JDBC inbound lab – Wrapper business object

What this exercise is about

The objective of this lab is to introduce you wrapper business object for inbound processing.

Lab requirements

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

- WebSphere Integration Developer V7.0 installed
- WebSphere Process Server V7.0 test environment installed
- WebSphere Adapter for JDBC V7.0 installed
- Sample code in the directory C:\Labfiles70\JDBC (Windows) or /tmp/LabFiles70/JDBC (Linux)

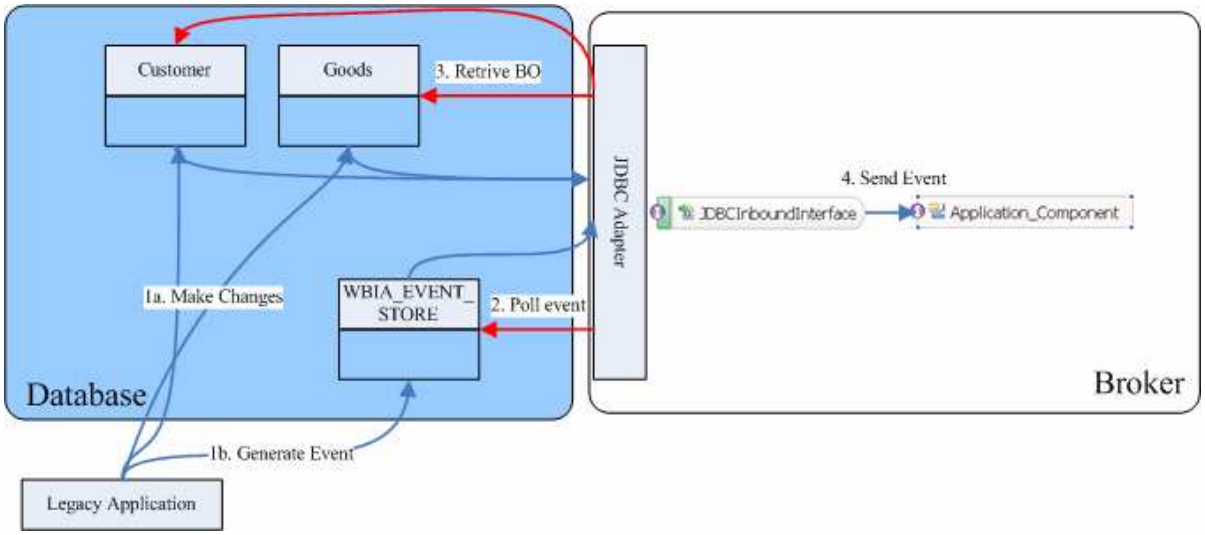
What you should be able to do

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

- Understand how WebSphere Adapter for JDBC 7.0.0.0 retrieves customer information from an application's database. A wrapper business object is used to retrieve records from multiple tables with one event entry.

Introduction

This lab illustrates the ability of WebSphere JDBC adapter to interact with database by polling database event from an event table. In this scenario, a legacy application makes some change of the CUSTOMER table and the GOODS table in a single operation. Then, insert an event entry record into the event table (WBIA_EVENT_TABLE). Then, the event is polled by JDBC adapter and sends it to one SCA component. JDBC adapter screen all the database operation details, event quality assuring details and provide a simple event interface for the application component. The following figure shows the whole scenario:



This case has three steps:

1. The legacy application will make the changes and then generate an event record. For simplify reason, you will insert records using SQL statement directly.

2. JDBC adapter will poll the event from database periodically. Thus, it will find the new events and fetch the event and corresponding business objects from database.

3. At last, JDBC adapter will convert the event to a SDO and send it to the destination SCA component.

Exercise instructions

Some instructions in this lab are 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 or .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>	JDBCInBoundWrapperBO	
<WID_HOME>	C:\IBM\WID7	
<WPS_HOME>	C:\IBM\WID7_WTE	
<JDBCADAPTER_HOME>	<WID_HOME>\Resource Adapters\JDBC_7.0.0.0\	
<LAB_FILES>	C:\Labfiles70	/tmp/Labfiles70
<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, replace C:\LabFiles70\ with C:/LabFiles70/

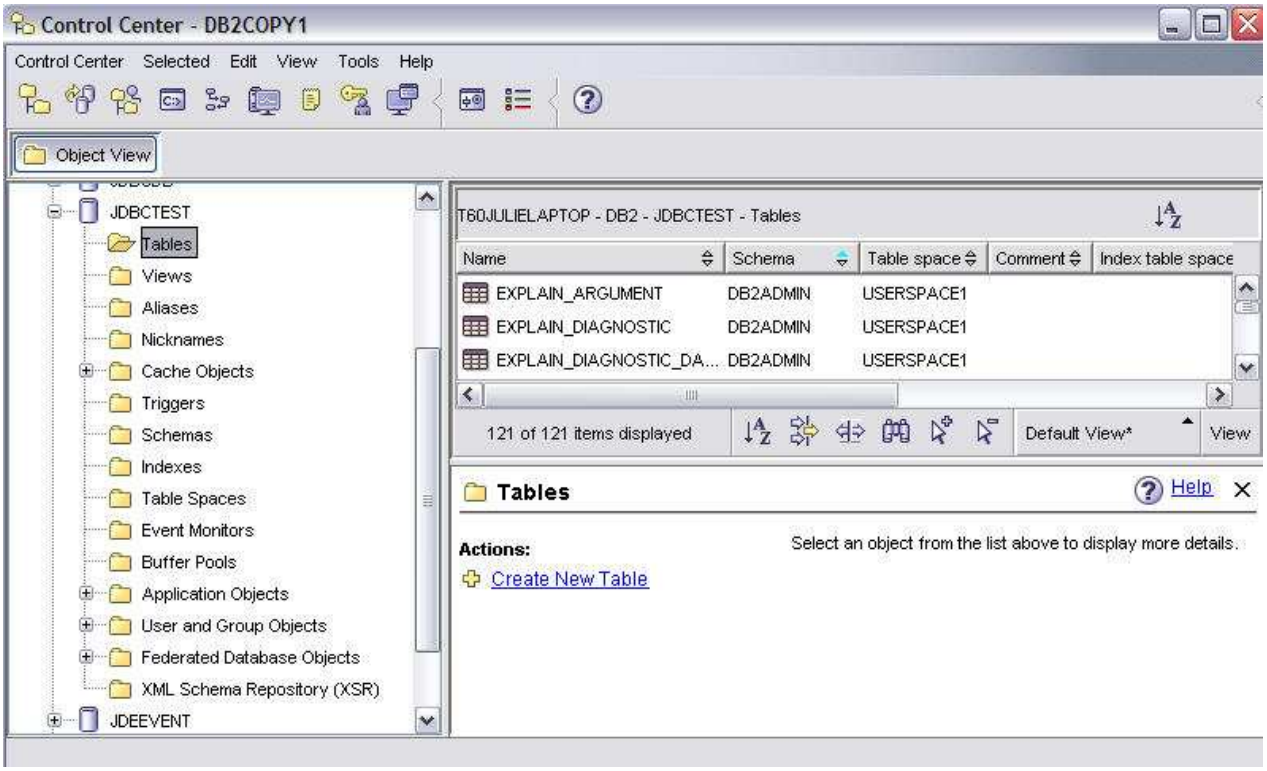
Part 1: Create the JDBCTEST database and tables using DB2

In this part you will create samples tables in DB2 in preparation for this tutorial. You will create table CUSTOMER and table ADDRESS along with event store table.

____ 1. Open **IBM DB2 → Control Center**

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

____ 2. Create a new database **JDBCTEST** if it does not already exist.



____ 3. Click  to open command editor

__ a. Add **JDBCTEST** as target database

____ 4. Paste these following scripts to create two tables and one event table for recording events

---

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

__ a. Create CUSTOMER table

```
CREATE TABLE CUSTOMER (
PKEY VARCHAR(10) NOT NULL PRIMARY KEY,
FNAME VARCHAR(20) ,
LNAME VARCHAR(20) ,
CCODE VARCHAR(10) ) ;
```

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

__ b. Create CUSTADD table

```
CREATE TABLE ADDRESS (
ADDRID VARCHAR(10) NOT NULL PRIMARY KEY,
CUSTID VARCHAR(10) ,
CITY VARCHAR(20) ,
ZIPCODE VARCHAR(10) );
```

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

__ c. Create the WBIA_JDBC_EVENTSTORE table

```
CREATE TABLE WBIA_JDBC_EVENTSTORE (
EVENT_ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1) PRIMARY KEY,
XID VARCHAR(200),
OBJECT_KEY VARCHAR(80) NOT NULL,
OBJECT_NAME VARCHAR(40) NOT NULL,
OBJECT_FUNCTION VARCHAR(40) NOT NULL,
EVENT_PRIORITY INTEGER NOT NULL,
EVENT_TIME TIMESTAMP,
EVENT_STATUS INTEGER NOT NULL,
EVENT_COMMENT VARCHAR(100) );
```

