

## IBM WEBSPHERE STUDIO 5.1 Skills Transfer - LAB EXERCISE

# Debugging Applications

### **What This Exercise is About**

Application troubleshooting is an important feature. This exercise presents some of the debugging features available in WebSphere Studio Application Developer. The debugger in Application Developer enables you to detect and diagnose errors in your programs. The client/server design makes it possible to debug programs running on systems accessible through a network connection as well as debug programs running on your local workstation. In this exercise you will carry out remote debugging against a running WebSphere Application Server. Debugging actions such as setting breakpoints and inspecting variables will also be featured.

### **User Requirement**

User must have IBM DB2 Universal Database (version 7.2) and IBM WebSphere Studio Application Developer Version 5.1 installed on a Windows 2000 workstation with Service Pack 3. The WebSphere Studio Application Developer should be installed at C:\Program Files\IBM\WebSphere Studio. The WebSphere Application Server Version 5.0.1 is also required to complete all parts of the lab. 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 are also required. Throughout this lab, a userid profile named wsdemo is assumed to be created on the system.

### **What You Should Be Able to Do**

After completing this lab you will understand the steps required to perform basic debugging steps using IBM WebSphere Studio Application Developer. You will also know how to set breakpoints and inspect and change values of variables in Enterprise Javabeans.

### **Introduction**

In this exercise, you will use Application Developer to debug a simple problem in the Transfer Bean. The problem that was introduced has the effect of subtracting twice the amount from the From Account and adding twice the amount to the To Account.

Debugging can be carried out both remotely and locally. In this exercise you will perform remote debugging. The workbench includes a debugger that enables you to view and inspect code. The debugger runs on the same system as the application you want to debug, which can

be on your workstation. Breakpoints are temporary markers you place in your program to tell the debugger to stop your program at a given point. At a breakpoint, the current line of execution is highlighted in the editor in the Debug perspective. In this exercise, you will set breakpoints, inspect and change values of variables and perform basic debugging actions.

## **Exercise Instructions**

**\*\* NOTE \*\*** Solution instructions are normally provided at the end. The solution is not provided in this case because you need to do the exercises in order to understand Java tools and there is no final solution to import. To go through the lab start at Part One assuming you have met the requirements in the section “User Requirements” stated above.

### **Part One: Lab Setup**

- \_\_ 1. In order to perform this lab successfully, you need to do a lab setup. The restoreConfig command line utility can be used to restore the original backed up WebSphere configuration files.
  - \_\_ a. At a Command Prompt, **CD \WebSphere\AppServer\bin**.
  - \_\_ b. Type **restoreConfig.bat baseWASnode.zip**. This will recover the contents of several directories and configuration files from the **baseWASnode.zip** file, which was created as part of the Installation lab.
- \_\_ 2. The MyBank application uses a database called BankData. This database needs to be created. At a Command Prompt, type **CD \LabFiles50\Debugging\Bankdata**. Then type **BankData.bat wsdemo password Table.ddl** where *password* is the password to your wsdemo account.

**NOTE:** If you do not have DB2 installed at C:\SQLLIB, you must update the variables in the \LabFiles50\setupVARs.bat file according to your installation.

- \_\_ a. This batch file drops any existing database called BankData, re-creates the BankData database, and populates the Account table with a few records.
- \_\_ b. After the BankData batch file completes, you should see the following message:  
  
BankData database has been created and loaded.

### **Part Two: Install the MyBank application into WebSphere Application Server**

- \_\_ 1. Now you are ready to install the MyBank EAR file on the WebSphere V5 Application Server. At a Command Prompt, type **CD \WebSphere\AppServer\bin**. If the

WebSphere Application Server is installed elsewhere, update the path accordingly -- this statement applies to this entire section of the lab exercise.

- \_\_2. Start the WebSphere Application Server by typing **startserver server1**. After successful startup of the application server, you should see resulting console messages like the following:

```
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\logs\server1\startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1324
```

- \_\_3. At a Command Prompt, type **CD \LabFiles50\Debugging**.
- \_\_4. In the same Command Prompt window, type **InstallMyBankJDBC.bat <db2userid> <db2password> <NODENAME>**. For example, for this lab you would be typing: **InstallMyBankJDBC.bat wsdemo password NODENAME**, where “*NODENAME*” is the name of your Websphere Application server instance and *password* is the password to your wsdemo account.

