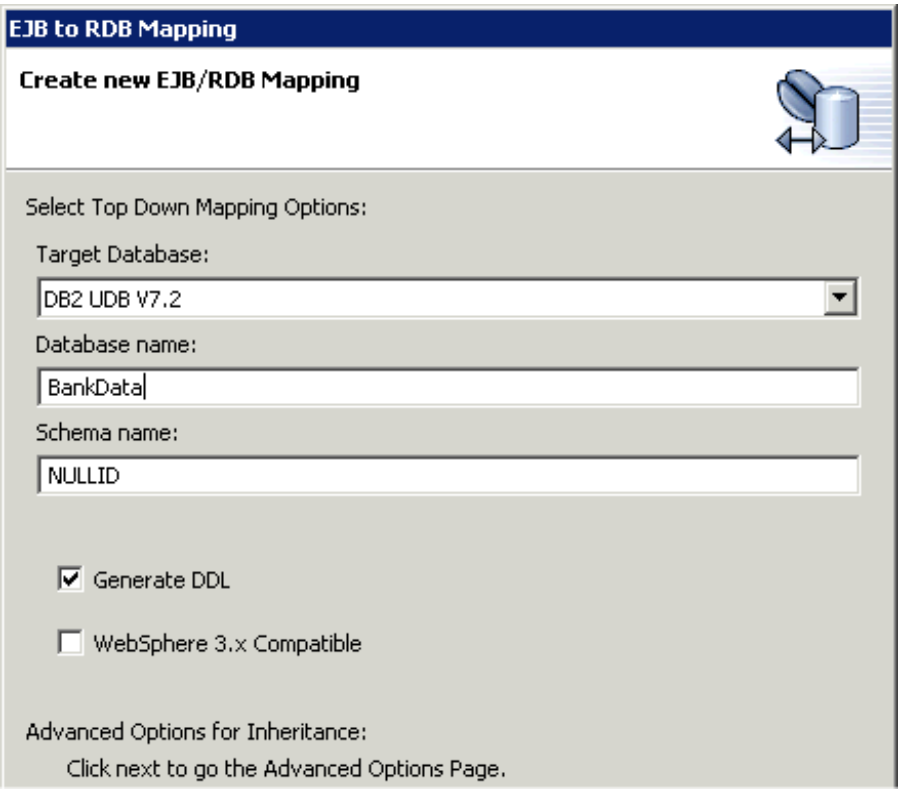
__ a. In the **J2EE Navigator** view of the J2EE Perspective, select **MyBankEJB**.

__ b. Right-click **MyBankEJB** and select **Generate > EJB to RDB Mapping...**

__ c. Select **Create a new backend folder**. Click **Next**.

__ d. Select **Top Down**. Click **Next**.

__ e. On the next panel, type **BankData** for the Database name. Accept the rest of the defaults.



__ f. Click **Finish**.

__ g. Next, **Close** the newly created mapping file by clicking on the X on the Right.

__10. Deploy the EJB code.

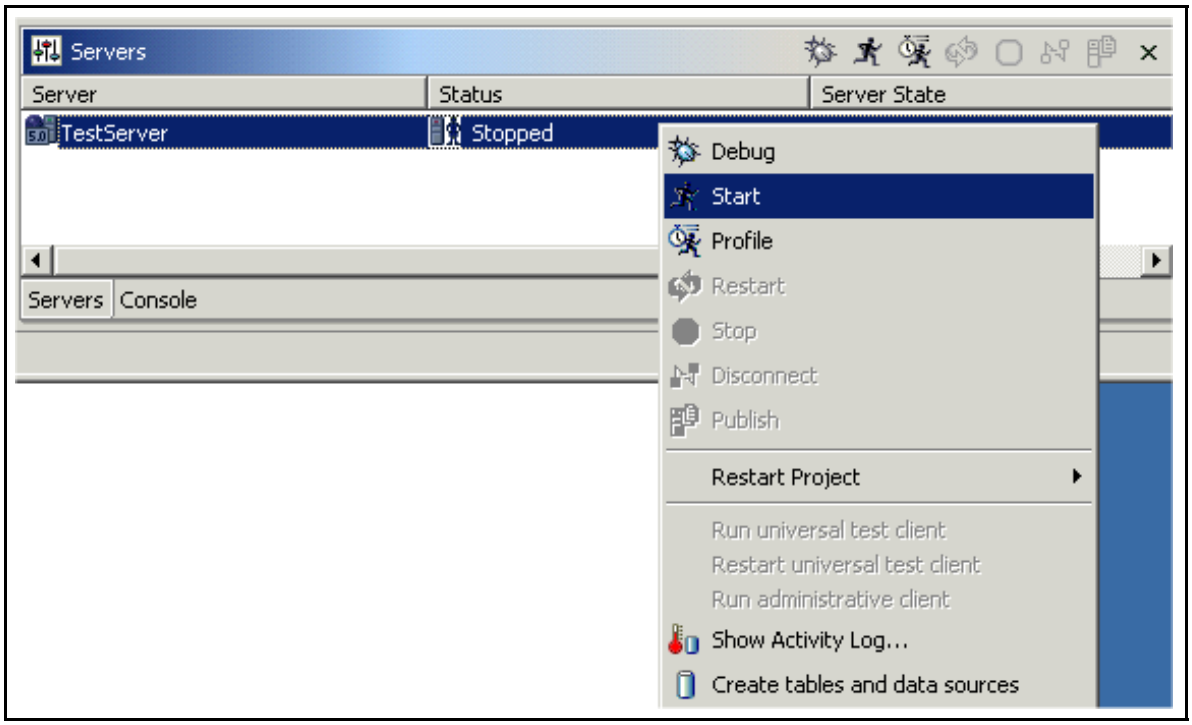
__ a. Right-click **MyBankEJB** again and select **Generate > Deploy and RMIC Code...**

__ b. Select all EJBs on the Generate Deploy and RMIC Code panel by clicking the **Select All** button.

__ c. Click **Finish**.

__11. To publish a Web Service, the WebSphereTest Environment server has to be running.

__ a. Switch to the **Server Perspective**. Select **TestServer** from the **Server tab**. Right click on **TestServer** and select **Start**. This will start the WebSphere Test Environment Server.

