© Copyright IBM Corporation 2005 All rights reserved

IBM RATIONAL APPLICATION DEVELOPER 6.0 – LAB EXERCISE

Component Test with WebSphereBank

What this exercise is about

The objective of this lab is to provide you with an understanding of the tools support for the automated component test. This lab also demonstrates how to use this technology in association with the Profiling support

Lab Requirements

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

- IBM Rational Application Developer v6.0 with embedded WebSphere Application Server v6.0 Test Environment.
- Rational Agent Controller installed and configured
- Lab source files (Labfiles60.zip) must be extracted to the local system (preferably C:\)

What you should be able to do

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

- Import an existing EAR file
- Create a new Component Test Project
- Create and run an EJB Component Test

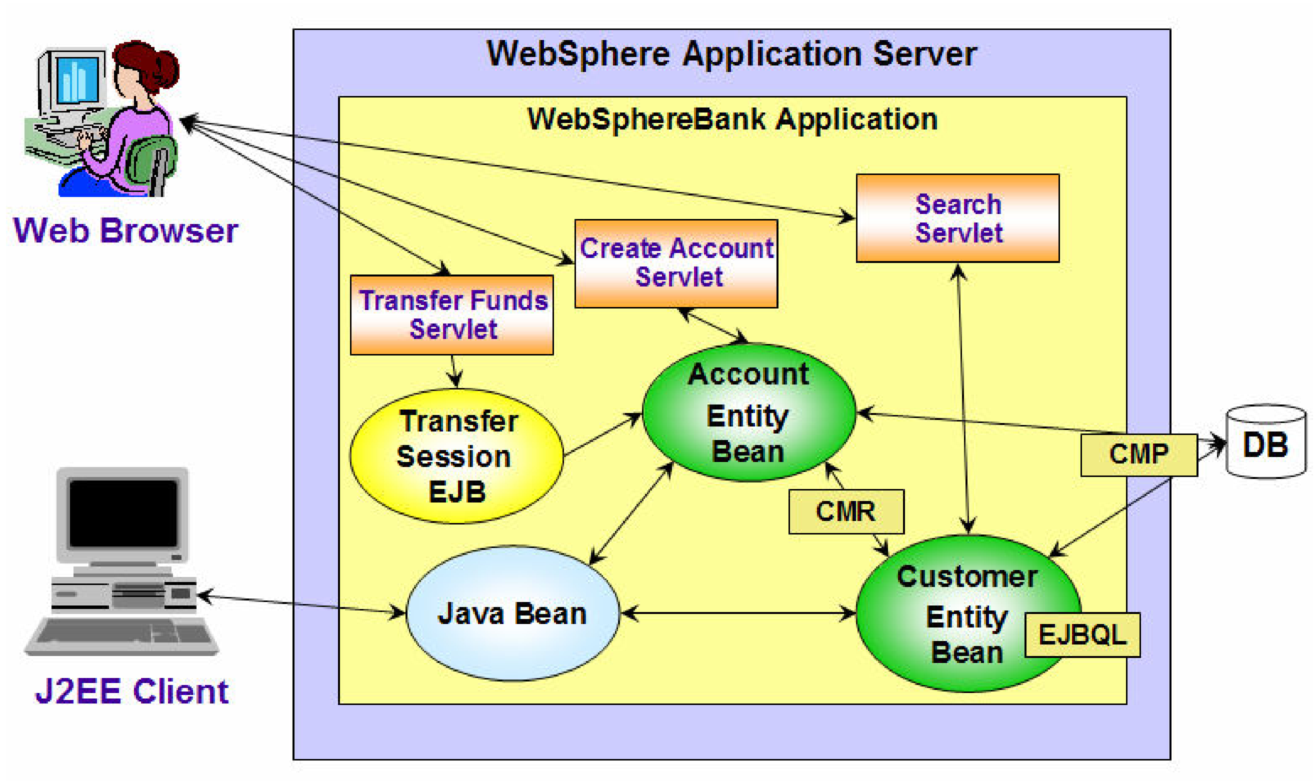
© Copyright IBM Corporation 2005. All rights reserved

- Review results of running an EJB Component Test
- Use profiling support to analyze code and line level coverage of a component test run

Introduction

The automated component test features available in the Rational Developer v6 products help improve the overall quality of Java and J2EE applications by helping developers find defects more easily. The component test capabilities provide an end to end test solution by helping developers create, edit, run, and view results for component tests. Another important advantage of the component test features is that it facilitates the reuse of test suites so that they can be used throughout the development process.

This exercise will highlight how to use the automated component testing features available with IBM Rational Application Developer v6. Specifically, you will use these features to test the WebSphereBank enterprise application. You will start by importing the WebSphereBank EAR file and deploying this application to the application server. You will then create a new Component Test Project to test the business methods on the Transfer session bean in the WebSphereBank application. As a final step in this exercise, you will run a component test using the profiling capabilities to analyze the method and line level code coverage for the component test.



Exercise Instructions

Some instructions in this lab may be Windows operating-system specific. If you plan on running the lab on an operating-system other than Windows, you will need to execute the appropriate commands, and use appropriate files (.sh vs. .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references, as follows:

© Copyright IBM Corporation 2005. All rights reserved

Reference Variable	Windows Location	AIX/UNIX Location
<WAS_HOME>	C:\WebSphere60\AppServer	/usr/WebSphere60/AppServer /opt/WebSphere60/AppServer
<IRAD_HOME>	C:\Program Files\IBM\Rational\SDP\6.0	
<LAB_FILES>	C:\Labfiles60	/tmp/Labfiles60
<TEMP>	C:\temp	/tmp
<LAB_NAME>	IRAD_ComponentTest	

Windows users please note: When directory locations are passed as parameters to a Java program such as EJBdeploy or wsadmin, it is necessary to replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles60\ would be replaced by C:/LabFiles60/

© Copyright IBM Corporation 2005. All rights reserved

Part 1: Import the WebSphereBank EAR file

____ 1. Start IBM Rational Application Developer.

__ a. Select **Start > Programs > IBM Rational > Rational Software Development Platform**.

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

__ c. When IBM Rational Application Developer v6 opens, close the welcome page.

NOTE: If the Auto Launch Configuration Change Alert window appears click the **Yes** button to change the auto launch eclipse instance to use when opening IBM Rational Software Development Platform in the future.

____ 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_ComponentTest\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 start working with the EJB Mediator support.

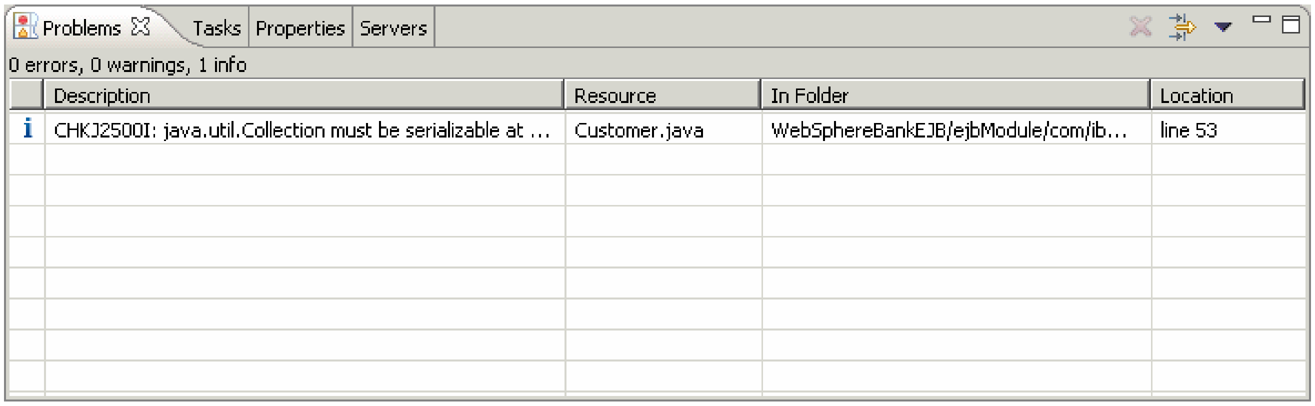
__ a. In the Project Explorer view, expand Dynamic Web Projects and right click on **WebSphereBankWeb**. From the context menu select **Properties**.

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

__ c. Click **OK**.

© Copyright IBM Corporation 2005. All rights reserved

__ d. Verify that there are no errors in the Problems view. Your Problems view should look like the following screen capture:



0 errors, 0 warnings, 1 info

	Description	Resource	In Folder	Location
i	CHKJ2500I: java.util.Collection must be serializable at ...	Customer.java	WebSphereBankEJB/ejbModule/com/ib...	line 53

© Copyright IBM Corporation 2005. All rights reserved

Part 2: Application and Cloudscape Setup

____ 1. Backup your server configuration. This will preserve your current server configuration. You will restore your server configuration at the end of the lab exercise.

__ a. Open a Windows **Command Prompt** and navigate to the following directory:

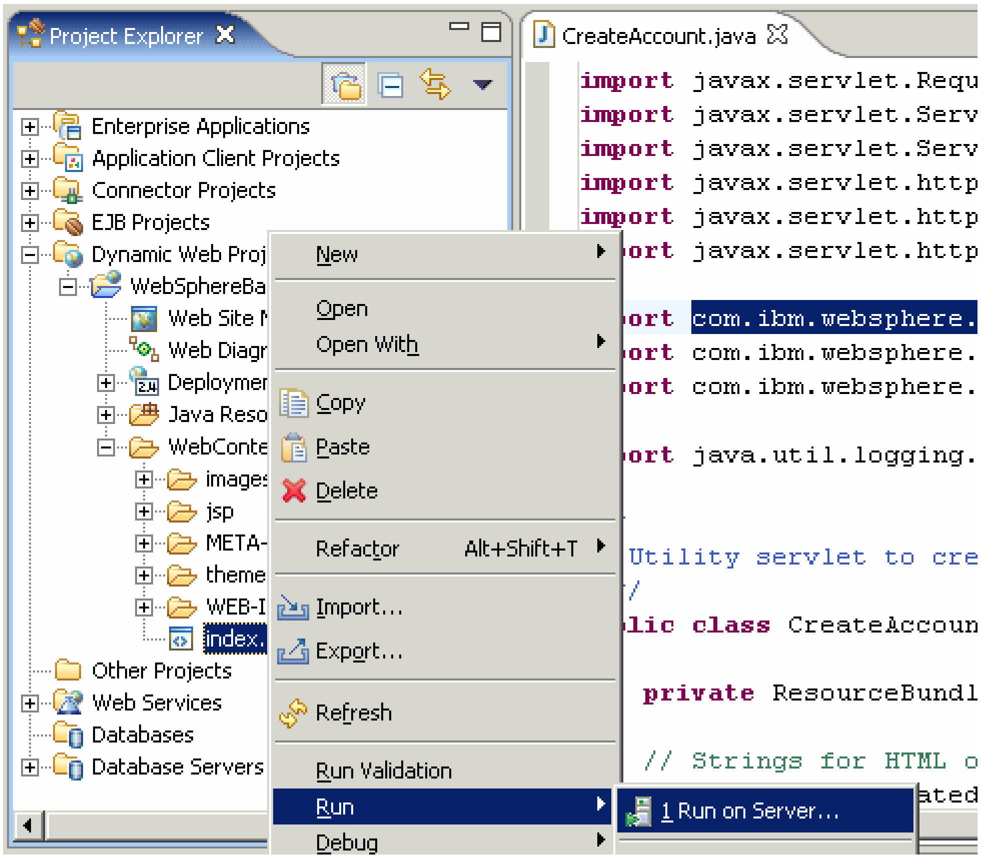
```
<RAD_HOME>\runtimes\base_v6\bin
```

__ b. Backup the server configuration by issuing the following command:

```
backupConfig "C:\Program Files\IBM\Backup.zip"
```

____ 2. Start the server with the WebSphereBank project while initializing the database and datasource.

__ a. In the Project Explorer view, navigate to **Dynamic Web Projects** > **WebSphereBankWeb** > **WebContent** and right click on index.html.

