IBM® WEBSPHERE® APPLICATION SERVER V6.0– LAB EXERCISE

Testing a JMS application

What this exercise is about

The objective of this lab is to demonstrate installing and configuring the WebSphere Bank MDB enterprise application to run on WebSphere application server.

Lab Requirements

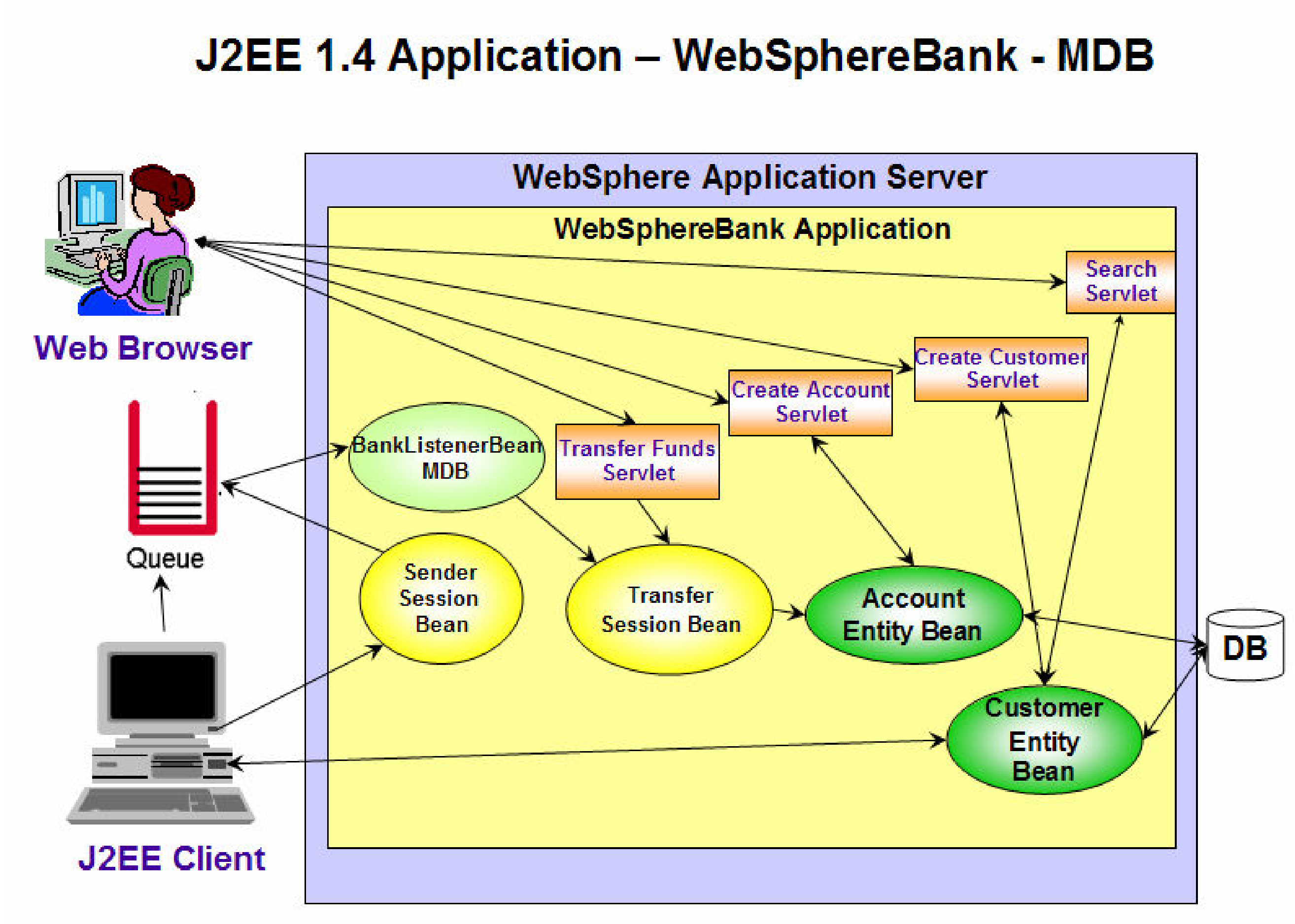
List of system and software required for the student to complete the lab.

- WebSphere Application Server Base version 6 installed and an unfederated application server
- Installation of lab sample code into the following directories:
  - Windows workstation: **C:\Labfiles60**
  - z/OS host system: **/etc/Labfiles60**

What you should be able to do

- Install an enterprise application with JMS resources
- Configure Cloudscape JDBC Driver and Data Source necessary.
- Prepare JMS Resources necessary for WebSphereBank to run.

Introduction



The WebSphereBank sample application is a collection of simple EJBs, servlets, and JSPs designed to showcase the various features in WebSphere. This sample consists of 2 entity beans, CustomerBean and AccountBean and session beans called Transfer and Sender. The abstract schema names for the entity beans are Customer and Account. Each entity bean has both remote/local interfaces and remote/local home interfaces. The entity bean CustomerBean has one-to-many relationships with AccountBean.

The SenderBean.java session bean is responsible for sending message to the destination, and the MDB BankListenerBean.java is the consumer for the message. A java client application mdbTest.java invokes the Sender session bean to send messages to the Queue **BankJMSQueue**. The message that is sent has a JMSType='transfer' and is of the format "<fromaccount> <toaccount> <transferamount>. The MDB BankListenerBean.java is only interested in messages of JMSType='transfer', and when the MDB receives the message it invokes the Transfer session bean to perform the money transfer.

In order to use a JMS Destination with the Default Messaging Provider, you need 2 different layers of administratively created objects. One of the layers is the **SIBDestination**. This Destination is invisible to the user application, which should only see the **JMS Destination** (which basically acts as a proxy/pointer to the SIBDestination). Destinations are called **queues** (point to point messaging domain) or **topics** (pub/sub messaging domain). In this scenario, we are exercising Point-to-Point messaging. Users will also need a **JMS Connection Factory** to create a connection. The JMS Destinations and JMS Connection Factories are bound into JNDI and the user code (that puts the message) does lookups on them and uses them to connect to the underlying JMS Provider through the standard JMS Interface.