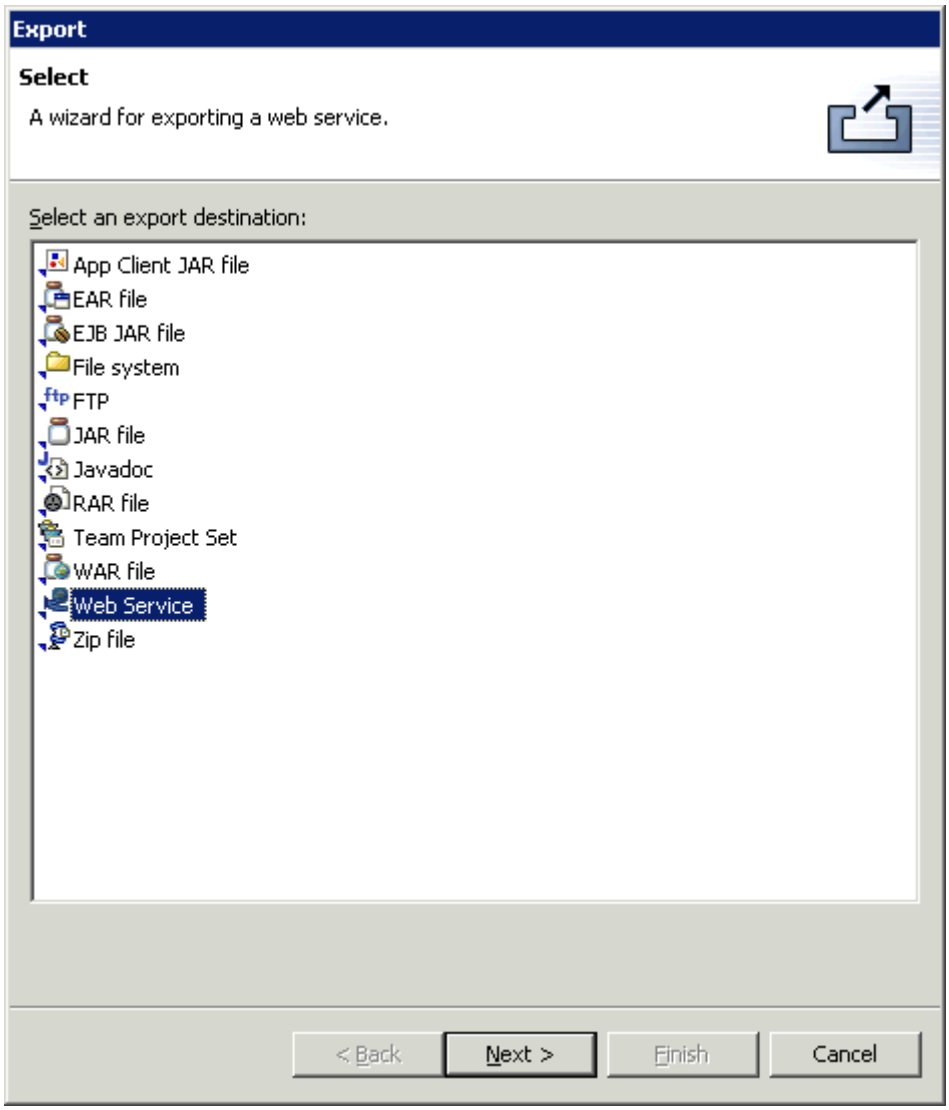


__ b. In the console messages you should see messages similar to the following:

```
[9/17/02 15:54:44:049 CDT] 17a0c09f HttpTransport A SRVE0171I:
Transport http is listening on port 9,080.
[9/17/02 15:54:46:502 CDT] 17a0c09f HttpTransport A SRVE0171I:
Transport https is listening on port 9,443.
[9/17/02 15:54:46:582 CDT] 17a0c09f JMXSoapAdapte A ADMC0013I:
SOAP connector available at port 8880
[9/17/02 15:54:46:632 CDT] 17a0c09f RMIConnectorC A ADMC0026I:
RMI Connector available at port 2809
[9/17/02 15:54:46:653 CDT] 17a0c09f WsServer      A WSVR0001I:
Server server1 open for e-business
```

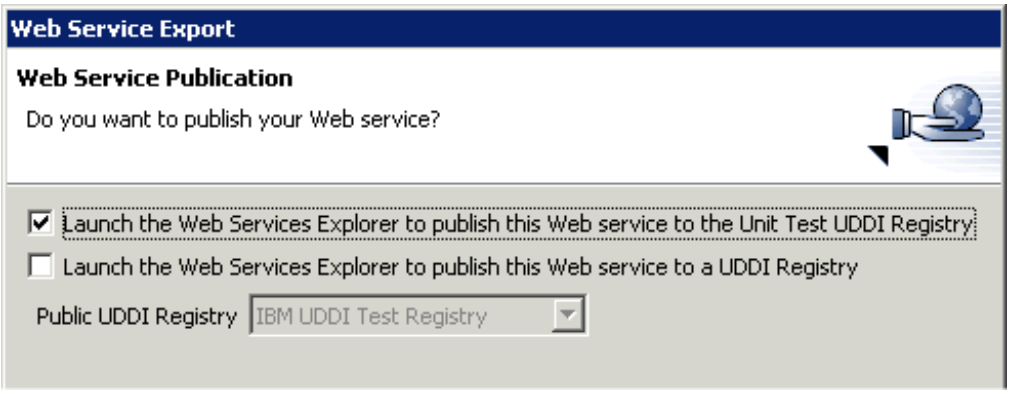
__12. Launch the Web Services Explorer.