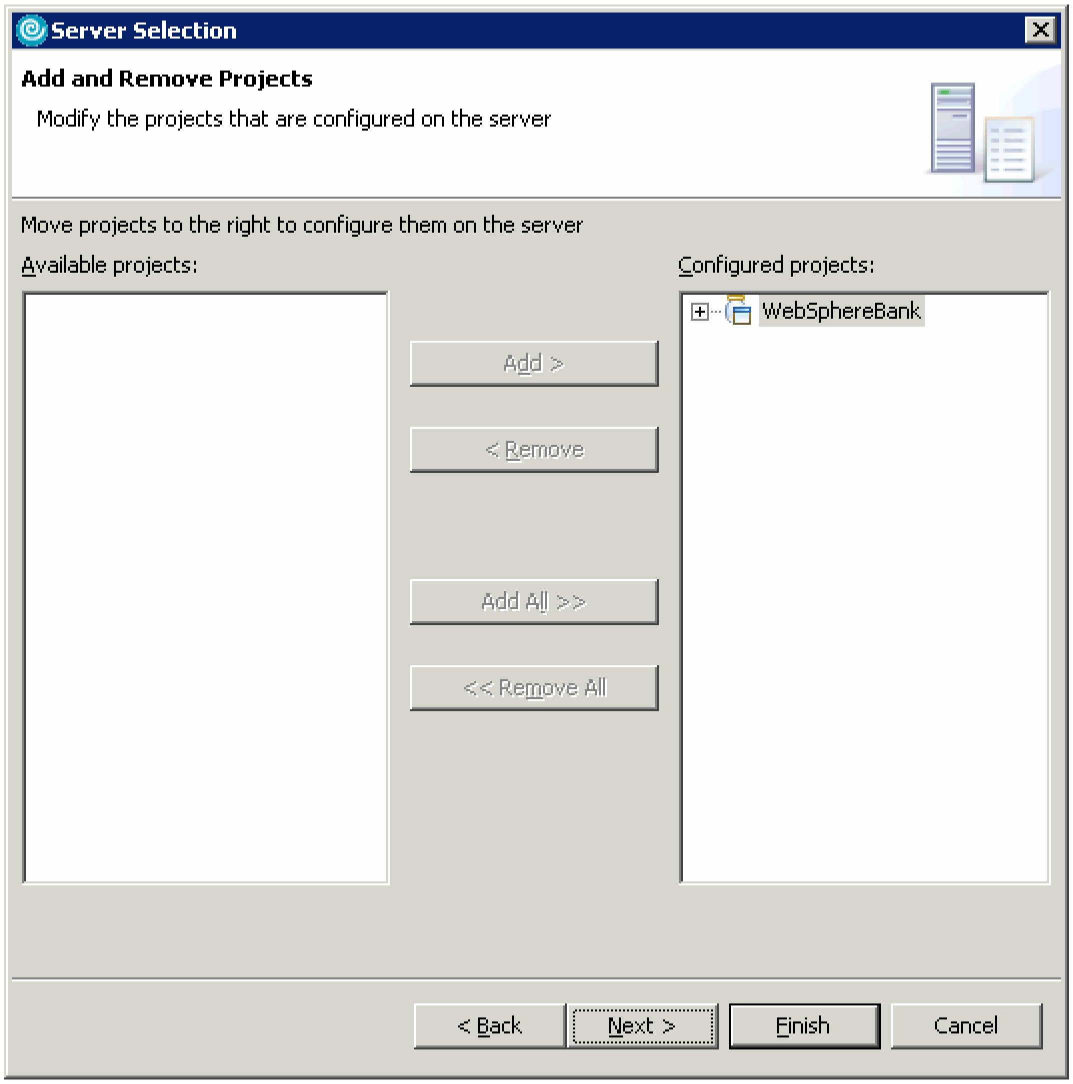


```
import javax.servlet.Requ
import javax.servlet.Serv
import javax.servlet.Serv
import javax.servlet.http
import javax.servlet.http
ort javax.servlet.http

ort com.ibm.websphere.
ort com.ibm.websphere.
ort com.ibm.websphere.

ort java.util.logging.

 Utility servlet to cre
/
lic class CreateAccoun

 private ResourceBundl

 // Strings for HTML o
ated
```

__ b. Select the **Run** > **Run on Server** option to open the Server Selection window.

__ c. On the Define a New Server page of the Server Selection window make sure the **Choose an existing server** option is selected as well as the server WebSphere Application Server v6.0. Then click on the **Next** button.

__ d. On the **Add and Remove Projects** page, make sure that WebSphereBank is added to the Configured projects list on the right side of the window. Then click on the **Next** button.


Server Selection

Add and Remove Projects

Modify the projects that are configured on the server

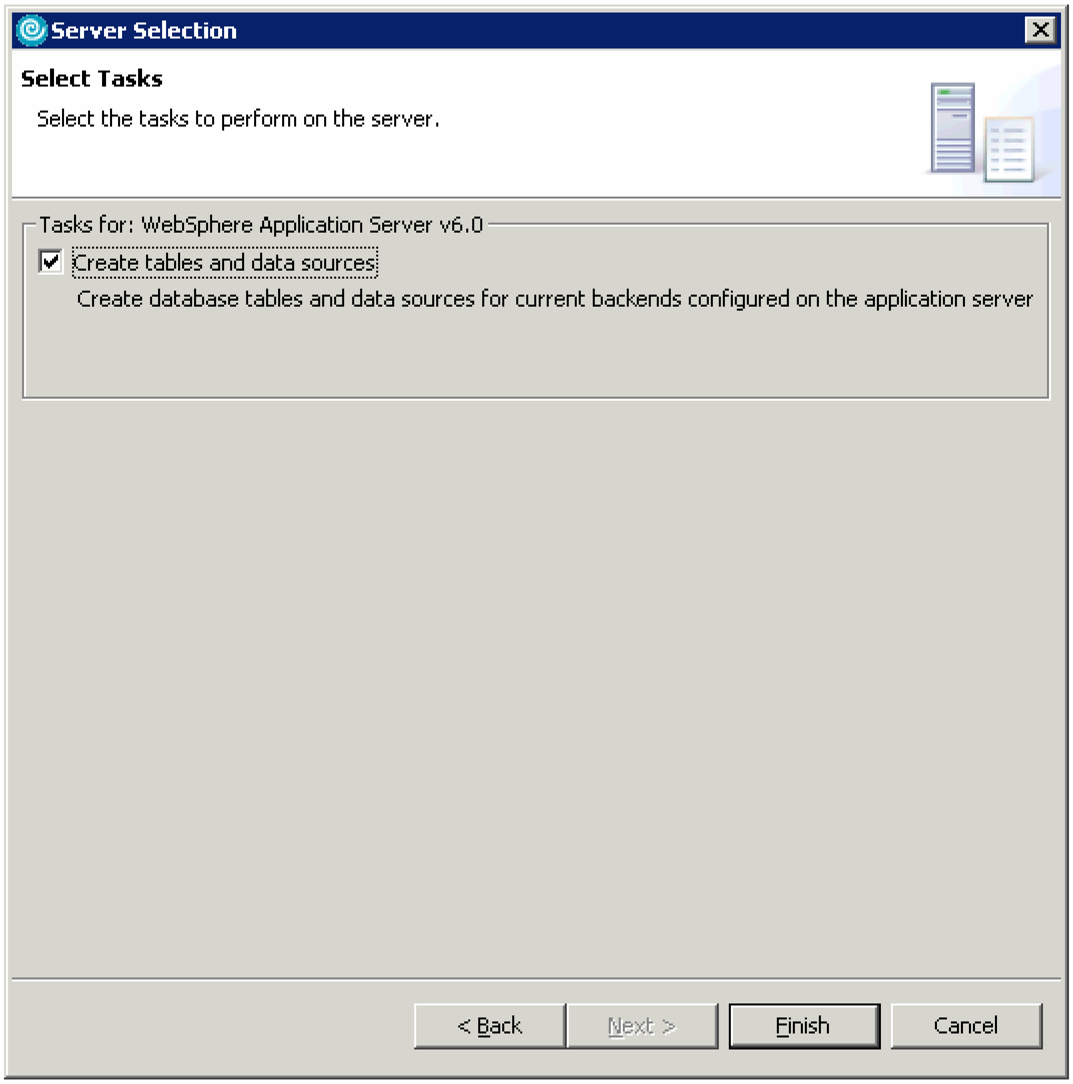
Move projects to the right to configure them on the server

Available projects:

Configured projects:

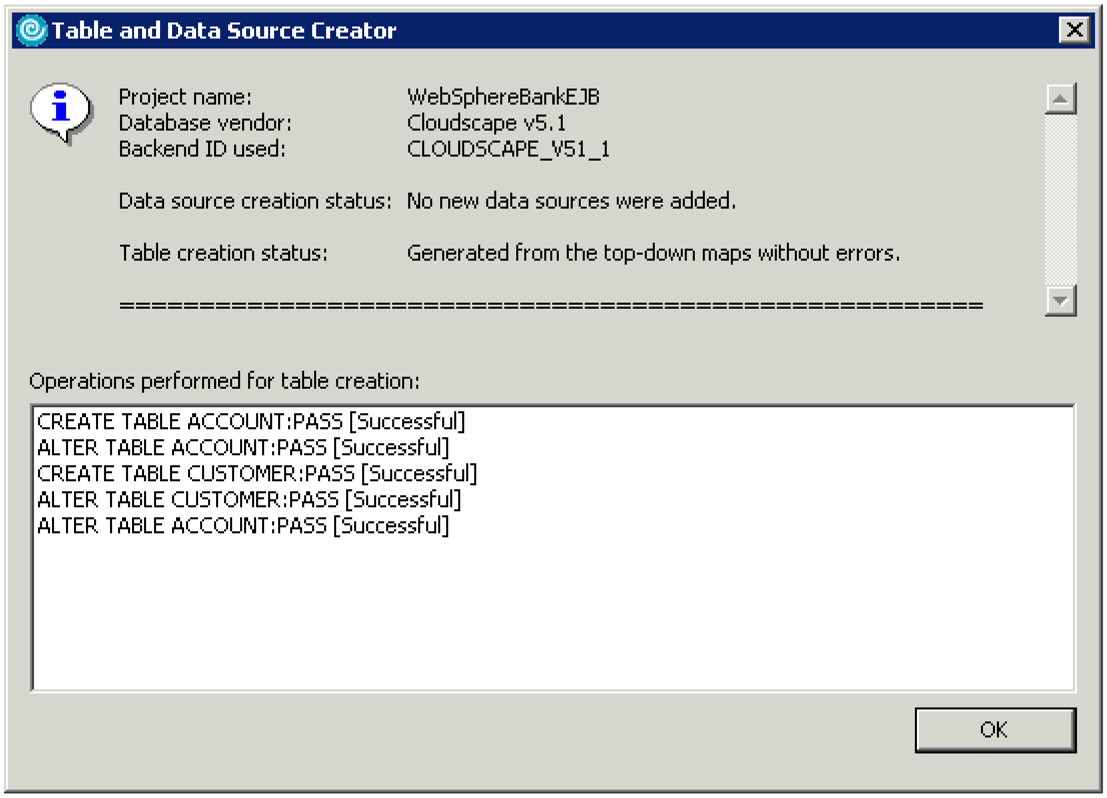
WebSphereBank

__ e. On the **Select Tasks** page, click on the Create tables and data sources checkbox. Then click the **Finish** button.


