# JDBCInbound Lab



## What this exercise is about

The objective of this lab is to provide you with an understanding of the WebSphere Adapter for JDBC and inbound event processing.

## Lab requirements

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

- WebSphere Integration Developer V6.0.2 installed
- WebSphere Process Server V6.0.2 test environment installed
- WebSphere Adapter for JDBC V6.0.2 installed
- Sample code in the directory C:\Labfiles602\JDBC (Windows®) or /tmp/LabFiles602/JDBC (Linux®)

## What you should be able to do

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

- Install and deploy the Adapter for JDBC and integrate it into an SCA application for use with inbound event processing.

# Introduction

This lab introduces you to the WebSphere Adapter for JDBC and the processing of inbound events that have occurred in a database table. It uses a JDBCTEST Cloudscape database along with a CUSTOMER table for user data and a WBIA_JDBC_EVENTSTORE table to record events. In the lab, you will import the JDBC Adapter into WebSphere Integration Developer and run the Enterprise Service Discovery wizard to input connection information, create a service description, and discover objects existing in the specified table. You will then assemble an SCA application, wiring together a Java™ component and the EIS export file. In the Java component, you will add implementations for create, update and delete methods.

Before working with the adapter, you will first add a record to the CUSTOMER table, creating a create event. The create trigger runs, inserting a record for that event into the event store table. The event has occurred and is now waiting in the table for the JDBCTestInbound application to start and the JDBC Adapter to start polling for events. To test the inbound application, you will use the WebSphere Test Environment.

# 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 run 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:

| Reference Variable | Windows Location | AIX®/UNIX® Location |
|---|---|---|
| <LAB_NAME> | JDBCInBound | |
| <WID_HOME> | C:\Program Files\IBM\WebSphere\ID**6.0.2** | |
| <WPS_HOME> | <WID_HOME>\runtimes\bi_v6 | |
| <JDBCADAPTER_HOME> | <WID_HOME>\Resource Adapters\JDBC | |
| <LAB_FILES> | C:\Labfiles602 | /tmp/Labfiles602 |
| <TEMP> | C:\temp | /tmp |

**Windows users 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:\LabFiles602\ would be replaced by C:/LabFiles602/

# Part 1: Create the JDBCTEST Cloudscape database and tables

In this part you will create the JDBCTEST database along with the CUSTOMER table for user data and the WBIA_JDBC_EVENTSTORE table for recording events. You will also add triggers to the CUSTOMER table that will fire whenever create, update, or deletes occur to the CUSTOMER table, inserting a record into the event table. To cause an event, you will create at least one record in the CUSTOMER table. This will fire the trigger for the create operation which will add a record to the WBIA_JDBC_EVENTSTORE.

____ 1. Start Cloudscape CView Graphical User Interface (GUI) by running the cview.bat.

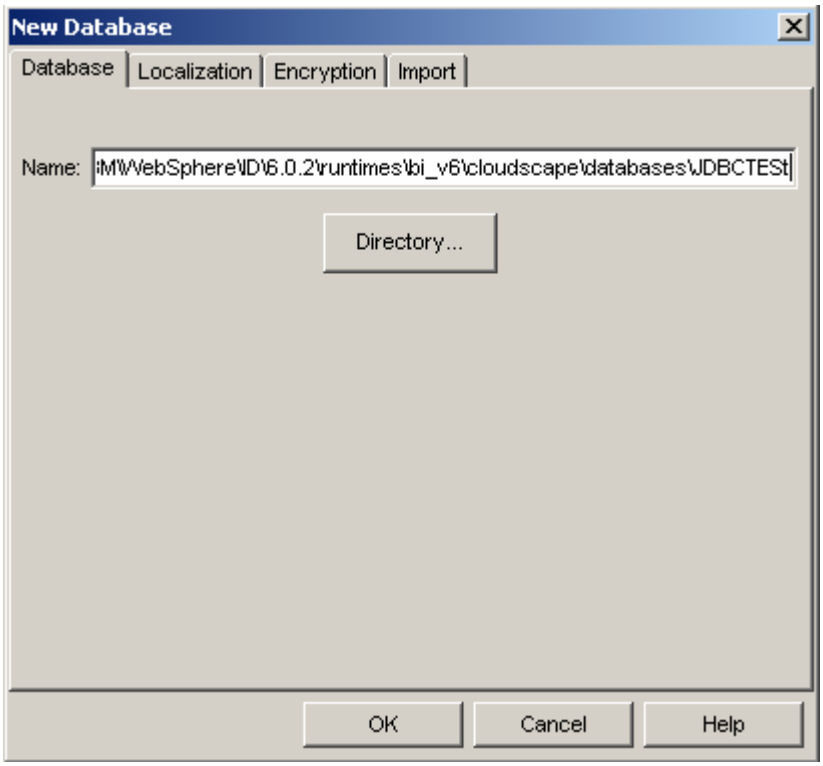
Note: The Cloudscape embedded driver that is being used in the lab, supports a connection from only one JVM at a time. You can have either the server running and connected to the JDBCTEST database, or the Cview GUI connected to the JDBCTEST database; but not both at the same time.

__ a. Open a command prompt window, navigate to the following subdirectory, and run the cview.bat program.

```
<WPS_HOME>\cloudscape\bin\embedded>cview
```

____ 2. Using the CView GUI, create the JDBCTEST database if it doesn't already exist. If you've already completed the JDBCOutbound lab, the database, tables, and triggers are the same. You can skip to step 7. where you add records to the CUSTOMER table.

__ a. Select from the menu, **File -> New -> Database** Click the **Directory …** button and browse to **<WPS_HOME>\cloudscape\databases** subdirectory , enter the name of the database to create, **JDBCTEST,** and click **Open.** You will return to the **New Database** panel as shown: Click **OK**



____ 3. Select the JDBCTEST Database, select the Database tab, and enter SQL to create two tables: CUSTOMER table for user data and WBIA_JDBC_EVENTSTORE table for recording events.

---

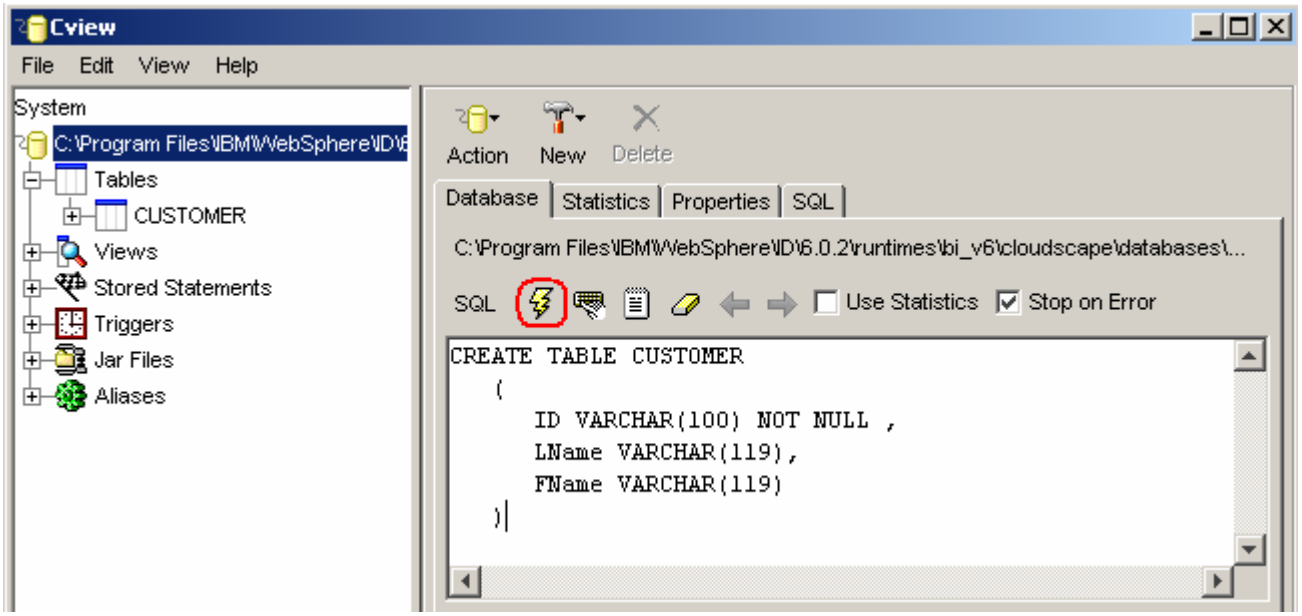
**NOTE:** For your convenience, the following SQL code snippets can be found in **<LAB_FILES>\JDBC\snippets\CUSTOMERSQL.txt**

---

__ a. Select the JDBCTEST Database, select the Database tab on the right, paste the following into the SQL window:

```
CREATE TABLE CUSTOMER
   (
      ID VARCHAR(100) NOT NULL ,
      LName VARCHAR(119),
      FName VARCHAR(119)
   )
```

__ b. Select the "lightening bolt" icon to run the SQL (The SQL text will appear highlighted after executing.)




__ c. Set the key for the CUSTOMER table by pasting the following into the SQL window (you can paste over the top of and replace the previous existing SQL shown in the window)

```
ALTER TABLE CUSTOMER
ADD CONSTRAINT NEW_KEYCU Primary Key (
ID)
```

__ d. Select the "lightening bolt" icon to run the SQL

____ 4. Create the WBIA_JDBC_EVENTSTORE table and create the primary key.

---

**NOTE:** For your convenience, the following SQL code snippets can be found in **<LAB_FILES>\JDBC\snippets\EVENTSTORESQL.txt**

---

__ a. Create the WBIA_JDBC_EVENTSTORE table by pasting the following into the SQL window:

```
CREATE TABLE WBIA_JDBC_EVENTSTORE
  (
    EVENT_ID INT DEFAULT AUTOINCREMENT  INITIAL 1 INCREMENT 1 NOT NULL ,
    OBJECT_KEY VARCHAR(80) NOT NULL ,
    OBJECT_NAME VARCHAR(40) NOT NULL ,
    OBJECT_FUNCTION VARCHAR(40) NOT NULL ,
    EVENT_PRIORITY INT NOT NULL ,
    EVENT_TIME TIMESTAMP DEFAULT CURRENT TIMESTAMP NOT NULL ,
    EVENT_STATUS INT NOT NULL ,
    EVENT_COMMENT VARCHAR(100),
    XID VARCHAR (255)
  )
```