__ a. Select **File** > **Export...** > **Web Service**.



__ b. Click **Next.**

__ c. Select **Launch the Web Services Explorer to publish this Web service to the Unit Test UDDI Registry.**



__ d. Click **Finish.**

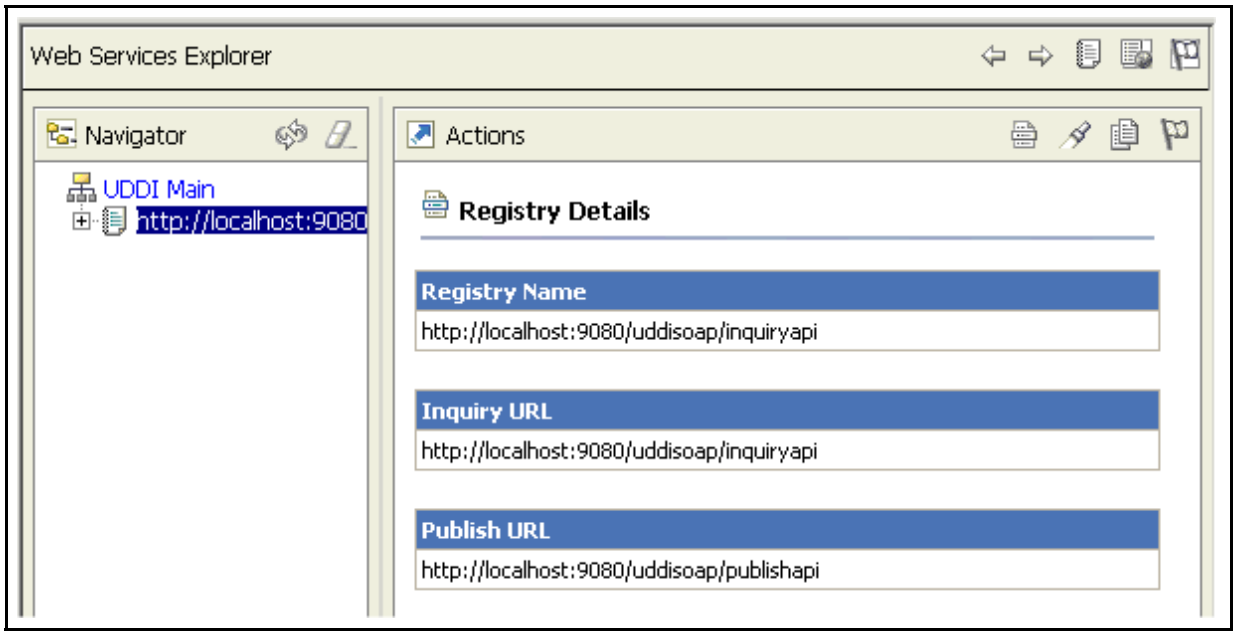
__ e. Another way to launch the Web Services Explorer is to select the icon that resembles a notebook on the toolbar. If you launch it with the icon, you have to specify the Registry Details. No action is necessary for this step.

Part Two: Publish Web Service to Unit Test UDDI Registry

__1. The Web Services Explorer is launched in an browser window within the Application Developer. For learning purposes of this lab, you will publish the CreateAccount Web Service to the Unit Test UDDI Registry. Cloudscape will be used as the database to store this information into. The Web Services wizard has the capability to create the sample application, however, when the wizard was run, the options to generate a proxy and sample were not selected. After publishing the service to the Unit Test UDDI Registry, you will create a sample application.

__ a. The Web Services Explorer is launched in a browser window and looks like the following graphic. Since the Web Services Explorer was launched from the wizard which was associated with the Unit Test UDDI Registry, the values for the registry name, inquiry and publish values are already populated.

Here's a graphic of the Web Services explorer that will be visible:



__ b. In the Status Frame along the bottom of the browser window, you should see a message similar to:

"The registry named http://localhost:9080/uddisoap/inquiryapi located at http://localhost:9080/uddisoap/inquiryapi was opened successfully."

__ c. Select the **Publish** Action. This is the second icon from the right that resembles two sheets of paper.

__ d. All Web Services published belong to a Business Entity. For the purpose of this lab, you publish a BusinessEntity object called “MyBusinessEntity.” You will publish the CreateAccount Web Service into this Business Entity.

__ e. For **Publish** type, select **Business** from the list box. The dialog in the frame changes based on the type of Publish selected. The Business publish type is the default selection for this entry field.

__ f. The Publish URL is already filled in and is the publish access point for the UDDI Registry.

__ g. Enter **wsdemo** and **wsdemo1** for the User ID and Password respectively.

__ h. For the Business Entity Name, type **MyBusinessEntity**. For the Description, type **Here’s the description.**

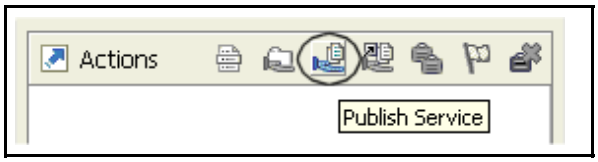
__ i. Click **Go**.