The Message Driven Bean is associated with the destination through an administratively created object called the Activation Spec. The MDBs onMessage method is called whenever a message reaches the destination that the MDB is associated with.

Packaged within the WebSphereBank.ear file is a database directory. This directory is a Cloudscape database containing the tables that are used by the WebSphereBank application. Some data already exists within the tables in this directory, such as Account numbers 1 and 2. The BANKDS datasource points to this directory, which is why when installing this sample application, no scripts are needed to create or populate the data tables. This should also explain why any changes you make to bank accounts are lost when the application is uninstalled and re-installed.

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:

Location Reference	Windows example	z/OS example
<WAS_HOME>	C:\WebSphere\AppServer	/etc/c6cellB/AppServer
<LAB_FILES>	C:\LabFiles60	/etc/LabFiles60
<DB_LOCATION>		/etc/LabFiles60/CloudscapeDB/BankDB
<SERVER_NAME>	server1	C6Server01
<HOST_NAME>	localhost	mvs221.rtp.raleigh.ibm.com

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/

Part 1: Lab Setup

Note: In this part we will run a wsadmin command to install the WebSphereBank sample and a jacl script to create the JDBC resources required for WebSphereBank.

____ 1. Open a command prompt and navigate to your instance bin directory.

```
cd <WAS_HOME>\profiles\default\bin
```

____ 2. Check to see if the server is running, and if it is, stop it.

__ a. Run the command:

```
./serverStatus.sh <SERVER_NAME>
```

__ b. If the server status indicates STARTED, then run the command:

```
./stopServer.sh <SERVER_NAME>
```

Note: The reason for completing these steps with the server stopped is that some changes to the namespace are picked up only on server startup.

____ 3. Run the clean-up script. This script will test to see if you have installed WebSphereBank from previous lab and will remove the WebSphereBank application and the BANKDS Data Source.

__ a. From a Command Prompt, navigate to **<WAS_HOME>/profiles/default/bin**

__ b. Run the command:

./wsadmin.sh –conntype none -f <LAB_FILES>/common/prepWSBank.jacl

____ 4. Create BANKDS data source

__ a. Navigate to **<WAS_HOME>\profiles\default\bin**

__ b. Run the following command.

<u>(Please see the notes below before you proceed with the command)</u>

./wsadmin.sh –conntype none –f <LAB_FILES>/common/setupBankDS.jacl <myCell> <myNode> <DB_LOCATION> <SERVER_NAME>

Notes:

a. The BANKDB database and tables have already been created for you.
b. <DB_LOCATION> must be a fully qualified path to the database.
   For example: /etc/LabFiles60/CloudscapeDB/BankDB
c. Substitute your cell name and your node for <myCell> and <myNode>. You can also look at the file structure under <WAS_HOME>/profiles/default/config directory to determine the Cell Name and Node Name for your Application Server installation. The directory structure is
   <WAS_HOME>/profiles/default/config/cells/<myCell>/nodes/<myNode>
d. <SERVER_NAME> is optional and will default to the value “server1” if it is not supplied.

____ 5. Start the application server

```
./startServer.sh <SERVER_NAME>
```

____ 6. Install WebSphereBank application

From same directory, run the following command. Replace LAB_FILES with your lab file path.

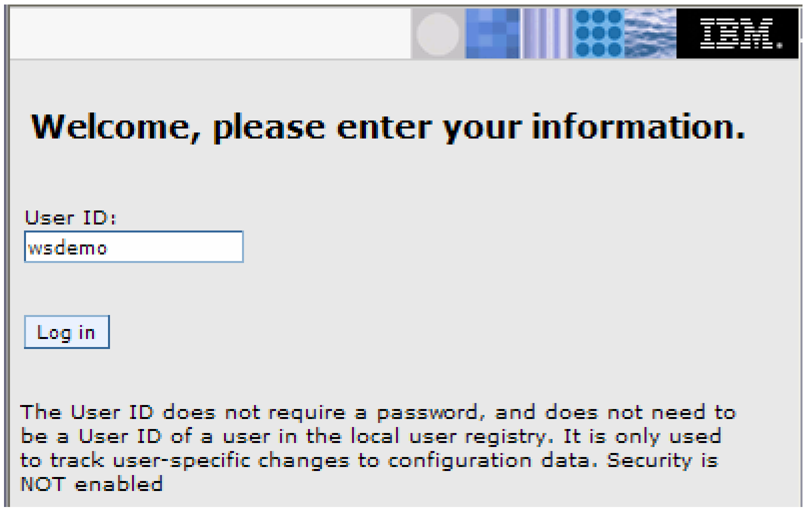
./wsadmin.sh –conntype none -c '$AdminApp install <LAB_FILES>/WASv6_MDBAppLab/WebSphereBank.ear {-appname WebSphereBank –server <SERVER_NAME> –usedefaultbindings –deployejb –deployejb.dbtype CLOUDSCAPE_V5}'

____ 7. Start the Administrative Console.

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

http://<HOST_NAME>:9080/ibm/console

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


Welcome, please enter your information.

User ID: wsdemo

The User ID does not require a password, and does not need to be a User ID of a user in the local user registry. It is only used to track user-specific changes to configuration data. Security is NOT enabled

Part 2: Creating the BANKDB Cloudscape database

This step demonstrates the creation of the Cloudscape BANKDB database, which you will need for the WebSphereBank application. You will use commands to create the database and tables. The database will be created in the <LAB_FILES>/CloudscapeDB directory.