__ b. Select the “lightning bolt” icon to run the SQL.

__ c. Create the primary key by pasting the following into the SQL window:

```
ALTER TABLE WBIA_JDBC_EVENTSTORE
  ADD CONSTRAINT "c3350098-0104-7683-90b5-ffffe0415391" Primary Key (
    EVENT_ID)
```

__ d. Select the “lightening bolt” icon to run the SQL.

____ 5. Create triggers on the CUSTOMER database for create, update, and delete events.

---

**NOTE:** For your convenience, the following SQL code snippets can be found in **<LAB_FILES>\JDBC\snippets\CUSTOMERTRIGGERSQL.txt**

---

__ a. Create the trigger for the create event by pasting the following into the SQL window:

```
CREATE TRIGGER event_create
AFTER INSERT ON CUSTOMER REFERENCING NEW AS N
FOR EACH ROW MODE DB2SQL
INSERT INTO wbia_jdbc_eventstore (object_key, object_name,
object_function, event_priority, event_status)
VALUES (N.id, 'AppCustomerBG', 'Create', 1, 0)
```

__ b. Select the “lightening bolt” to run the SQL.

__ c. Create the trigger for the update event by pasting the following into the SQL window:

```
CREATE TRIGGER event_update
AFTER UPDATE ON CUSTOMER REFERENCING NEW AS N
FOR EACH ROW MODE DB2SQL
INSERT INTO wbia_jdbc_eventstore (object_key, object_name,
object_function, event_priority, event_status)
VALUES (N.id, 'AppCustomerBG', 'Create', 1, 0)
```

__ d. Select the “lightening bolt” to run the SQL.

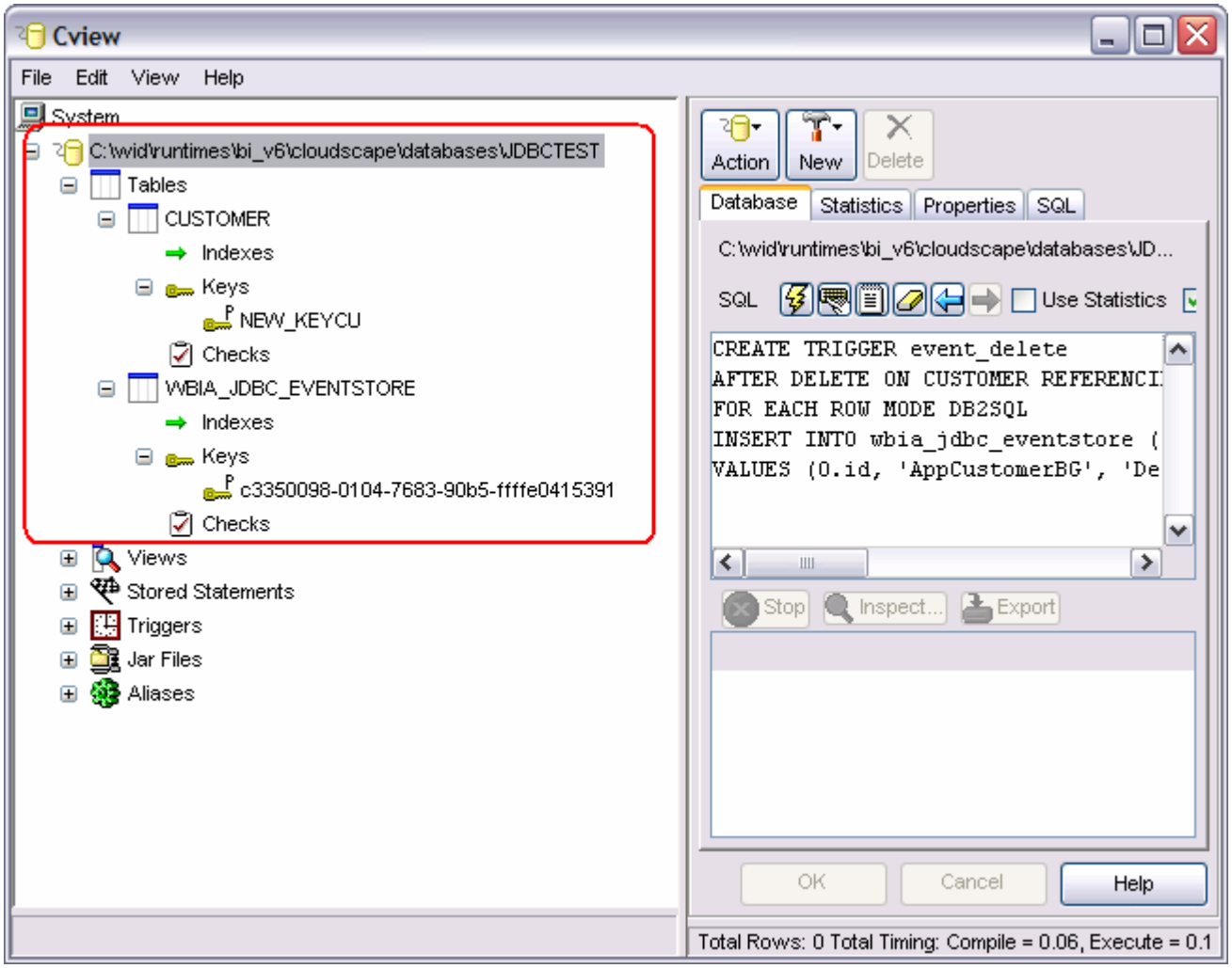
__ e. Create the trigger for the delete event by pasting the following into the SQL window:

```
CREATE TRIGGER event_delete
AFTER DELETE ON CUSTOMER REFERENCING OLD AS O
```

```
FOR EACH ROW MODE DB2SQL
INSERT INTO wbia_jdbc_eventstore (object_key, object_name,
object_function, event_priority, event_status)
VALUES (O.id, 'AppCustomerBG', 'Delete', 1, 0)
```

__ f. Select the “lightening bolt” to run the SQL.

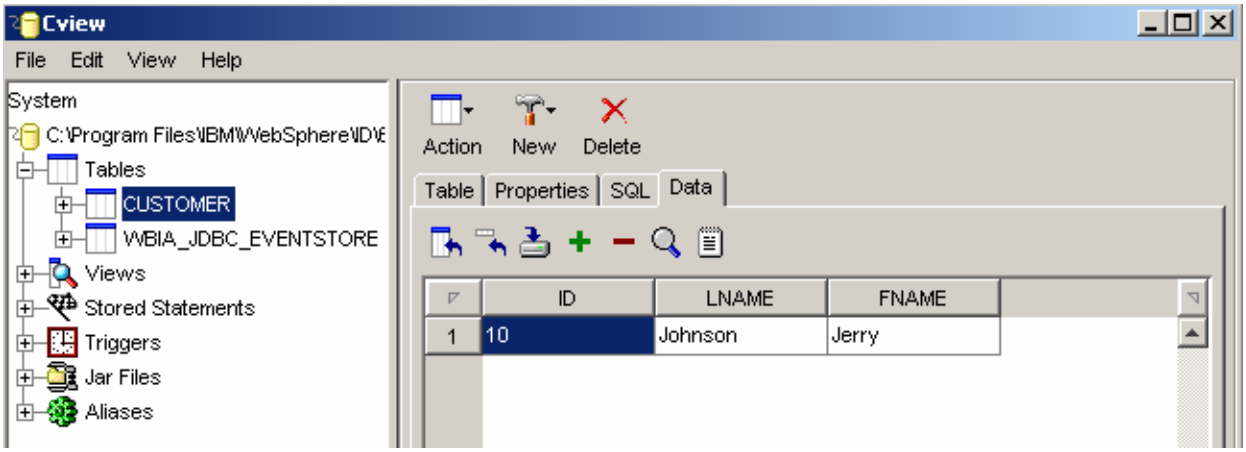
____ 6. Verify that your JDBCTEST database and tables look similar to the following.



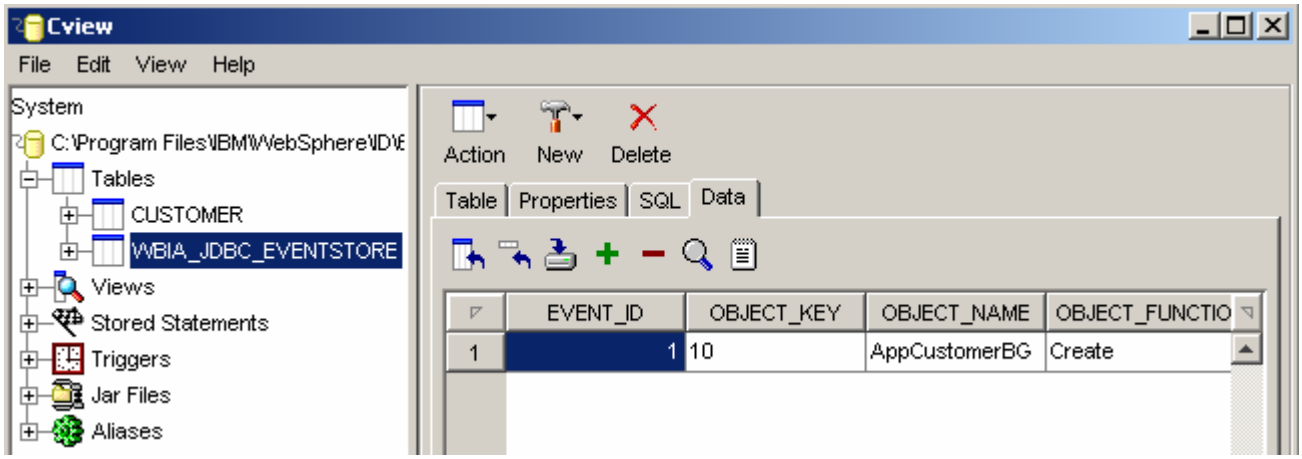

____ 7. Using the CView GUI, create a record in the CUSTOMER Table.

__ a. Expand the Tables tree underneath the JDBCTEST database. Select the CUSTOMER Table. Select the Data tab at the far right. Click in the open area below the column names.