Server Selection
Select Tasks
Select the tasks to perform on the server.
Tasks for: WebSphere Application Server v6.0
☑ Create tables and data sources
Create database tables and data sources for current backends configured on the application server
< Back
Next >
Finish
Cancel

© Copyright IBM Corporation 2005. All rights reserved

__ f. You should see a window like the one below if the database has been setup successfully.



```
Project name:                 WebSphereBankEJB
Database vendor:              Cloudscape v5.1
Backend ID used:              CLOUDSCAPE_V51_1

Data source creation status:  No new data sources were added.

Table creation status:        Generated from the top-down maps without errors.

========================================================
```

Operations performed for table creation:

```
CREATE TABLE ACCOUNT:PASS [Successful]
ALTER TABLE ACCOUNT:PASS [Successful]
CREATE TABLE CUSTOMER:PASS [Successful]
ALTER TABLE CUSTOMER:PASS [Successful]
ALTER TABLE ACCOUNT:PASS [Successful]
```

__ g. Click **OK**.

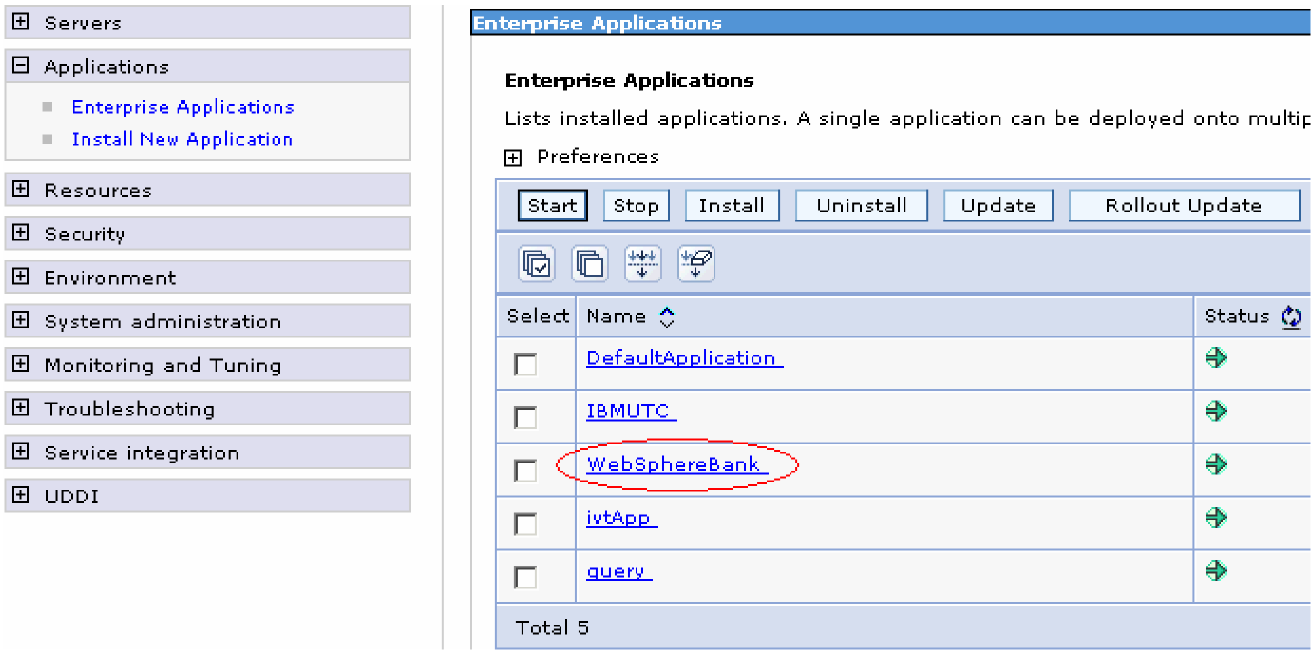
____ 3. Go to the administrative console and wait for WebSphereBank application to show up.

__ a. Open a Web Browser and navigate to the following URL:

`http://localhost:9060/ibm/console`

__ b. When prompted for a User ID, enter **wsdemo** to log in.

__ c. Select Applications -> Enterprise Applications. Watch for WebSphereBank to appear in the list of applications.



Enterprise Applications

Lists installed applications. A single application can be deployed onto multip

Select	Name	Status
☐	DefaultApplication	➜
☐	IBMUTC	➜
☐	WebSphereBank	➜
☐	ivtApp	➜
☐	query	➜

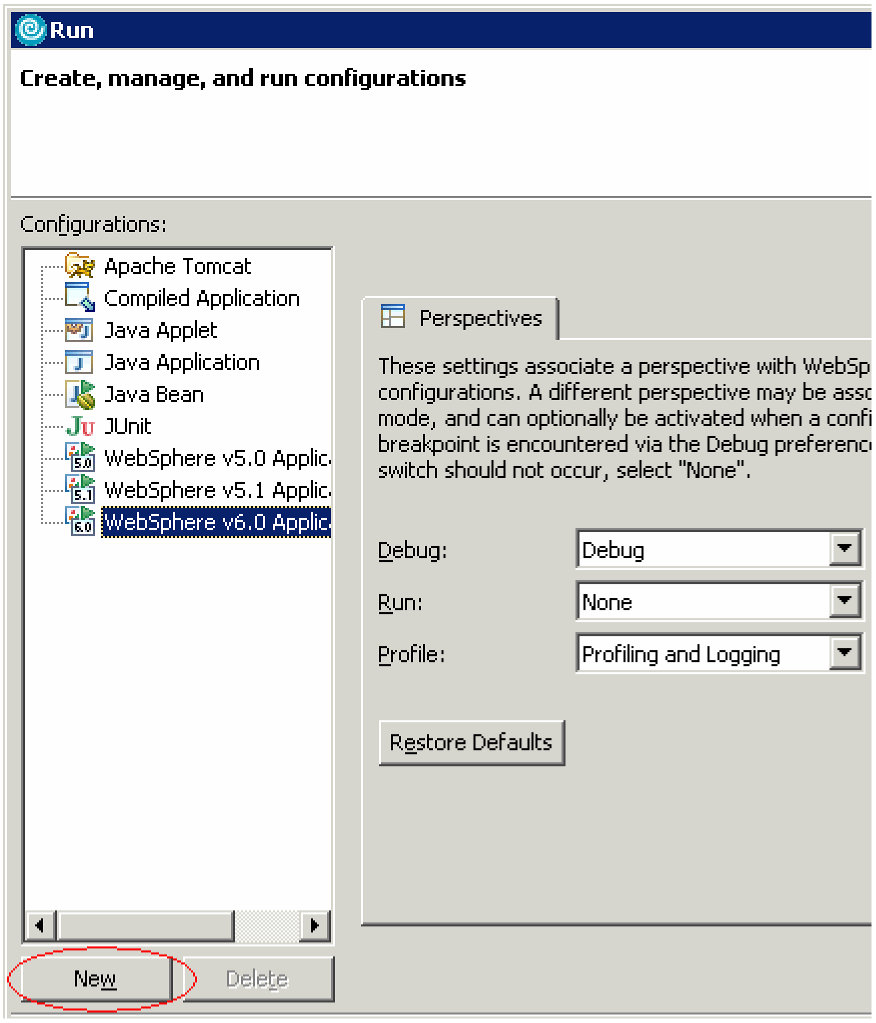
Total 5

____ 2. Populate the WebSphereBank database with Customer and Account data. For your convenience, and Application Client Project called CreateAccounts has been included with the WebSphereBank EAR file you imported at the beginning of this lab exercise.

__ a. From the Project Explorer view, expand **Application Client Projects > CreateAccounts > com.ibm.websphere.samples.bank.clients**.

__ b. Right click on CreateAccounts.java, and select Run > **Run** from the context menu.

__ c. In the Configurations list, select WebSphere v6.0 Application Client and click the New button at the bottom of the list.


Run

Create, manage, and run configurations

Configurations:

- Apache Tomcat
- Compiled Application
- Java Applet
- Java Application
- Java Bean
- JUnit
- WebSphere v5.0 Applic
- WebSphere v5.1 Applic
- WebSphere v6.0 Applic

Perspectives

These settings associate a perspective with WebSp
configurations. A different perspective may be asso
mode, and can optionally be activated when a confi
breakpoint is encountered via the Debug preferenc
switch should not occur, select "None".

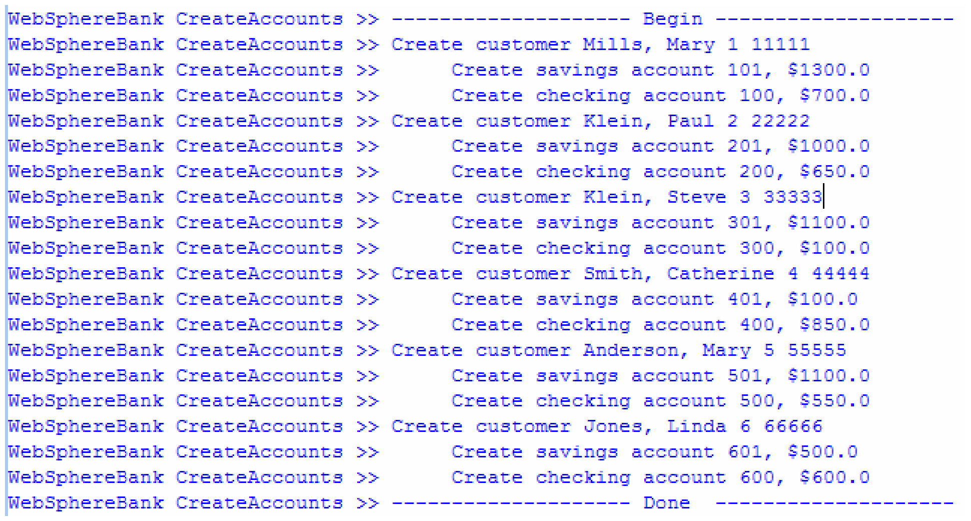
Debug: Debug

Run: None

Profile: Profiling and Logging

__ d. Enter CreateAccounts for the Name and click **Run**.

__ e. When the process has completed you should see the following message in the Console view.



```
WebSphereBank CreateAccounts >> -------------------- Begin --------------------
WebSphereBank CreateAccounts >> Create customer Mills, Mary 1 11111
WebSphereBank CreateAccounts >>      Create savings account 101, $1300.0
WebSphereBank CreateAccounts >>      Create checking account 100, $700.0
WebSphereBank CreateAccounts >> Create customer Klein, Paul 2 22222
WebSphereBank CreateAccounts >>      Create savings account 201, $1000.0
WebSphereBank CreateAccounts >>      Create checking account 200, $650.0
WebSphereBank CreateAccounts >> Create customer Klein, Steve 3 33333
WebSphereBank CreateAccounts >>      Create savings account 301, $1100.0
WebSphereBank CreateAccounts >>      Create checking account 300, $100.0
WebSphereBank CreateAccounts >> Create customer Smith, Catherine 4 44444
WebSphereBank CreateAccounts >>      Create savings account 401, $100.0
WebSphereBank CreateAccounts >>      Create checking account 400, $850.0
WebSphereBank CreateAccounts >> Create customer Anderson, Mary 5 55555
WebSphereBank CreateAccounts >>      Create savings account 501, $1100.0
WebSphereBank CreateAccounts >>      Create checking account 500, $550.0
WebSphereBank CreateAccounts >> Create customer Jones, Linda 6 66666
WebSphereBank CreateAccounts >>      Create savings account 601, $500.0
WebSphereBank CreateAccounts >>      Create checking account 600, $600.0
WebSphereBank CreateAccounts >> -------------------- Done  --------------------
```

© Copyright IBM Corporation 2005. All rights reserved

Part 3: Create a Component Test

____ 1. Create a new component test project.

__ a. From the menu select **File** > **New** > **Other** > **Component Test** > **Component Test Project** and click **Next.**

NOTE: You may need to click the check box next to Show All Wizards.

__ b. Click OK for the Confirm Enablement dialog that enables component test capabilities.

__ c. Enter **WebSphereBankTest** for the project name and click Next

__ d. Check the box next to **WebSphereBankEJB** for the test scope for this component test project and click **Finish**

__ e. When the Confirm Perspective Switch dialog appears, click **Yes**

__ f. Notice that the Test Perspective is now open

____ 2. Enable capabilities for EJB component test.

__ a. From the menu select **Window > Preferences**.

__ b. Expand Workbench and click on **Capabilities**.

__ c. In the capabilities list expand Advanced J2EE and check the box next to **Component Test for EJB**.

__ d. Click **OK**.

____ 3. Create a component test for business methods on the Transfer session bean.

__ a. From the Test navigator view, right click on WebSphereBankTest and select **New > Component Test**.

__ b. From the wizards list, expand EJB and select EJB Component Test. Click **Next**.

__ c. From the Test projects list select **WebSphereBankTest** and click **Next**

__ d. Check the box next to the Transfer session bean. For the Test name and location accept the defaults and click **Next**.