____ 8. Generate the Cloudscape BANKDB database and tables.

1. In a command window, navigate to

   /etc/c6cellB/AppServer/cloudscape/bin/embedded

2. Issue the Cloudscape **ij** command to start the Cloudscape utility

   ./ij.sh

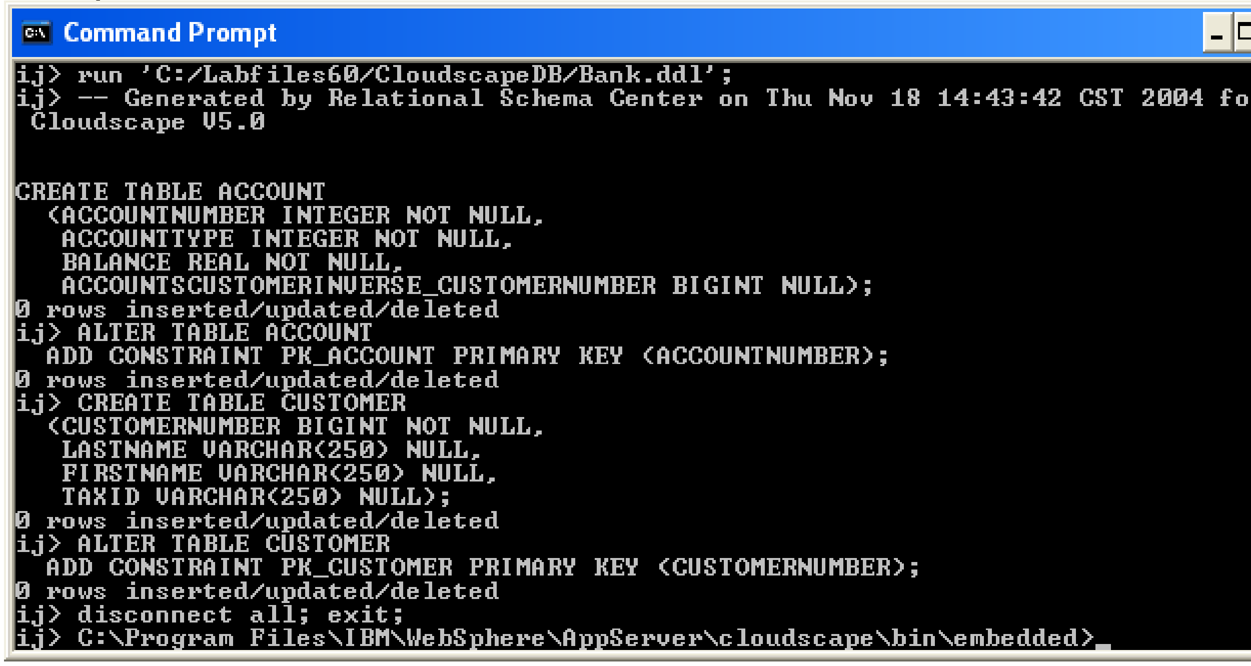
3. Issue the following commands, replacing <LAB_FILES> with your lab file path:

   mkdir /tmp/LabFiles60/CloudscapeDB

   connect ‘jdbc:db2j:/tmp/LabFiles60/CloudscapeDB/BANKDB;create=true’;

   Note: this command may run for 10 to 30 seconds; when the command completes, you will see the “ij>” prompt with no messages

4. **run ‘tmp/LabFiles60/common/Bank.ddl’;**



```
ij> run 'C:/Labfiles60/CloudscapeDB/Bank.ddl';
ij> -- Generated by Relational Schema Center on Thu Nov 18 14:43:42 CST 2004 fo
 Cloudscape V5.0

CREATE TABLE ACCOUNT
  (ACCOUNTNUMBER INTEGER NOT NULL,
   ACCOUNTTYPE INTEGER NOT NULL,
   BALANCE REAL NOT NULL,
   ACCOUNTSCUSTOMERINVERSE_CUSTOMERNUMBER BIGINT NULL);
0 rows inserted/updated/deleted
ij> ALTER TABLE ACCOUNT
  ADD CONSTRAINT PK_ACCOUNT PRIMARY KEY (ACCOUNTNUMBER);
0 rows inserted/updated/deleted
ij> CREATE TABLE CUSTOMER
  (CUSTOMERNUMBER BIGINT NOT NULL,
   LASTNAME VARCHAR(250) NULL,
   FIRSTNAME VARCHAR(250) NULL,
   TAXID VARCHAR(250) NULL);
0 rows inserted/updated/deleted
ij> ALTER TABLE CUSTOMER
  ADD CONSTRAINT PK_CUSTOMER PRIMARY KEY (CUSTOMERNUMBER);
0 rows inserted/updated/deleted
ij> disconnect all; exit;
ij> C:\Program Files\IBM\WebSphere\AppServer\cloudscape\bin\embedded>_
```

5. **disconnect all; exit;**

6. **Go to the <LAB_FILES> directory and issue ‘chmod -R 777 CloudscapeDB’**

The BANKDB database and tables have been created.

Part 3: View Application using Application Server Toolkit(AST) or IRAD

Note: This part is OPTIONAL. You can view the WebSphereBank application using Application Server Toolkit (AST) or IRAD. You may skip to the next section if you choose not to view the application information.

____ 1. Open Application Server Tool Kit, or IRAD.

__ a. Using Windows Explorer, navigate to the c:\eclipse\ folder, right click on the eclipse.exe icon, and select **Open** from the menu.

__ b. Enter the workspace as **c:\labfiles60\WASv6_MDBAppLab\workspace.**

__ c. Click **OK**.

__ d. Close the Welcome window.

____ 2. Import WebSphereBank EAR file into the workspace.

__ a. Click **File > Import.**

__ b. Select **EAR file**. Click **Next.**

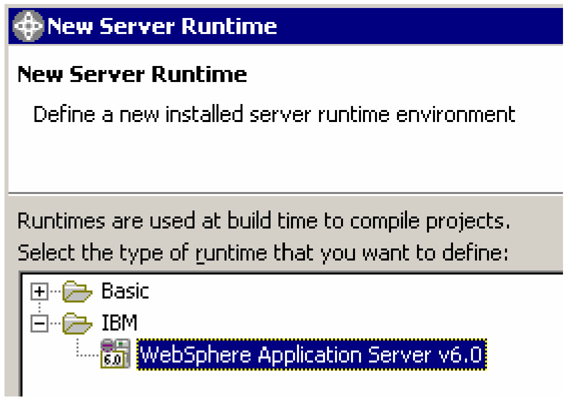
__ c. Click **Browse** to select <LAB_FILES>\WASv6_MDBAppLab\WebSphereBank.ear.