____ 5. Click  to run the SQL scripts. Make sure all SQL commands completed successfully

```
------------------------------ Commands Entered ------------------------------
CREATE TABLE WBIA_JDBC_EVENTSTORE (
EVENT_ID INTEGER NOT NULL PRIMARY KEY,
XID VARCHAR(200),
OBJECT_KEY VARCHAR(80) NOT NULL,
OBJECT_NAME VARCHAR(40) NOT NULL,
OBJECT_FUNCTION VARCHAR(40) NOT NULL,
EVENT_PRIORITY INTEGER NOT NULL,
EVENT_TIME TIMESTAMP,
EVENT_STATUS INTEGER NOT NULL,
EVENT_COMMENT VARCHAR(100) );
------------------------------------------------------------------------------
CREATE TABLE WBIA_JDBC_EVENTSTORE ( EVENT_ID INTEGER NOT NULL PRIMARY KEY, XID VARCHAR(200), OBJECT_KEY VARCHAR(80) NOT NU
DB20000I  The SQL command completed successfully.
```

____ 6. Insert records into two tables you have just created. This will add two entries to the CUSTOMER and ADDRESS tables.

```
INSERT INTO CUSTOMER (PKEY, FNAME, LNAME, CCODE)
      VALUES ('C1', 'JONE', 'SMITH', '1');
INSERT INTO CUSTOMER (PKEY, FNAME, LNAME, CCODE)
      VALUES ('C2', 'ROTH', 'GREEN', '1');
INSERT INTO ADDRESS (ADDRID, CUSTID, CITY, ZIPCODE)
      VALUES ('A1', 'C1', 'AUSTIN', '100000');
INSERT INTO ADDRESS (ADDRID, CUSTID, CITY, ZIPCODE)
      VALUES ('A2', 'C2', 'HOUSTON', '200000');
```

____ 7. Exit DB2 command editor and control center.

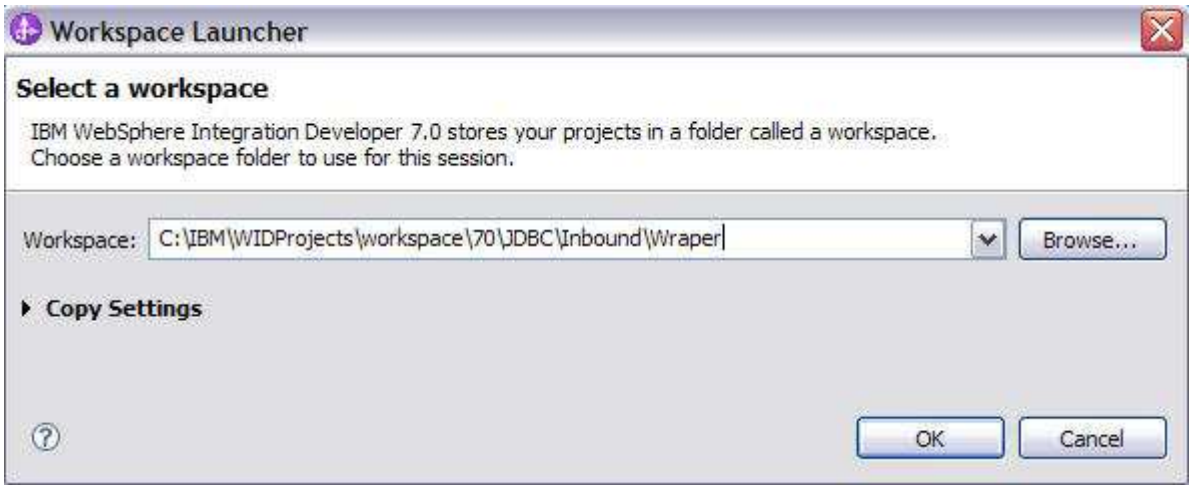
Part 2: Set up the development environment

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 (WTE).

____ 1. Start WebSphere Integration Developer V7.0 with a new workspace.

__ a. From the start menu select **Start > Programs → IBM WebSphere Integration Developer → IBM WebSphere Integration Developer 7.0 → IBM WebSphere Integration Developer V7.0**

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



__ c. When WebSphere Integration Developer V7.0 opens, close the **Getting Started** page

Part 3: Use External Service wizard to generate business objects and other artifacts

In this part you will run External Service to discover objects and create the necessary SCA artifacts, and assemble into an SCA application.

____ 1. In the Business Integration Perspective, run the External Service wizard. A Business Integration project is created for you during this process.

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

____ 2. From within the Business Integration Perspective, select **File > New > External Service**. This opens an External Service wizard that helps you obtain a service which establishes connectivity with other systems. The wizard provides four connectivity options – Adapters, Java, Registers, and Messaging

__ a. Expand **Adapters,** select **JDBC,** and click **Next**

____ 3. Highlight **IBM WebSphere Adapter for JDBC (IBM: 7.0.0.0)** and click **Next**

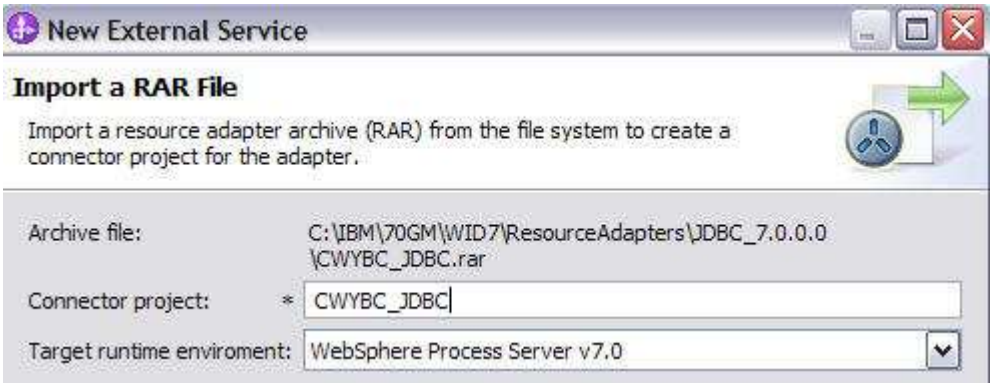
____ 4. Adapter Import screen:

In this step, you will import a connector resource adapter archive from the file system into your WebSphere Integration Developer workspace. The adapter RAR file already exists under **<JDBCADAPTER_HOME**>.

__ a. The default Connector file is selected which is shipped along with WebSphere Integration Developer

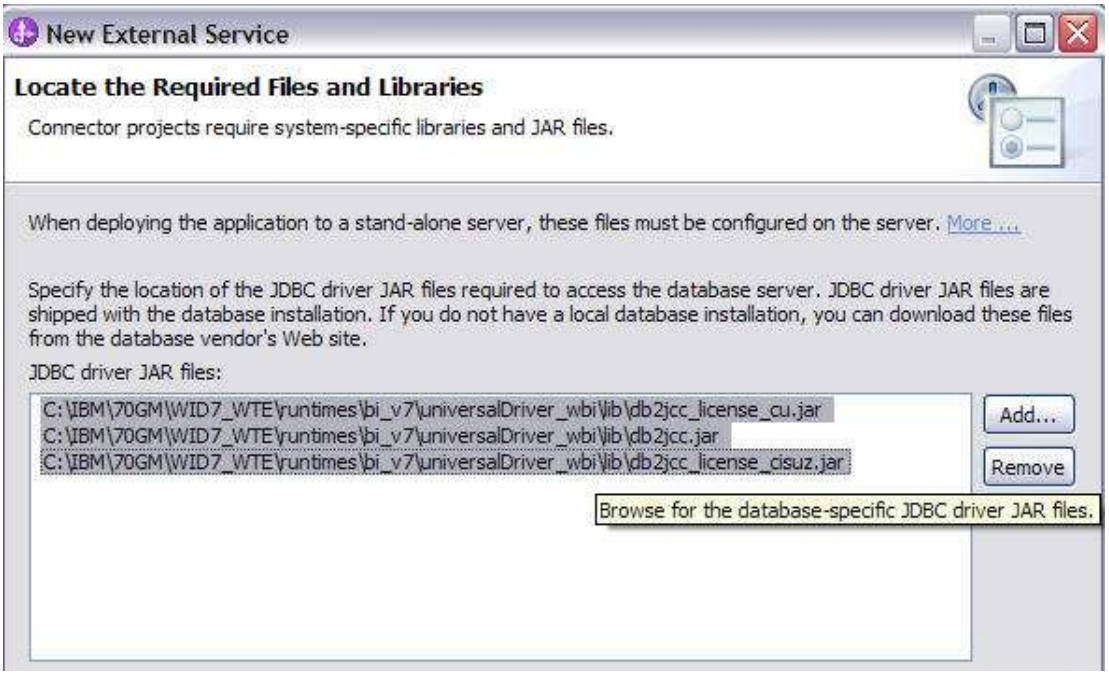
__ b. Accept the default name for Connector project, **CWYBC_JDBC**. You can change it to any other name, but for this lab, you can leave the default name.

__ c. For Target server, ensure that **WebSphere Process Server V7.0** is selected



____ 5. Add any external dependencies your adapter has to the imported project. These are dependencies that the adapter can have on the JDBC applications (adapter–specific).

__ a. Click **Add** and browse to the location of c:\<WPS_HOME>\universalDriver_wbi\lib and select the **db2jcc.jar, db2jcc_license_cu.jar,** and **db2jcc_license_cisuz.jar**



__ b. Click **Next**

____ 6. Select the type of processing the adapter will perform at runtime

__ a. Select **Inbound** and click **Next**

____ 7. Complete the Connection Configuration for Discovery Agent Configuration panel to connect to the JDBCTEST database and discover the available services. To connect to the database, these information are necessary: username, password, database URL and JDBC driver class. Check the driver manual for the appropriate values for driver URL and driver class.