<p><b>NOTE:</b> The parameter &lt;NODENAME&gt; is case-sensitive.</p>
---

- \_\_5. The setupMyBank.jacl script configures the JDBC Provider, DataSource, and Connection Factory for the MyBank application. The last few lines received from running the setupMyBank JACL script, which is called by the InstallMyBankJDBC bat file, should be similar to the following:

```
Created CMPConnectorFactory for WebSphere Relational Resource
Adapter(cells/BaseApplicationServerCell/nodes/NODENAME:resources.
xml#builtin_rra)
- Attempting to save new configuration for the Connection
  Factory
- Changes saved successfully.
```

- \_\_6. Ensure your path in the Command Prompt says **\WebSphere\AppServer\bin**.
- \_\_7. Type **wsadmin** and press Enter. When the **wsadmin>** prompt is visible, type: **\$AdminApp install C:/LabFiles50/Debugging/Materials/MyBank.ear** and press Enter.
- \_\_8. After the install completes, you should see messages like those below.

```
ADMA5016I: Installation of MyBankCMP started.
ADMA5005I: Application MyBankCMP configured in WebSphere
repository
ADMA5001I: Application binaries saved in C:\WebSphere\AppServer\
wstemp\Scriptf0eA03c3f7\workspace\cells\NODENAME\applications\
MyBankCMP.ear\MyBankCMP.ear
```

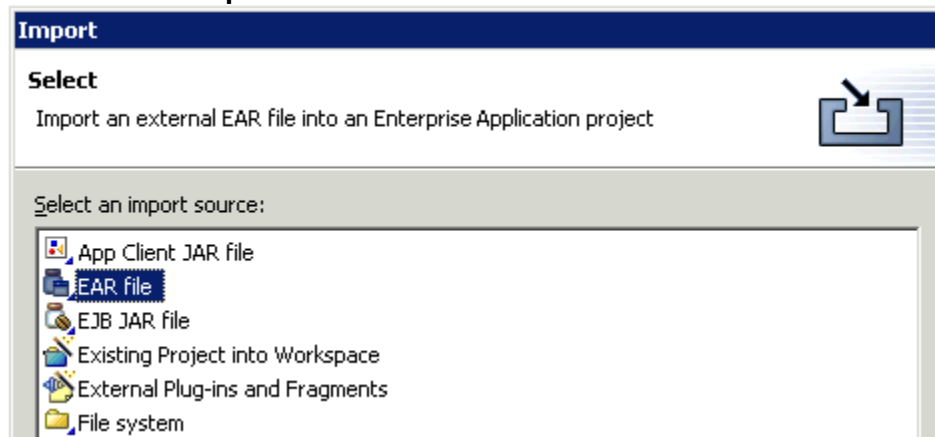
ADMA5011I: Cleanup of temp dir for app MyBankCMP done.  
ADMA5013I: Application MyBankCMP installed successfully.

- \_\_9. Type **\$AdminConfig save** and press Enter.
- \_\_10. Type **exit** in the Command Prompt to exit wsadmin.

### **Part Three: Import MyBank application source code into Application Developer**

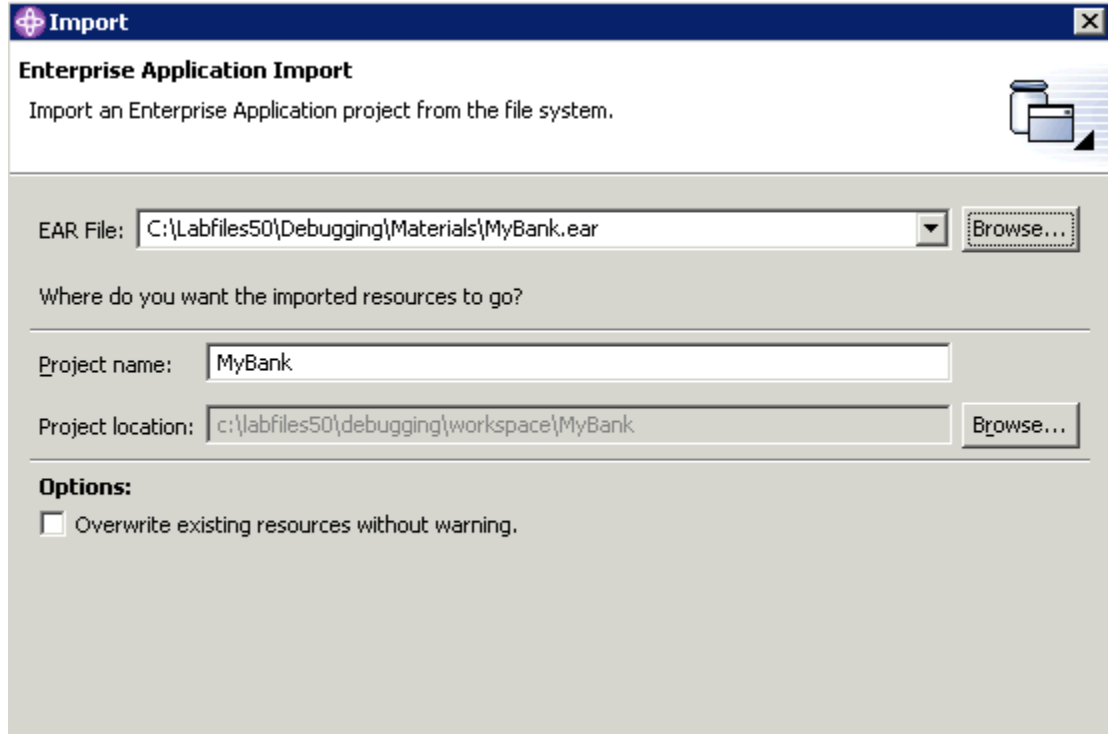
- \_\_1. Start WebSphere Studio Application Developer.
  - \_\_ a. Select **Start > Programs > IBM WebSphere Studio > Application Developer 5.1**.
  - \_\_ b. A dialog box will be displayed allowing you to select the location where you would like the workspace directory to be stored. Enter **C:\LabFiles50\Debugging\workspace** for the location and select **OK**. Application Developer will start with an empty workspace. An empty workspace will leave your existing workspace untouched and help avoid name conflicts between what you may already have in your workspace and what you will be creating in this lab.
- \_\_2. When WebSphere Application Studio Developer finishes loading, import the MyBank EAR file.