__ j. In the Status Frame, you should see a message saying “Business MyBusinessEntity was successfully published.”

__ k. After the Business Entity has been created, the Actions icons change.

__ l. Now you will publish the Web Service and it will be associated with the Business Entity called MyBusinessEntity.

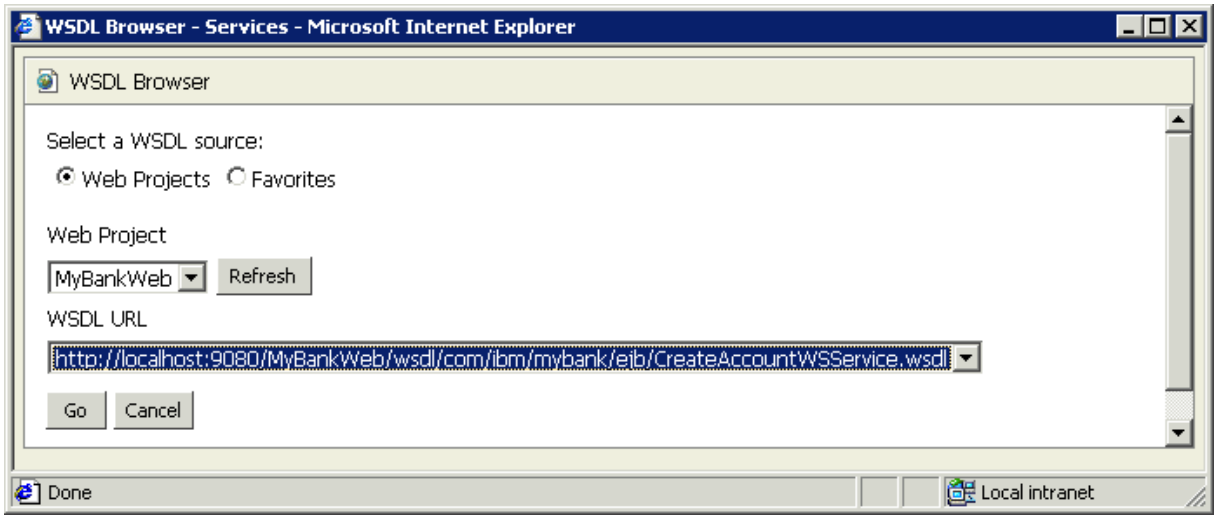
__ m. With the Business Details pane still present, select the **Publish Service** Action (the third Action icon from the left). This icon resembles a hand underneath two sheets of paper.



__ n. On the resulting Publish Service panel that is displayed, click **Browse...** link next to WSDL URL.

__ o. You have two options to choose from when Browsing WSDL URLs. The first is to have the Web Services Explorer browse Web Projects (within WebSphere Studio Application Developer) and the second is to browse the Favorites you have set up for the Web Services Explorer. By default, the Favorites have the default public registries for IBM, HP, Microsoft and SAP already loaded.

__ p. The default is Web Projects and you see our WSDL service definition for our CreateAccount Web Service is selected.



__ q. Click **Go**. This populates the WSDL URL in the main Web Services explorer browser window.

__ r. For the service **Name**, type **CreateAccount**. For the description, type **My CreateAccount Web Service**.

__ s. Click **Go**.

__ t. In the Status Frame, you should see a message similar to the following.

__ u. The Service Key shown above uniquely identifies the service just published in the “MyBusinessEntity” Business Entity. You do have the option of adding additional details to this service entry. For example, you could make associations with multiple categories and languages. For this lab, no additional changes will be made.

__ v. Next, **Close** the Web Services Explorer window by clicking on the X on the right.

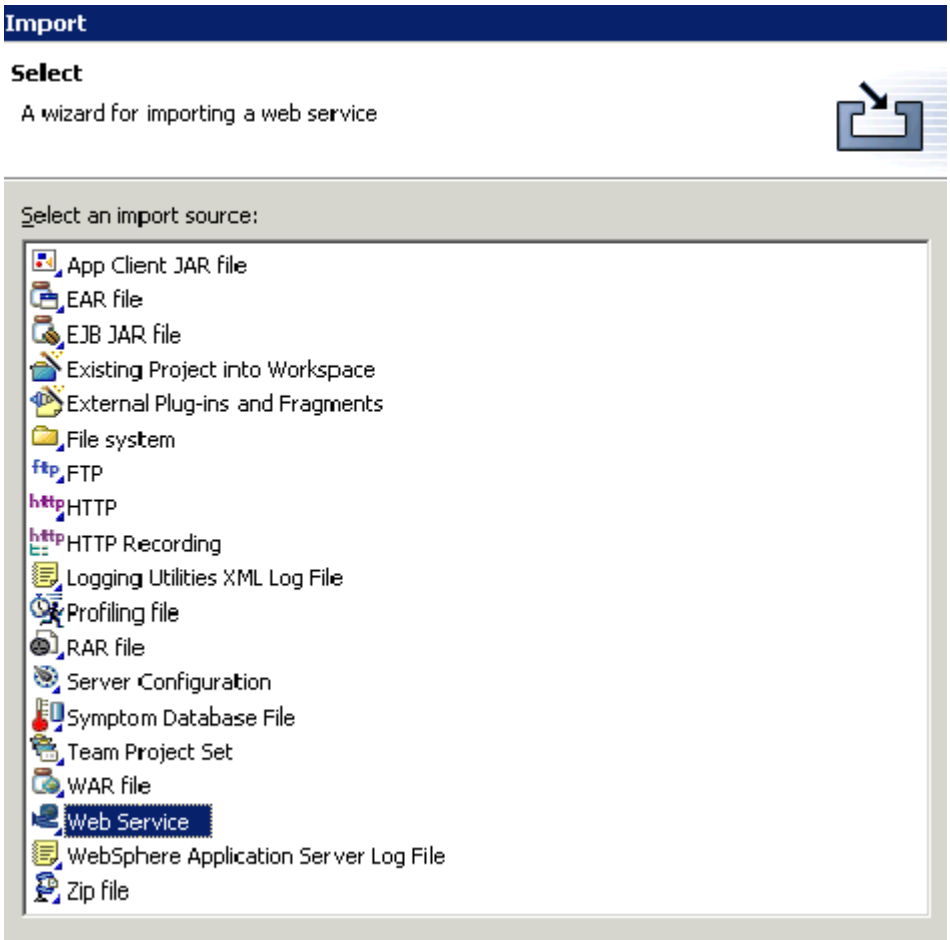
__ w. Congratulations, you have successfully published a Web Service in the IBM Unit Test UDDI Registry on your local machine.