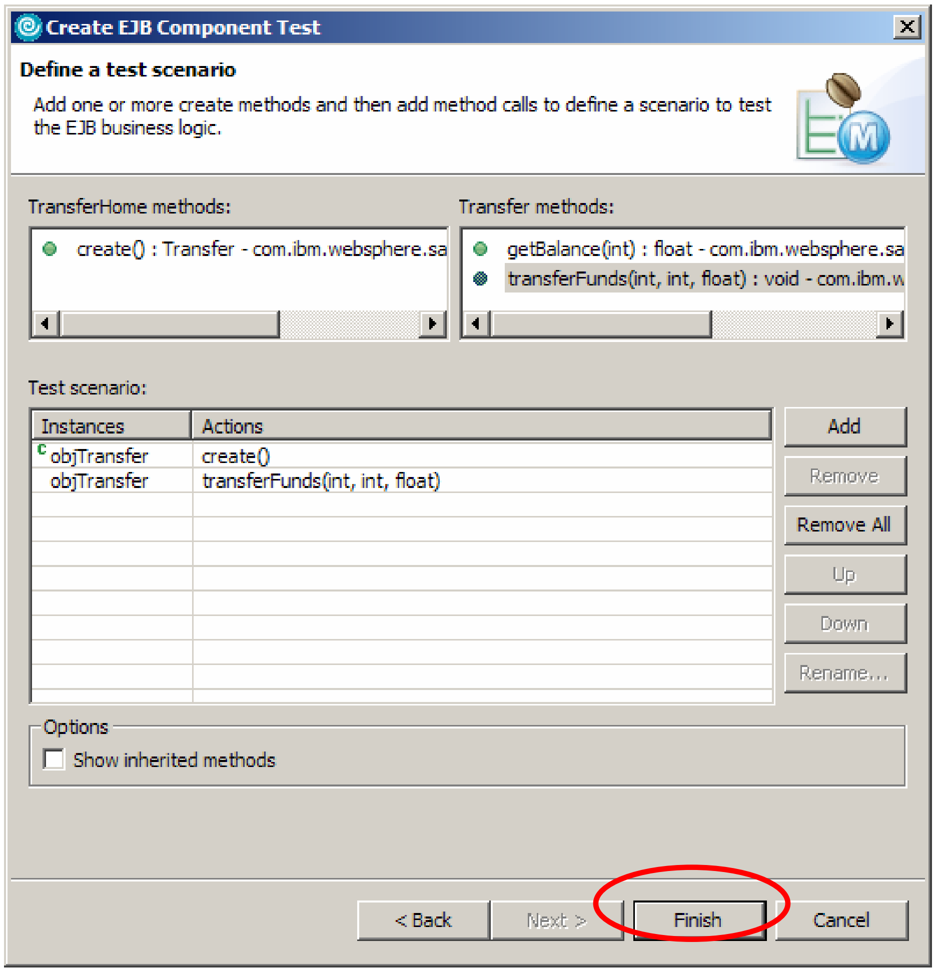
__ e. From the available test patterns list select **EJB business logic testing** and click **Next**.

__ f. For this test we will test the remote interfaces, check the radio button next to Test remote interfaces, and Click **Next**.

__ g. From the TransferHome methods list, select **create()** and click the **Add** button.

__ h. From the Transfer methods list select **transferFunds(int, int, float)** and click the **Add** button.

__ i. Click **Finish**.



Create EJB Component Test

Define a test scenario

Add one or more create methods and then add method calls to define a scenario to test the EJB business logic.

TransferHome methods:

- ● create() : Transfer - com.ibm.websphere.sa

Transfer methods:

- ● getBalance(int) : float - com.ibm.websphere.sa
- ● transferFunds(int, int, float) : void - com.ibm.w

Test scenario:

Instances	Actions
c objTransfer	create()
objTransfer	transferFunds(int, int, float)

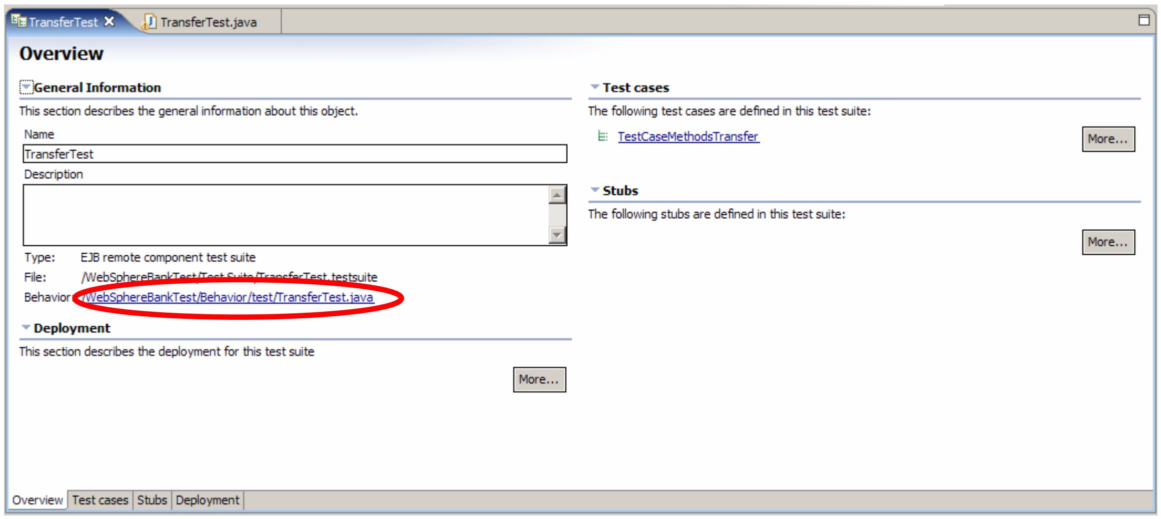
Options

☐ Show inherited methods

____ 4. Browse the Test Suite Editor and Behavior file for the component test created in the previous step.

© Copyright IBM Corporation 2005. All rights reserved

__ a. After creating the component test in the previous step the Test Suite Editor and Behavior file should automatically open. On the Overview tab of the Test Suite Editor, notice the link to the behavior file.



This section describes the general information about this object.

Name: TransferTest

Description

Type: EJB remote component test suite

File: /WebSphereBankTest/Test Suite/TransferTest.testsuite

Behavior: /WebSphereBankTest/Behavior/test/TransferTest.java

This section describes the deployment for this test suite

The following test cases are defined in this test suite:

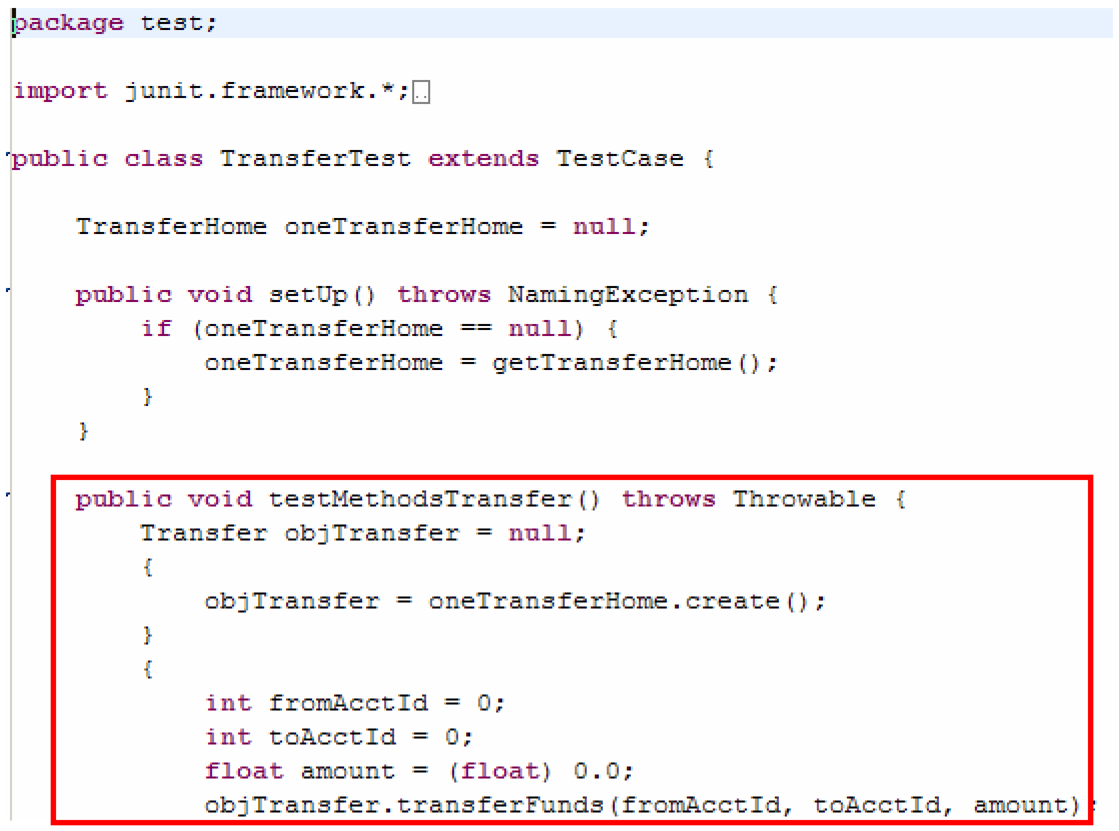
TestCaseMethodsTransfer

The following stubs are defined in this test suite:

____ 6. Open the Test Data Table view for the TestCaseMethodsTransfer test case

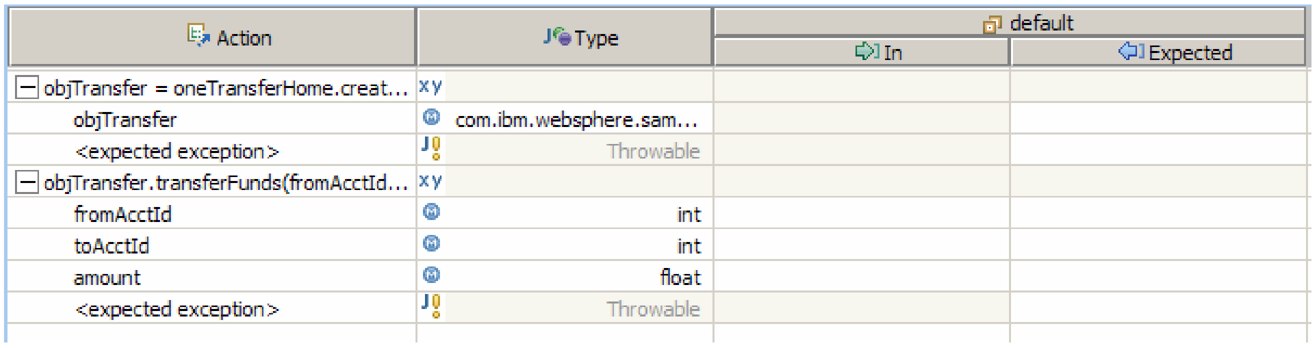
__ a. From the Test Suite Editor click on the behavior link to open the behavior file.

__ b. In order to open the Test Data Table for a particular data set, you need to click in the appropriate method in the behavior file. In this case, click inside the testMethodsTransfer() method within the behavior file. The following screen capture is the TransferTest behavior file and the testMethodsTransfer() method has been highlighted.



```
package test;

import junit.framework.*;

public class TransferTest extends TestCase {

    TransferHome oneTransferHome = null;

    public void setUp() throws NamingException {
        if (oneTransferHome == null) {
            oneTransferHome = getTransferHome();
        }
    }

    public void testMethodsTransfer() throws Throwable {
        Transfer objTransfer = null;
        {
            objTransfer = oneTransferHome.create();
        }
        {
            int fromAcctId = 0;
            int toAcctId = 0;
            float amount = (float) 0.0;
            objTransfer.transferFunds(fromAcctId, toAcctId, amount)
```

__ c. After clicking inside the testMethodsTransfer() method, notice that the Test Data Table view opens. Verify that the information in the Test Data Table looks like the following.



Action	Type	default	
		In	Expected
objTransfer = oneTransferHome.creat...	xy		
objTransfer	com.ibm.websphere.sam...		
<expected exception>	Throwable		
objTransfer.transferFunds(fromAcctId...	xy		
fromAcctId	int		
toAcctId	int		
amount	float		
<expected exception>	Throwable		

____ 7. Create initialization points for fromAcctId, toAcctId, and amount variables above the objTransfer.transferFunds(fromAcctId, toAcctId, amount) action in the Test Data Table.

