

IBM WEBSHERE ADAPTER FOR JDBC V6.1 – LAB EXERCISE

JDBC inbound lab

What this exercise is about	1
Lab requirements	1
What you should be able to do	1
Introduction	2
Exercise instructions	2
Part 1: Create the JDBCTEST Derby database and tables.....	4
Part 2: Set up the development environment	8
Part 3: Use External Service wizard to generate business objects and other artifacts	10
Part 4: Create the authentication alias and configure the data source.....	24
Part 5: Test the application using the WebSphere test environment.....	29
What you did in this exercise	32
Task: Adding remote server to the WebSphere Integration Developer test environment	33

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 V6.1 installed
- WebSphere Process Server V6.1 test environment installed
- WebSphere Adapter for JDBC V6.1 installed
- Sample code in the directory C:\Labfiles61\JDBC (Windows) or /tmp/LabFiles61/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 Derby 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 External Service 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 Process Server Test Environment.

Exercise instructions

Some instructions in this lab can 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 versus .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:\IBM\WebSphere\ID\6.1	
<WPS_HOME>	<WID_HOME>\runtimes\bi_v6	
<JDBCADAPTER_HOME>	<WID_HOME>\Resource Adapters\JDBC_6.1.0.0\deploy	
<LAB_FILES>	C:\Labfiles61	/tmp/Labfiles61
<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:\LabFiles61\ is replaced by C:/LabFiles61/

Note that the previous table is relative to where you are running WebSphere Integration Developer.

Instructions if using a remote server for testing

Note that the previous table is relative to where you are running WebSphere Integration Developer. The following table is related to where you are running the remote test environment:

Reference variable	Example: Remote Windows test server location	Example: Remote z/OS [®] test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	sssr011	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/sscell/AppServer	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<SOAP_PORT>	8880	8880	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	ssadmin	
<PASSWORD>	N/A	fr1day	

Instructions for using a remote testing environment, such as z/OS, AIX[®] or Solaris, can be found at the end of this document, in the section "[Task: Adding remote server to the WebSphere Integration Developer test environment](#)".

Part 1: Create the JDBCTEST Derby 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.

If you are using a remote testing environment on z/OS, it is recommended to create the Derby databases on your Windows machine and then FTP the data files to the z/OS environment. Be sure to upload the files in the /log and /seg0 directories in binary format, and the "service.properties" file in ASCII format. You used a directory on the host called /tmp/LabFiles61/DerbyDB. Also be sure to use the CHMOD command to change all the files to 777.

1. Start Derby ij command

Note: The Derby 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 vendor GUI tool connected to the JDBCTEST database; but not both at the same time.

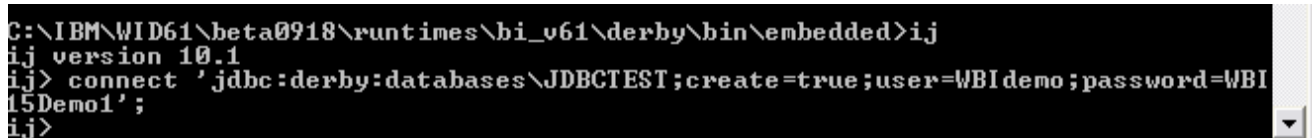
- a. Open a command prompt window, navigate to this subdirectory, and run Derby's interactive JDBC scripting tool.

```
<WPS_HOME>\derby\bin\embedded>ij
```

2. Using the command line, create the JDBCTEST database if it does not already exist. If you have already completed the JDBCOutbound lab, the database, tables, and triggers are the same. You can skip to step7, where you add records to the CUSTOMER table.