- \_\_ a. Select **File > Import...**



- \_\_ b. Select **Ear file** and click **Next**.
- \_\_ c. Select **Browse...** to the right of EAR file.

- \_\_ d. Navigate to `\LabFiles50\Debugging\Materials`. Select **MyBank.ear** and select **Open**.
- \_\_ e. Type **MyBank** for the Project Name.



- \_\_ f. Select **Finish**.

#### **Part Four: Configure WebSphere Application Server for debugging**

- \_\_ 1. Start Internet Explorer and go to WebSphere Application Server Administrative Console by entering <http://localhost:9090/admin> in the Address field.
- \_\_ 2. When the login screen comes up, fill in the User ID field with **wsdemo** and press **OK**.
- \_\_ 3. Expand Servers and select **Application Servers**.
- \_\_ 4. The available servers will be displayed on the right. Select the **server1** link in the Application Servers table.

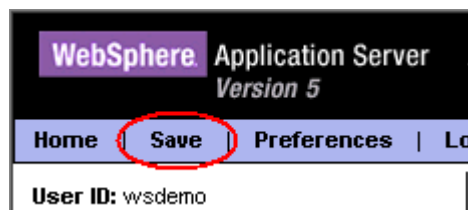
- \_\_5. The settings for server1 will now be displayed on the right. Scroll the window down and select **Process Definition** (see screen capture below) in the Additional Properties table.

<a href="#">Custom Services</a>	Define custom service classes that will run within this server properties.
<a href="#">Server Components</a>	Additional runtime components which are configurable.
<a href="#">Process Definition</a>	A process definition defines the command line information for a process.
<a href="#">Performance Monitoring Service</a>	specify settings for performance monitoring, including enabling and selecting the PML module and setting monitoring levels.

- \_\_6. The settings for the process that WebSphere runs in will now be displayed on the right. Scroll the window down and select **Java Virtual Machine** in the Additional Properties table.
- \_\_7. The additional setting for the JVM that WebSphere runs in will be displayed on the right. Scroll the window down and place a check the box next to **Debug Mode** (see screen capture below). Also take note of the values that are predefined in the Debug Arguments field. This is where the port number that debugging will take place on is declared.

		mac serv argu is en
<b>Debug Mode</b>	<input checked="" type="checkbox"/>	i S debu enab
Debug arguments	-Djava.compiler=NONE -Xdebug -Xnc	i S argu

- \_\_8. Scroll to the end of the page and select **OK**.
- \_\_9. Save the configuration.
- \_\_ a. Select **Save** at the top of the Administrative Console (see screen capture below).



- \_\_ b. Select **Save** again in the right panel to confirm that you want to permanently save the changes.

\_\_10. Select **Logout** at the top of the Administrative Console and then close the Administrative Console window.