__ d. Make sure that EAR project is **WebSphereBank.**

__ e. If using IRAD, skip to step k.

__ f. For the Target Server, Click **New.**

__ g. Make sure **WebSphere Application Server v6.0** is highlighted.



New Server Runtime

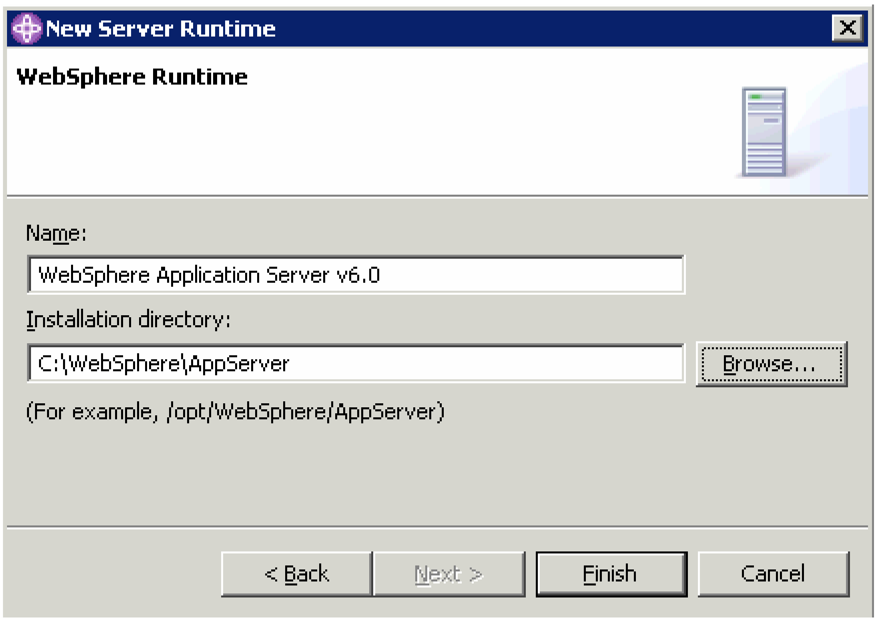
New Server Runtime

Define a new installed server runtime environment

Runtimes are used at build time to compile projects.
Select the type of runtime that you want to define:

__ h. Click **Next.**

__ i. Enter the Installation Directory as **C:\WebSphere\AppServer.**



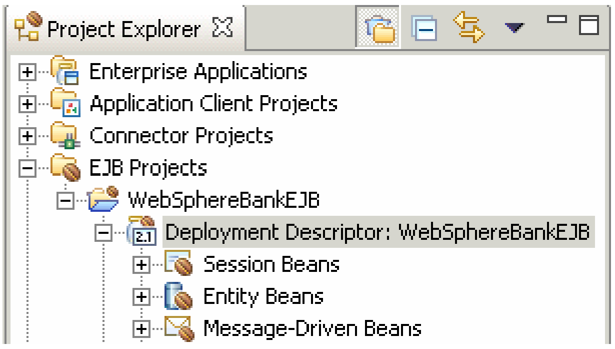
__ j. Click **Finish.**

__ k. Click **Finish** again.

__ l. Click **Yes** when prompted to switch to the J2EE perspective. Wait for import to complete.

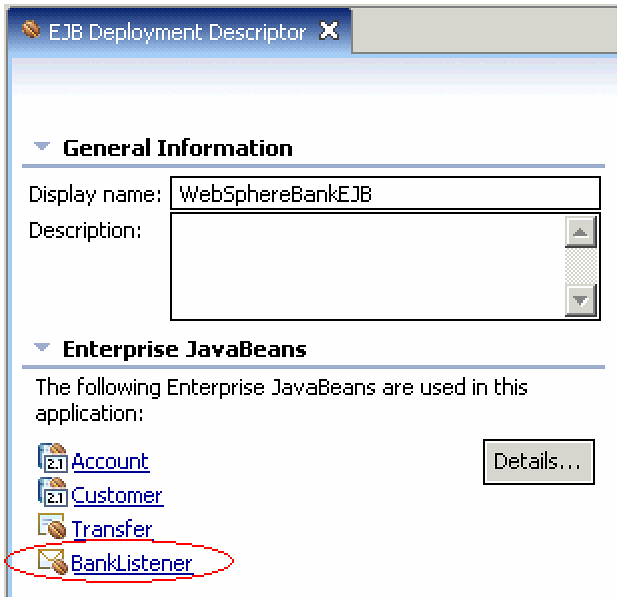
____ 3. Open the WebSphereBankEJB Deployment Descriptor.

__ a. In the J2EE perspective, expand **EJB Projects > WebSphereBankEJB** and double click on **Deployment Descriptor: WebSphereBankEJB**. This will open the Deployment Descriptor in the edit window to the right.