__ a. From the left panel, expand **DB2 UDB** and select version of your db2

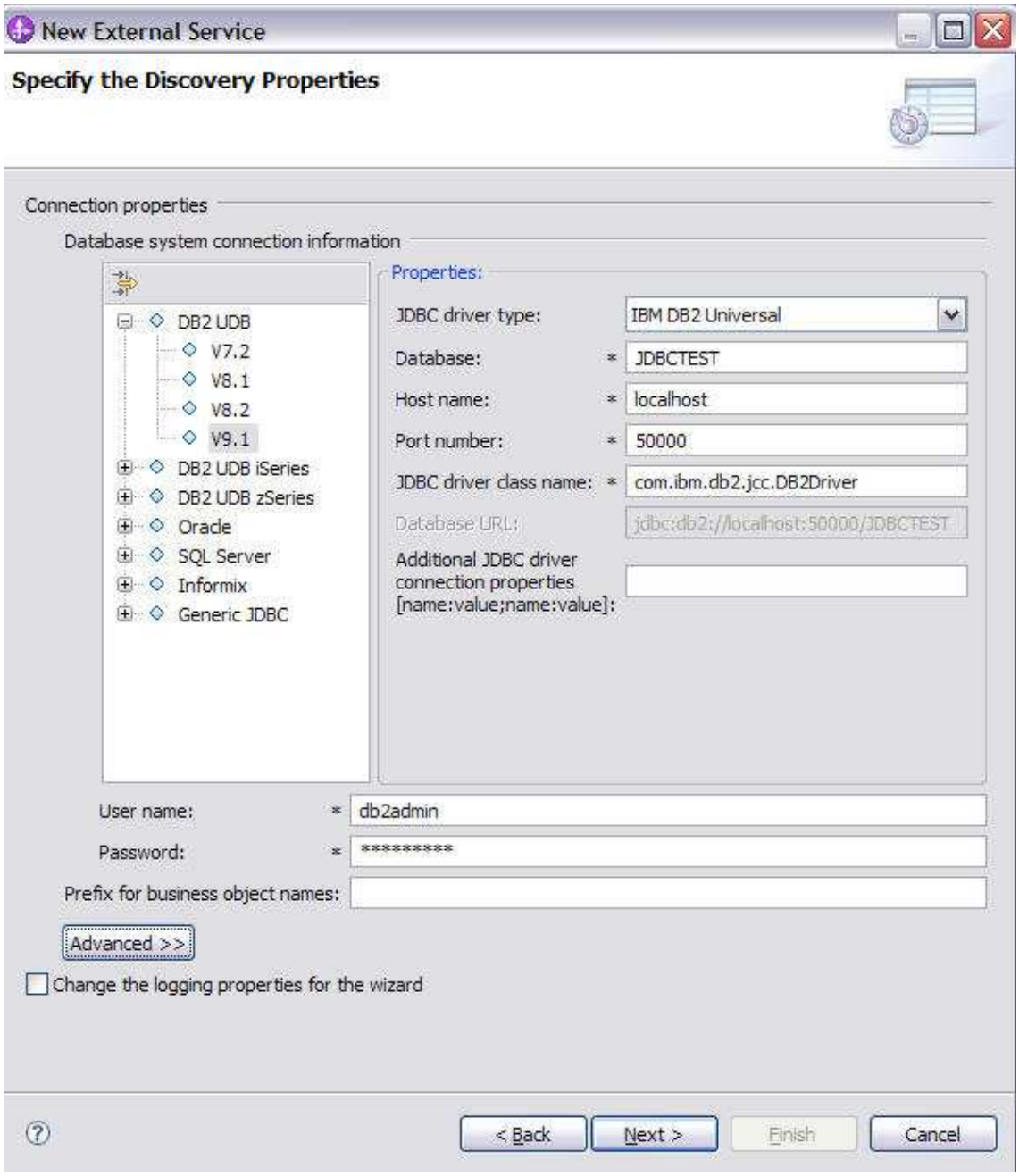
__ b. From the right panel, enter/select these following values

1) Database - **JDBCTEST**

2) Hostname - **LOCALHOST**

3) Port number - **50000**

__ c. Enter valid user ID and password values to access JDBCTEST database



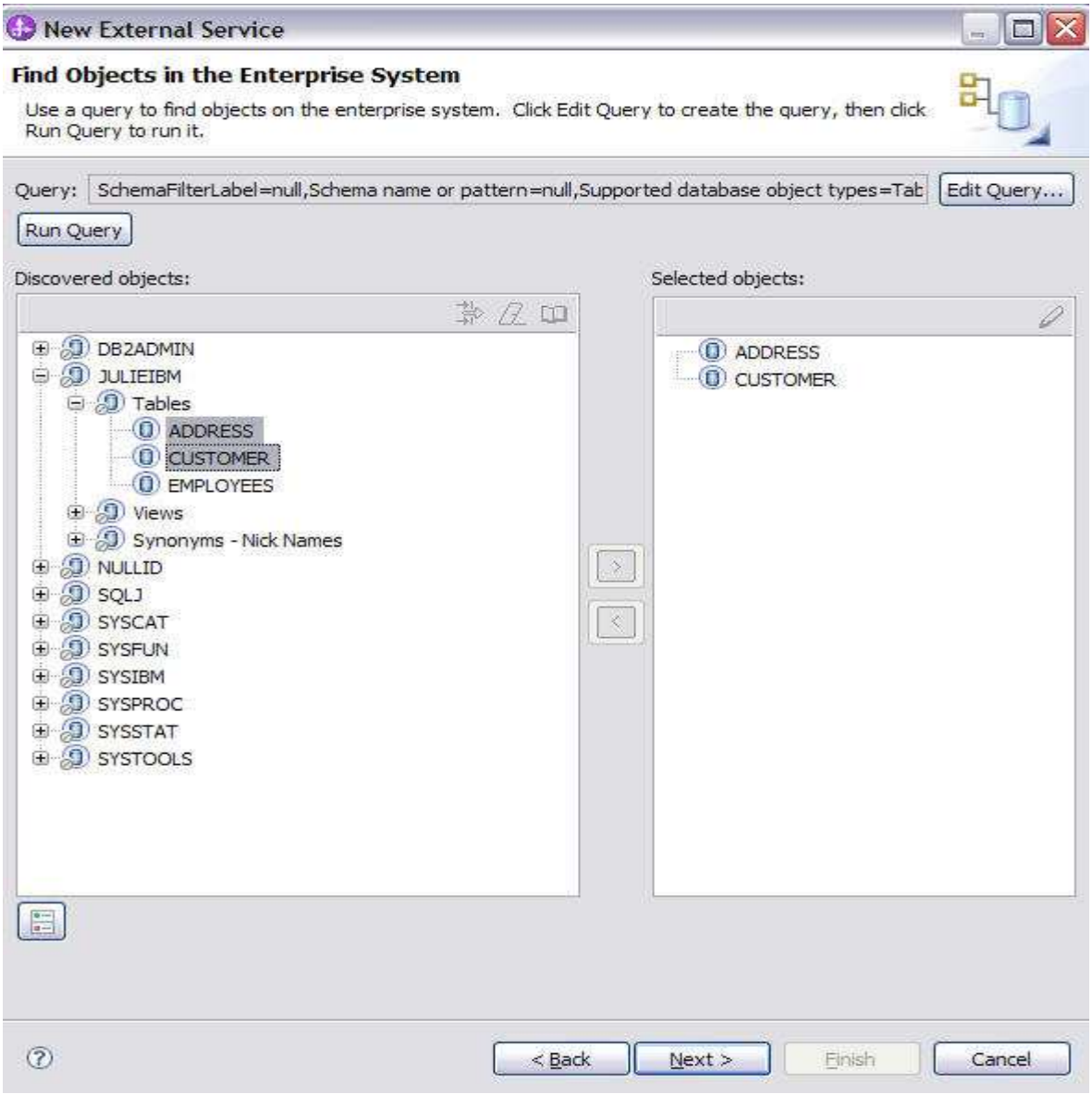
__c. Click **NEXT**

____ 8. Complete the **Discover Objects and Services** panel

__ a. Select the **Run Query** button. A connection is made to the DB2 JDBCTEST database and a selection of Meta data objects is presented in a tree-like structure.

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

__ b. Expand the JDBCTEST database, then select **Tables**, and highlight **CUSTOMER** and **ADDRESS**, click button. **CUSTOMER** and **ADDRESS** now open in the **Objects to be imported** window. Click **Next**



____ 9. Complete the **Configure Objects** panel

__ a. For wrapper object names, click **Add** and enter **Wrapper** for the new wrapper business object.

__ b. In the **Table, view, synonym, or nickname child objects for the selected wrapper** area, click **Add** to add **CUSTOMER** and **ADDRESS** table business objects for the wrapper.

__ c. In the **Service functions for selected wrapper object** area, click **Add** to add functions for the wrapper.

__ d. Leave the default value for Namespace.

__ e. Note that Business Graph is optional - It determines if you want to generate business graph or not. If the property is checked only then business graphs is generated. And by default the property is checked.