\_\_11. At a Command Prompt, type **CD WebSphere\Appserver\bin**. Stop the WebSphere Application Server by typing **stopserver server1**.

```
ADMU0116I: Tool information is being logged in
           C:\WebSphere\AppServer\logs\server1\stopServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.
```

\_\_12. At a Command Prompt, ensure that you are in the **WebSphere\Appserver\bin** directory. Start the WebSphere Application Server by typing **startserver server1**.

```
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\logs\server1\startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1324
```

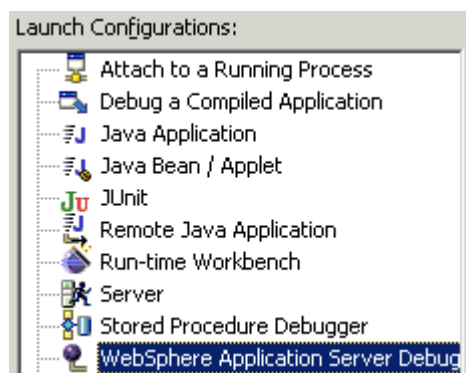
### **Part Five: Setup Application Developer to debug the remote server**

\_\_1. In Application Developer, open the Debug Perspective. Select **Window > Open Perspective > Debug**.

\_\_2. Create a launch configuration for the MyBank application on the remote server.

\_\_ a. Open the Launch Configuration window by selecting **Run > Debug...**

\_\_ b. In the Launch Configurations window, select **WebSphere Application Server Debug** (see screen capture below) from the panel on the left and select **New**.

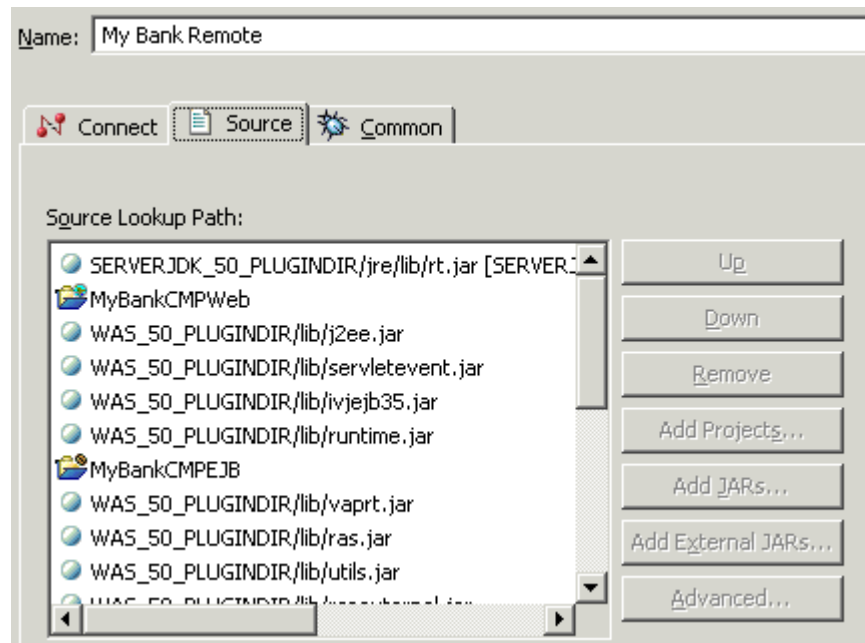


\_\_ c. On the right side, fill in the Name field with **My Bank Remote**.

- \_\_\_ d. In the **Connect** panel, select **Browse...** next to the Project field and select **MyBankCMPWeb**.

**NOTE:** If you have the debug port set to a different value other than the default, you will need to modify the JVM Debug Port setting on this panel.

- \_\_\_ e. Select the **Source** tab. Ensure the projects **MyBankCMPEJB** and **MyBankCMPWeb** are listed within the Source Lookup Path.



- \_\_\_ f. Select the **Common** tab. Check the **Debug** box under the Display in favorites menu section (see screen capture below).



- \_\_\_ g. Select **Apply**.

- \_\_\_ h. Select **Close**.

### **Part Six: Debug MyBank application on the remote server**

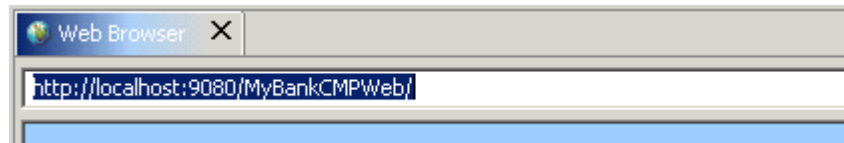
- \_\_\_ 1. Bring the Navigator view into the Debug Perspective. Select **Window > Show View > Other....** In the Show View window, expand **Basic** and select **Navigator**. Select **OK**.



- \_\_2. Access the web interface of the MyBank app. Open a web browser inside of Application Developer.
- \_\_ a. Open the Server Perspective. Select **Window > Open Perspective > Other...**
  - \_\_ b. In the Select Perspective window, select **Server**. Select **OK**.
  - \_\_ c. Select the **Open Web Browser** button (circled in the screen capture below) located in the toolbar.