__ **a.** Right click on the objTransfer.transferFunds(fromAcctId, toAcctId, amount) action in the Test Data Table view and select **Insert Initialization Point Above…** from the context menu.

__ **b.** Select the radio button next to **Select a variable** and select fromAcctId. Click **OK**.

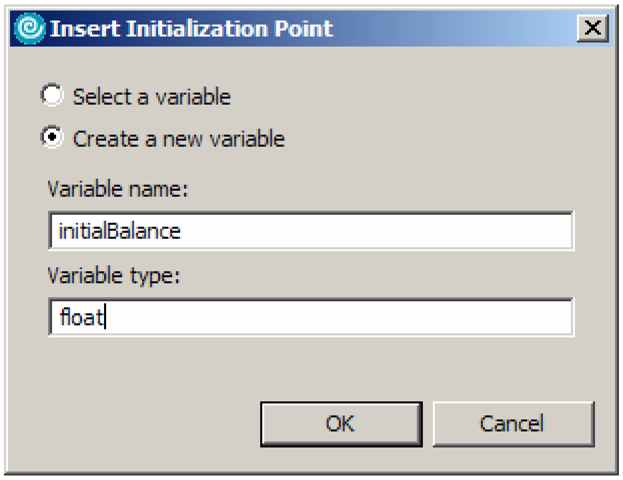
__ **c.** Repeat Steps a and b above for the **toAcctId** and **amount** variables

© Copyright IBM Corporation 2005. All rights reserved

____ **8.** Create an initialization point for a new variable called **initialBalance** above the objTransfer.transferFunds(fromAcctId, toAcctId, amount) action in the Test Data Table

__ **a.** Right click on the objTransfer.transferFunds(fromAcctId, toAcctId, amount) action and select **Insert Initialization Point Above…** from the context menu

__ **b.** Select the radio button next to **Create a new variable**. Enter initialBalance for the Variable name and float for the Variable type. Click **OK**.

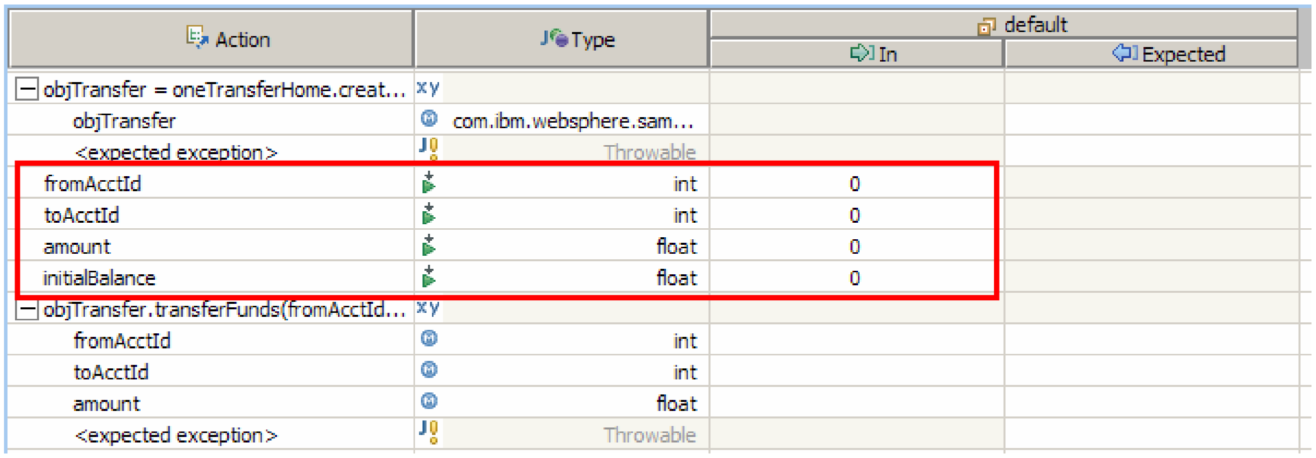

Insert Initialization Point

( ) Select a variable
[✓ Create a new variable]

Variable name: initialBalance

Variable type: float

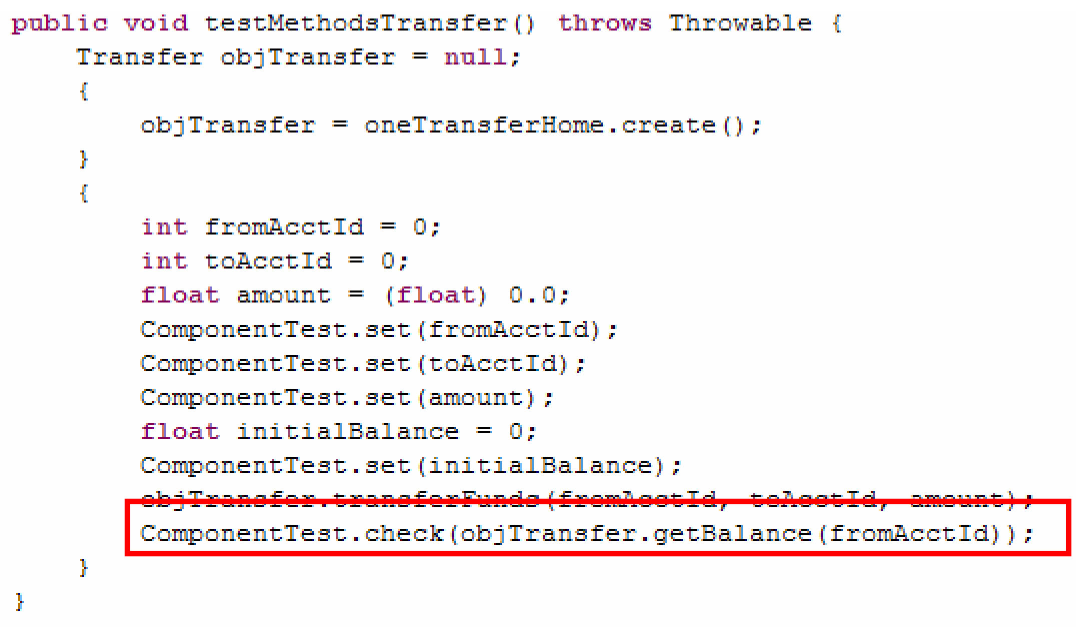
____ **7.** Verify that your Test Data Table looks like the following



Action	Type	default In	default Expected
objTransfer = oneTransferHome.creat…	xy		
objTransfer	com.ibm.websphere.sam…		
<expected exception>	Throwable		
fromAcctId	int	0	
toAcctId	int	0	
amount	float	0	
initialBalance	float	0	
objTransfer.transferFunds(fromAcctId…	xy		
fromAcctId	int		
toAcctId	int		
amount	float		
<expected exception>	Throwable		

____ **9.** Insert a validation action after the call to transferFunds(). This validation action will verify that the transfer action was successful.

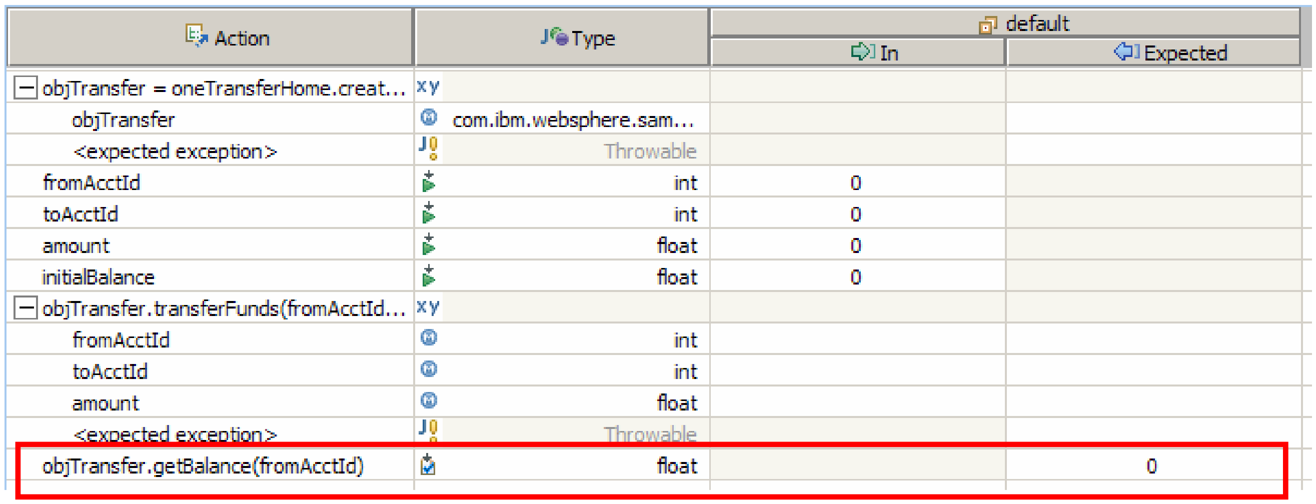
__ **a.** In the TransferTest.java behavior file, add the following line of code to the testMethodsTransfer() method under the objTransfer.transferFunds(fromAcctId, toAcctId, amount). The appropriate line of code is highlighted in the following screen shot. For your convenience, this line of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet1.txt**



```
public void testMethodsTransfer() throws Throwable {
    Transfer objTransfer = null;
    {
        objTransfer = oneTransferHome.create();
    }
    {
        int fromAcctId = 0;
        int toAcctId = 0;
        float amount = (float) 0.0;
        ComponentTest.set(fromAcctId);
        ComponentTest.set(toAcctId);
        ComponentTest.set(amount);
        float initialBalance = 0;
        ComponentTest.set(initialBalance);
        objTransfer.transferFunds(fromAcctId, toAcctId, amount);
        ComponentTest.check(objTransfer.getBalance(fromAcctId));
    }
}
```

__ **b.** Save the TransferTest.java behavior file.

__ **c.** Notice the validation item that appears underneath the objTransfer.transferFunds(fromAcctId, toAcctId, amount) action in the Test Data Table view.

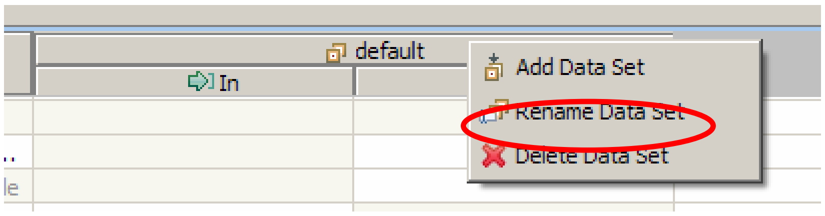


Action	Type	default	
		In	Expected
⊟ objTransfer = oneTransferHome.creat...	xy		
objTransfer	com.ibm.websphere.sam...		
<expected exception>	Throwable		
fromAcctId	int	0	
toAcctId	int	0	
amount	float	0	
initialBalance	float	0	
⊟ objTransfer.transferFunds(fromAcctId...	xy		
fromAcctId	int		
toAcctId	int		
amount	float		
<expected exception>	Throwable		
objTransfer.getBalance(fromAcctId)	float		0

____ **10.** Rename the **default** data set.

__ **a.** Right click on the title bar for the default data set.

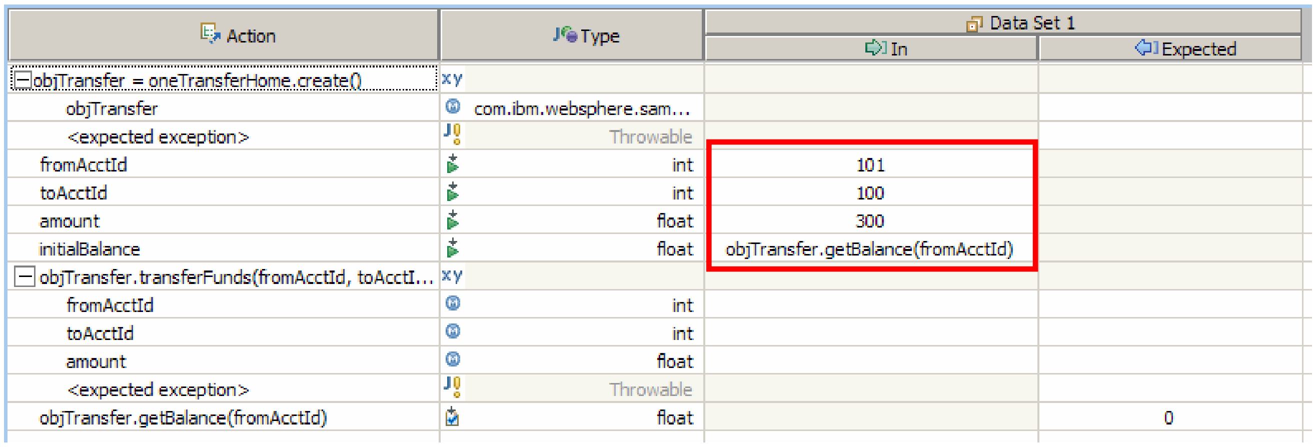
__ **b.** Select **Rename Data Set** from the context menu.