__ b. Add an entry to the CUSTOMER table by typing in values for **ID, LNAME, and FNAME**. Specify ID of **10**, LNAME of **Johnson**, and FNAME of **Jerry**. Click **OK.**




__ c. Verify that the WBIA_JDBC_EVENTSTORE now has an event record for the customer record just created. Select the WBIA_JDBC_EVENTSTORE Table. Select the Data tab at the far right. You should see a record with OBJECT_KEY of **10** and OBJECT_FUNCTION of **Create**.




__ d. Close the JDBCTEST database by selecting File -> Close. Exit Cview GUI by selecting File -> Exit.

---

Note: You must exit the Cview GUI, before starting the server as the Cloudscape “embedded” database driver used in this lab allows for only one JVM connection at a time.

---

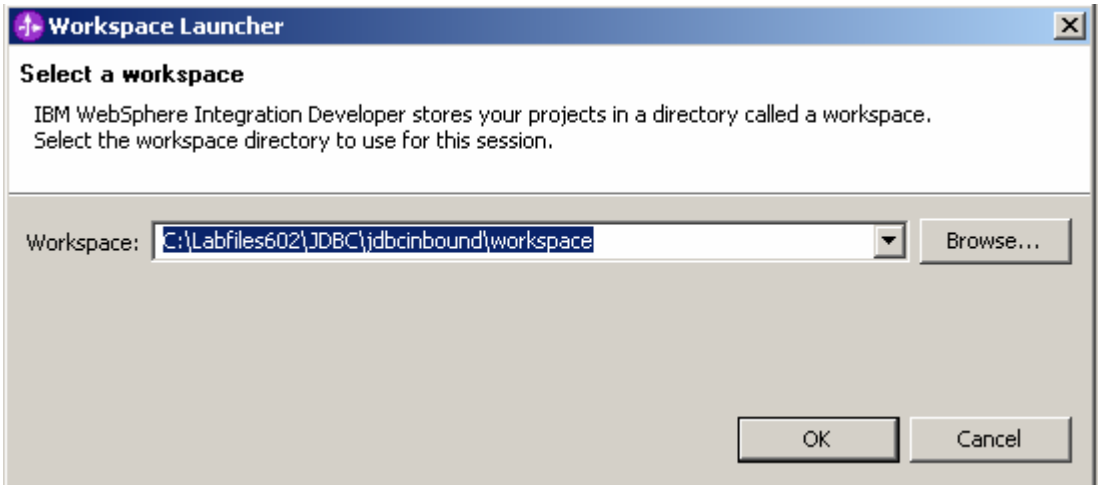
# Part 2: Initialize workspace and import RAR into WebSphere Integration Developer

In this part, you will start WebSphere Integration Developer, using a new workspace, and set up the WebSphere Process Server to be used as the WebSphere test environment.

____ 1. Start WebSphere Integration Developer V6 with a new workspace.

__ a. From the start menu select **Start > Programs > IBM WebSphere > Integration Developer V6.0.1> WebSphere Integration Developer V6.0.1**

__ b. When prompted enter **<LAB_FILES>\JDBC\jdbcinbound\workspace** for your workspace and click **OK**




__ c. When WebSphere Integration Developer V6.0.1 opens, close the **Welcome page**


____ 2. Import the JDBC Adapter Resource Archive (RAR) file. This will create a J2EE Connector Project.

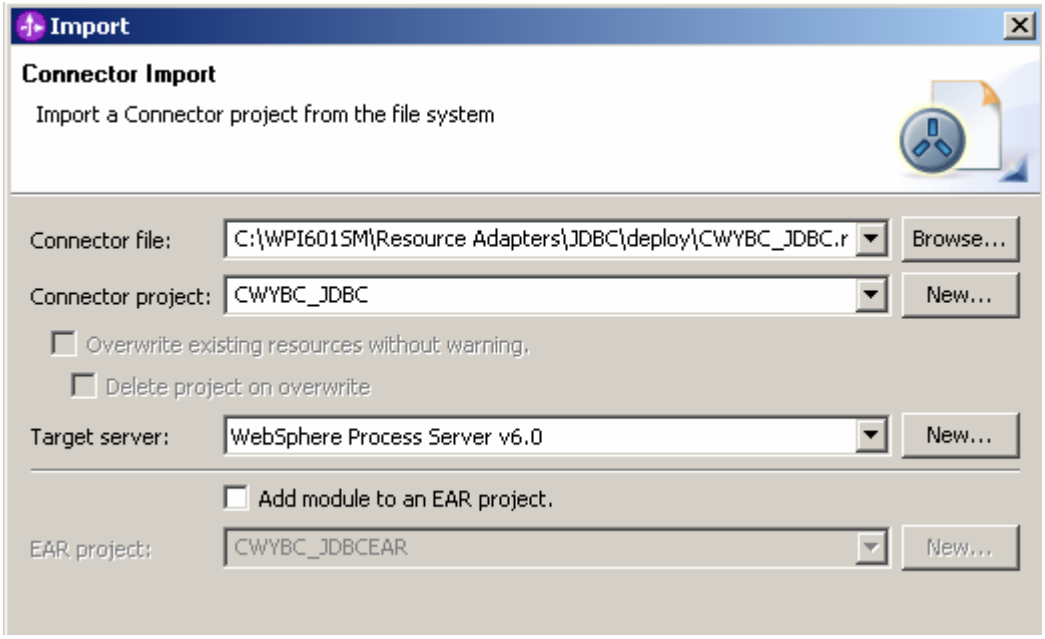
__ a. From the top Menu bar, select **File -> Import**

__ b. Select **RAR file,** click **next**

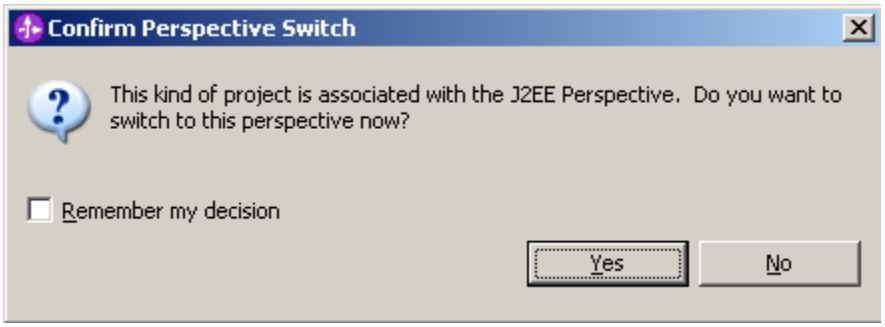
__ c. Complete the Connector Import panel

**1)** Connector file: **Browse…** to the location of the CWYBC_JDBC.rar adapter file and select it, for example: c:\<WID_HOME>\Resource Adapters\jdbc\deploy\CWYBC_JDBC.rar