- \_\_ d. When the web browser opens, point the browser to the MyBank application that is running on the remote application server. Fill in the address line with <http://localhost:9080/MyBankCMPWeb/> (see screen capture below) and press Enter.



- \_\_3. The portion of the application that you will want to debug is the Transfer functionality of the web site. To get an idea of what the problem is, you will first execute one transfer. To access the transfer functionality of the web site, select the **Transfer Funds** link.
- \_\_4. Fill in the table with the values specified in the screen capture below and select **Transfer**.

Amount	From Account	To Account
<input type="text" value="100"/>	<input type="text" value="1"/>	<input type="text" value="2"/>
	<b>Balance</b>	<b>Balance</b>
<input type="button" value="Transfer"/>	***	***
<input type="button" value="Clear"/>	<input type="button" value="Get Balance"/>	<input type="button" value="Get Balance"/>

- \_\_ a. Notice the new values that returned for the two accounts (account 1's starting balance was \$1000 and account 2's was \$2000). Instead of \$100 dollars being added and subtracted, \$200 has been added and subtracted from each account.
- \_\_5. Return to the **Debug Perspective** within Application Developer.

- \_\_6. Place a breakpoint in TransferFunds.java. The breakpoint will be placed on the line which calls the transferFunds function of the Transfer bean. In this way we can stop the execution of the program before the call is made to our Transfer bean. In the Navigator view, expand **MyBankCMPWeb/Java Source/com/ibm/mybank/web**. Double-click **TransferFunds.java** to open the file.
- \_\_ a. Open the Outline view by selecting **Window > Show View > Outline**. In the Outline view, select the **performTask** function.
- \_\_ b. Scroll the window down to the highlighted line indicated in the screen capture below (should be line 223).

```

try {
    // Create a Transfer object.
    transfer = transferHome.create();
    // Based on user request type, invoke appropriate transfer me
    if (request == 1) {
        transfer.transferFunds(fromKey, toKey, amountFloat);
        fromBalance = "" + formatFloat(transfer.getBalance(fromKey)
        toBalance = "" + formatFloat(transfer.getBalance(toKey))
        transferFlag = true;
    } else if (request == 2) {
        fromBalance = "" + formatFloat(transfer.getBalance(fromKey)
        transferFlag = true;
    }
}

```

- \_\_ c. Place your cursor on the line, and then select **Run > Add/Remove Breakpoint** (you could also double-click in the margin to the left of the line or press Ctrl+Shift+B).

```

try {
    // Create a Transfer object.
    transfer = transferHome.create();
    // Based on user request type, invoke appropriate transfer
    if (request == 1) {
        transfer.transferFunds(fromKey, toKey, amountFloat);
        fromBalance = "" + formatFloat(transfer.getBalance(fromK
        toBalance = "" + formatFloat(transfer.getBalance(toKey)
        transferFlag = true;
    } else if (request == 2) {
        fromBalance = "" + formatFloat(transfer.getBalance(fromK
    }
}

```

- \_\_ d. Select the Breakpoints view in the upper right panel. This view displays every breakpoint that you have set in your workspace. The breakpoint that you have just set will show up in this view.
- \_\_ e. Close the TransferFunds.java file.
- \_\_7. Also place a breakpoint in TransferFundsJSP.jsp, so that you are able to see how Application Developer is able to debug .jsp pages as well. In the TransferFunds JSP, you will place the breakpoint on the first line that is capable of being debugged. In the Navigator view, expand **MyBankCMPWeb/Web Content**. Double-click **TransferFundsJSP.jsp** to open the file.

**NOTE:** For a complete list of lines that can be debugged within a JSP, see Application Developer's Help Content.

- \_\_ b. Select the **Source** tab.
- \_\_ c. Scroll the window down to the highlighted line indicated in the screen capture below (should be line 13).

```
</HEAD>
<BODY>

  <jsp:useBean id="JSPBean"
              type="com.ibm.mybank.web.JSPBean"
              scope="session">
    </jsp:useBean>

  <H1 ALIGN="center"> Transfer Funds between Accounts </H1>
```

```
</HEAD>
<BODY>

  <jsp:useBean id="JSPBean"
              type="com.ibm.mybank.web.JSPBean"
              scope="session">
    </jsp:useBean>

  <H1 ALIGN="center"> Transfer Funds between Accounts </H1>
```

- \_\_ d. Place your cursor on the line. Right-click on the highlighted line and select **Add Breakpoint**.