__ **c.** Enter **Data Set 1** as the new name for the data set.

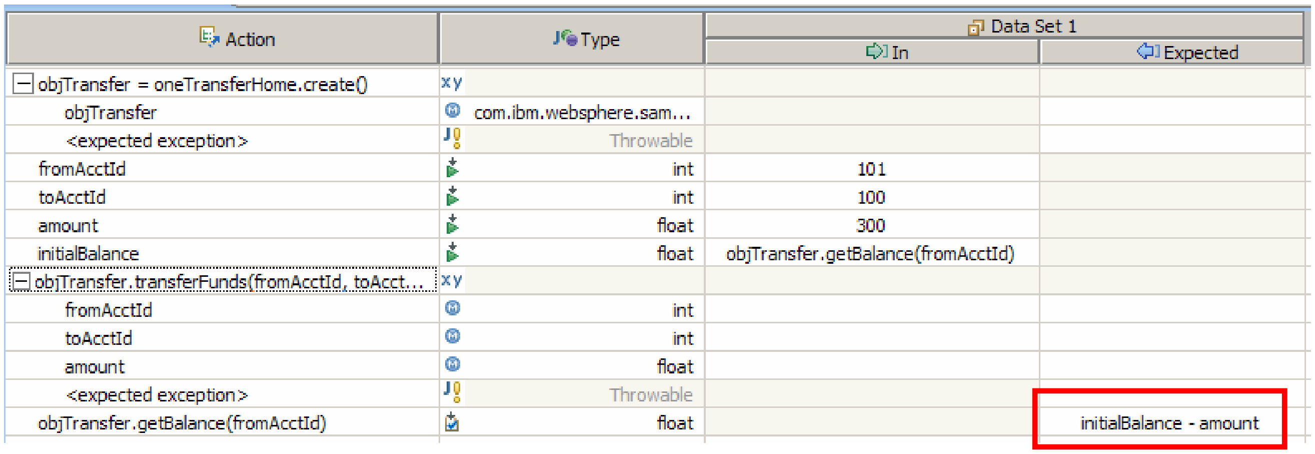
____ **11.** Enter data for Data Set 1.

__ **a.** Underneath objTransfer = oneTransferHome.create() action enter the data as indicated in the screen shot below



Action	Type	Data Set 1 In	Data Set 1 Expected
objTransfer = oneTransferHome.create()			
objTransfer	com.ibm.websphere.sam...		
<expected exception>	Throwable		
fromAcctId	int	101	
toAcctId	int	100	
amount	float	300	
initialBalance	float	objTransfer.getBalance(fromAcctId)	
objTransfer.transferFunds(fromAcctId, toAcctI...			
fromAcctId	int		
toAcctId	int		
amount	float		
<expected exception>	Throwable		
objTransfer.getBalance(fromAcctId)	float		0

__ **b.** For the objTransfer.getBalance(fromAcctId) validation action at the bottom of the data table, enter **initialBalance – amount** for the expected value.



Action	Type	Data Set 1 In	Data Set 1 Expected
objTransfer = oneTransferHome.create()			
objTransfer	com.ibm.websphere.sam...		
<expected exception>	Throwable		
fromAcctId	int	101	
toAcctId	int	100	
amount	float	300	
initialBalance	float	objTransfer.getBalance(fromAcctId)	
objTransfer.transferFunds(fromAcctId, toAcct...			
fromAcctId	int		
toAcctId	int		
amount	float		
<expected exception>	Throwable		
objTransfer.getBalance(fromAcctId)	float		initialBalance - amount

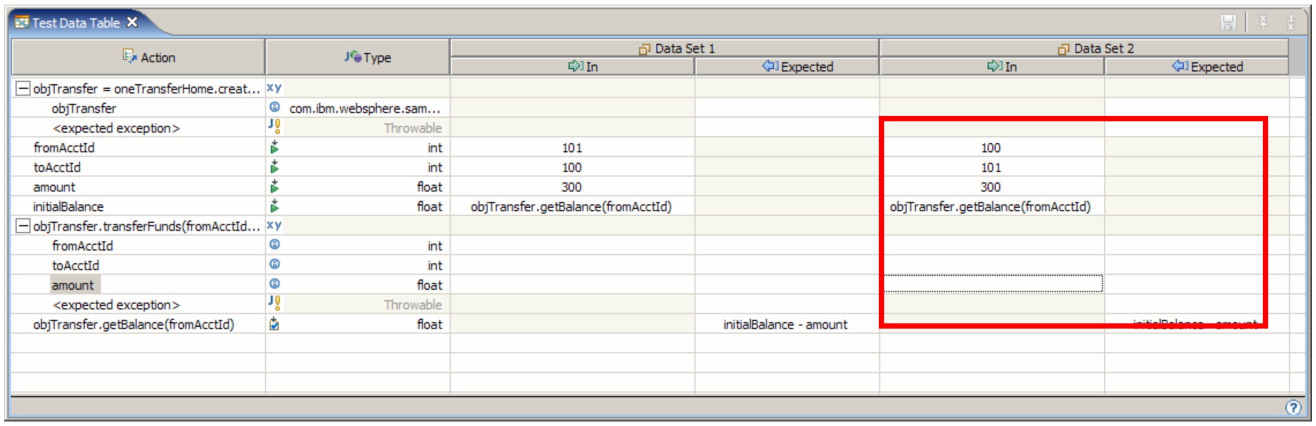
____ **12.** Create a new Data Set.

__ **a.** Right click on the title bar for Data Set 1.

__ **b.** Select **Add Data Set** from the context menu.

__ **c.** You will see a new data set appear to the right of Data Set 1 called New Test Data. Rename this data set to **Data Set 2**.

__ **d.** Repeat Step 10, but reverse the account numbers for the fromAcctId and toAcctID. The Test Data Table should appear like the following screen capture.

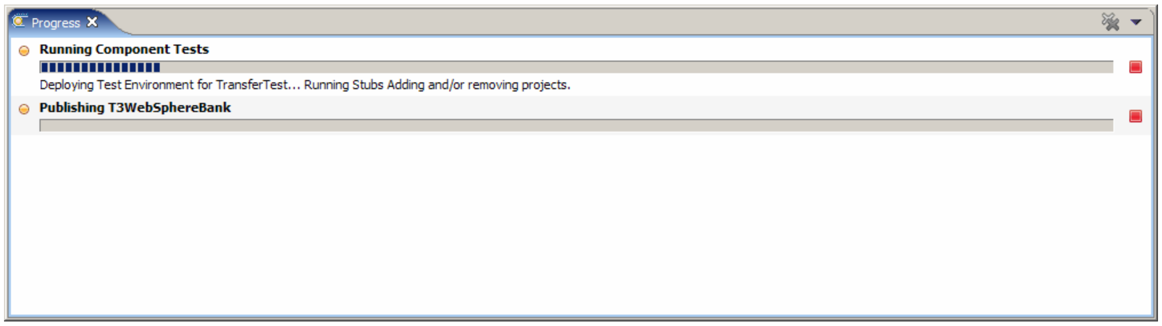

Test Data Table

Action	Type		Data Set 1		Data Set 2	
			In	Expected	In	Expected
objTransfer = oneTransferHome.creat…	xy					
objTransfer	com.ibm.websphere.sam…					
<expected exception>	Throwable					
fromAcctId	int		101		100	
toAcctId	int		100		101	
amount	float		300		300	
initialBalance	float		objTransfer.getBalance(fromAcctId)		objTransfer.getBalance(fromAcctId)	
objTransfer.transferFunds(fromAcctId…	xy					
fromAcctId	int					
toAcctId	int					
amount	float					
<expected exception>	Throwable					
objTransfer.getBalance(fromAcctId)	float			initialBalance - amount		initialBalance - amount

____ **13.** Run the TransferTest component test

__ **a.** From the Test Navigator View right click on TransferTest under WebSphereBankTest > Test Suite and select **Run > Component Test**.

__ **b.** Running an EJB Component test may take several minutes. To view the progress, open the Progress view by clicking the ( ) icon in the lower right hand corner of the workspace.


Progress

Running Component Tests

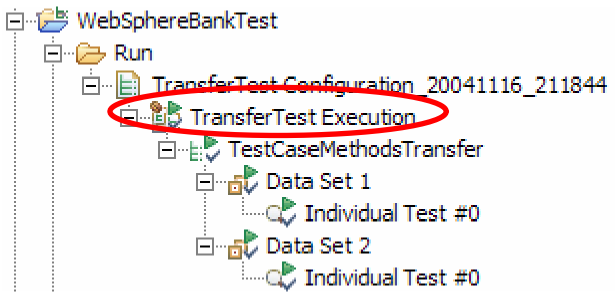
Deploying Test Environment for TransferTest… Running Stubs Adding and/or removing projects.

Publishing T3WebSphereBank

NOTE: You may get an error message indicating there was a problem cleaning the execution project. You can dismiss this message and continue.

____ **14.** View the results of the component test run.

__ **a.** From the Test Navigator View, expand **WebSphereBankTest > Run > Run** and double click on the TransferTestExecution data set underneath the appropriate test run.

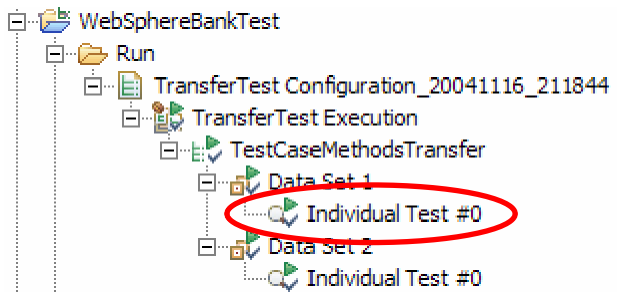

```
WebSphereBankTest
  Run
    TransferTest Configuration_20041116_211844
      TransferTest Execution
        TestCaseMethodsTransfer
          Data Set 1
            Individual Test #0
          Data Set 2
            Individual Test #0
```

© Copyright IBM Corporation 2005. All rights reserved

NOTE: Each test run will have a unique name that includes a timestamp for when the test case was run. At this point in the lab there should only be one test case listed underneath the Run folder.

__ **b.** You will notice that the Test Run editor opens in the editor area. This editor contains the results of the execution of a test.

__ **c.** You can also review the results of a particular component test run by double clicking on either Individual Test #0 underneath a particular Data Set in the Test Navigator view. You will notice the Data Comparator view opens providing a review of the results for the particular data set.



© Copyright IBM Corporation 2005. All rights reserved

Part 4: Profiling a Component Test (Optional)

____ 1. Restart the Server in Profile mode.

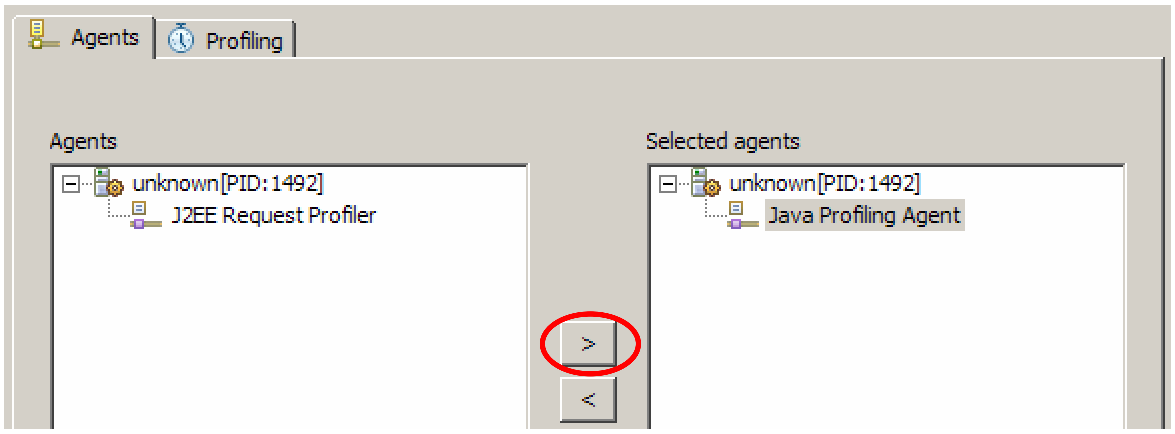
__ **a.** Switch to the J2EE perspective.