Part Three: Create a Sample Application to Exercise Web Service

__1. Launch the Web Services Explorer.

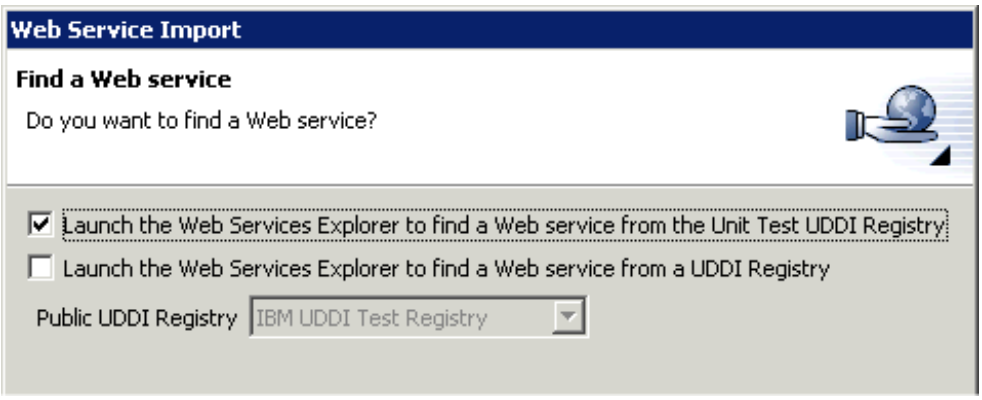
__ a. In the WebSphere Application Studio Application Developer workbench, select **File > Import...**.

__ b. Select **Web Service.**



__ c. Click **Next**.

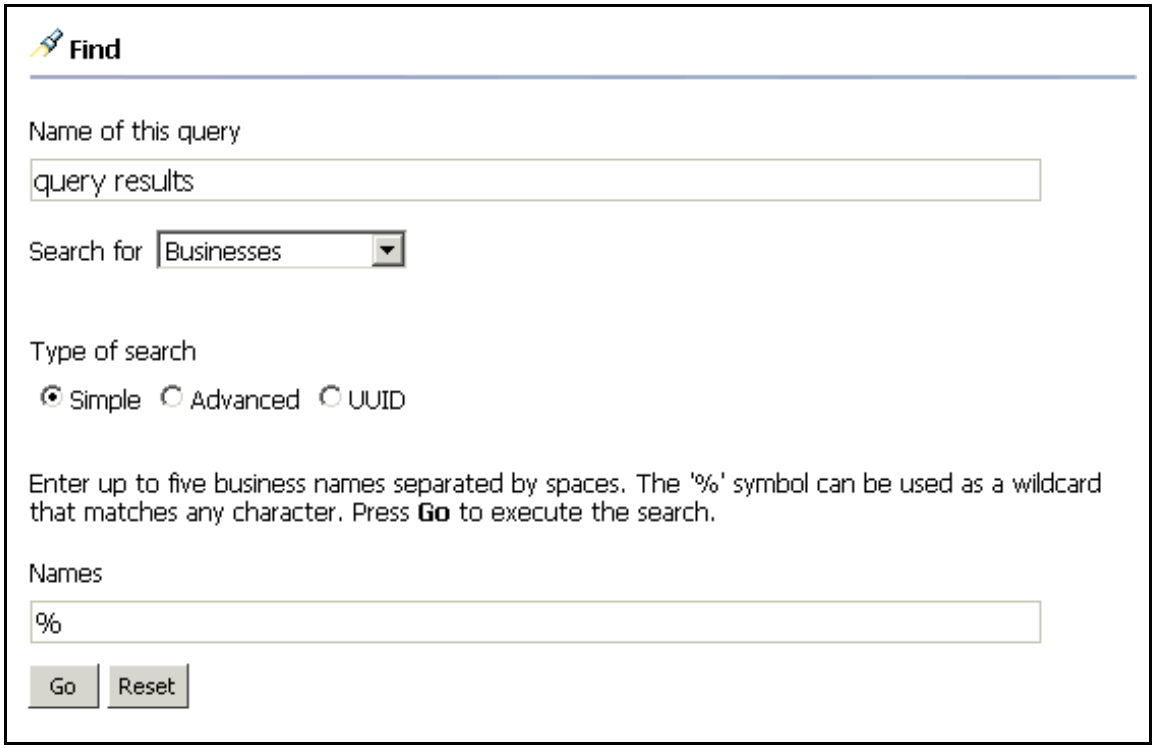
__ d. Select **Launch the Web Services Explorer to find the Web service from the Unit Test Registry**. This will launch the explorer allowing you to discover Web Services.



__ e. Click **Finish**.

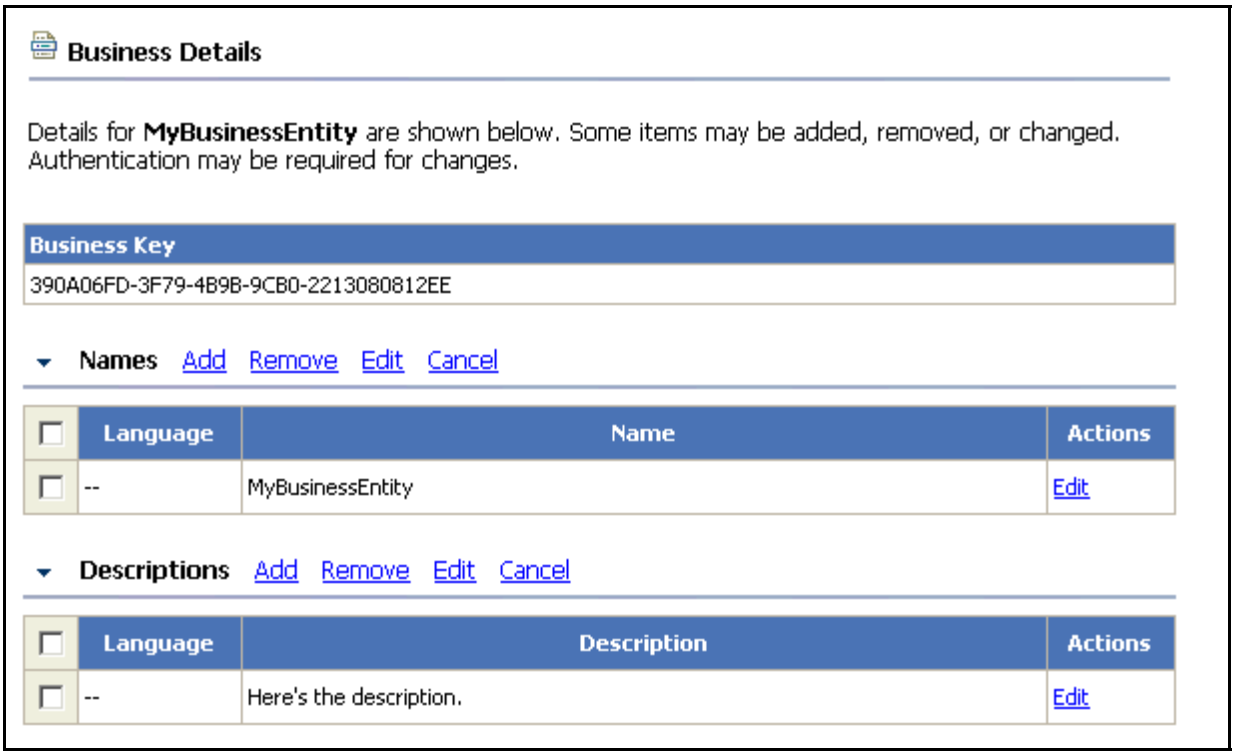
__ f. Select the **Find** Action. The icon resembles a flashlight.

