Building WebSphereBank: Introduction to Message Driven Beans

What this exercise is about

The objective of this lab is to add a Message Driven Bean (MDB) to the WebSphereBank application that you developed in an earlier lab.

Lab Requirements

List of system and software required for the student to complete the lab.

- Windows 2000 Professional Service Pack 4 is required for this lab exercise.
- IBM Rational Application Developer v6 with WebSphere Application Server v6 test Environment installed
- The lab source files (LabFiles60.zip) must be extracted to the root directory (i.e., C:\).
- Experience with previous versions of Rational Application Developer and the J2EE programming model are also required.

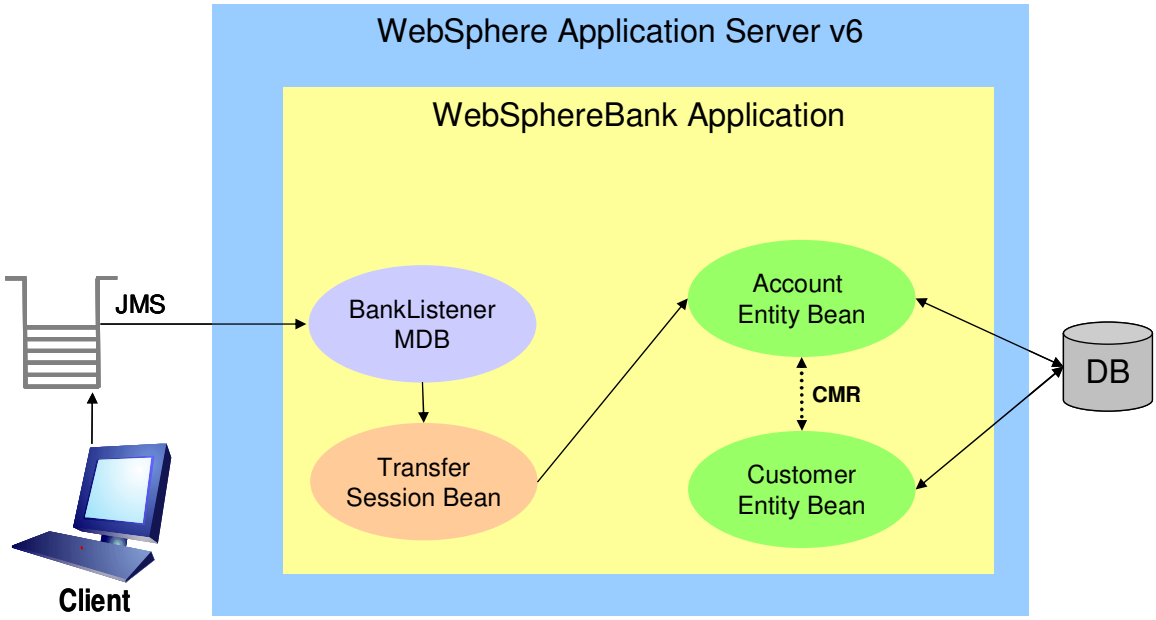
What you should be able to do

At the end of this lab you should be able to use IBM Rational Application Developer v6 to:

- Import an existing EAR file
- Create and configure a Message-driven Bean (MDB)
- Create a new Application Client Project
- Prepare JMS resources using the administrative console
- Test the Message-driven Bean

Introduction

In this exercise you will start by importing an EAR file that contains the base WebSphereBank application. You will then add a Message-Driven Bean (MDB) to the application that is used to process a message coming from a Queue. The message will contain a string with three elements in it: source account number, target account number, and an amount of money to transfer. The MDB will process the message by calling the Transfer session bean to perform a money transfer between the two accounts specified in the message. In this exercise you will also write an Application Client Project that will be used to initiate the transfer by placing a message on the messaging Queue. The following picture illustrates the WebSphereBank application you will be building.



Exercise Instructions

Because these instructions are not operating-system specific, the directory locations will be specified in the lab instructions using symbolic references, as follows:

Reference Variable	Windows Location	AIX/UNIX Location
<LAB_FILES>	C:\Labfiles60	/tmp/Labfiles60

Solution instructions are provided at the end in case you are unable to complete the lab.