__ f. Click **Next**

____ 10. Complete Publishing Object Configuration Properties

__ a. Specify the security crendentials

1) Select **Other** to define the data source JNDI name

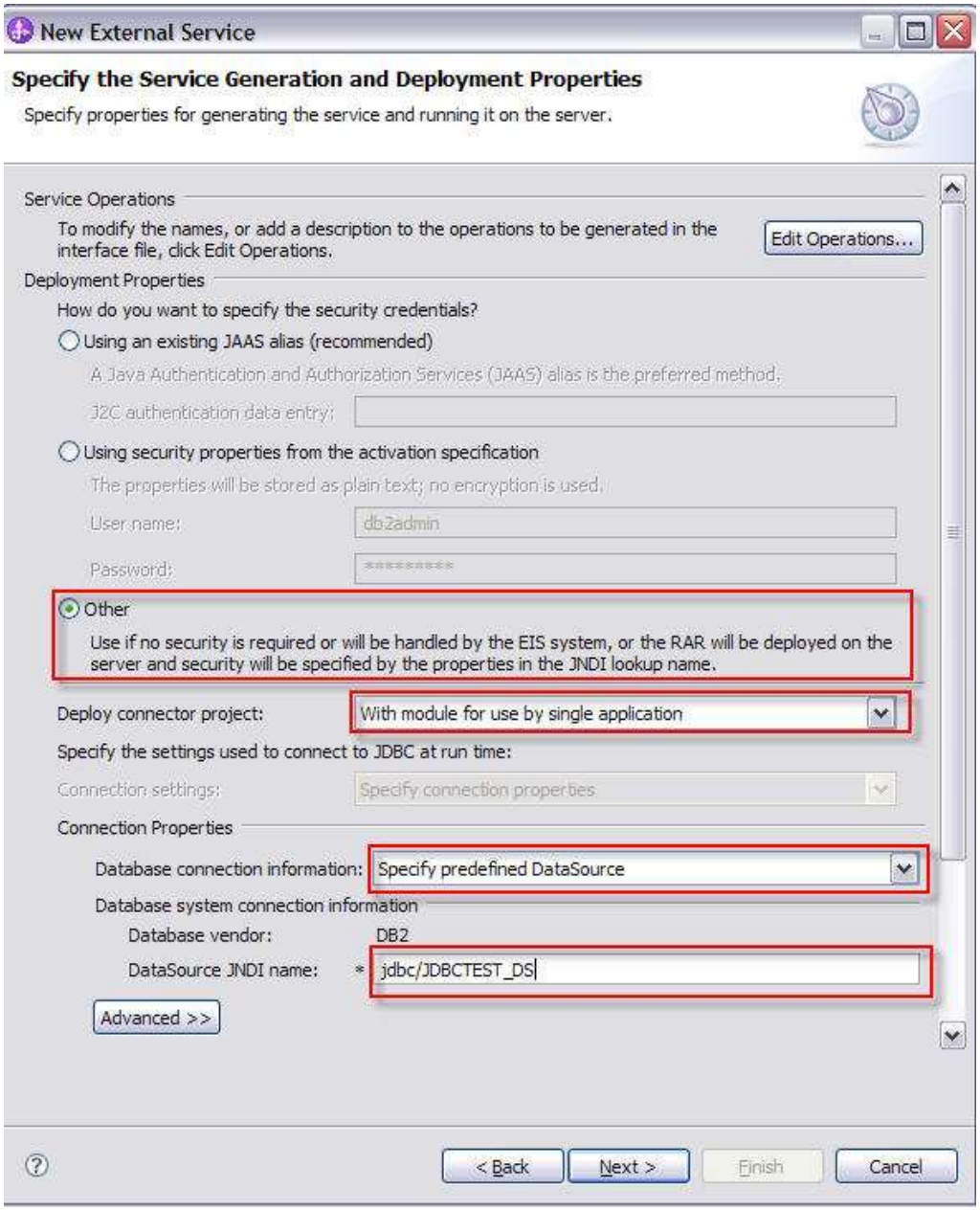
__ b. Leave the default **"With module for use by single application"** option in this exercise.

__ c. Under **Connection properties**

1) Leave the default "**Specify predefined Datasource"**

2) Database Vendor is pre-populated with value **DB2** (If you were using Oracle, or MSSQLServer, you see those as specific adapter processing is available with those specific databases.)

3) Enter **jdbc/JDBCTEST_DS** for Data source JNDI name



__ d. Expand by clicking on **Advanced** button for additional properties options

__ e. Expand **Event delivery configuration**

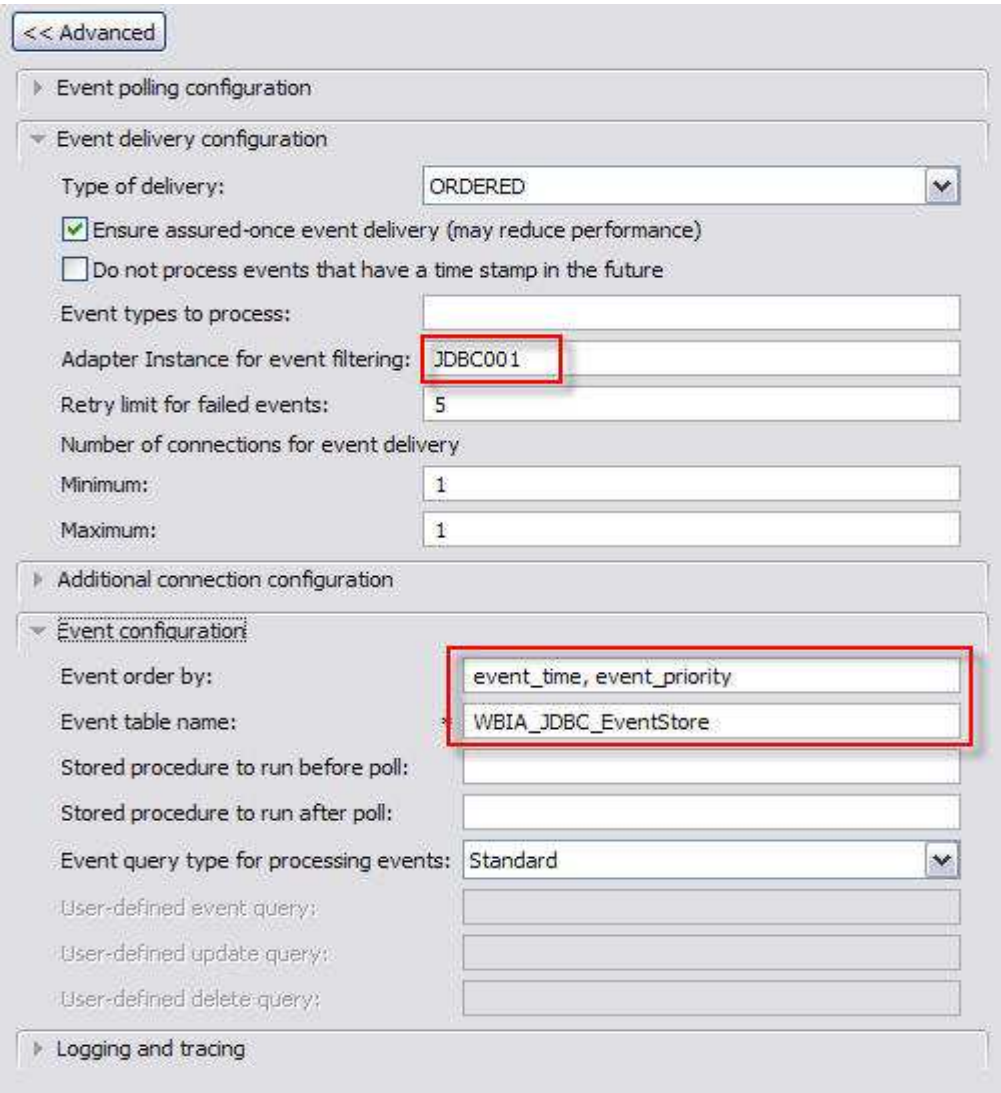
1) Verify the **Ensure once-only event delivery** is checked

2) Enter **JDBC001** for event filtering

__ f. Expand Event Configuration

1) Event Order By: **event_time, event_priority**

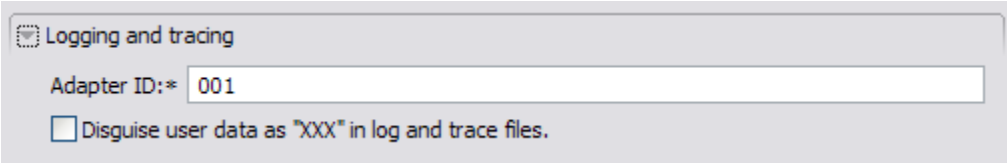
2) Event Table Name: **WBIA_JDBC_EventStore**



__ g. Expand **Logging and Tracing**

1) **Optional** - Enter any numeric value for **Adapter ID**. This property identifies the adapter instance in log and trace files and for PMI events. The adapter ID is used with an adapter-specific identifier, MyAdapterRA, to form the component name used by Log and Trace Analyzer.

2) **Optional** - If you set the property to disguise user data as “XXX’, the adapter will replace user data with a string of X's when writing to log and trace files. For inbound processing, this property is retrieved from the resource adapter properties. For outbound processing, it is retrieved from the managed connection factory properties.