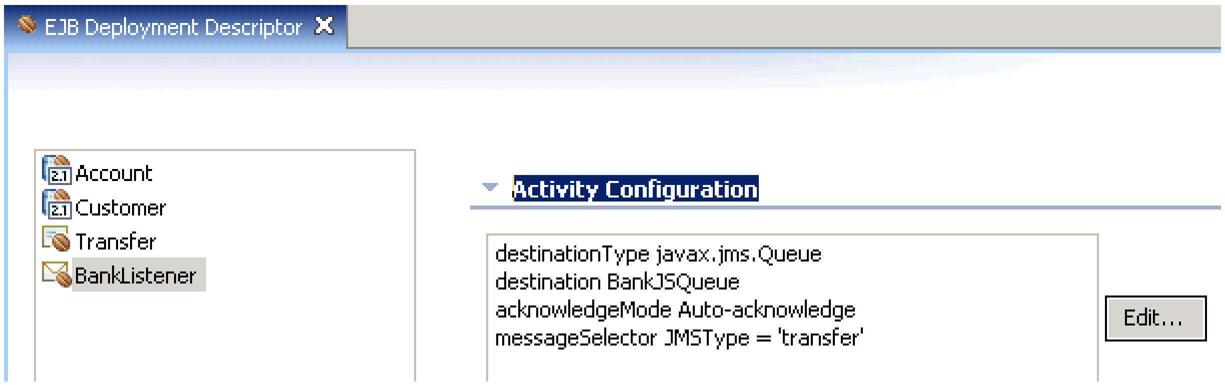


____ 4. View the Activity Configuration and Binding information

__ a. Click on **BankListener**.

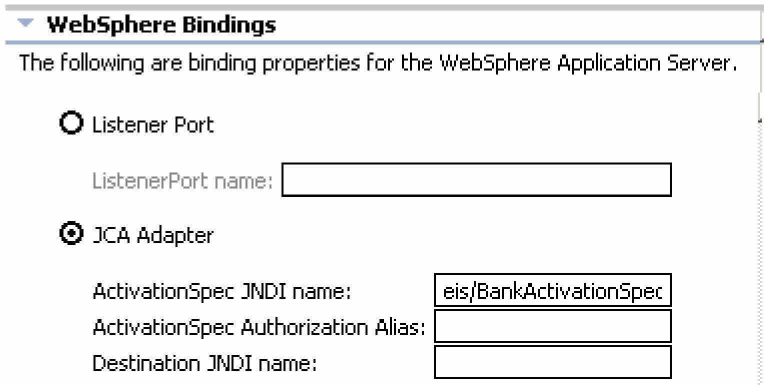

EJB Deployment Descriptor
General Information
Display name: WebSphereBankEJB
Description:
Enterprise JavaBeans
The following Enterprise JavaBeans are used in this application:
Account
Customer
Transfer
BankListener
Details...

__ b. View the Activity Configuration information. Notice the destination type is queue and the JMS type is transfer.



```
destinationType javax.jms.Queue
destination BankJSQueue
acknowledgeMode Auto-acknowledge
messageSelector JMSType = 'transfer'
```

__ c. View the binding information. Notice this uses an Activation Spec. Also notice that Listener Port is still available.



WebSphere Bindings

The following are binding properties for the WebSphere Application Server.

○ Listener Port

ListenerPort name:

◉ JCA Adapter

ActivationSpec JNDI name: eis/BankActivationSpec

ActivationSpec Authorization Alias:

Destination JNDI name:

____ 5. Exit AST.

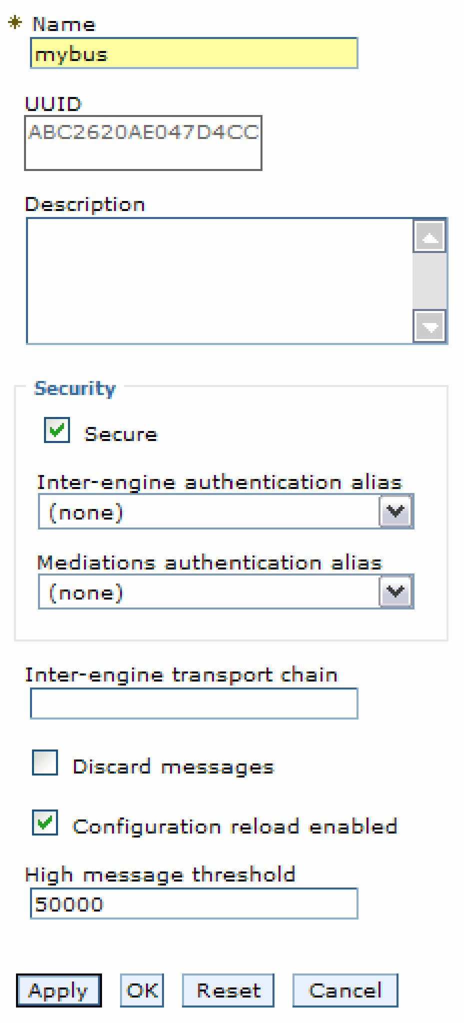
Part 4: Preparing the JMS Resources

____ 1. Create a service integration bus.

__ a. In the Administrative Console, expand **Service Integration**. Click **Buses.**

__ b. Click **New** to create a new bus.

__ c. Enter the name as **mybus.**



* Name: mybus

UUID: ABC2620AE047D4CC

Description:

Security

[✓] Secure

Inter-engine authentication alias: (none)

Mediations authentication alias: (none)

Inter-engine transport chain:

[ ] Discard messages

[✓] Configuration reload enabled

High message threshold: 50000

__ d. Click **OK**.

____ 2. Save changes.

__ a. In the Messages area, click **Save**.

__ b. Click **Save** again to save changes to master configuration.

____ 3. Make <SERVER_NAME> a member of mybus

__ a. Under **Service Integration**, click **Buses.**

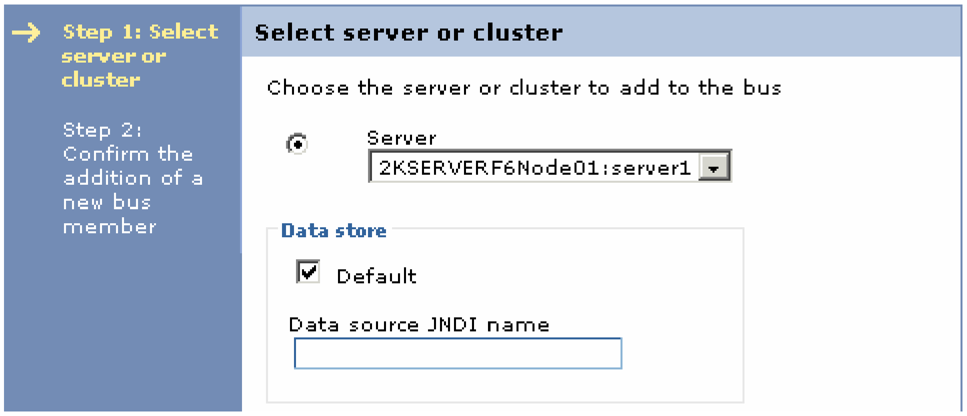
__ b. Click **mybus.**

__ c. Under Additional Properties, select **Bus Members.**

__ d. Click **Add** to add a new bus member.

__ e. Make sure that the **Server** radio button is selected.

__ f. From the server selection box, select **<mynodename>:<SERVER_NAME>.**


Step 1: Select server or cluster
Step 2: Confirm the addition of a new bus member

Select server or cluster

Choose the server or cluster to add to the bus

● Server: 2KSERVERF6Node01:server1

Data store

[✓] Default

Data source JNDI name

__ g. Click **Next.**

__ h. Click **Finish.**

_____ 4. Save changes.

__ a. In the Messages area, click **Save**.

__ b. Click **Save** again to save changes to master configuration.

_____ 5. Let's now create the SIB destination. In this exercise, we are using point-to-point messaging, i.e., our MDB needs to listen to a queue. So, let's create a SIB queue.