Part 1: Import the WebSphereBank EAR file

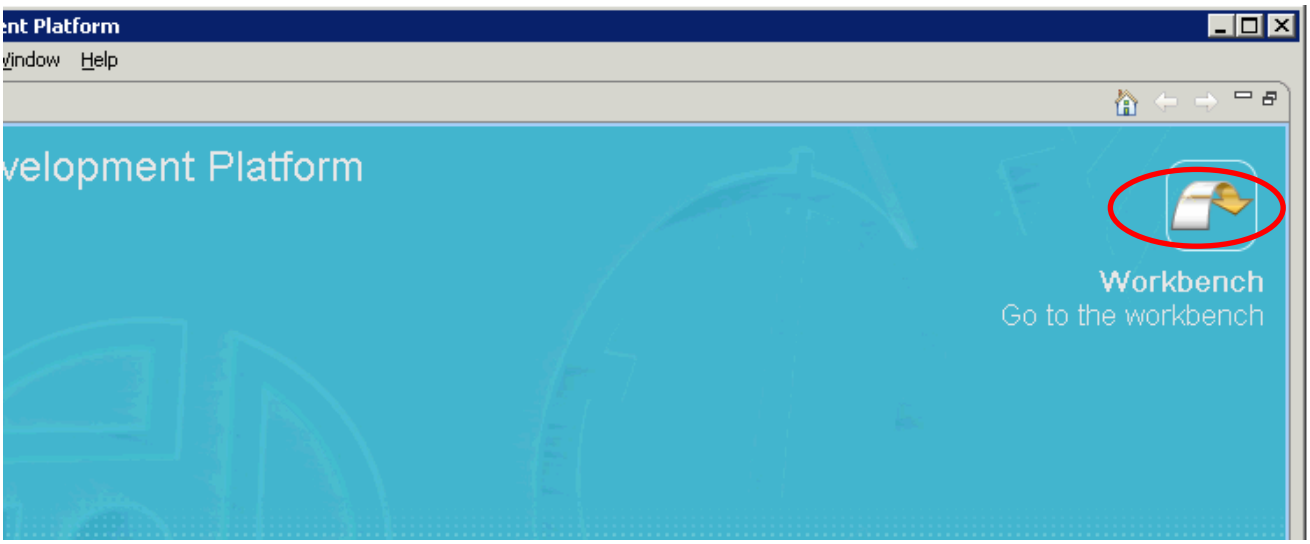
____ 1. Start Rational Application Developer.

__ a. Select **Start > Programs > IBM Rational > IBM Rational Application Developer V6.0 > Rational Application Developer.**

__ b. For the workspace, specify **<LAB_FILES>\IRAD_WebSphereBankMDB\workspace**.

NOTE: If you receive an Auto Launch Configuration Change Alert window, select **Yes** to change the auto- launch workspace to the path of the current lab.

____ 1. When IBM Rational Application Developer v6 opens, click the curved arrow at the top right corner of the welcome page.



____ 2. Import the WebSphereBank application into Rational Application Developer for testing.

__ a. Select **File > Import...**

__ b. Select **EAR file** and click **Next**.

__ c. Select **Browse...** and navigate to **<LAB_FILES>\IRAD_WebSphereBankMDB\WebSphereBankInitial.ear** and click **Open**.

__ d. Change the EAR project to **WebSphereBank**.

__ e. Click **Finish**.

Note : if you receive a Confirm Perspective Switch window, click on the **Yes** option to continue.

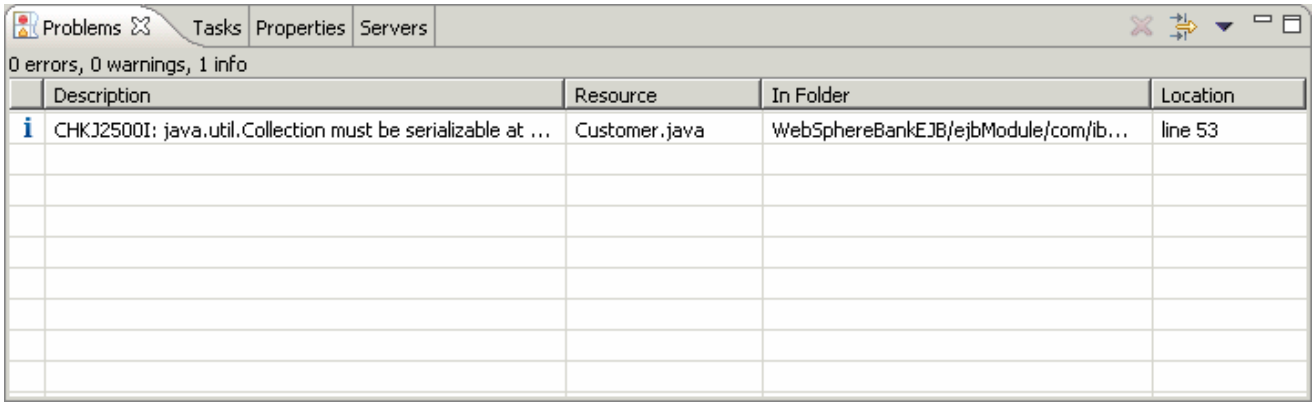
____ 3. When the import is complete, you will notice several errors in the Problems view. These errors arise because the WebSphereBankWeb Dynamic Web Project does not include the WebSphereBankEJB EJB project in the build path. Resolve these errors before you add the message-driven bean support.