- a. Type in this script to create JDBC database as user 'Wbidemo' and password 'Wbi15Demo1' in the Console view.

```
connect 'jdbc:derby:databases\JDBCTEST;create=true;user=Wbidemo;password=Wbi15Demo1';
```



```
C:\IBM\WID61\beta0918\runtimes\bi_v61\derby\bin\embedded>ij
ij version 10.1
ij> connect 'jdbc:derby:databases\JDBCTEST;create=true;user=WBI demo;password=WBI
15Demo1';
ij>
```

3. Enter SQL to create two tables: CUSTOMER table for user data and WBIA_JDBC_EVENTSTORE table for recording events.

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

- a. Paste this script into the command window:

```
CREATE TABLE CUSTOMER
(
    pkey VARCHAR(10) NOT NULL PRIMARY KEY,
    LName VARCHAR(20),
    FName VARCHAR(20),
    ccode VARCHAR(10)
);
```

```

C:\IBM\WID61\beta0918\runtimes\bi_v61\derby\bin\embedded>ij
ij version 10.1
ij> connect 'jdbc:derby:databases\JDBCTEST;create=true;user=WBI demo;password=WBI
15Demol';
ij> CREATE TABLE customer
(
    pkey          VARCHAR(10) NOT NULL    PRIMARY KEY,
    fname         VARCHAR(20),
    lname         VARCHAR(20),
    ccode         VARCHAR(10)
);
0 rows inserted/updated/deleted

```

- ___ 4. Create the WBIA_JDBC_EVENTSTORE table and create the primary key.

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

- ___ a. Create the WBIA_JDBC_EVENTSTORE table by pasting this script into the SQL window:

```

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 default CURRENT_TIMESTAMP NOT NULL,
    event_status INTEGER NOT NULL,
    event_comment VARCHAR(100)
);

```

```

ij> CREATE TABLE WBIA_JDBC_EventStore
(
    event_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, IN
CREMENT 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 default CURRENT_TIMESTAMP NOT NULL,
    event_status INTEGER NOT NULL,
    event_comment VARCHAR(100)
);
0 rows inserted/updated/deleted
ij>

```

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

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

- ___ a. Paste this script 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.pkey, 'WbidemoCustomerBG', 'Create', 1, 0);
```

__ b. Paste this script into the command window:

```
CREATE TRIGGER event_udpate
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.pkey, 'WbidemoCustomerBG', 'Update', 1, 0);
```

__ c. Paste this script into the command 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.pkey, 'WbidemoCustomerBG', 'Delete', 1, 0);
```

```
event_comment VARCHAR(100)
>;
0 rows inserted/updated/deleted
ij> 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, even
t_priority, event_status)
VALUES (N.pkey, 'WbidemoCustomerBG', 'Create', 1, 0);
0 rows inserted/updated/deleted
ij> CREATE TRIGGER event_udpate
AFTER UPDATE ON CUSTOMER REFERENCING NEW AS N
FOR EACH ROW MODE DB2SQL
INSERT INTO wbia_jdbc_eventstore (object_key, object_name, object_function, even
t_priority, event_status)
VALUES (N.pkey, 'WbidemoCustomerBG', 'Update', 1, 0);
0 rows inserted/updated/deleted
ij> 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, even
t_priority, event_status)
VALUES (O.pkey, 'WbidemoCustomerBG', 'Delete', 1, 0);
0 rows inserted/updated/deleted
ij>
```

___ 6. From command window, create a record in the CUSTOMER Table.

__ a. Type these into command window. This will add three entries to the CUSTOMER table by typing in values for **PKEY**, **LNAME**, **FNAME**, and **CCODE**. Specify key (10, 20, 30) LNAME (Johnson, Smith, Walker), FNAME (Jerry, John, Jane), and CCODE (Office, Office, Office) respectively.

```
INSERT INTO CUSTOMER VALUES
('10', 'Johnson', 'Jerry', 'Office');
INSERT INTO CUSTOMER VALUES
('20', 'Smith', 'John', 'Office');
INSERT INTO CUSTOMER VALUES
('30', 'Walker', 'Jane', 'Office');
```

```
ij> INSERT INTO CUSTOMER VALUES ('10','Johnson','Jerry','Office');
1 row inserted/updated/deleted
ij> INSERT INTO CUSTOMER VALUES ('20','Smith','John','Office');
1 row inserted/updated/deleted
ij> INSERT INTO CUSTOMER VALUES ('30','Walker','Jane','Office');
1 row inserted/updated/deleted
ij>
```

___ b. Verify that the WBIA_JDBC_EVENTSTORE now has same number of event records for customer records that have just created.