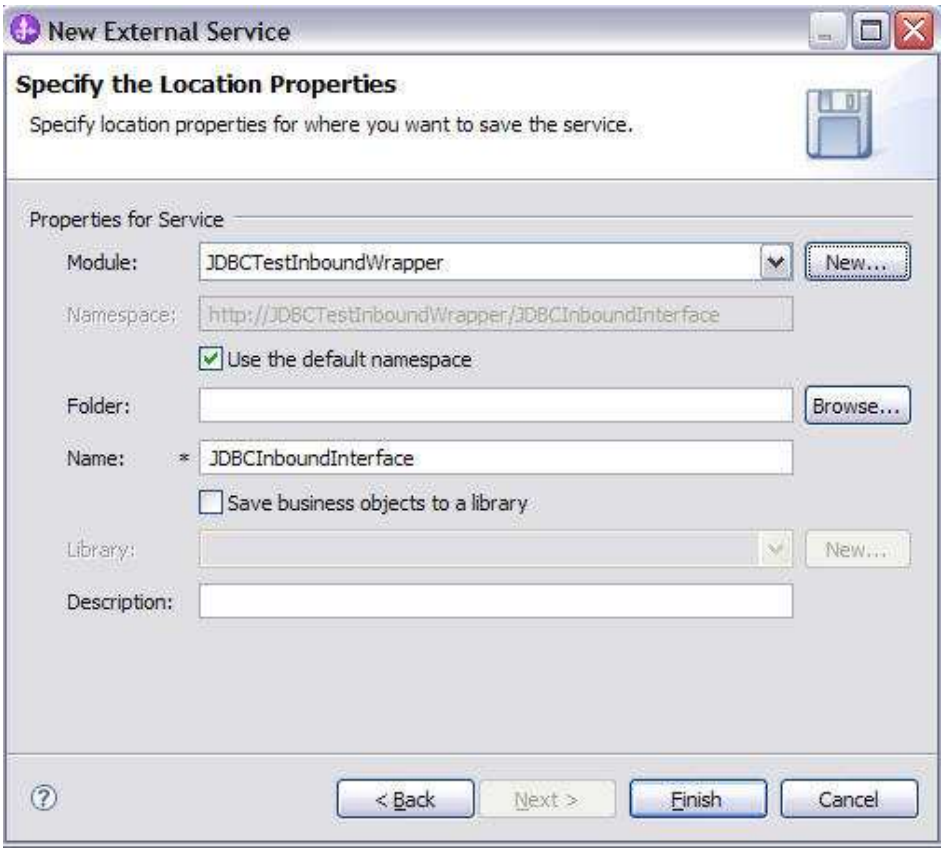


__ h. Click **Next**

____ 11. Complete **Publishing Properties** panel

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

__ **b.** Leave the default **JDBCInboundInterface** for the Name value.



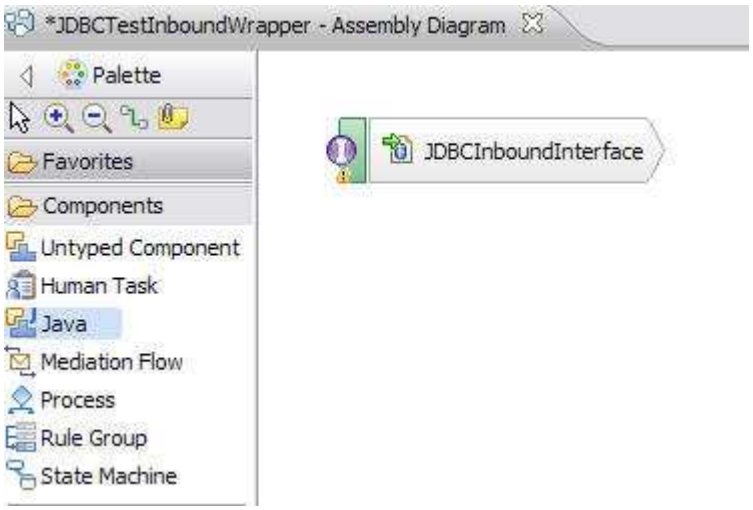
__ c. Click **Finish**

____ 12. Use the Assembly Diagram to wire the **JDBCTestInboundWrapper** Interface to a Java Component.

__ a. Expand the **JDBCTestInBoundWrapper** folder.

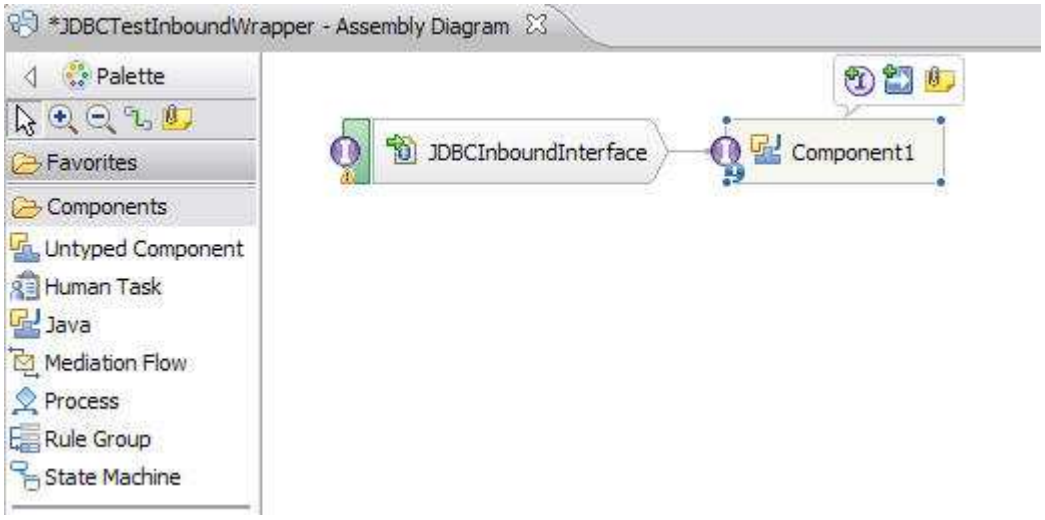
__ b. From the **Business Integration** view, double click the **JDBCTestInBoundWrapper** module. This will open the module in the Assembly Diagram.

__ c. From the palette, expand **Component**, then select the **Java Component** and drop it on the Assembly Diagram.



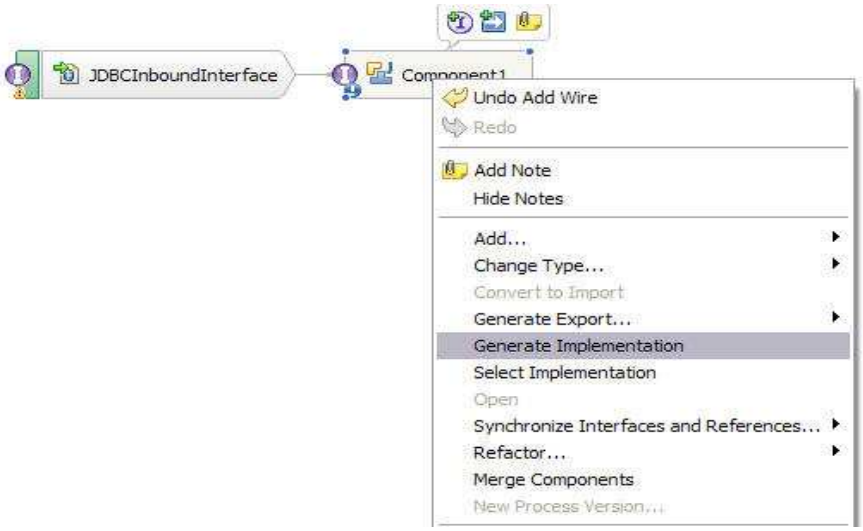
__ d. Wire the **JDBCInboundInterface** to the **Java Component.** At the Add Wire popup window, select **OK**.

__ e. The Assembly Diagram now looks as follows:



____ 13. Generate the implementation for the Java Component

__ a. Right click the **Java Component1**, select **Generate Implementation** from the pop-up menu



__ b. Highlight the **default** package and select **OK** (Click New Package and create new package if you want)

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

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

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

```
public void createWrapperBG(DataObject createWrapperBGInput) {
   DataObject wrapper = createWrapperBGInput.getDataObject("Wrapper");
   System.out.println("-----------------------------");
   System.out.println("Wrapper was created.");
   DataObject customer = (DataObject) wrapper.getList("customerobj").get(0);
   DataObject addr = (DataObject) wrapper.getList("addressobj").get(0);
   System.out.println("CUSTOMER info as below:");
   System.out.println("PKEY is: "+ customer.getString("pkey"));
   System.out.println("FNAME is: "+ customer.getString("fname"));
   System.out.println("LNAME is: "+ customer.getString("lname"));
   System.out.println("CCODE is: "+ customer.getString("ccode"));
   System.out.println();
   System.out.println("CUSTOMER ADDRESS info as below:");
   System.out.println("ADDRID is: "+ addr.getString("addrid"));
   System.out.println("CUSTID is: "+ addr.getString("custid"));
   System.out.println("CITY is: "+ addr.getString("city"));
   System.out.println("ZIPCODE is: "+ addr.getString("zipcode"));
   System.out.println("-----------------------------");
   System.out.println();
   }
```