__ g. In the Find panel displayed, type '**%**' for Names.



__ h. Click **Go**.

__ i. The Query Results returned will display the “MyBusinessEntity” business details previously published. If multiple Business Entities exist, multiple rows in a table are presented.



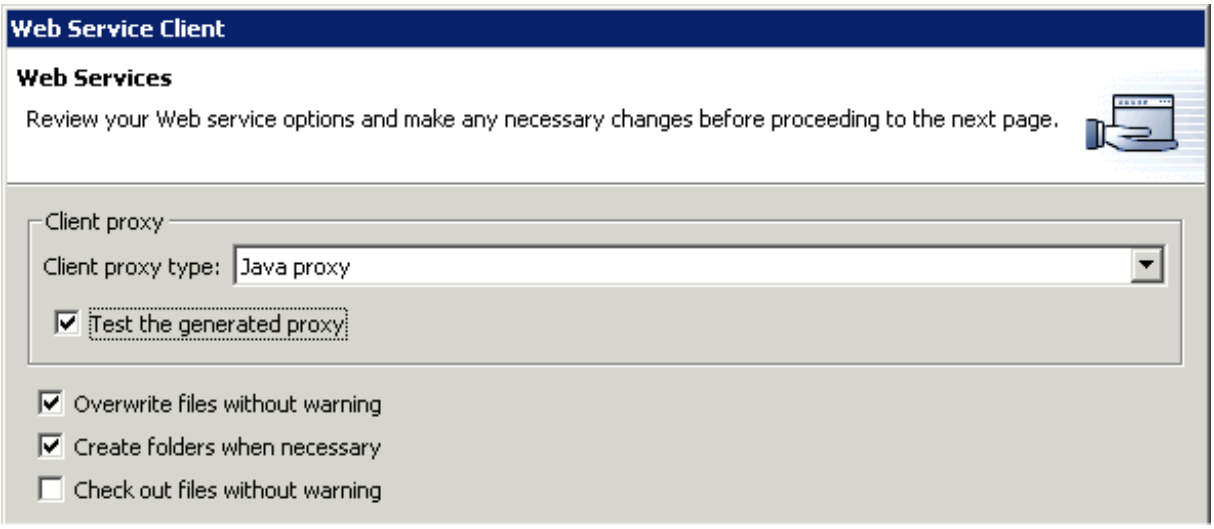
__ j. Select the **Get Services** action icon. The icon resembles a hand under a folder. The Status Frame should indicate the Number of Services found is 1.

__ k. The **CreateAccount** Service Details are displayed. Select the **Launch Web Services Wizard** Action.

__ l. On the Launch Web Service Wizard, select **Web Service Client**.

__ m. Click **Go**. This launches the Web Service Client wizard.

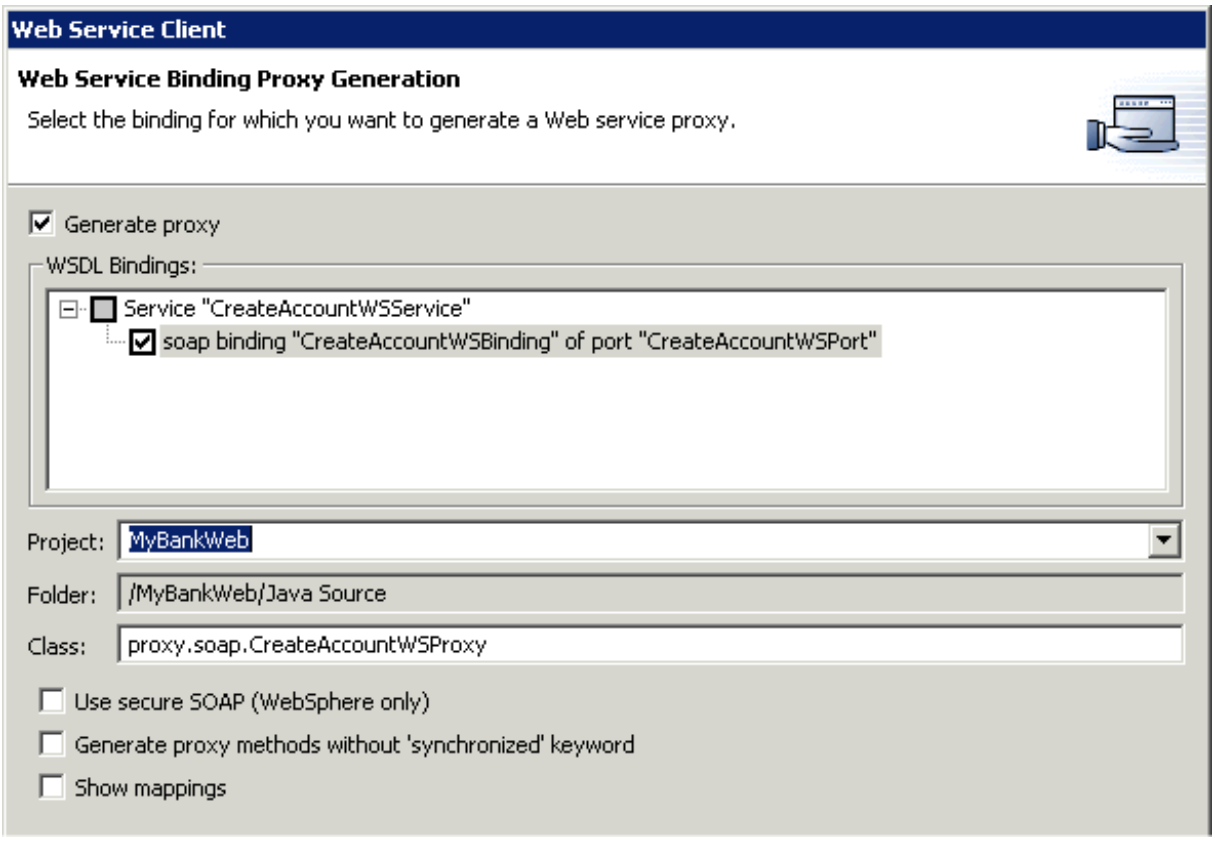
__ n. The Web Service Client window is now active. Select **Test the generated proxy**. Ensure **Overwrite files without warning** and **Create folders when necessary** are also selected.