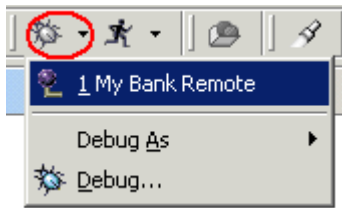
```
</HEAD>
<BODY>

  <jsp:useBean id="JSPBean"
              type="com.ibm.mybank.web.JSPBean"
              scope="session">
    </jsp:useBean>

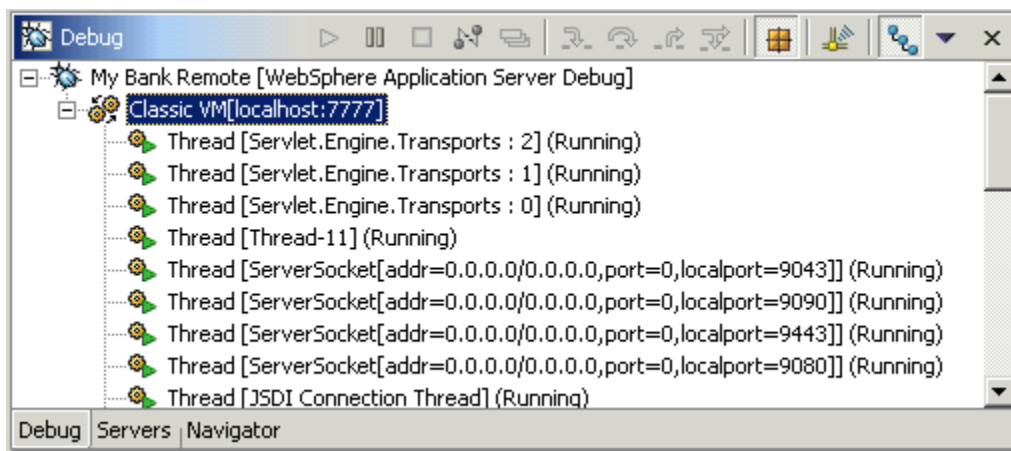
  <H1 ALIGN="center"> Transfer Funds between Accounts </H1>
```

- \_\_ e. Notice the breakpoint that you have just added also shows up in the Breakpoints view.
  - \_\_ f. Close the TransferFundsJSP.jsp file.
- \_\_ 8. Start debugging the application by launching the Launch Configuration created in Part Five. Select the arrow next to the **Debug** button (see screen capture below) in the

Application Developer's toolbar and select **My Bank Remote**.



- \_\_9. When the launch configuration is selected, Application Developer attaches to the running Application Server using the debug port specified in the launch configuration. The process where the application server is running is then brought into the Debug view (see screen capture below). Select the **Debug** view in the upper left panel of the Debug perspective. From this view you can control the execution of the MyBank application.

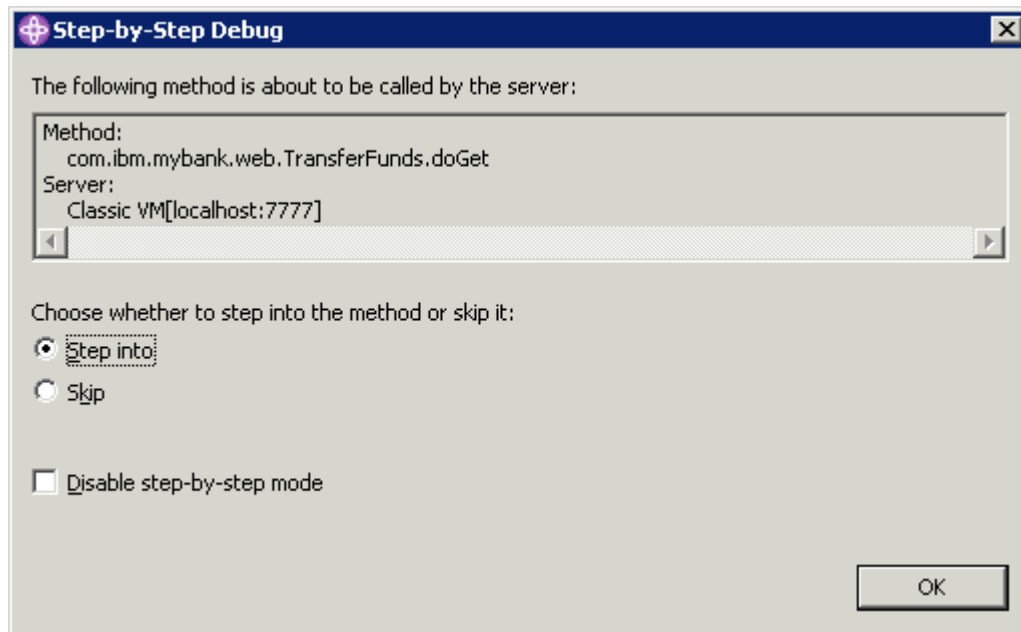



- \_\_10. In the Web Browser window, fill in the table with the values specified in the screen capture below and select **Transfer**.