**2)** Deselect the "**Add module to an EAR project**" and select **Finish**

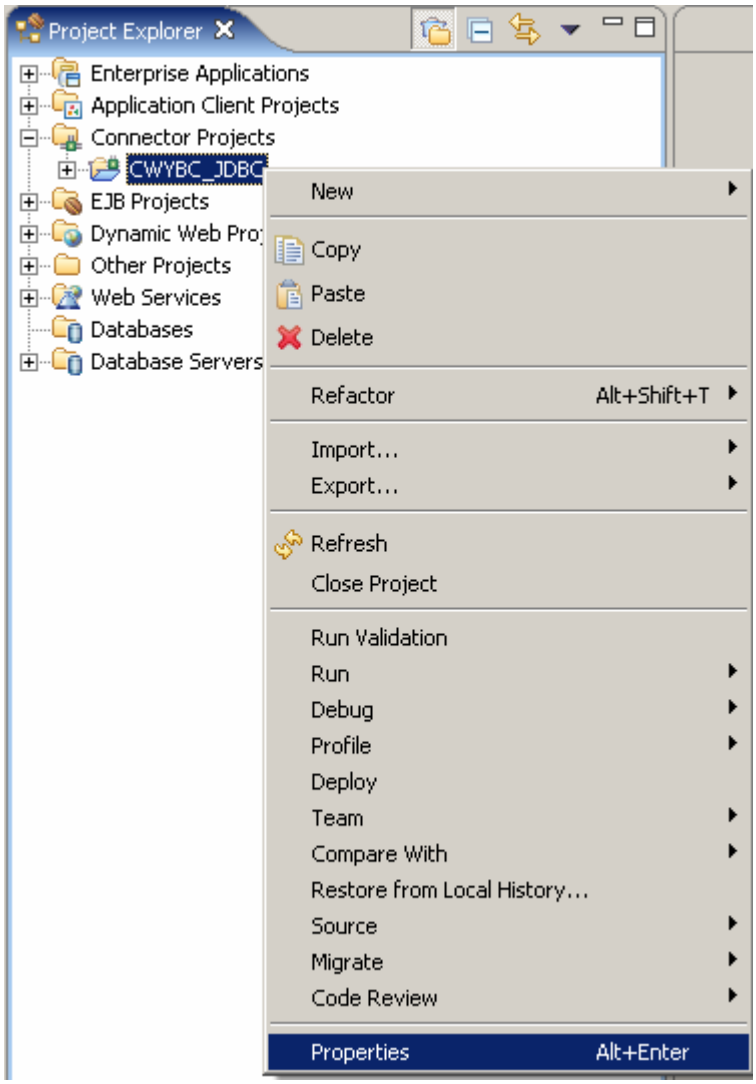



__ d. At the Confirm Perspective Switch prompt, click **Yes**

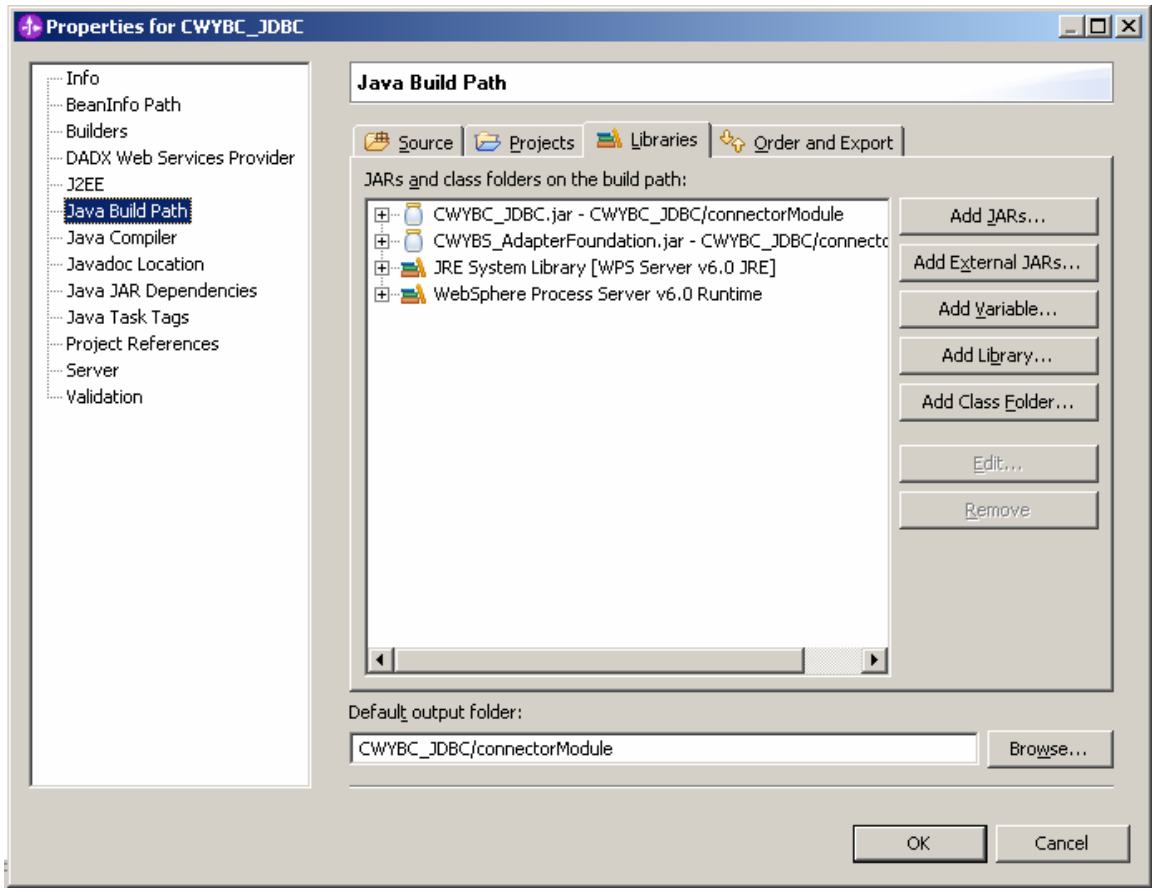



____ 3. Add any external dependencies your adapter has to the imported project. These are dependencies that the adapter may have on the JDBC applications (adapter–specific). Add the db2j.jar to the build path of the Connector project.

__ a. Expand the **Connector Projects** folder, right click the **CWYBC_JDBC** connector project, select **properties**.

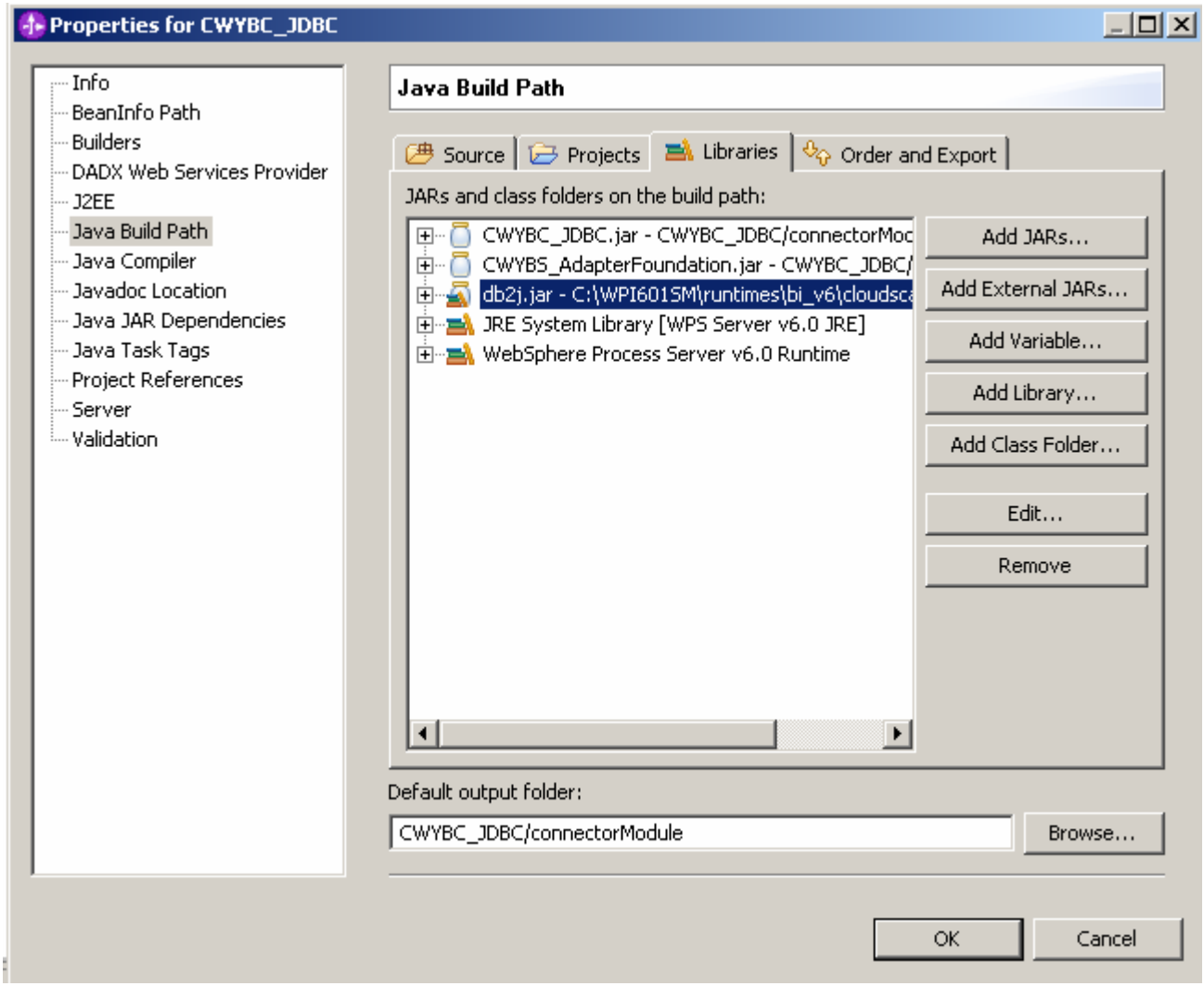


__ b. Select **Java Build Path** from the list at the left, select the **Libraries tab** from the panel at the right, select **Add External JARs...** button



__ c. Browse to the location of c:\<WPS_HOME>\cloudscape\lib and select the **db2j.jar**, click **Open**.

__ d. You will now see the db2j.jar added in your JARS and class folders in the build path: list. Click **OK**



# Part 3: Use Enterprise Service Discovery wizard to generate business objects and other artifacts

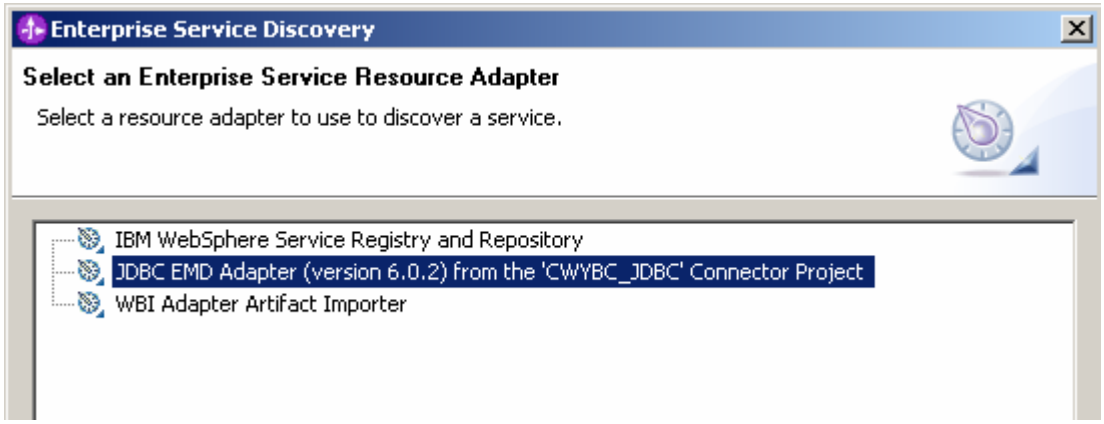
In this part you will import the WebSphere Adapter for JDBC, run Enterprise Service Discovery to discover objects and create the necessary SCA artifacts, and assemble into an SCA application.

____ 1. Switch to the Business Integration Perspective and run the Enterprise Service Discovery wizard. A Business Integration project will be created for you during this process.