__ o. Click **Next**.

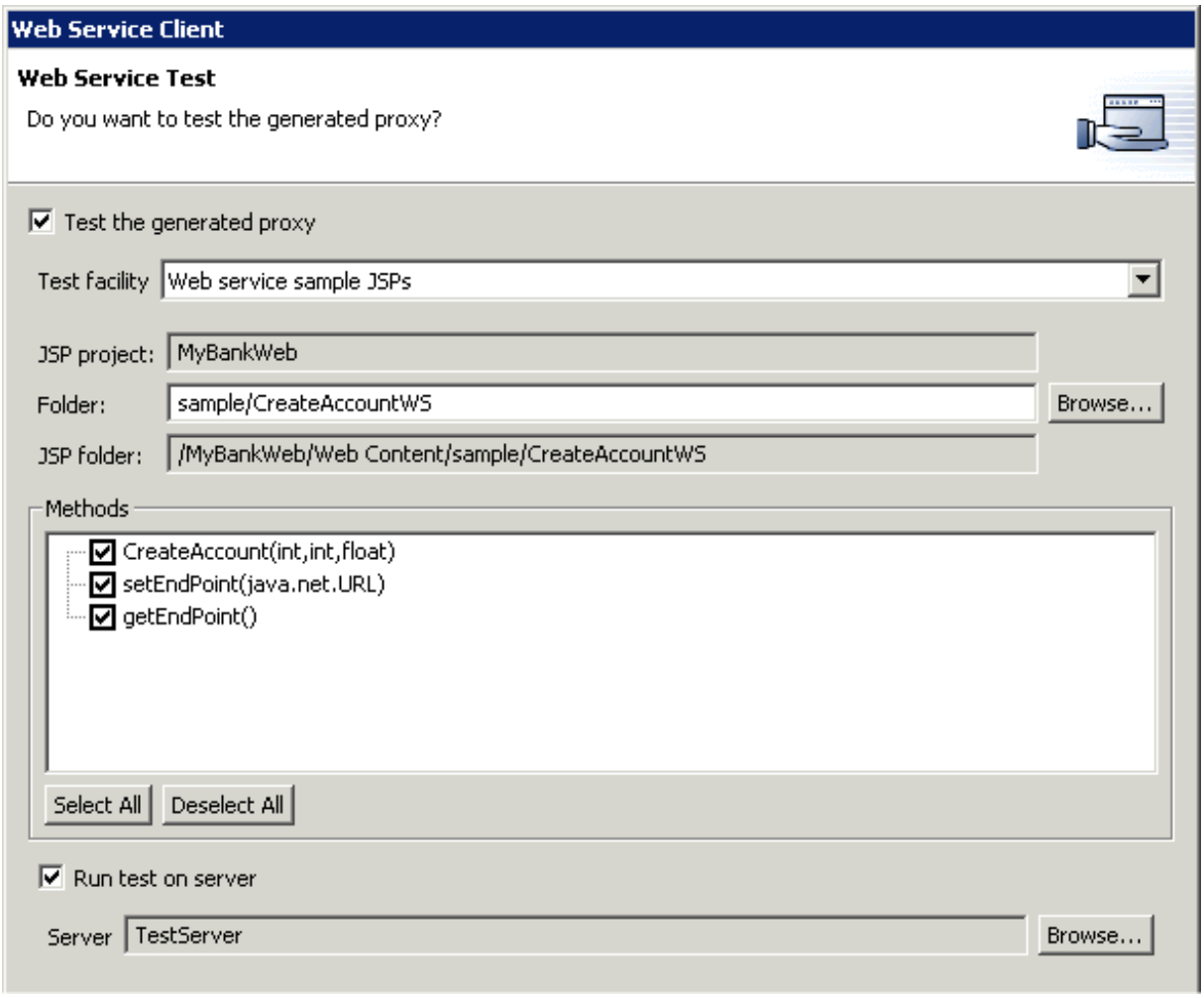
__ p. The WSDL file name or URL on the Web Service WSDL File Selection panel is pre-populated with the CreateAccountwSService wsdl file. Click **Next**.

__ q. On the Web Service Binding Proxy Generation panel, select **MyBankWeb** for the Project name. This changes the folder name to MyBankWeb/Java Source.



__ r. Click **Next**.

__ s. On the Web Service Test panel, ensure the **Test the generated proxy** option is selected. Accept the remaining default.