Amount	From Account	To Account
100	2	1

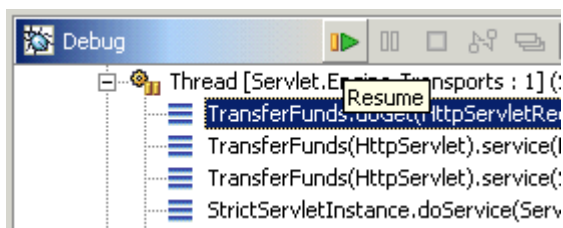
- \_\_11. The Step by Step Debug window will be displayed. This box is displayed every time a new object is placed on the stack. In this case the application server is placing the TransferFunds object on the stack. Leave the radio button next to **Step Into** and select **OK**. This will make Application Developer display the TransferFunds source code with

the first line of the doGet method highlighted.

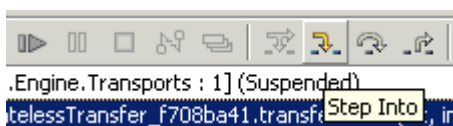


**NOTE:** When you start debugging the application, debugging is started in step-by-step mode. As each object (that is not filtered) is loaded onto the stack, a dialogue box pops up and asks the user if he/she wants to step into or skip over the object being loaded. This allows the user to quickly move through his/her code at a high level until the problematic class or object is reached. This can be disabled or enabled by selecting the  button in the Debug view.

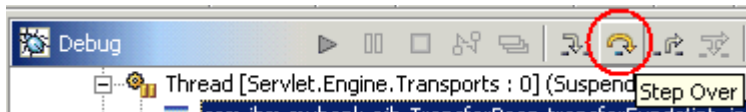
- \_\_12. After the TransferFunds class is displayed, switch to the Debug view and press **Resume** to continue execution until your breakpoint is reached.



- \_\_13. Press the **Step Into** button in the Debug view to step into this function call.

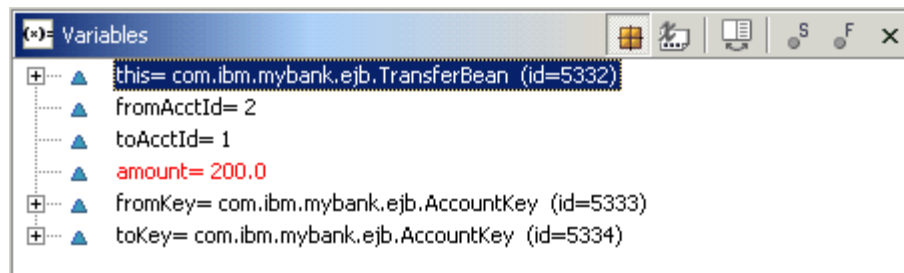


- \_\_14. The Step Into function, will take you into the EJB deployment code. Because you don't want to debug the deployment code source, press **Resume** again to execute the deployment code until it reaches the call to the TransferBean.transferFunds method.
- \_\_ a. The Step-by-Step Debug window will be displayed, indicating that it wants to step into the TransferBean.transferFunds method. Leave the radio button next to **Step Into** and select **OK**.
- \_\_15. The Step-by-Step function will take you into the transferFunds function of the TransferBean class. Make sure that the **Variables** view is displayed. In the upper right panel, select the **Variables** tab to display the view. The view displays the current variables active on the stack. Notice the values of fromAcctId, toAcctId, and amount. These are the values that you initially entered into the table.
- \_\_16. Press **Step Over** in the Debug view until you reach the following line:



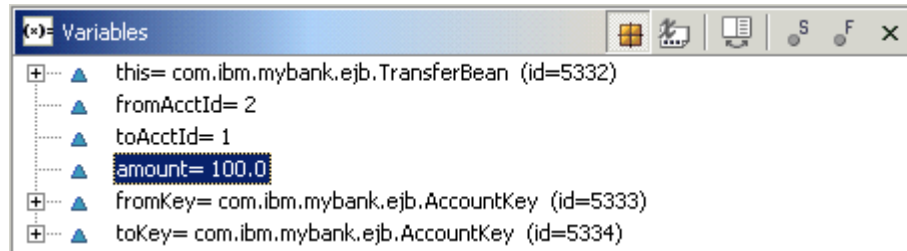
```
amount=(float)someFunction(amount);
```

- \_\_17. Again press **Step Over** in the Debug view, but pay special attention to the value of **amount** in the Variables view after the button is pressed.
- \_\_ a. Notice that after the line is executed, the amount variable in the Variables view changes to red indicating that the value has changed. If you look at what the someFunction does you will see that it simply multiplies the value of amount by two and returns it.