___ c. Disconnect and exit the ij command window

```
ij> disconnect;
ij> exit;
C:\IBM\WID61\beta0918\runtimes\bi_v61\derby\bin\embedded>
```

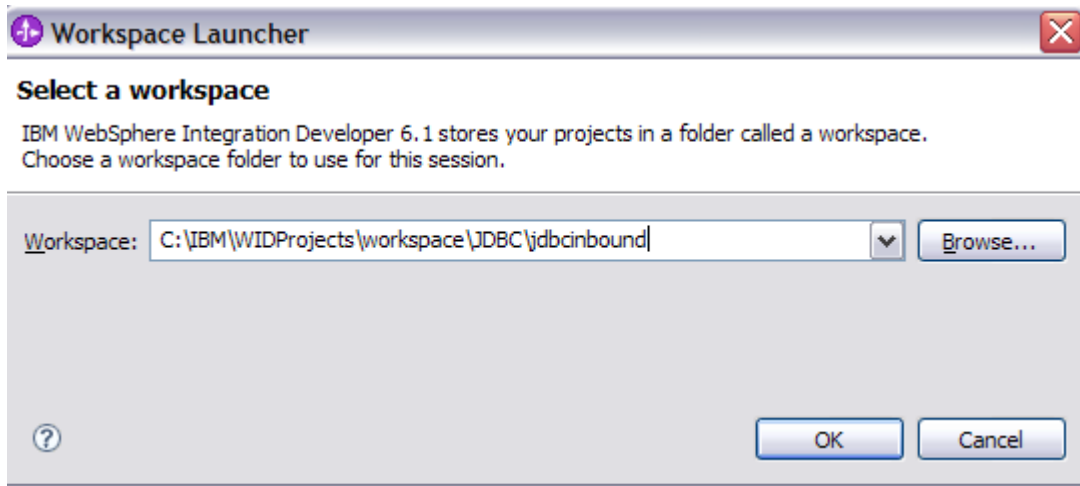
Note: You must exit the ij command window, before starting the server as the Derby “embedded” database driver used in this lab allows for only one JVM connection at a time.

Remember that if you are using a remote test environment for this lab, you need to FTP the Derby database files to the remote system, as explained at the beginning of this section.

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 V6.1 with a new workspace.
 - ___ a. From the start menu select **Start > Programs > IBM Software Development Platform > IBM WebSphere Integration Developer 6.1 > IBM WebSphere Integration Developer V6.1**
 - ___ b. When prompted enter **<LAB_FILES>\JDBC\jdbcinbound** for your workspace and click **OK**



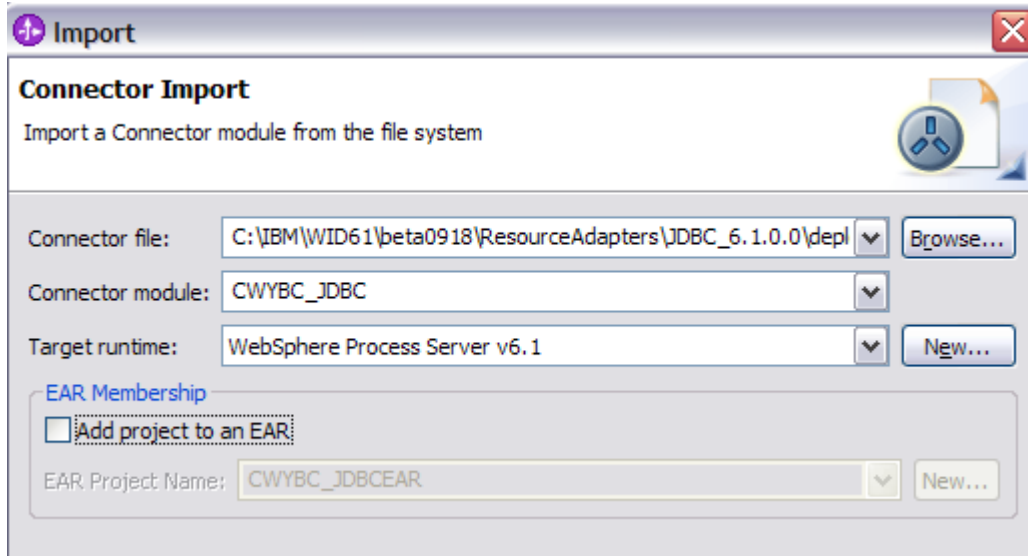
- ___ c. When WebSphere Integration Developer V6.1 opens, close the **Welcome page**