__ a. From the top Menu bar, select **Window > Open Perspective > Other … > Business Integration (default)** click **OK**

__ b. From within the Business Integration Perspective, select **File > New > Enterprise Service Discovery**

__ c. Highlight the **JDBC EMD Adapter** and click **Next**




____ 2. Complete the Configure Settings for Discovery Agent panel to connect to the JDBCTEST database and discover the available services. Click **Next**

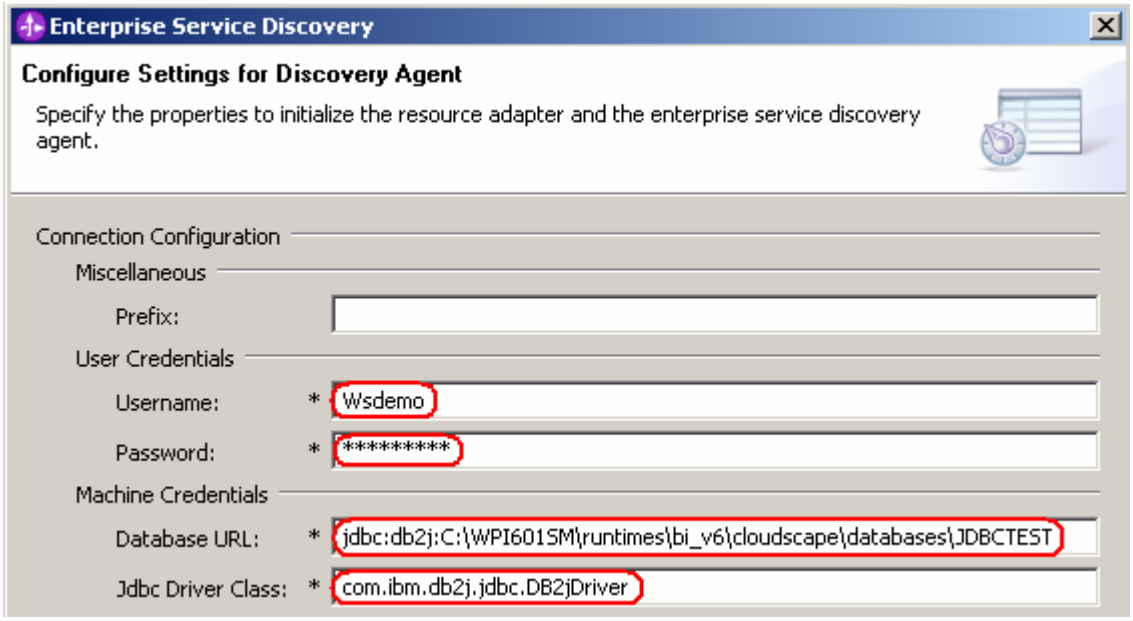
__ a. Enter valid user name and password values, for example

User name: **WSdemo**
Password: **WS15Demo1**

__ b. Enter the following values for DatabaseURL and JdbcDriverClass

DatabaseURL: **jdbc:db2j:<WPS_HOME>\runtimes\bi_v6\cloudscape\databases\JDBCTEST**

JdbcDriverClass: **com.ibm.db2j.jdbc.DB2jDriver**




__c. Click **NEXT**

____ 3. Complete the **Find and Discover Enterprise Services** panel

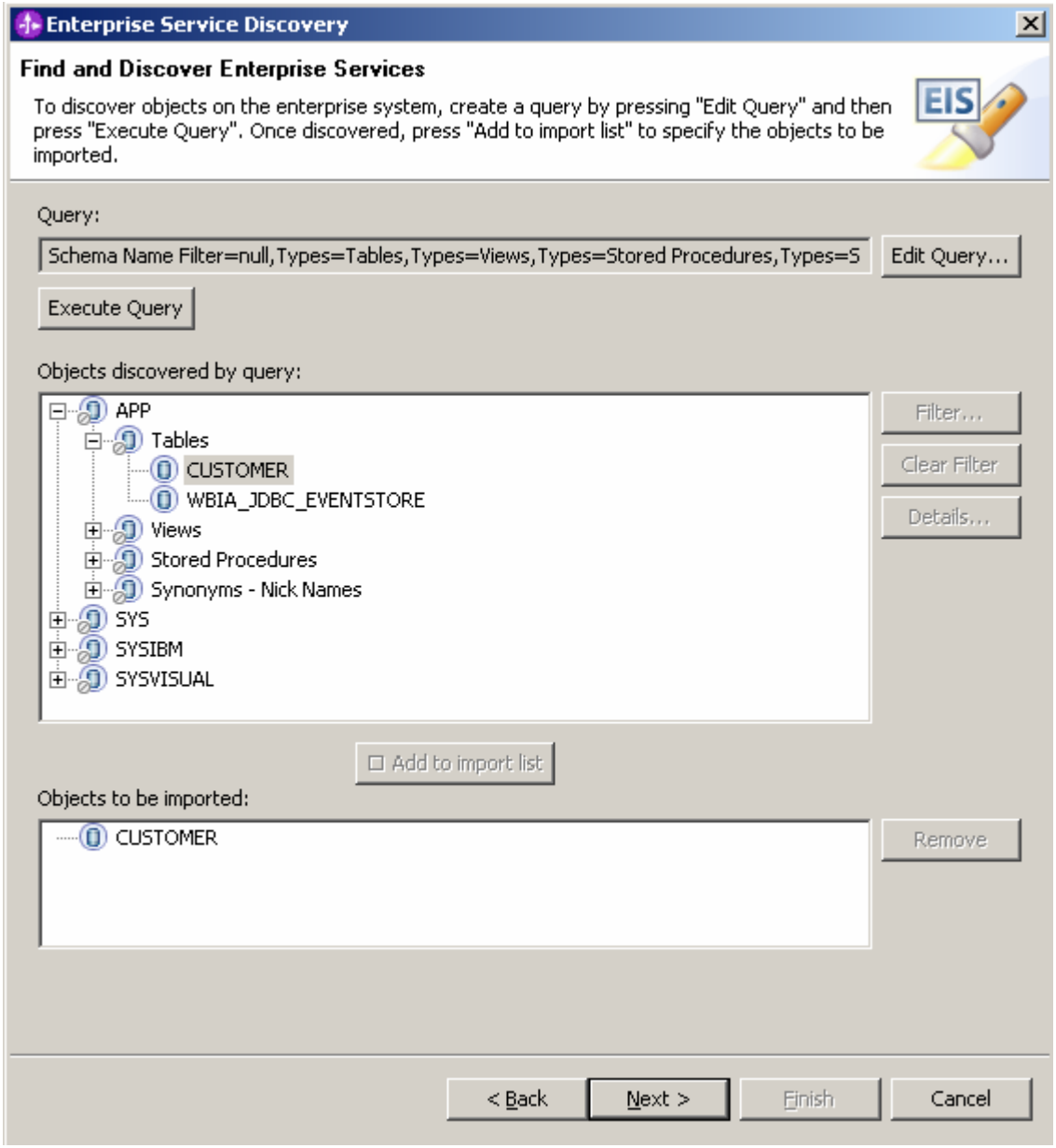
__ a. Select the **Run Query** button. A connection will be made to the Cloudscape JDBCTEST database and a selection of Meta data objects will be presented in a tree-like structure.

---

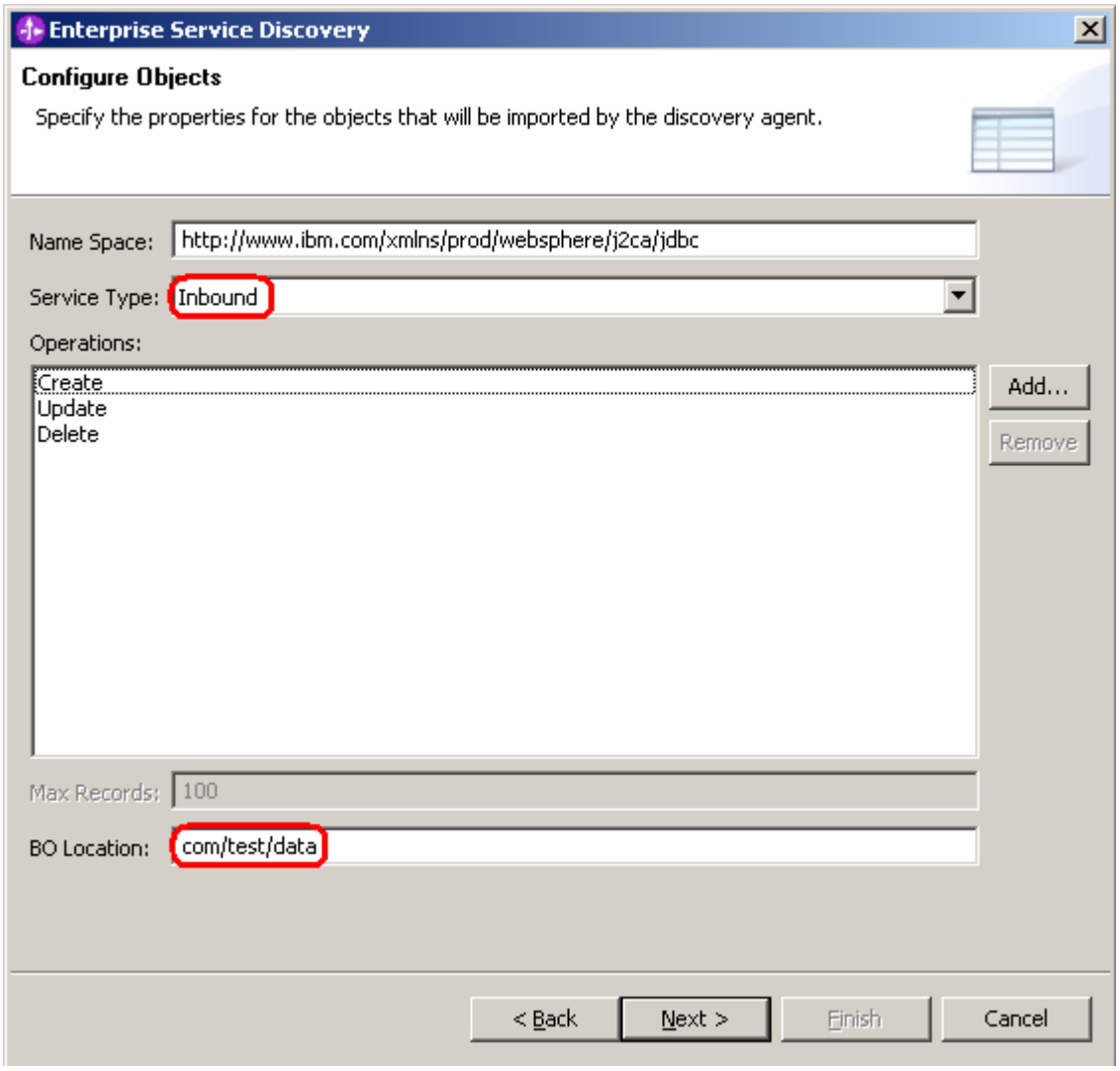
Note: Select the **Edit Query …** button to see the available options, however, don't change any of the defaults for this lab exercise.