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

```
public void deleteWrapperBG(DataObject deleteWrapperBGInput) {
   DataObject wrapper = deleteWrapperBGInput.getDataObject("Wrapper");
   System.out.println("-----------------------------");
   System.out.println("Wrapper was deleted.");
   System.out.println("PKEY is: "+  wrapper.getString("wrapcustomerpkey"));
   System.out.println("ADDRESS is: "+ wrapper.getString("wrapaddressaddrid"));
   System.out.println("-----------------------------");
   System.out.println();
   }
```

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

```
public void updateWrapperBG(DataObject updateWrapperBGInput) {
   DataObject wrapper = updateWrapperBGInput.getDataObject("Wrapper");
   System.out.println("-----------------------------");
   System.out.println("Wrapper was updated.");
   DataObject customer = (DataObject) wrapper.getList("customerobj").get(0);
   DataObject addr = (DataObject) wrapper.getList("addressobj").get(0);
   System.out.println("CUSTOMER info as below:");
   System.out.println("PKEY is: "+ customer.getString("pkey"));
   System.out.println("FNAME is: "+ customer.getString("fname"));
   System.out.println("LNAME is: "+ customer.getString("lname"));
   System.out.println("CCODE is: "+ customer.getString("ccode"));
   System.out.println();
   System.out.println("CUSTOMER ADDRESS info as below:");
   System.out.println("ADDRID is: "+ addr.getString("addrid"));
   System.out.println("CUSTID is: "+ addr.getString("custid"));
   System.out.println("CITY is: "+ addr.getString("city"));
   System.out.println("ZIPCODE is: "+ addr.getString("zipcode"));
   System.out.println("-----------------------------");
   System.out.println();
}
```

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

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

Part 4: Create the authentication alias and configure the data source

Note: If you have already completed this section in any of JDBC Outbound labs, you can skip this and proceed to Part 5.

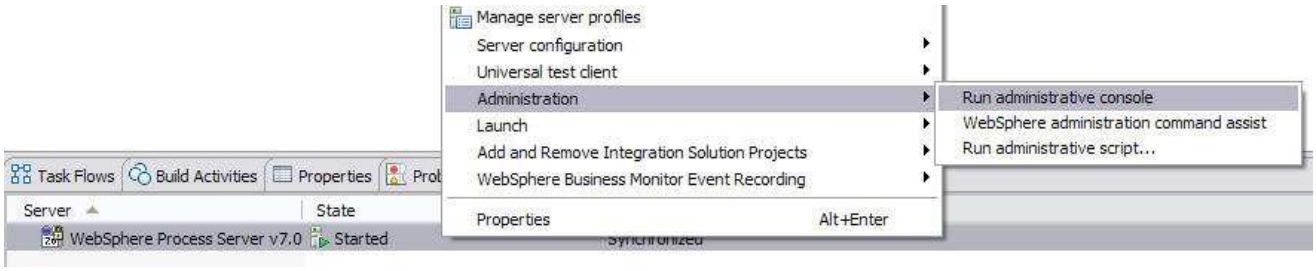
In this part, the authentication alias needs to be set since the data source uses the username/password set in the authentication alias to connect to the database. This data source connects to the database and is used later when generating the artifacts for the module. Here are the steps to set the authentication alias in WebSphere Process Server administrative console.

____ 1. Switch to Servers view by selecting Windows → Show View → Servers.

____ 2. Set the authentication alias from the administrative console

__ a. In the **Servers** tab in the lower-right corner pane, right click the **Server** and then select **Start**

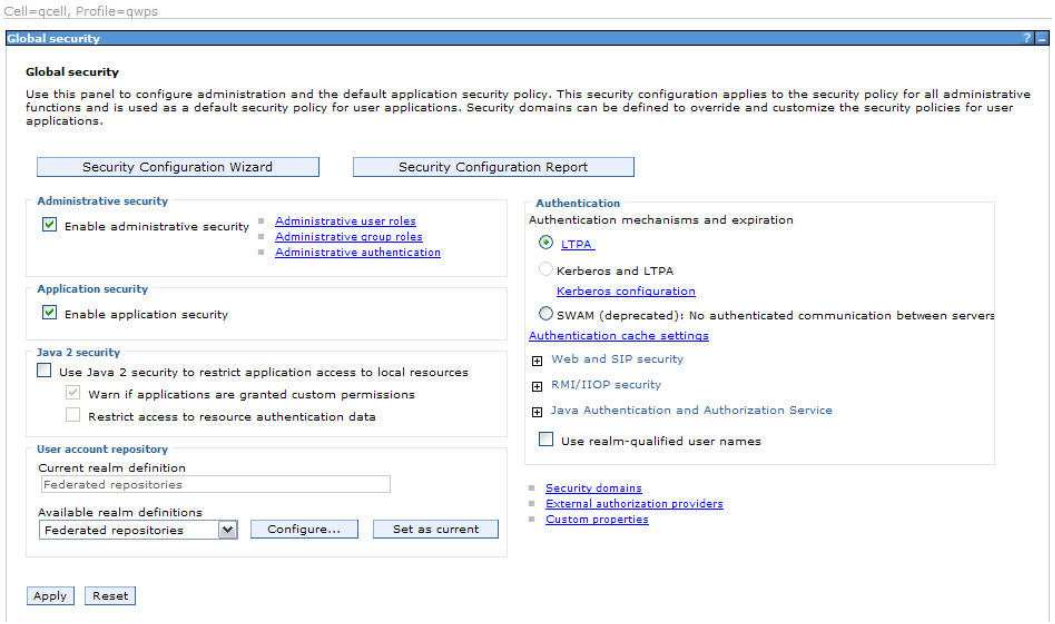
__ b. When the server status is **Started**, right click the **Servers**, and then select **Administration →Run Administrative console**



__ c. Log in to the administrative console by clicking the "Log in" button

__ d. Click **Security → Global Security**

__ e. On the right, expand **Java Authentication and Authorization Service** under the Authentication heading.



__ f. Click **J2C authentication alias**. It gives the list of existing aliases.

a) Click **New** and type in alias name, user ID, and password to connect to the database. Click **Ok**