__ a. In the Project Explorer view, expand Dynamic Web Projects and right click on **WebSphereBankWeb**. From the context menu select **Properties**.

__ b. Select **Java Build Path** on the left, and click the **Projects** tab. Select the checkbox next to **WebSphereBankEJB**.

__ c. Click **OK**.

__ d. Verify that there are zero errors in the Problems view. Your Problems view should look like the following screen capture:



Part 2: Create a Message Driven Bean

____ 1. Create the BankListener message-driven bean.

__ a. From the menu select **File > New > Other**. Select the checkbox next to **Show All Wizards**.

__ b. Expand **EJB** and select **Enterprise Bean** and click **Next**.

__ c. If the **Confirm Enablement** message dialog appears, click **OK**.

__ d. From the Create an Enterprise Bean wizard, select the radio button next to **Message-driven bean**.

__ e. Select **WebSphereBankEJB** for the EJB project.

__ f. Enter **BankListener** for the Bean name.

__ g. Enter **com.ibm.websphere.samples.bank.ejb** for the Default package field and click **Next**.

__ h. On the Message Driven Bean type page select **JMS Type** and click **Next**.

__ i. On the Enterprise Bean Details page, ensure that **Container** is specified as the Transaction type and **Queue** is specified as Destination type. Accept the default for Bean class, and click **Next**.

__ j. Continue to click **Next** until you reach the Select Class Diagram for Visualization page. Uncheck the **Add bean to Class Diagram** check box and click **Finish**.

____ 2. Add Code snippets to the BankListener class.

__ a. From Project Explorer view, expand **EJB Projects > WebSphereBankEJB > ejbModule > com.ibm.websphere.samples.bank.ejb** and right click on the **BankListenerBean.java** file. Select **Open** from the context menu.

__ b. Add the member data circled below to the top of the BankListenerBean class. For your convenience this code can be found in the file: **<LAB_FILES>\IRAD_WebSphereBankMDB\snippets\snippet1.txt**

```
public class BankListenerBean implements javax.ejb.MessageDrivenBean,
        javax.jms.MessageListener {

    private javax.ejb.MessageDrivenContext fMessageDrivenCtx;

    //Code added from snippet1.txt
    private final static String TRANSFER_JNDI_NAME = "java:comp/env/ejb/Transfer";
    static final String className = BankListenerBean.class.getName();
    private Logger messageLogger;
    private static final String rbName = null;
```

__ c. Add the following code to the **ejbCreate()** method for the BankListenerBean. For your convenience this code can be found in the file: **<LAB_FILES>\IRAD_WebSphereBankMDB\snippets\snippet2.txt**

```
public void ejbCreate() {
    try {
        messageLogger = Logger.getLogger(className, rbName);
        LoggerHelper.setAttributes(messageLogger, "IBM", "WebSphere",
                "Samples", WsLevel.DETAIL);
        LoggerHelper.addLoggerToGroup(messageLogger, "WebSphereBank");
    } catch (Exception e) {
        // Problem initializing logging. Continue anyway.
    }
}
```