__ **b.** From the Server view right click on WebSphere Application Server v6.0 and select **Restart > Profile** from the context menu.

NOTE: If your server status is indicated as "Stopped", right click on WebSphere Application Server v6.0 and select Profile from the context menu.

__ **c.** If a Confirm Enablement dialog appears to enable the Profiling and Logging capabilities, click **OK**.

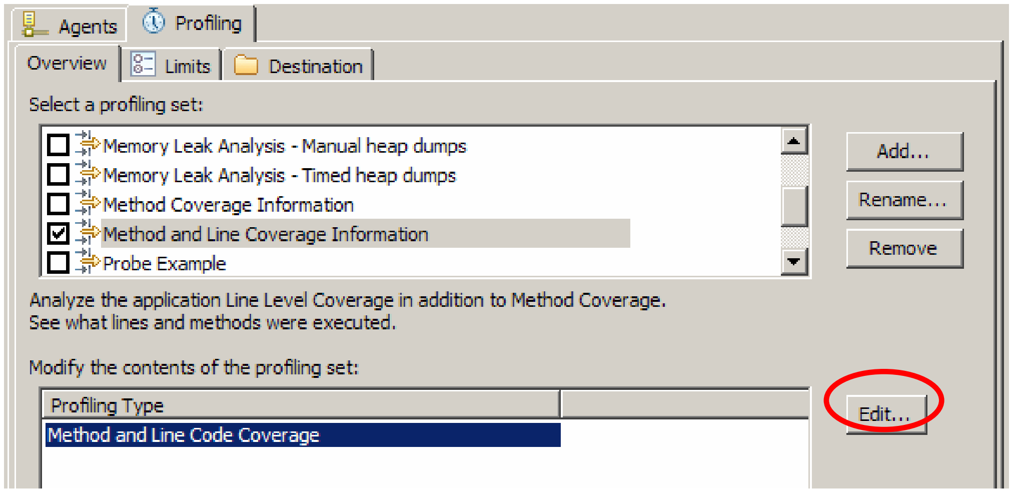
__ **d.** You will see a Profile on server configuration window appear. The first step in the configuration process is to select the appropriate agent. From the Agents tab, expand the agent listed under unknown[PID:xxxxx]. Select the Java Profiling Agent and click the Add button to add this agent to the selected agents list.



__ **e.** Click on the Profiling tab. On the Overview sub tab, check the box next to **Method and Line Coverage Information** from the list of profiling sets.

© Copyright IBM Corporation 2005. All rights reserved

__ **f.** To modify the contents of the profiling set, select Method **and Line Code Coverage** from the lower list under Profiling Type and click **Edit**.



Select a profiling set:

- [ ] Memory Leak Analysis - Manual heap dumps
- [ ] Memory Leak Analysis - Timed heap dumps
- [ ] Method Coverage Information
- [x] Method and Line Coverage Information
- [ ] Probe Example

Analyze the application Line Level Coverage in addition to Method Coverage.
See what lines and methods were executed.

Modify the contents of the profiling set:

Profiling Type
Method and Line Code Coverage

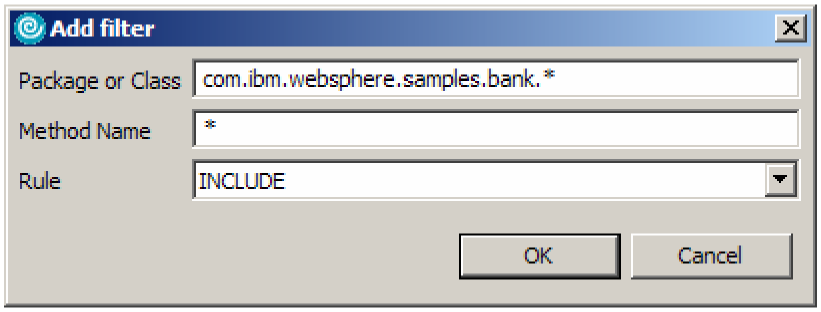
__ **g.** Highlight **Method and Line Code Coverage** from the list of profile sets in the left window and click **Next** to continue configuring this profiling set.

__ **h.** From the Select a filter set list, check the box next to **WebSphere J2EE**.

__ **i.** Click the Add button next to the list of the contents of the selected filter set.

NOTE: This is the Add button next to the bottom window.

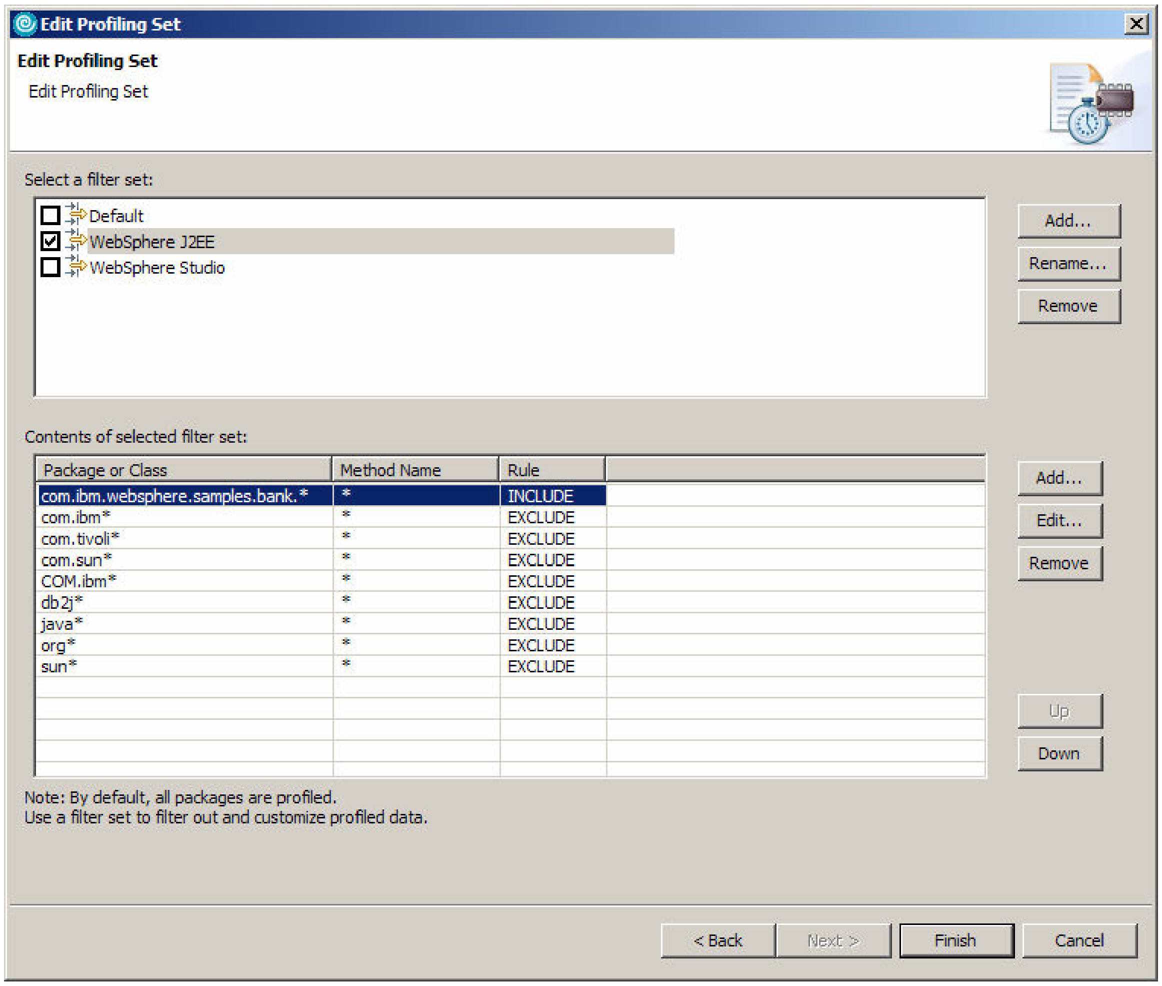
__ **j.** For the package or class enter **com.ibm.websphere.samples.bank.***



Package or Class: com.ibm.websphere.samples.bank.*
Method Name: *
Rule: INCLUDE

© Copyright IBM Corporation 2005. All rights reserved

__ **k.** Verify that the filter looks like the following and click **Finish**.



Select a filter set: [ ] Default [✓ WebSphere J2EE] [ ] WebSphere Studio

Contents of selected filter set:

Package or Class	Method Name	Rule
com.ibm.websphere.samples.bank.*	*	INCLUDE
com.ibm*	*	EXCLUDE
com.tivoli*	*	EXCLUDE
com.sun*	*	EXCLUDE
COM.ibm*	*	EXCLUDE
db2j*	*	EXCLUDE
java*	*	EXCLUDE
org*	*	EXCLUDE
sun*	*	EXCLUDE

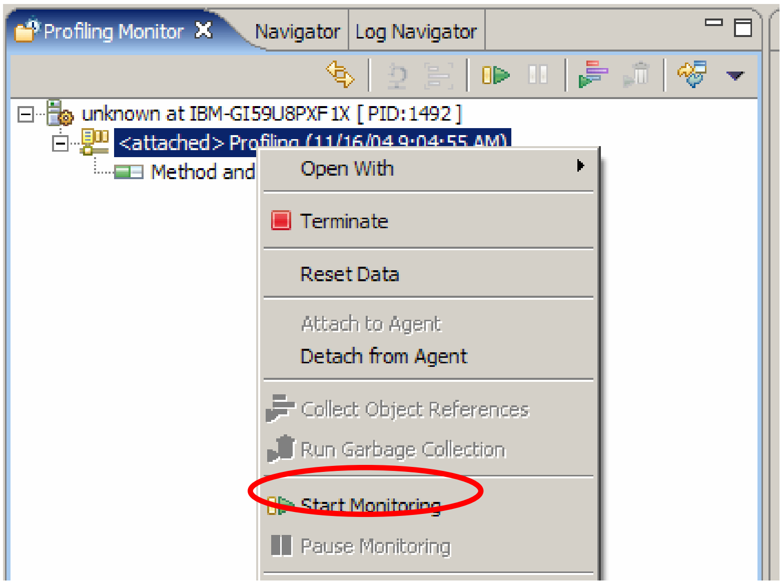
Note: By default, all packages are profiled.
Use a filter set to filter out and customize profiled data.

__ **l.** Click **Finish** on the Profile on server configuration dialog.

__ **m.** Notice that the Profiling and Logging perspective opens. A Dialog will appear reminding you to start Monitoring by selecting “Start Monitoring” from the pop-up menu of the agent in the Profiling Monitor view. Click **OK** for this message. You will perform this task in the next step.

© Copyright IBM Corporation 2005. All rights reserved

__ **n.** Go to the Profiling Monitor view and right click on the agent and select **Start Monitoring** from the context menu.



____ **2.** Run the TransferTest component test.

__ **o.** Return to the Test perspective.

__ **p.** From the Test Navigator view right click on TransferTest under WebSphereBankTest > Test Suite and select **Profile > Component Test**

__ **q.** As before, you can monitor the progress by switching to the progress view.

____ **3.** When the component test has completed, go to the Profiling Monitor view and Pause Monitoring for the agent

__ **a.** From the Profiling Monitor view, right click on the agent and select **Pause Monitoring** from the context menu.