__ g. Click **Ok** and save the changes

____ 3. Create a JDBC Provider to which is used to create data source

__ a. From administrative console, click **Resources → JDBC → JDBC Providers**

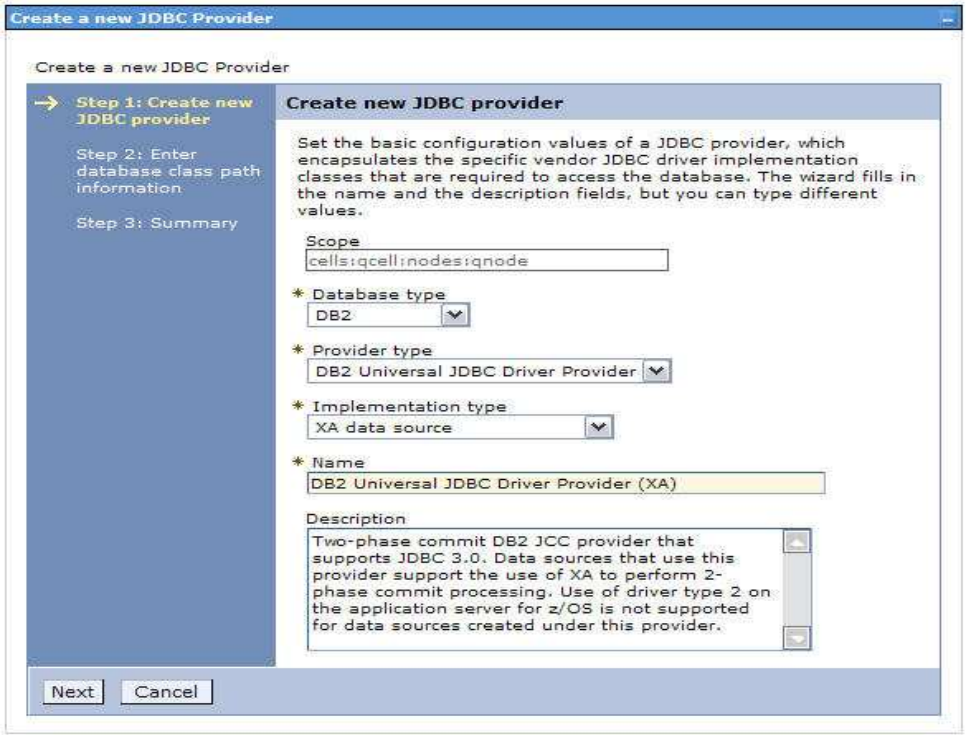
__ b. Select **Node**

__ c. On the right, click **New** and choose these following values

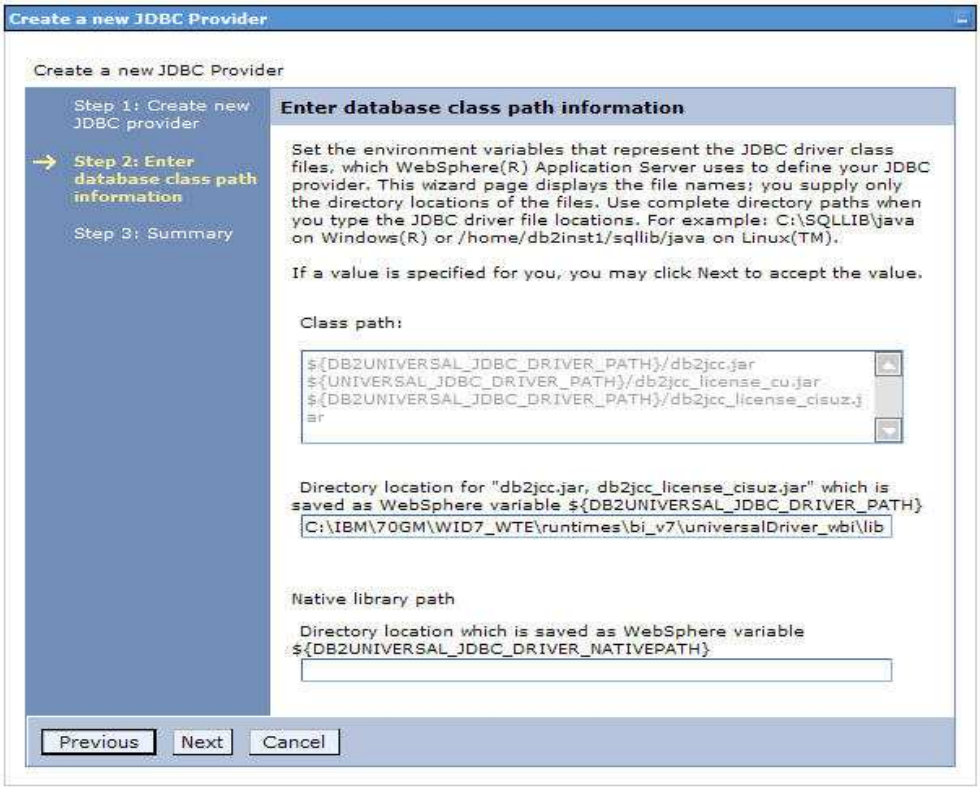
1) Database type – **DB2**

2) Provider type – **DB2 Universal JDBC Driver Provider**

3) Implementation type: **XA data source**



__ d. Click **Next** and enter the database class path information



__ e. Click **Next**

__ f. Click **Finish** in the summary page.

__ g. Click **Save** to save the changes.

____ 4. Create JDBC data source

__ a. On the right, click **Data Sources** under **Additional Properties** heading

__ b. Click **New** to set a data source and enter these following values

1) Data source name – DB2 Universal JDBC Driver XA Data source

2) JNDI name: **jdbc/JDBCTEST_DS**

__ c. Click **Next** to Step 2

__ d. In **Database specific properties for the data source,** enter these values

1) Database name – **JDBCTEST**

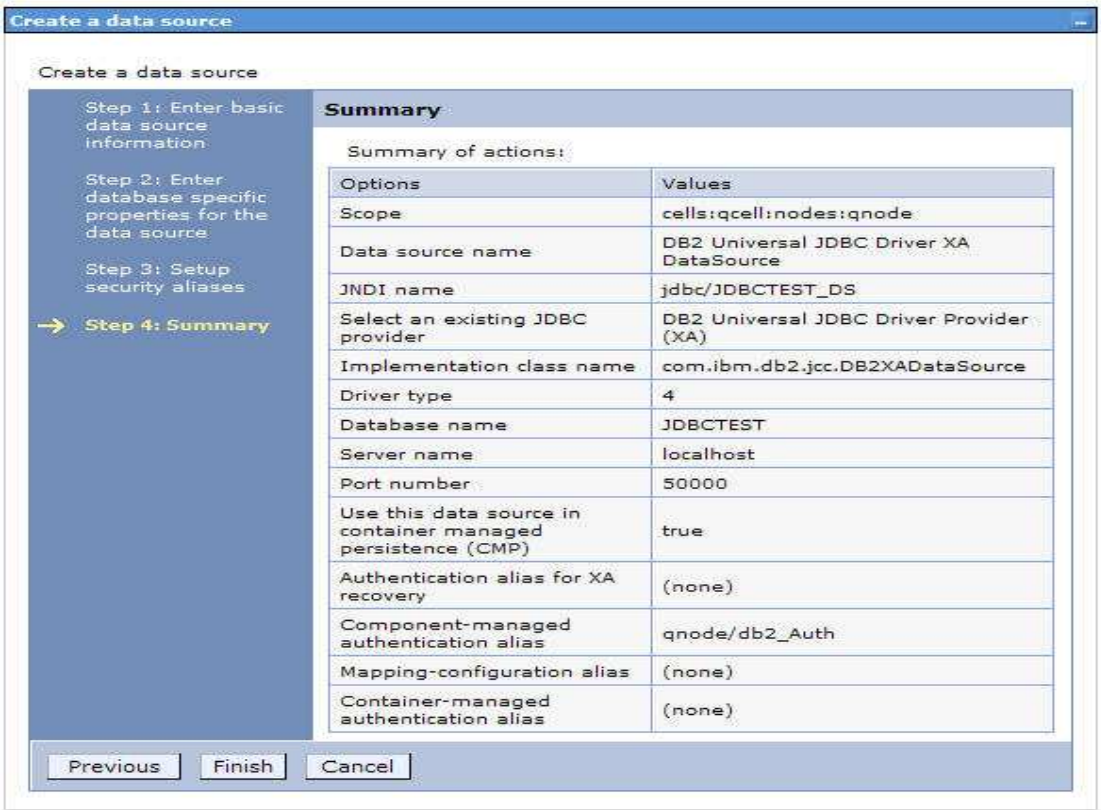
2) Server name – **LOCALHOST**

3) Port number – accept the default (50000)

__ e. Click **Next** to Step 3

__ f. Set the Component-managed authentication alias to the one created in the earlier section.

__ g. Click **Next** and view the Summary



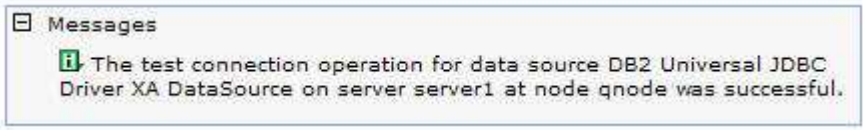
Options	Values
Scope	cells:qcell:nodes:qnode
Data source name	DB2 Universal JDBC Driver XA DataSource
JNDI name	jdbc/JDBCTEST_DS
Select an existing JDBC provider	DB2 Universal JDBC Driver Provider (XA)
Implementation class name	com.ibm.db2.jcc.DB2XADataSource
Driver type	4
Database name	JDBCTEST
Server name	localhost
Port number	50000
Use this data source in container managed persistence (CMP)	true
Authentication alias for XA recovery	(none)
Component-managed authentication alias	qnode/db2_Auth
Mapping-configuration alias	(none)
Container-managed authentication alias	(none)

__ h. Click **Finish** and save the changes.

____ 5. Test the JDBC Data Source connection

__ a. Check the box next to **DB2 Universal JDBC Driver XA DataSource** and click on **Test connection** from the top of the screen.

__ b. You should be these success message on the top of the screen



Part 5: Test the application using the WebSphere test environment

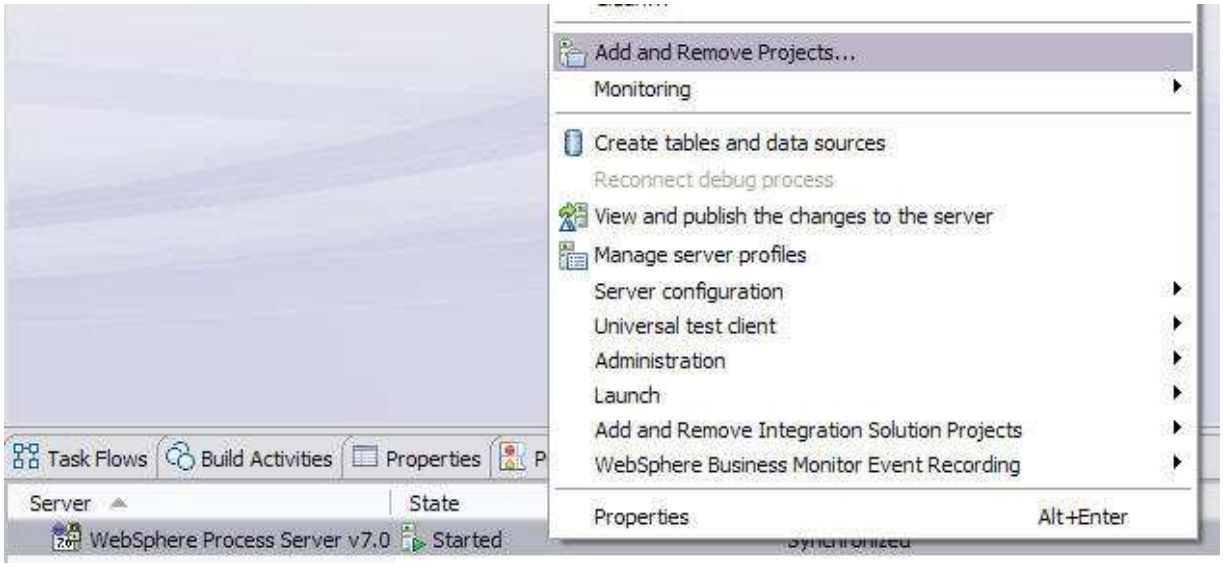
In this part you will use the WebSphere Test Environment to test the SCA application event processing by starting the application and verifying that it polls for and picks up the Create event already waiting in the WBIA_JDBC_EVENTSTORE table.