__ d. Add the following code to the **onMessage()** method for the BankListenerBean. For your convenience this code can be found in the file: **<LAB_FILES>\IRAD_WebSphereBankMDB\snippets\snippet3.txt**

```
public void onMessage(javax.jms.Message msg) {
    final String methodName = "onMessage";
    int fromKey = 0;
    int toKey = 0;
    float amountFloat = 0f;

    try {
        TextMessage textMessage = (TextMessage) msg;
        String text = textMessage.getText();
        messageLogger.logp(Level.INFO, className, methodName, "Processing the following message: " + text + "\n");
        StringTokenizer tokens = new StringTokenizer(text, " ", false);
        fromKey = Integer.parseInt(tokens.nextToken());
        toKey = Integer.parseInt(tokens.nextToken());
        amountFloat = (float) Integer.parseInt(tokens.nextToken());
    } catch (Exception e) {
        messageLogger.logp(Level.INFO, className, methodName, "ERROR: Could not parse message\n");
        e.printStackTrace();
    }

    try {
        TransferLocalHome transferHome = null;
        Context ctx = new InitialContext();
        Object homeObject = ctx.lookup(TRANSFER_JNDI_NAME);
        transferHome = (TransferLocalHome) homeObject;
        TransferLocal transfer = null;
        transfer = transferHome.create();
        messageLogger.logp(Level.INFO, className, methodName, "Calling transferFunds.\n");
        transfer.transferFunds(fromKey, toKey, amountFloat);
        transfer.remove();
    } catch (Exception e) {
        messageLogger.logp(Level.INFO, className, methodName,"ERROR: Unable to complete transfer.\n");
        e.printStackTrace();
    }

}
```

__ e. At this point there may be a number of errors indicating that the appropriate import statements have not been included in the BankListener class definition. An easy way to add these import statements is to right click anywhere in the editor area, and select **Source > Organize Imports** from the context menu.

NOTE: You may be prompted to choose the appropriate Logger and Context class. For these you should choose **java.util.logging.Logger** and **javax.naming.Context**, respectively.

__ f. Format your code by pressing **Ctrl+Shift+F**.

__ g. **Save and Close** BankListenerBean.java.

____ 3. Open the EJB Deployment Descriptor.

__ a. From Project Explorer view, expand **EJB Projects > WebSphereBankEJB**.

__ b. Right click on **Deployment Descriptor: WebSphereBankEJB** and select **Open**.

____ 4. Configure the BankListener Bean.

__ a. From the deployment descriptor editor, select the **Bean** tab.

__ b. Select **BankListener** from EJB list on the left.

__ c. Under WebSphere Bindings, select the **JCA Adapter** radio button and enter **eis/BankActivationSpec** for the Activation Spec JNDI name.

__ d. Switch to the Source tab and add the code circled below to your BankListener definition. For your convenience this code can be found in the file: **<LAB_FILES>\IRAD_WebSphereBankMDB\snippets\snippet4.txt**

```
<message-driven id="BankListener">
    <ejb-name>BankListener</ejb-name>
    <ejb-class>com.ibm.websphere.samples.bank.ejb.BankListenerBean</ejb-class>
    <messaging-type>javax.jms.MessageListener</messaging-type>
    <transaction-type>Container</transaction-type>
    <message-destination-type>javax.jms.Queue</message-destination-type>
    <activation-config>
      <activation-config-property>
          <activation-config-property-name>destination</activation-config-property-name>
          <activation-config-property-value>BankJSQueue</activation-config-property-value>
      </activation-config-property>
      <activation-config-property>
          <activation-config-property-name>acknowledgeMode</activation-config-property-name>
          <activation-config-property-value>Auto-acknowledge</activation-config-property-value>
      </activation-config-property>
      <activation-config-property>
          <activation-config-property-name>messageSelector</activation-config-property-name>
          <activation-config-property-value>JMSType = 'transfer'</activation-config-property-value>
      </activation-config-property>
      <activation-config-property>
          <activation-config-property-name>destinationType</activation-config-property-name>
          <activation-config-property-value>javax.jms.Queue</activation-config-property-value>
      </activation-config-property>
    </activation-config>
</message-driven>
```