---

__ b. Expand the schema named **APP**, (this is the default schema given to the Cloudscape tables) expand **Tables**, highlight **CUSTOMER**, click the **Add to import list** button. **CUSTOMER** now appears in the **Objects to be imported** window. Click **Next.**

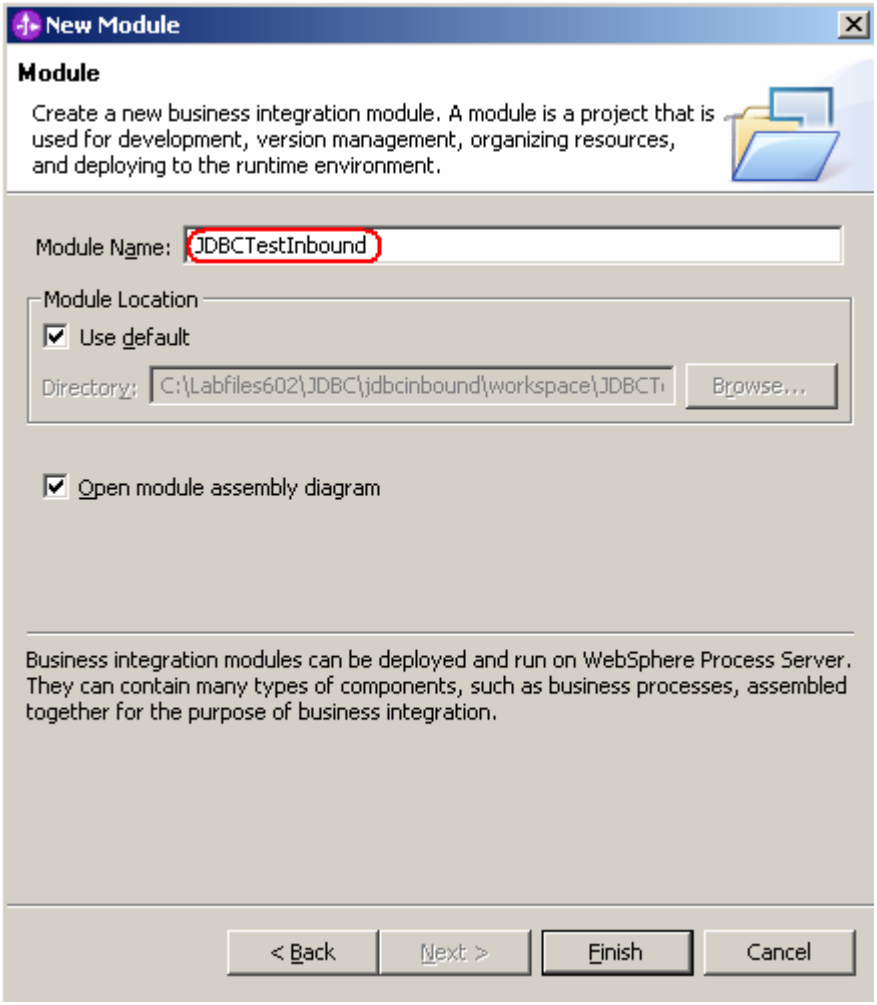


__ **c.** On the **Configure Objects** panel, leave the default value for Namespace, leave the Service Type: set to **Inbound**, and enter **com/test/data** for **BOLocation.** Note the Operations available. Click **Next.**




____ **4.** Complete the **Generate Artifacts** Panel.

__ **a.** A Business Integration Module has not yet been created, select the **New…** button and enter in the name **JDBCTestInbound** for the Module Name. Click **Finish.**




__ **b.** Enter **com/test/data** for the Folder value. Leave the default **JDBCInboundInterface** for the Name value.

__ **c.** Leave the **Deploy connector with module** box checked.

__ d. Select the radio button to the left of **Use discovered connection properties**. Additional properties options appear to complete for Activation Spec, ResourceAdapterProperties, and Miscellaneous properties.

---

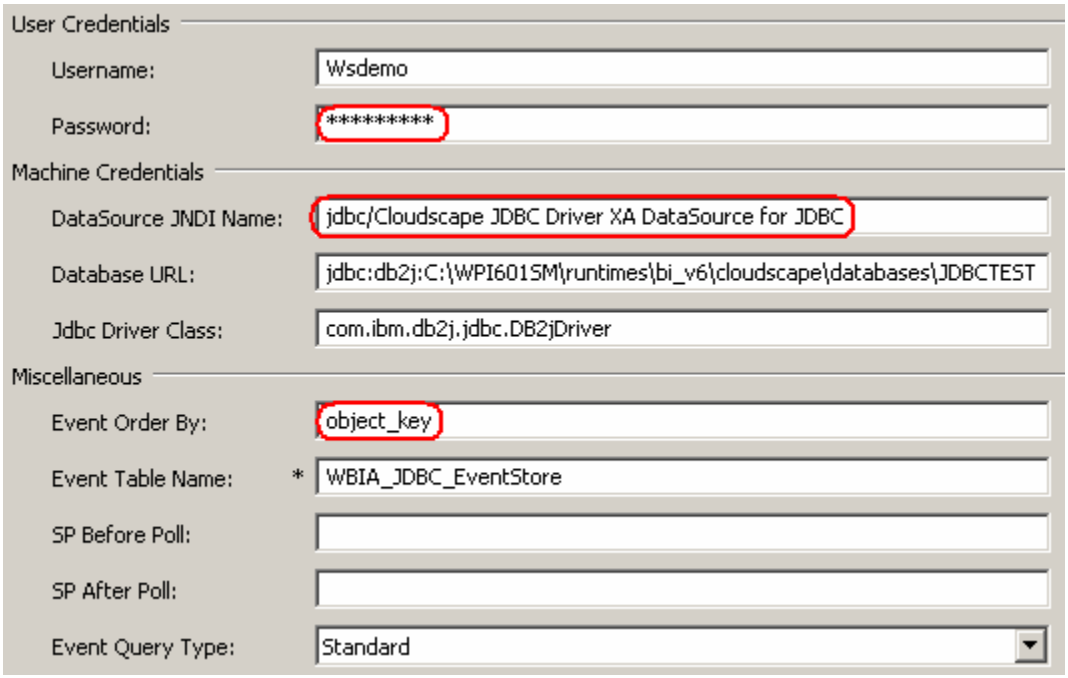
**Note:** All IBM WebSphere Adapters are supported as "Deploy connector with module" only, meaning the adapter is deployed within the module which is packaged as an Enterprise Archive file (EAR file). Therefore, the "Deploy connector with module" check box should always be selected and the "Use discovered connection properties" check box should always be selected.

---

__ e. Scroll down the list of properties available to set for the Activation Specification. Scroll down to the UserCredentials section, and enter the following values for the required fields. Click **Finish.** Wait for the workspace to complete building.

Username: **valid value** to connect to the database
Password: **valid value** to connect to the database
Data Source JNDI Name: **jdbc/Cloudscape JDBC Driver XA DataSource for JDBC**
EventOrderBy: **OBJECT_KEY**
EventTableName: **WBIA_JDBC_EVENTSTORE**

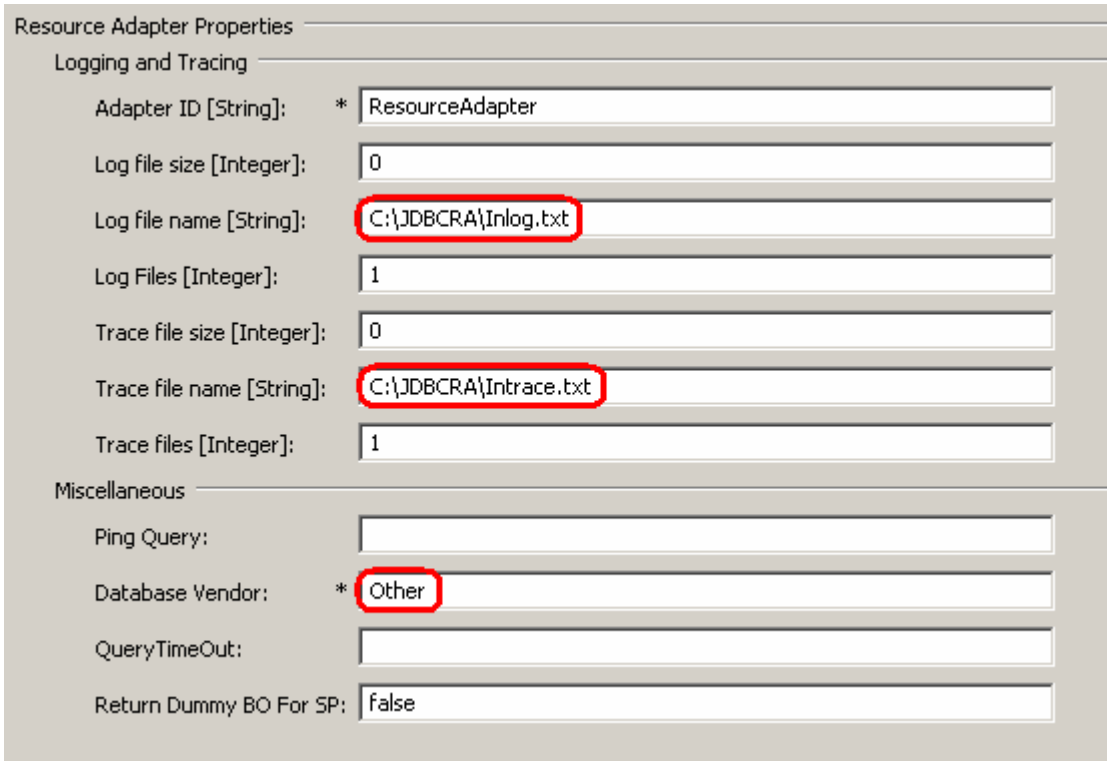
DatabaseVendor: **Other** (Enter Other since you are using Cloudscape. If you were using DB2®, Oracle, or MSSQLServer, you would enter those values instead as specific adapter processing is available with those specific databases.).