____ 1. In the **Servers** tab in the lower-right corner pane, right click the **Server** and then select **Start**

____ 2. Add the project to the server for the WebSphere Test Environment.

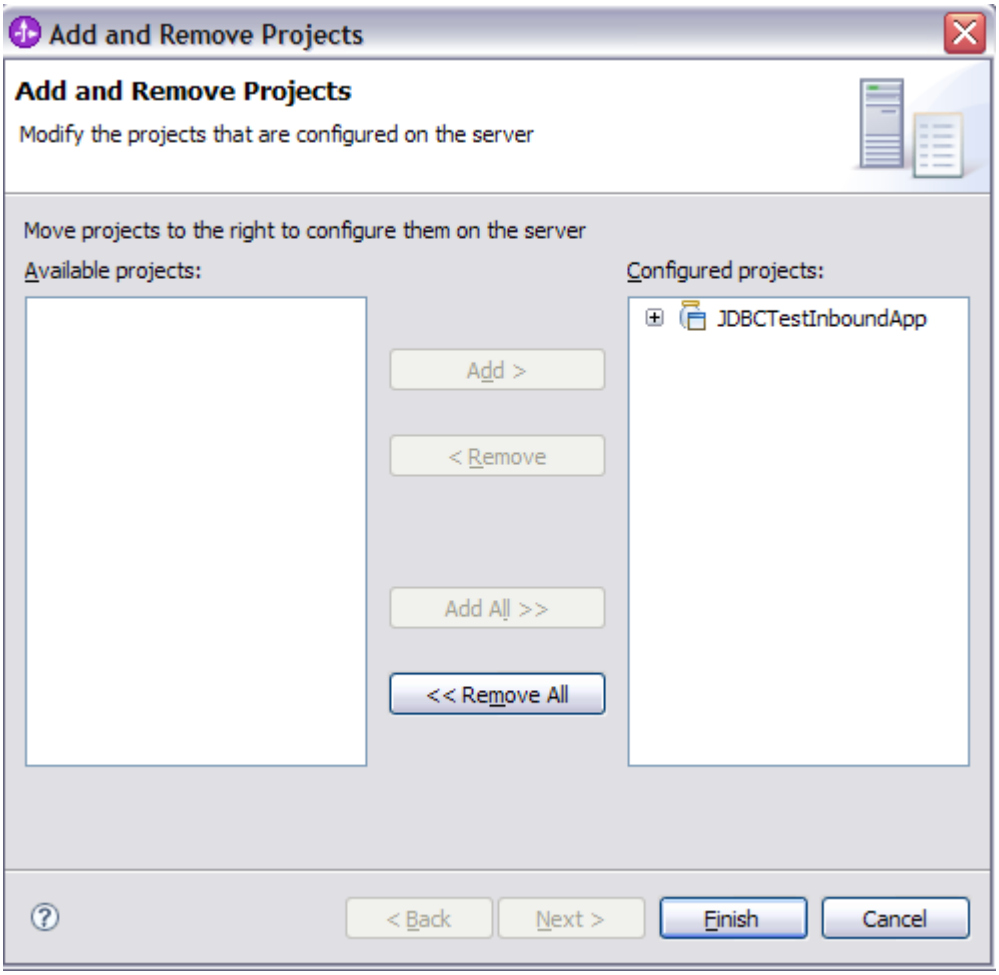
__ a. Start the server by right click Server and select **Start.**

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



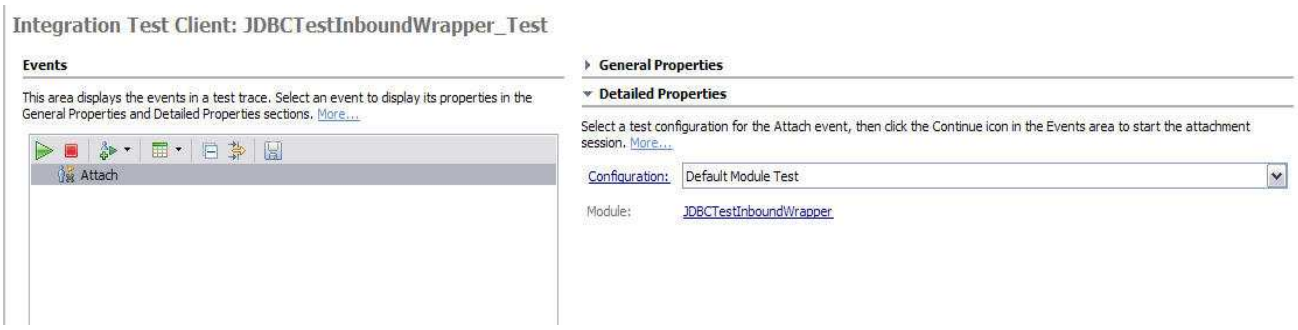
__ c. In the Add and Remove Projects dialog, select the **JDBCTestInboundApp** project from the Available projects panel.

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



__ e. In Business Integration view, right-click the JDBCTestInboundWrapper module, and select **Test → Attach**

__ f. This will open **Integration Test Client**



__ g. Click  to start new service

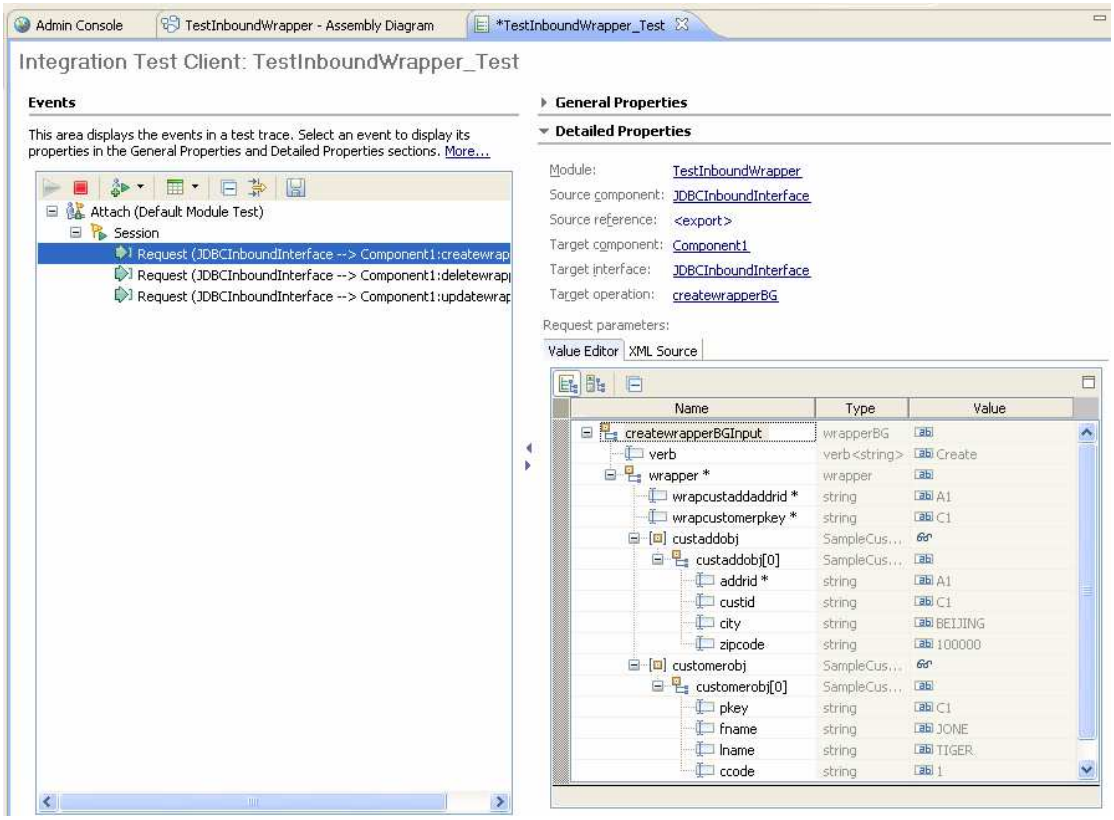
__ h. Open **JDBCTEST** database through DB2 command editor

1) Run these SQL scripts for **WBIA_JDBC_EVENTSTORE**

```
INSERT INTO WBIA_JDBC_EVENTSTORE
 (OBJECT_KEY,OBJECT_NAME,OBJECT_FUNCTION,EVENT_PRIORITY,EVEN
T_STATUS, CONNECTOR_ID) VALUES
 ('A1;C1','WrapperBG','Create',1,0,'JDBC001');
INSERT INTO WBIA_JDBC_EVENTSTORE
 (OBJECT_KEY,OBJECT_NAME,OBJECT_FUNCTION,EVENT_PRIORITY,EVEN
T_STATUS, CONNECTOR_ID) VALUES
 ('A2;C2','WrapperBG','Update',1,0,'JDBC001');
INSERT INTO WBIA_JDBC_EVENTSTORE
 (OBJECT_KEY,OBJECT_NAME,OBJECT_FUNCTION,EVENT_PRIORITY,EVEN
T_STATUS, CONNECTOR_ID) VALUES
 ('A3;C3’,'WrapperBG','Delete',1,0,'JDBC001');
```

Note: In a real environment, a trigger or another application, which can access the database, can insert the event record.

__ i. Review the output of the service from test client or from server log



__ j. 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 wrapper business object in inbound processing. You have learned how to retrieve records from multiple tables with one event entry using wrapper business object feature.