____ 5. Add a reference definition for the BankListener EJB.

__ a. Switch to the **Reference** tab of the EJB deployment descriptor.

__ b. Select **BankListener** from the list on the left and click the **Add** button.

__ c. From the Add reference wizard, ensure the **EJB reference** radio button is selected, and click **Next**.

__ d. Expand **WebSphereBank > WebSphereBankEJB**, select the **Transfer** bean and click **Finish**.

____ 6. **Save** and **Close** the EJB deployment descriptor.

Part 3: Build an Application Client Project

____ 1. Launch the New Application Client Project wizard.

__ a. From the menu select **File > New > Project**.

__ b. Check the **Show All Wizards** check box.

__ c. Expand **J2EE** and select **Application Client Project**. Click **Next**.

__ d. If the **Confirm Enablement** message dialog appears, click **OK**.

__ e. Enter **TransferClient** for the Name and click the **Show Advanced** button.

__ f. Select **WebSphereBank** as the EAR Project and uncheck the **Create a default Main Class** box. Click **Next**.

__ g. On the Module Dependencies page, check the box next to **WebSphereBankEJB.jar**, and click **Finish**.

____ 2. Create the TransferClient class.

__ a. From Project Explorer view, expand **Application Client Projects > TranferClient**.

__ b. Right click on **appClientModule** and select **New > Class.**

__ c. Enter **com.ibm.websphere.samples.bank.client** for the Package and **TransferClient** as the class Name.

__ d. Click **Finish**.

____ 3. Update the MANIFEST.MF file to specify the TransferClient class you just created as the main class.

__ a. From Project Explorer view, expand **Application Client Projects> TranferClient > appClientModule > META-INF**.

__ b. Right click on **MANIFEST.MF** and select **Open**.

__ c. Add the following line to the file.

```
Main-Class: com.ibm.websphere.samples.bank.client.TransferClient
```

__ d. **Save** and **Close** MANIFEST.MF.

```
Manifest-Version: 1.0
Class-Path: WebSphereBankEJB.jar
Main-Class: com.ibm.websphere.samples.bank.client.TransferClient
```

____ 4. Include some constants at the top of the file.

__ a. From Project Explorer view, expand **Application Client Projects > TranferClient > appClientModule > com.ibm.websphere.samples.bank.client** and right click on the **TransferClient.java** file. Select **Open** from the context menu.

__ b. Add the code to the TransferClient class. For your convenience this code can be found in the file **<LAB_FILES>\IRAD_WebSphereBankMDB\snippets\snippet5.txt**. Copy and paste the contents of this file in between the open and closing brackets of the class definition. Review the code, and pay particular attention to the **sendMessage()** method.

```
public static void sendMessage(String message) {
    System.out.println("---> Begin sendMessage() with: " + message);
    try {
        Context ctx = new InitialContext();
        System.out.println("---> Finding the WebSpere Application    Server QueueConnectionFactory");
        javax.jms.ConnectionFactory qcf = (javax.jms.ConnectionFactory) ctx.lookup(BANKJMSCF_JNDI_NAME);

        System.out.println("---> Finding the Queue Destination");
        Destination q = (Destination) ctx.lookup(BANKJMSQ_JNDI_NAME);

        System.out.println("---> Creating JMS Connection");

        Connection connection; // Create JMS Connection
        connection = qcf.createConnection();
        System.out.println("---> Connection created successfully.");

        Session session; // Create JMS Session
        boolean transacted = false;
        session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);


        MessageProducer queueSender = session.createProducer(q);
        TextMessage outMessage = session.createTextMessage();

        String type;
        type = new String("transfer");
        outMessage.setText(message);
        outMessage.setJMSType(type);
        outMessage.setJMSDestination(q);
        queueSender.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
        System.out.println("---> Sending message: "+ outMessage.getText());
        queueSender.send(outMessage);

        connection.close();
    } catch (NamingException ne) {
        ne.printStackTrace();
    } catch (JMSException jmse) {
        jmse.printStackTrace();
    }
    System.out.println("---> End sendMessage()");
}
```

__ c. Add the appropriate import statements by right clicking anywhere in the editor area and selecting **Source > Organize Imports**.

NOTE: You will need to choose the appropriate class for importing during this process. The following lists those that need to be added: **javax.naming.Context, javax.jms.Destination, javax.jms.Connection, javax.jms.Session, and javax.jms.MessageProducer**.

__ d. **Save and close** the TransferClient.java file.

____ 5. Configure the Application Client.

__ a. From Project Explorer view, expand **Application Client Projects > TranferClient**.

__ b. Right click on **Deployment Descriptor: TransferClient** and select **Open**.

____ 6. Configure the Resource Reference.

__ a. Click on the **References** tab and click **Add**.

__ b. From the Add Reference wizard select the **Resource reference** radio button and click **Next**.

__ c. The follow table lists the values you should enter for each field.

Field	Value
Name	jms/BankJMSConnFactory
Type	javax.jmx.QueueConnFactory
Authentication	Container
Sharing scope	Shareable

__ d. Click **Finish**.

NOTE: You may see an error message after clicking Finish. If this occurs click OK, and move on to the next step.

__ e. Open the **WebSphere Bindings** section. This should already be open for you. If it is not, click the arrow noted below.

__ f. Click on the resource reference you just added. You may see an error message appear. If this occurs, again click **OK** and continue.