__ t. Click **Finish**.

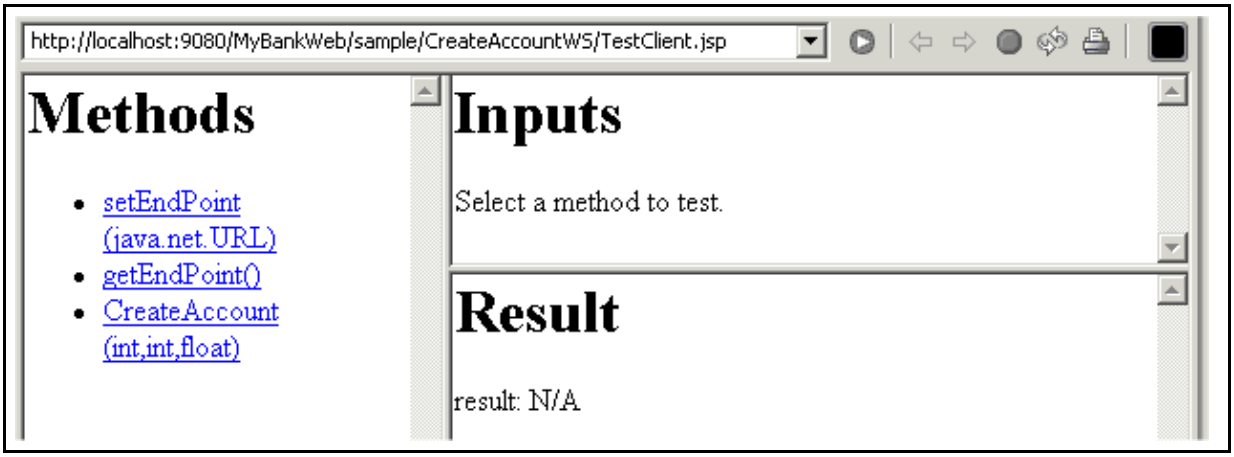
__ u. Next, **Close** the Web Services Explorer window by clicking on the X on the right.

Part Four: Test a Web Service within WebSphere Studio Application Developer

__1. In the Application Developer, a web browser should be active as a result of running the Web Service wizard and selecting the "Test the generated proxy" option.

__2. Test the Web Service using the sample application.

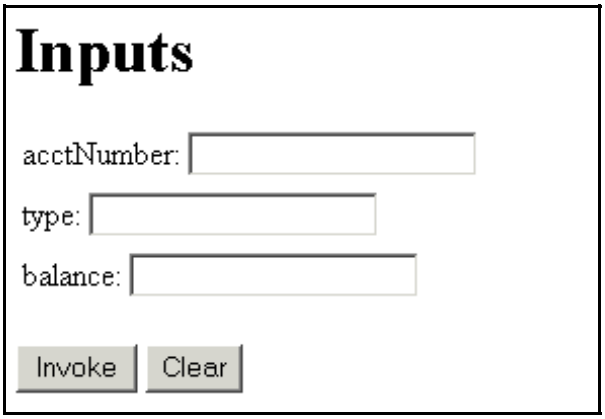
__ a. The following sample application (TestClient.jsp) is generated for you. By selecting the CreateAccount method, you can test the execution of this Web Service. The desired outcome is to create an account in the DB2 database.



__ b. Before executing the CreateAccount method, verify the accounts that are currently in the Account table. At a Command Prompt, **CD \LabFiles50\WebServices\Solutions**. Type **CheckAccountBalance.bat wsdemo wsdemo1**. The result of this batch file should be as follows:

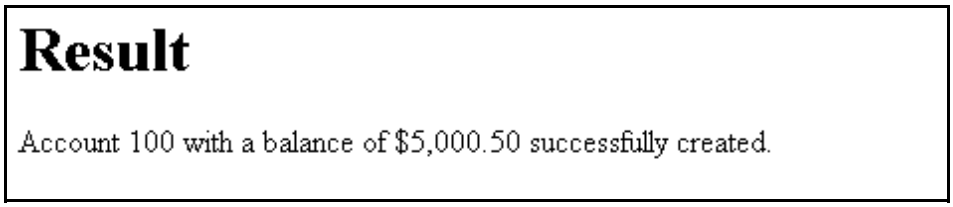
```
ACCOUNTNUMBER BALANCE
------------- -----------

1        1000.00
2        2000.00
3        3000.00
4        4000.00
5        5000.00
6        6000.00
7        7000.00
8        8000.00
9        9000.00
10      10000.00

10 record(s) selected.
```

__ a. In the Application Developer, select the **CreateAccount** link in the Web Browser that is running. This will modify the Inputs frame to the following.



__ b. You will create an account to test the CreateAccount Web Service. In the Inputs frame, type **100** for the acctNumber, **1** for the type (which really means a checking account - a 2 would mean a savings account in the MyBank application), and for the beginning balance, type **5000.50**. Click **Invoke**.

__ c. The Results frame should read as follows.



__ d. To verify the correct result, you will run the database query again. At a Command Prompt, **CD \LabFiles50\WebServices\Solutions**. Type **CheckAccountBalance.bat wsdemo wsdemo1**. The result of this batch file should be as follows:

```
ACCOUNTNUMBER BALANCE
------------- -----------

1        1000.00
2        2000.00
3        3000.00
4        4000.00
5        5000.00
6        6000.00
7        7000.00
8        8000.00
9        9000.00
10      10000.00
100      5000.50 <<--- here's the newly created account.

10 record(s) selected.
```

__3. Stop the Test Environment server. It is no longer needed and will cause a port conflict if it is left running when you move to the WebSphere Application Server runtime environment

__ a. If you are not currently in the Server Perspective, switch to the **Server Perspective**. Select **TestServer** from the **Server** tab. Right click on **TestServer** and select **Stop**. This will stop the WebSphere Test Environment server.

__4. Close WebSphere Studio Application Developer.

What you did in this exercise

In this lab exercise, you acted as a Service Provider and a Service Requester. As a Service provider, you made the CreateAccountWS method in a stateless session bean into a Web Service. You also published this Web Service definition in the IBM UDDI Test Registry (private UDDI Registry) running on your local machine using Cloudscape as the backed database. You also tested the Web Service sample application the wizard created and ensured the correct results were obtained in the DB2 database.