__ f. Enter the following for the **Logging and Tracing** of Resource Adapter properties

1) LogFileName: **C:\JDBCRA\Inlog.txt**

2) TraceFileName: **C:\JDBCRA\Intrace.txt**

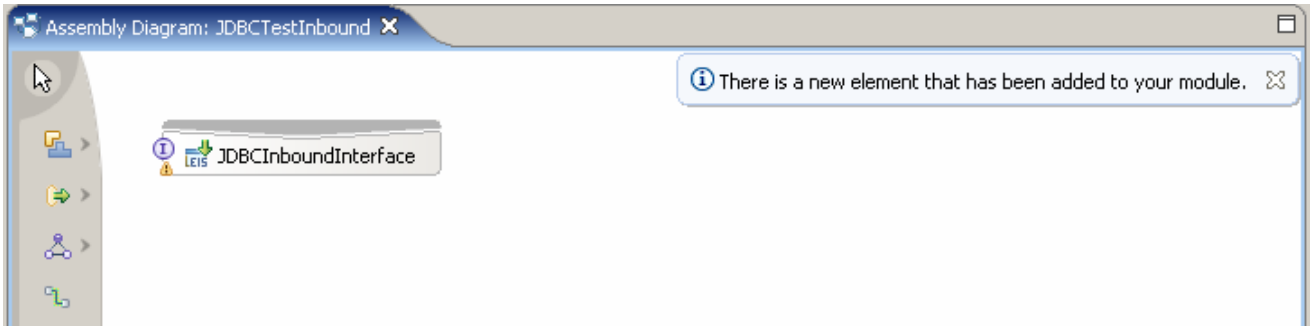



__ g. Click Finish.

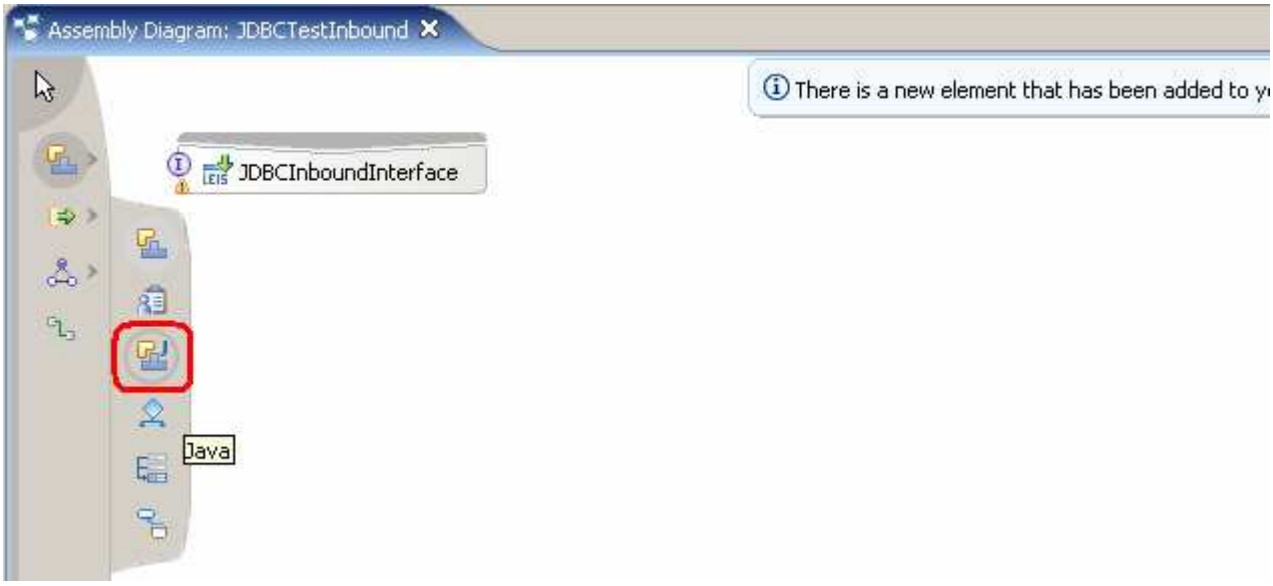
____ 5. Use the Assembly Diagram to wire the **JDBCTestInbound** Interface to a Java Component.

__ a. Change to the **Business Integration** perspective. Expand the **JDBCTestInBound** folder.

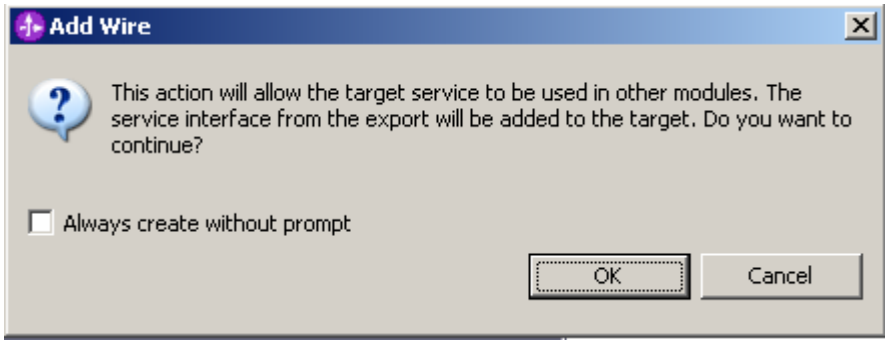
__ b. From the **Business Integration** view, double click the **JDBCTestInbound** module. This will open the module in the Assembly Diagram. You will see a message that there is one new element added to the module.



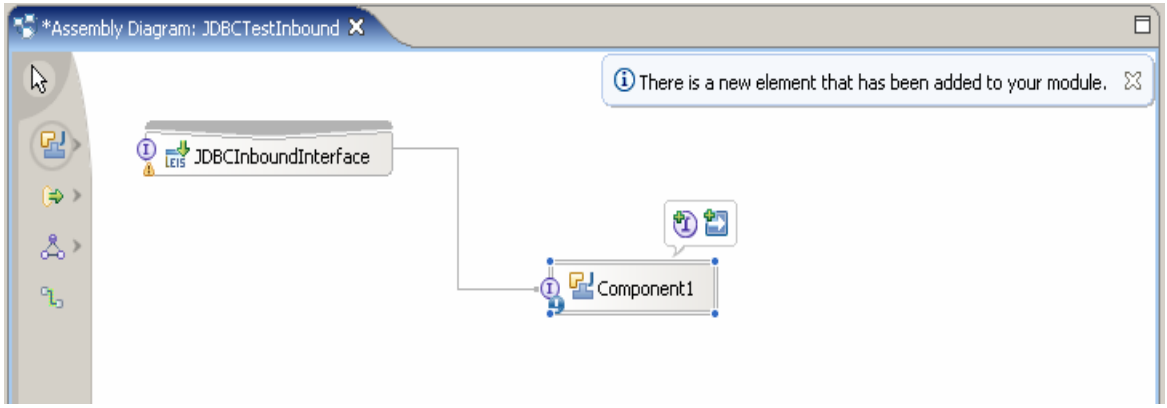

__ c. From the palette, select the **Component (with no implementation type)** icon, then select the **Java Component** and drop it on the Assembly Diagram.




__ d. Wire the **JDBCInboundInterface** to the **Java Component.** At the Add Wire popup window, select **OK** in response:

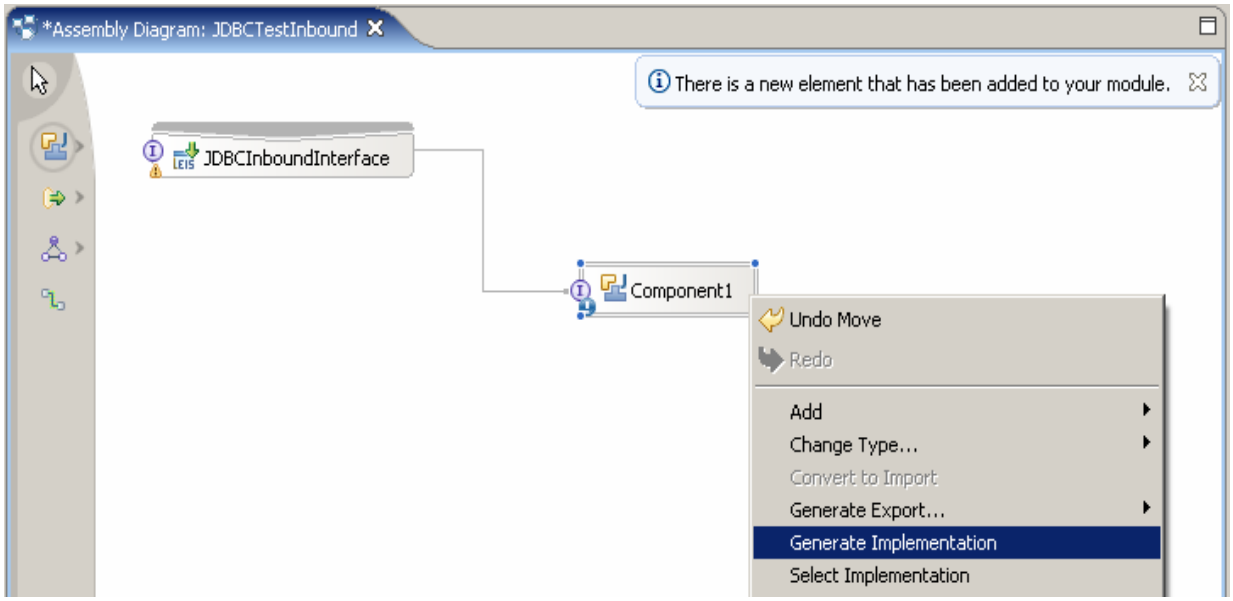



__ e. The Assembly Diagram now looks as follows:

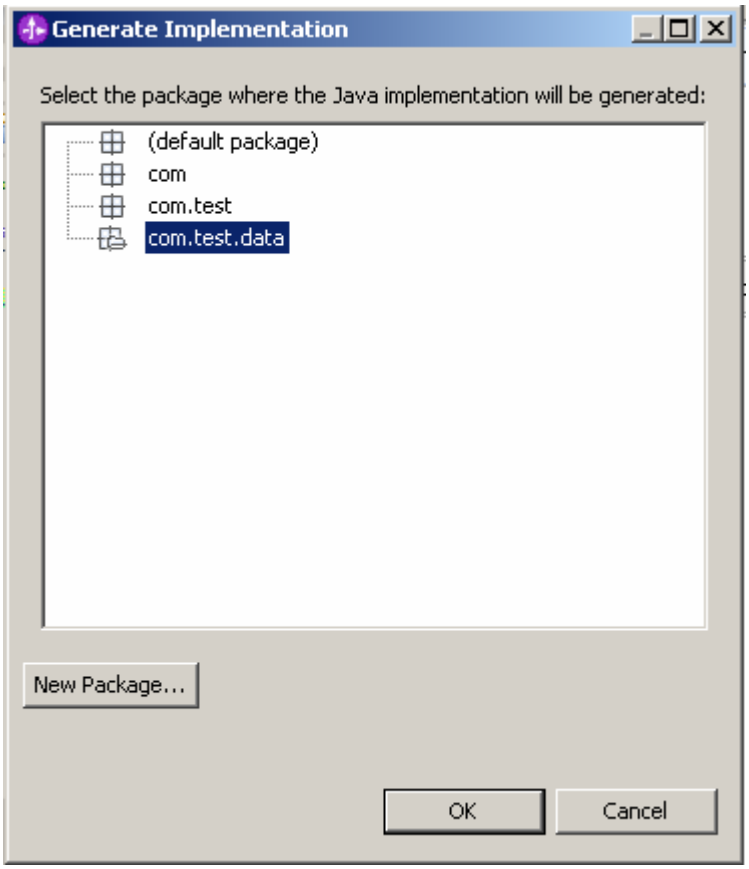



____ 6. Generate the implementation for the Java Component

__ a. Right click the **Java Component1**, select **Generate Implementation** from the context menu




__ b. Highlight the **com.test.data** package and select **OK** (Click New Package and create the com.test.data package if it doesn't already exist.)