- \_\_18. Change the value of the amount back to the amount that was entered. In the Variables view, right-click on amount=200.0 and select **Change Variable Value**.

- \_\_\_ a. Type the value **100.0** and press Enter. The end result should look like the screen capture below.



- \_\_\_ 19. Disable Step by Step debugging by pressing the Disable Step by Step Debugging button in the Debug view (see screen capture below). By disabling this feature, Application Developer will no longer interrupt the execution of the application every time a new object is placed on the stack.

**NOTE:** Step by step debugging is enabled when the Step by Step button is pressed in, and disabled when the button is pushed back out.



- \_\_\_ 20. Press **Resume** to continue execution until the breakpoint that you set in TransferFundsJSP.jsp file is reached. When the breakpoint is reached the TransferFundsJSP.jsp file will be opened in the Debug perspective. Step through the JSP file and take note of how Application Developer moves through the JSP file in debug mode.

- \_\_\_ a. Press **Step Over** to move to the next line that can be debugged in the JSP.

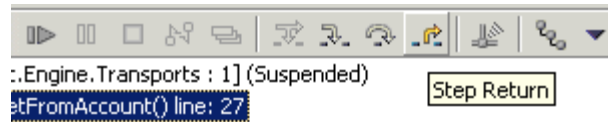
```

NAME="fromaccount"
SIZE="20"
MAXLENGTH="10"
VALUE=<jsp:getProperty name="JSPBean" property="fromAccount" />>
>
ALIGN="center">
INPUT TYPE="text"

```

- \_\_\_ b. Press **Step Into** to step into the JSPBean.getFromAccount method.

- \_\_ c. Press **Step Return** to run to end of the method and to return back to TransferFundsJSP.jsp.

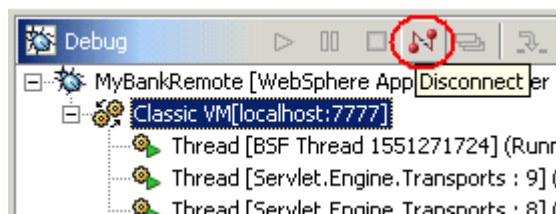


- \_\_ d. Press **Resume** to let the transfer funds action carry out to completion.

- \_\_21. Switch back to the Web Browser. Notice now that the amount added and subtracted from the two accounts was 100 (account 1's starting balance was \$800 and account 2's was \$2200 before the transaction), which was the amount that you specified while debugging. (If you got an application error after trying to transfer, you might have experienced some kind of JSP timeout error while debugging. In this case, you would attempt to transfer the funds again.)

Amount	From Account	To Account
<input type="text"/>	<input type="text" value="2"/>	<input type="text" value="1"/>
	<b>Balance</b>	<b>Balance</b>
<input type="button" value="Transfer"/>	2100.00	900.00

- \_\_22. Detach Application Developer from the remote application server. In the Debug view, **select** the line indicated in the screen capture below and press **Disconnect** (indicated in the screen capture below). This will only disconnect Application Developer from the remote application server, and not terminate the running application server. The server will remain in debug mode as well. To reconnect to the debugger, you would have to re-select the arrow next to the Debug icon and select My Bank Remote.



- \_\_23. Close Application Developer by selecting **File > Exit**.



## **Part Seven: Lab Cleanup**

- \_\_1. Disable debugging on your WebSphere Application Server by running the `disableDebugging.bat` file. At a Command Prompt, type **CD \LabFiles50\Debugging**. Type **disableDebugging <NODENAME> <SERVERNAME>**. For example, for this lab you would be typing: **disableDebugging NODENAME server1**, where “NODENAME” is your hostname.

**NOTE:** The parameter <NODENAME> is case-sensitive.

- \_\_2. Uninstall the MyBank application from the WebSphere Application Server. At a Command Prompt, type **CD \WebSphere\Appserver\bin**. Type **wsadmin -c “\$AdminApp uninstall MyBankCMP”** to uninstall the application.
- \_\_3. At a Command Prompt, ensure that you are in the **\WebSphere\Appserver\bin** directory. Stop the WebSphere Application Server by typing **stopserver server1**.

```
ADMU0116I: Tool information is being logged in
           C:\WebSphere\AppServer\logs\server1\stopServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.
```

## **What you did in this exercise**

In this lab exercise, you used Application Developer to attach to a remote WebSphere Application Server that was running in debug mode. Once connected, you used the step by step debug feature to move through the MyBank application at a high level. At a lower level, you used breakpoints and the step over, step into, and change variable value functions to effectively debug the MyBank application.