__ a. Under **Service Integration**, click **Buses.**

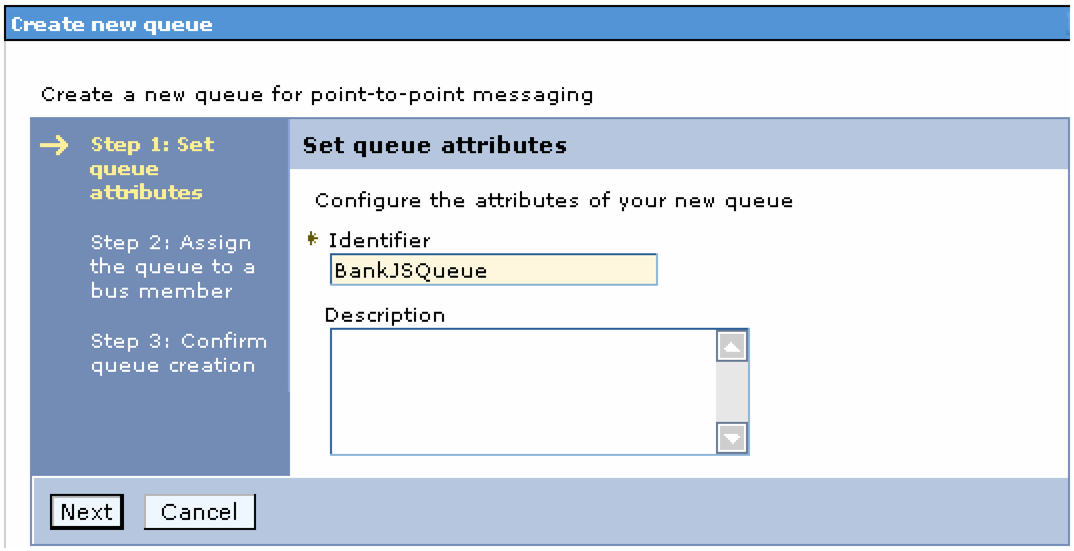
__ b. Click on the bus that we just created, **mybus.**

__ c. In the Additional Properties, click **Destinations.**

__ d. Click **New** to create a new destination.

__ e. Select **Queue** as the Destination Type and click **Next.**

__ f. In the Set Queue Attributes panel, enter the Identifier as **BankJSQueue.**


Create new queue
Step 1: Set queue attributes
Step 2: Assign the queue to a bus member
Step 3: Confirm queue creation
Next
Cancel

Create a new queue for point-to-point messaging

Set queue attributes

Configure the attributes of your new queue

Identifier: BankJSQueue

Description

__ g. Click **Next.**

__ h. In the Assign Bus member panel, <SERVER_NAME> is selected by default. Click **Next.**

__ i. Click **Finish** on the Confirmation Panel.

_____ 6. Save changes.

__ a. In the Messages area, click **Save** .

__ b. Click **Save** again to save changes to master configuration.

_____ 7. Let's now create the JMS resources that will point at the SIB Destination. Create a JMS Connection factory.

__ a. Expand **Resources > JMS Providers**. Click **Default Messaging**.

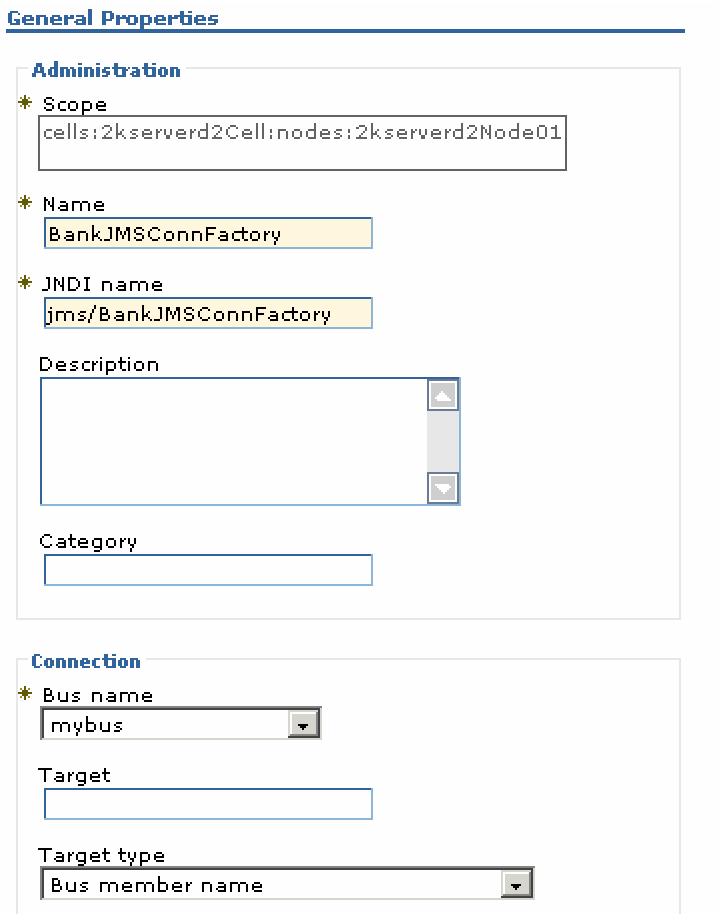
__ b. Under Connection Factories, click **JMS connection factory.**

__ c. Click **New.**

__ d. Enter the following:

Name:	**BankJMSConnFactory**
JNDI Name:	**jms/BankJMSConnFactory**
Bus Name:	**mybus**

Leave all the other values to the default.