__ c. The Java Editor will open with the **Component1Impl.java** file.

**NOTE:** For your convenience, the following code snippets can be found in **<LAB_FILES>\JDBC\snippets\Component1Impl.txt**

__ d. Scroll down and locate the **createAppCustomer(Object createCustomerInput)** method that needs to be implemented. Paste the following code into the method so the complete method looks as follows:

```
public void createAppCustomer(Object createCustomerInput) {

System.out.println("CREATE customer is:  " + createAppCustomerInput);
DataObject bg=(DataObject)createAppCustomerInput;
DataObject bo=bg.getDataObject("AppCustomer");
System.out.println("bo is:  "+bo);
System.out.println("verb is:  "+bg.get("verb"));
System.out.println("id is:  "+bo.get("id"));
System.out.println("LNAME is:  "+bo.get("lname"));
System.out.println("FNAME is:  "+bo.get("fname"));
}
```

__ e. Scroll down and locate the **updateAppCustomer(Object updateAppCustomerInput)** method that needs to be implemented. Paste the following code into the method so the complete method looks as follows:

```
public void updateAppCustomer(Object updateAppCustomerInput) {

System.out.println("UPDATE customer is:  " + updateAppCustomerInput);
DataObject bg=(DataObject)updateAppCustomerInput;
DataObject bo=bg.getDataObject("AppCustomer");
System.out.println("verb is:  "+bg.get("verb"));
System.out.println("bo is:  "+bo);
System.out.println("id is:  "+bo.get("id"));
System.out.println("LNAME is:  "+bo.get("lname"));
System.out.println("FNAME is:  "+bo.get("fname"));
}
```

__ f. Scroll down and locate the **deleteAppCustomer(Object deleteAppCustomerInput)** method that needs to be implemented. Paste the following code into the method so the complete method looks as follows:

```
public void deleteAppCustomer(Object deleteAppCustomerInput)  {

System.out.println("DELETE customer is:  " + deleteAppCustomerInput);
DataObject bg=(DataObject) deleteAppCustomerInput;
DataObject bo=bg.getDataObject("AppCustomer");
System.out.println("verb is:  "+bg.get("verb"));
System.out.println("bo is:  "+bo);
System.out.println("id is:  "+bo.get("id"));
System.out.println("LNAME is:  "+bo.get("lname"));
System.out.println("FNAME is:  "+bo.get("fname"));
}
```

__ g. Save your work by selecting **File -> Save** from the top menu, or using the shortcut key sequence **Ctrl + S.** Close the file.

__ h. Close and save the Assembly Diagram. Wait for the workspace to complete building.

____ 7. Before starting the WebSphere Test Environment Server, you first need to “Switch Workspaces” which will release the Enterprise Discovery process connection to the Cloudscape database. This is necessary, so the server JVM can get a connection to the database when running the test in the next Part.

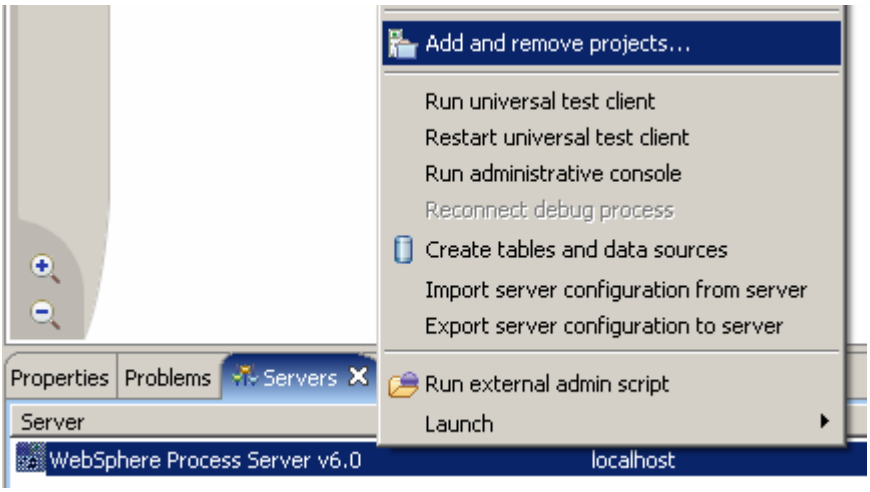
__ a. From the top menu bar, select **File > Switch Workspace** and select the same workspace from which you’ve been working.

# Part 4: Test the application using the WebSphere test environment

In this part you will use the WebSphere Test Environment to test the SCA application event processing by starting the application and verifying that it polls for and picks up the Create event already waiting in the WBIA_JDBC_EVENTSTORE Table.

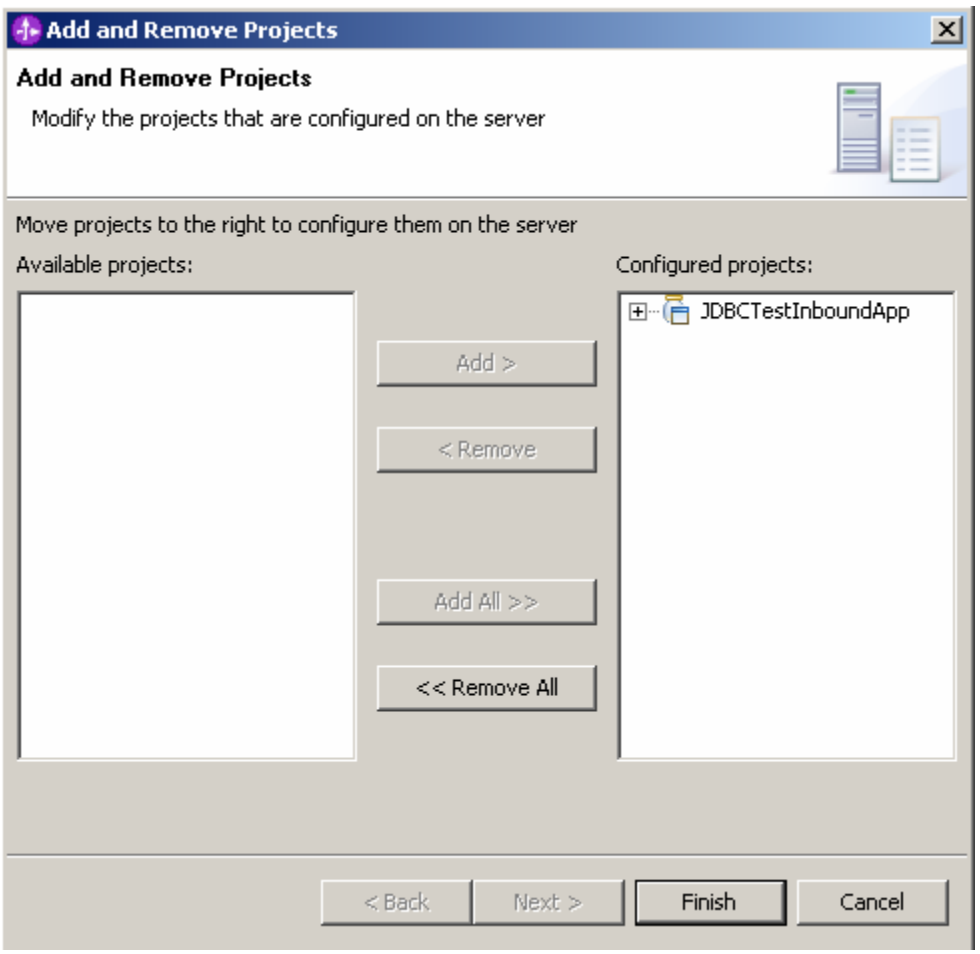
____ 1. Add the project to the server for the WebSphere Test Environment.

__ a. Right click on the server in the server view and select **Add and remove projects** …




__ b. In the Add and Remove Projects dialog, select the **JDBCTestInboundApp** project from the Available projects panel.

__ c. Click **Add >** to add it to the Configured projects panel. Click **Finish**




__ d. Wait for the project to be added. In the Console view, you will see messages that the application has successfully started and has received and processed the inbound Create event. (These messages are also reflected in the c:\<WPS_HOME>\prof\logs\server1\Systemout.log).

```
[10/12/06 22:48:13:000 CDT] 00000057 SystemOut     O verb is:  Create
[10/12/06 22:48:13:000 CDT] 00000057 SystemOut     O id is:  10
[10/12/06 22:48:13:000 CDT] 00000057 SystemOut     O LNAME is:  Johnson
[10/12/06 22:48:13:000 CDT] 00000057 SystemOut     O FNAME is:  Jerry
```

__ e. If you would like to try other events, such as a delete, you will need to first stop the server (right click on the server, and select **stop**) to release it's connection to the JDBCTEST database. Then, start the Cview.bat GUI to add, delete or update records to the database. Once completed, exit the Cview GUI, and then restart the server for the events to be processed.

__ f. Repeat steps a through c to remove the project from the server.

## What you did in this exercise

In this exercise, you learned how to install and deploy the WebSphere Adapter for JDBC. You also were introduced to inbound event processing.