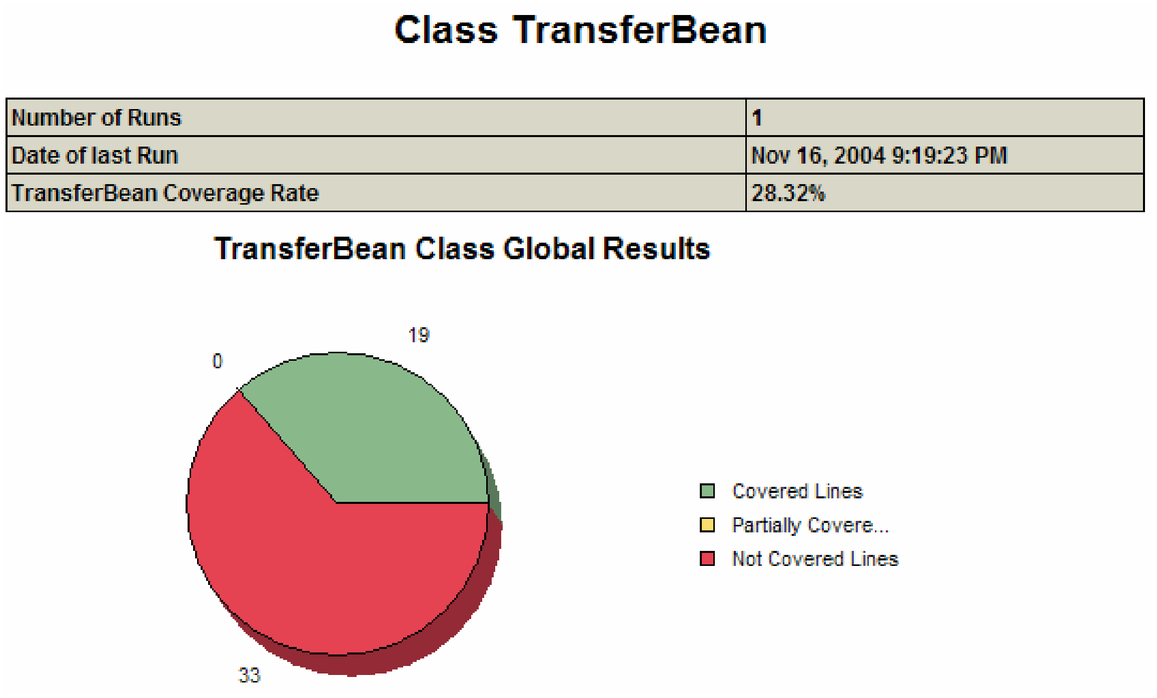
____ **4.** View results of profiling test run.

__ **a.** Right click on the Method and Line Code Coverage item in the Profiling Monitor view for the appropriate agent and select **Open With > Coverage Details**. Notice that this opens the Coverage Navigator and Annotated Source views.

__ **b.** From the Coverage Navigator view expand com.ibm.websphere.samples.bank.ejb

© Copyright IBM Corporation 2005. All rights reserved

__ **c.** Locate the TransferBean class in the list and double click on it. Notice that this updates what is shown in the Annotated source view. On the summary page for the Annotated Source view, notice that the coverage rate for the TransferBean class is roughly 28%.



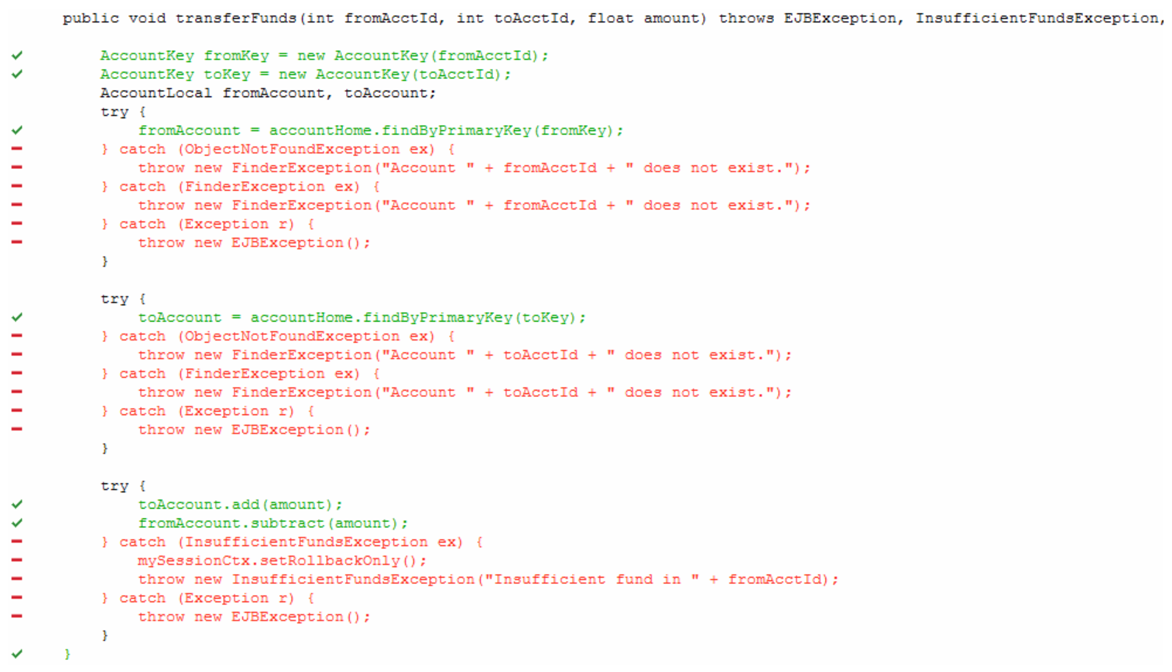
Class TransferBean

Number of Runs	1
Date of last Run	Nov 16, 2004 9:19:23 PM
TransferBean Coverage Rate	28.32%

TransferBean Class Global Results

__ **d.** Scroll through the summary page and notice the getBalance() and transferFunds() methods. To view the specific lines in these methods that have not been covered by the test click on the (📄) icon at the top of the view to switch to the source view.

__ **e.** Click on the source view option in the Annotated source view and notice the lines of code that have not been covered. These lines of code are listed in red below.



```
    public void transferFunds(int fromAcctId, int toAcctId, float amount) throws EJBException, InsufficientFundsException,

✓       AccountKey fromKey = new AccountKey(fromAcctId);
✓       AccountKey toKey = new AccountKey(toAcctId);
        AccountLocal fromAccount, toAccount;
        try {
✓           fromAccount = accountHome.findByPrimaryKey(fromKey);
-       } catch (ObjectNotFoundException ex) {
-           throw new FinderException("Account " + fromAcctId + " does not exist.");
-       } catch (FinderException ex) {
-           throw new FinderException("Account " + fromAcctId + " does not exist.");
-       } catch (Exception r) {
-           throw new EJBException();
        }

        try {
✓           toAccount = accountHome.findByPrimaryKey(toKey);
-       } catch (ObjectNotFoundException ex) {
-           throw new FinderException("Account " + toAcctId + " does not exist.");
-       } catch (FinderException ex) {
-           throw new FinderException("Account " + toAcctId + " does not exist.");
-       } catch (Exception r) {
-           throw new EJBException();
        }

        try {
✓           toAccount.add(amount);
✓           fromAccount.subtract(amount);
-       } catch (InsufficientFundsException ex) {
-           mySessionCtx.setRollbackOnly();
-           throw new InsufficientFundsException("Insufficient fund in " + fromAcctId);
-       } catch (Exception r) {
-           throw new EJBException();
        }
✓   }
```

____ **5.** Add a data set for the TransferTest component test to cover those lines of code that were missed with the previous test runs. Specifically, create an error path test case where a transfer is attempted for an account that has insufficient finds for the amount requested for the transfer.

__ **a.** Switch to the Test perspective.

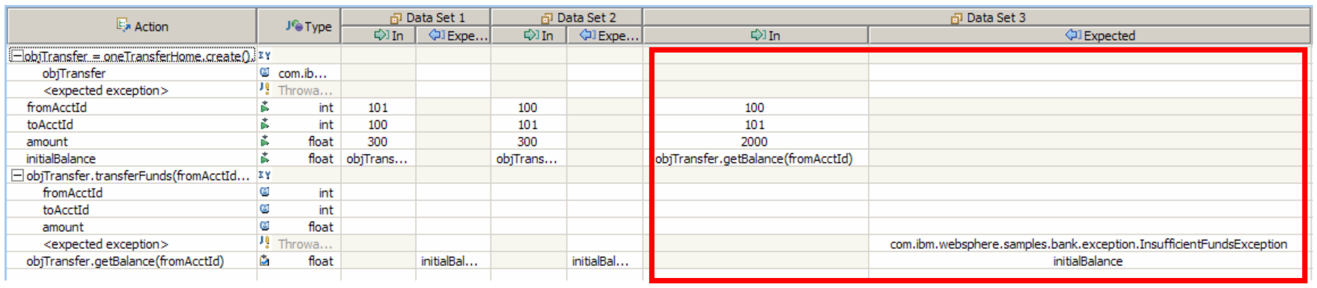
__ **b.** Open the Test Data Table view for the TestCaseMethodsTransfer test case

__ **c.** Create a new Data Set by right clicking on the title bar for Data Set 2.

__ **d.** Select **Add Data Set** from the context menu.

__ **e.** You will see a new data set appear to the right of Data Set 2 called New Test Data. Rename this data set to **Data Set 3**.

__ **f.** Enter data for this data set as indicated below in screen capture.



Action	Type	Data Set 1		Data Set 2		Data Set 3	
		In	Expe…	In	Expe…	In	Expected
⊟ objTransfer = oneTransferHome.create()…							
objTransfer	com.ib…						
<expected exception>	Throwa…						
fromAcctId	int	101		100		100	
toAcctId	int	100		101		101	
amount	float	300		300		2000	
initialBalance	float	objTrans…		objTrans…		objTransfer.getBalance(fromAcctId)	
⊟ objTransfer.transferFunds(fromAcctId…							
fromAcctId	int						
toAcctId	int						
amount	float						
<expected exception>	Throwa…						com.ibm.websphere.samples.bank.exception.InsufficientFundsException
objTransfer.getBalance(fromAcctId)	float		initialBal…		initialBal…		initialBalance

____ **6.** Start Monitoring again the application server process again.

__ **a.** From the Profiling and Logging perspective, right click on the appropriate agent and select Start Monitoring from the context menu.

____ **7.** Profile the modified component test.

__ **a.** From the Test Navigator view right click on TransferTest under WebSphereBankTest > Test Suite and select **Profile > Component Test**

____ **8.** View the results of profiling test run.

__ **a.** Right click on the Method and Line Code Coverage item in the Profiling Monitor view for the appropriate agent and select **Open With > Coverage Details**.

__ **b.** Locate the TransferBean class in the list and notice that on the summary page for the Annotated Source view the coverage rate for the TransferBean class is roughly 34.5% and notice that the coverage of the transferFunds() method is increased.

© Copyright IBM Corporation 2005. All rights reserved

Part 5: Restore the Server Configuration

____ 1. Remove the WebSphereBank bank application from the application server

__ **a.** From the Server view right click on the server and select Add and Remove projects

__ **b.** In the Configured projects list select WebSphereBank and click the Remove button

__ **c.** Click **Finish**

____ **2.** Stop the server

__ **a.** From the Server view, right click on the server and select Stop from the context menu

____ **3.** Restore your server configuration. This will return your server configuration to its original state.

__ **a.** Open a Windows **Command Prompt** and navigate to the following directory:

```
<RAD_HOME>\runtimes\base_v6\bin
```

__ **b.** Restore the server configuration by issuing the following command:

```
restoreConfig "C:\Program Files\IBM\Backup.zip"
```

© Copyright IBM Corporation 2005. All rights reserved

What you did in this exercise

In this exercise you used the Component Test features available in IBM Rational Application Developer v6 to test the WebSphereBank application. This exercise highlighted not only the steps needed to create and run an EJB component test on the WebSphere Application Server v6 test environment, but also locating and viewing the test results. An important part of component test is the ability to use the profiling capabilities while you run your component tests. Specifically, the “Method and Line Level Code Coverage” profiling set has been integrated with the component test metrics to allow developers to easily access the code coverage for a particular test suite. This exercise also demonstrated some of the important steps needed to profile a component test.

© Copyright IBM Corporation 2005. All rights reserved

This page is left intentionally blank.