General Properties

Administration

* Scope: cells:2kserverd2Cell:nodes:2kserverd2Node01

* Name: BankJMSConnFactory

* JNDI name: jms/BankJMSConnFactory

Description:

Category:

Connection

* Bus name: mybus

Target:

Target type: Bus member name

__ e. Click **OK.**

_____ 8. Save changes.

__ a. In the Messages area, click **Save**.

__ b. Click **Save** again to save changes to master configuration.

_____ 9. Create a JMS Queue.

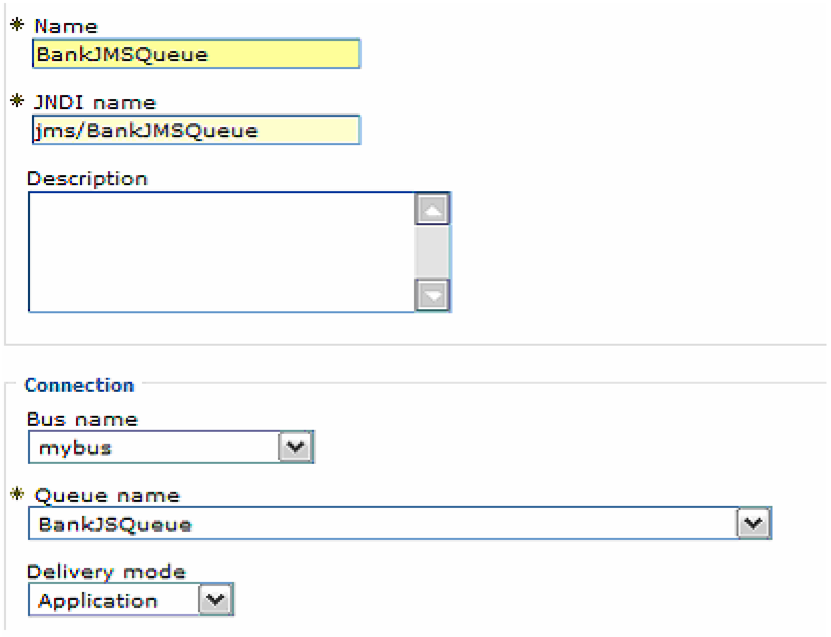
__ a. Under **Resources > JMS Providers**, select **Default Messaging.**

__ b. Under Destinations, click **JMS Queue**. Click **New.**

__ c. Enter the following:

Name:	**BankJMSQueue**
JNDI Name:	**jms/BankJMSQueue**
Queue Name:	**BankJSQueue**
Bus Name:	**mybus**

Leave all the other values to default.



* Name
BankJMSQueue

* JNDI name
jms/BankJMSQueue

Description

Connection

Bus name
mybus

* Queue name
BankJSQueue

Delivery mode
Application

__ d. Click **OK**.

_____ 10. Save changes.

__ a. In the Messages area, click **Save**.

__ b. Click **Save** again to save changes to master configuration.

_____ 11. Create an activation spec for the Message Driven Bean.

__ a. Under **Resources > JMS Providers**, select **Default Messaging.**

__ b. Under Activation Specs, click **JMS activation specifications**. Click **New.**

__ c. Enter the following:

Name:	**BankActivationSpec**
JNDI Name:	**eis/BankActivationSpec**
Destination JNDI Name:	**jms/BankJMSQueue**
Bus Name:	**mybus**
Authentication Alias:	**<cell name>\samples**

Leave all the other values to default.

__ d. Click **OK.**

_____ 12. Save changes.

__ a. In the Messages area, click **Save**.

__ b. Click **Save** again to save changes to master configuration.

____ 13. Stop the server.

__ a. Change directories to **<WAS_HOME>\profiles\default\bin.**

__ b. Run the following command:

```
./stopserver.sh <SERVER_NAME>
```

Part 5: Testing the Application

____ 1. Re-start the WebSphere server.

__ a. Change directories to **<WAS_HOME>\profiles\default\bin.**

__ b. Run the following command:

```
./startserver.sh <SERVER_NAME>
```

____ 2. Open a new browser window.

____ 3. Enter the URL http://<HOST_NAME>:9080/WebSphereBankWeb/.

____ 4. Test the datasource.

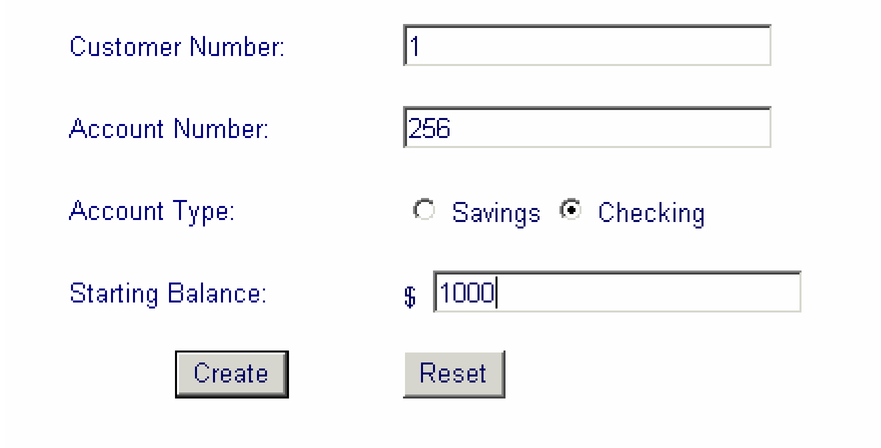
__ a. Click on **Create Customer**.