__ g. Under **WebSphere Bindings**, enter **jms/BankJMSConnFactory** for the JNDI name.

____ 7. Add a Message reference.

__ a. From the References tab and click **Add**.

__ b. From the Add Reference wizard select the **Message destination reference** radio button and click **Next**.

__ c. Expand **WebSphereBank > WebSphereBankEJB** and select **BankListener** bean in the Message destination reference window.

__ d. Change the Name to **jms/BankJMSQueue** and click **Next**.

__ e. The follow table lists the values you should enter for each field.

Field	Value
Destination Link	BankJSQueue
Type	javax.jms.Queue
Usage	Produces

__ f. Click **Finish**.

____ 8. **Save and Close** the Client Deployment Descriptor.

Part 4: Export the EAR File

____ 1. Export WebSphereBank EAR file.

__ a. Click on **File > Export** and select EAR File from the list. Click **Next**.

__ b. Select **WebSphereBank** from the drop down menu for the Enterprise Application project.

__ c. Browse to **<LAB_FILES>\IRAD_WebSphereBankMDB** and enter **WebSphereBankMDB** for the name of the EAR file. Click **Save**.

__ d. Check the **Export source files** and **Include project build paths and meta-data files** check boxes and click **Finish**.

What you did in this exercise

In this exercise you added a message-driven bean to the WebSphereBank application using IBM Rational Application Developer v6. This exercise highlights several important concepts associated with the using and configuring message-driven beans.

Solution Instructions

____ 1. Start Rational Application Developer.

__ a. Select **Start > Programs > IBM Rational > IBM Rational Application Developer 6.0 > Rational Application Developer**.

__ b. For the workspace, specify **<LAB_FILES>\IRAD_WebSphereBankMDB\solution\workspace**.

NOTE: If you receive an Auto Launch Configuration Change Alert window, select **Yes** to change the auto- launch workspace to the path of the current lab.

____ 2. Import the WebSphereBank application into Rational Application Developer for testing.

__ a. Select **File > Import...**

__ b. Select **EAR file** and select **Next**.

__ c. Select **Browse...** and navigate to **<LAB_FILES>\IRAD_WebSphereBankMDB\solution\WebSphereBank.ear** and select **Open**.

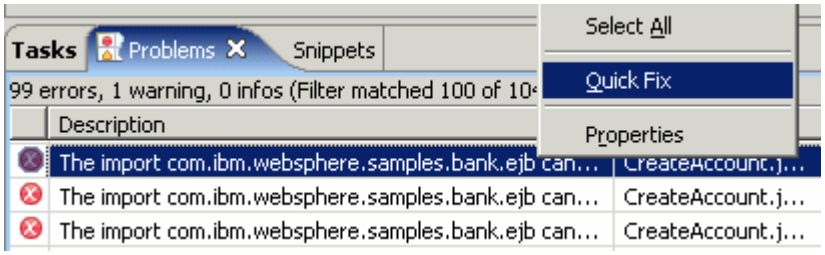
__ d. For the Project name, enter **WebSphereBank**.

__ e. Click **Finish**.

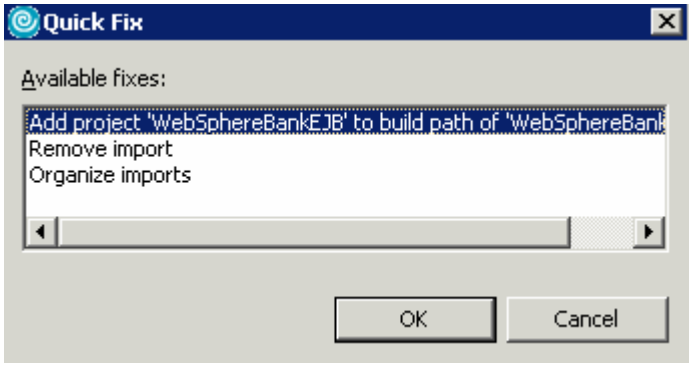
Note : if you receive a **Confirm Perspective Switch** window, click on the **Yes** option to continue.

____ 3. Add **Java Build Path** to clean up errors. If you select the Problems tab, you will notice a list of errors and warnings.

__ a. Right click on an error and select **Quick Fix.**



__ b. A list of available fixes will appear. Select **Add project 'WebSphereBankEJB' to build path of 'WebSphereBankWeb'** and select **OK.** The errors should disappear.



____ 4. Explore the WebSphereBankEJB project, and the various EJBs developed in this lab.

NOTE: The next lab in this series will test the functionality developed in this exercise for the WebSphereBank application.

This page is left intentionally blank.