__ b. Enter **1** for Customer Number, enter Name and Tax ID. Select **Create**.

__ c. Click on **Create Account.**

__ d. Enter these values:

Customer Number:	1
Account Number:	256
Account Type	Checking
Starting Balance	1000


Customer Number: 1

Account Number: 256

Account Type: Savings [✓ Checking]

Starting Balance: $ 1000

__ e. Click **Create.**

__ f. Create a second account using these values:

Customer Number:	1
Account Number:	257
Account Type	Saving

Starting Balance	1000

__ g. Click **Create.**

If the accounts are created without generating errors, then the datasource is working.

____ 5. Test the Account and Transfer entity beans.

__ a. Click **Transfer Funds.**

__ b. Enter these values:

From Account	256
To Account	257
Amount	5

__ c. Click **Transfer.**

__ d. The balances displayed should reflect that $5 was moved from account 256 to account 257.

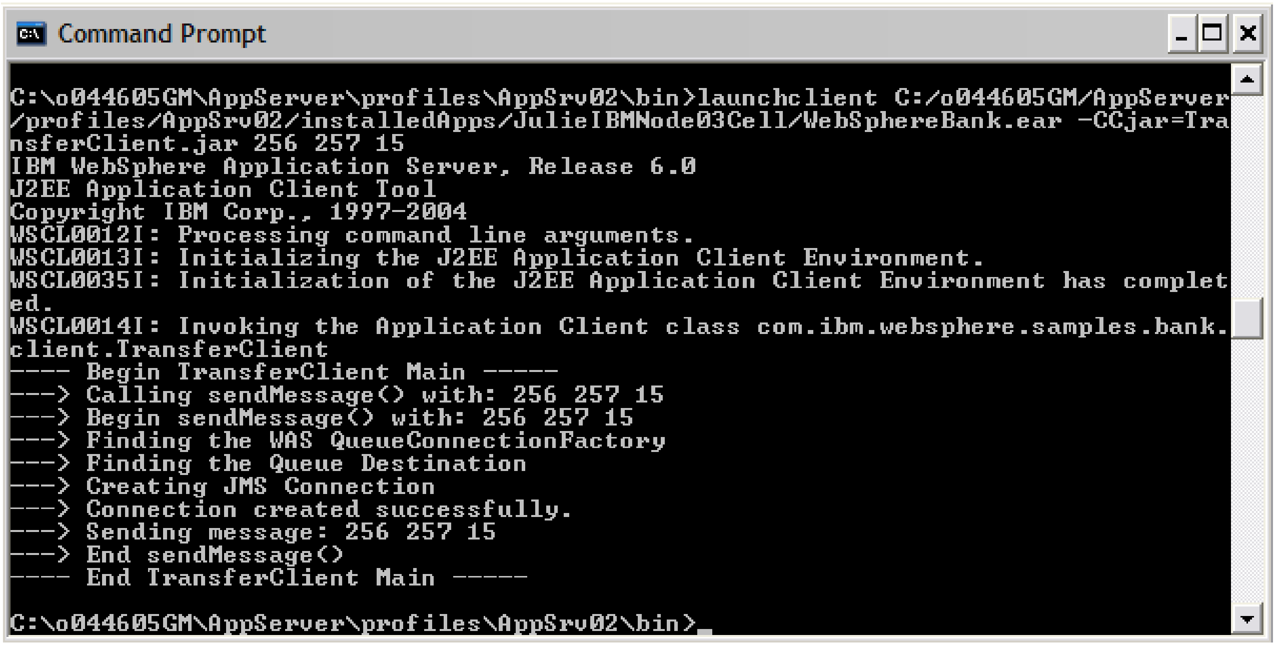
____ 6. Test the Message Driven Bean.

__ a. From a command prompt, navigate to **<WAS_HOME>\profiles\default\bin.**

__ b. Transfer $15 from account 256 to account 257 using the following command:

```
./launchClient.sh
<WAS_HOME>/profiles/default/installedApps/<MyCell>/WebSphereBank.ear –
CCjar=TransferClient.jar 256 257 15
```

Note: Remember to substitute your cell name with <MyCell>. You can also look at the file structure under <WAS_HOME>/profiles directory to determine the cell name for your Application Server installation.



Command Prompt

```
C:\o044605GM\AppServer\profiles\AppSrv02\bin>launchclient C:/o044605GM/AppServer
/profiles/AppSrv02/installedApps/JulieIBMNode03Cell/WebSphereBank.ear -CCjar=Tra
nsferClient.jar 256 257 15
IBM WebSphere Application Server, Release 6.0
J2EE Application Client Tool
Copyright IBM Corp., 1997-2004
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has complet
ed.
WSCL0014I: Invoking the Application Client class com.ibm.websphere.samples.bank.
client.TransferClient
---- Begin TransferClient Main -----
---> Calling sendMessage() with: 256 257 15
---> Begin sendMessage() with: 256 257 15
---> Finding the WAS QueueConnectionFactory
---> Finding the Queue Destination
---> Creating JMS Connection
---> Connection created successfully.
---> Sending message: 256 257 15
---> End sendMessage()
---- End TransferClient Main -----

C:\o044605GM\AppServer\profiles\AppSrv02\bin>_
```

____ 7. In the browser window for the WebSphereBank application, test the balances of the accounts.

__ a. Click **Get Balance.**

__ b. Enter the account number (**256**).

__ c. Click **Balance.**

__ d. The balance displayed should reflect a transfer of $15 from account 256 to account 257.

Successful completion of a transfer from the command line client indicates that the client code successfully placed a message on the queue, and the message-driven bean was driven by that message to contact the Transfer bean, which in turn accessed two instances of the Account bean and changed the balances of each. Since the Account and Transfer beans were tested in the previous step, this simply confirms the function of the MDB.

____ 8. Stop the server.

__ a. In the command window, enter:

```
./stopServer.sh <SERVER_NAME>
```

__ b. This completes the lab.

What you did in this exercise

In Part1, you installed the WebSphereBank sample and also created the JDC resources using a provided .jacl script. In part 2, you used the AST to view deployment description information. In Part3, you created the JMS resources using the Administrative Console.

In Part 4 of the exercise you tested the newly created datasource, the newly installed WebSphereBank application, and the message driven bean.