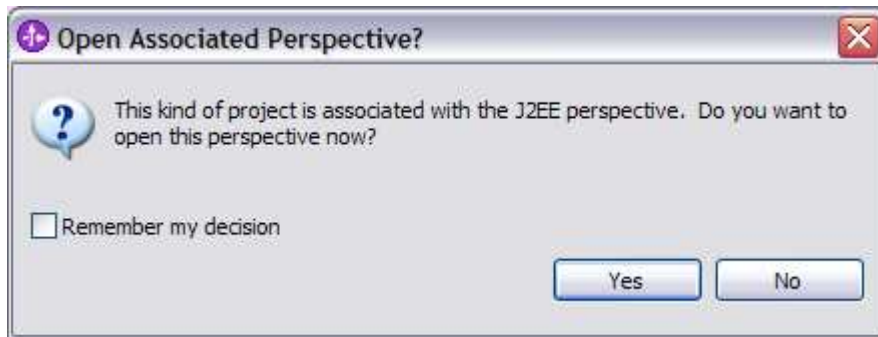
- ___ 2. Import the JDBC Adapter Resource Archive (RAR) file. This will create a J2EE Connector Project.
 - ___ a. From the top Menu bar, select **File -> Import**
 - ___ b. Select **RAR file**, click **next**

__ c. Complete the Connector Import panel

- 1) Connector file: **Browse...** to the location of the CWYBC_JDBC.rar adapter file and select it.
 - a) For example: c:\<WID_HOME>\Resource Adapters\jdbc\deploy\CWYBC_JDBC.rar
- 2) Make sure **Target Runtime** is **WebSphere Process Server V6.1**
- 3) Deselect the “**Add module to an EAR project**” and select **Finish**



__ d. At the Confirm Perspective Switch prompt, click **No**



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.

NOTE: If you are using a remote test environment on z/OS, you will need to enable your Derby database for network access using these steps:

On the host, login to the system using telnet.

In order to allow connections to the network server, you need to edit the `/<WPS_HOME>/cloudscape/db2j.properties` file. Delete the '#' (pound sign) to uncomment this line: **

```
db2j.drda.host=0.0.0.0
```

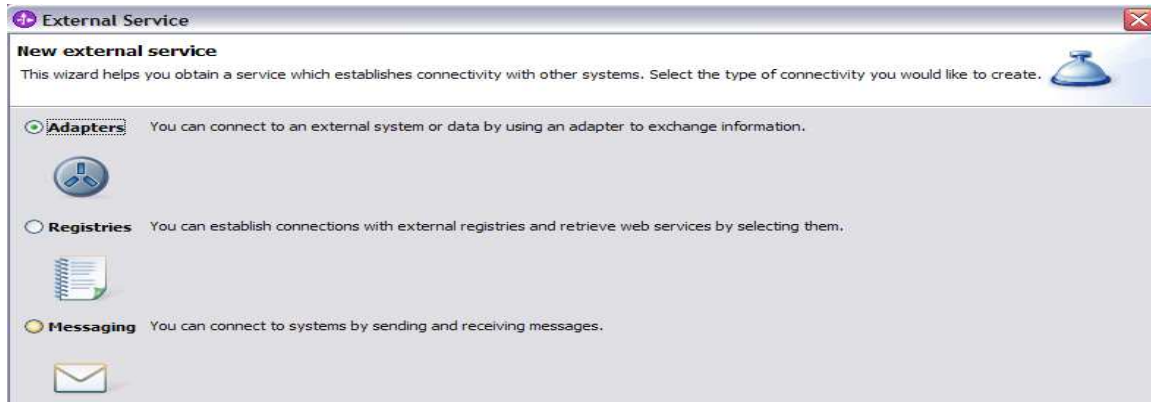
Navigate to `/<WPS_HOME>/derby/bin/networkServer`

Issue the command `./startNetworkServer.sh`

Keep this telnet window open during the next few steps; this window will show when a connection has been successfully made from WebSphere Integration Developer to the database server.

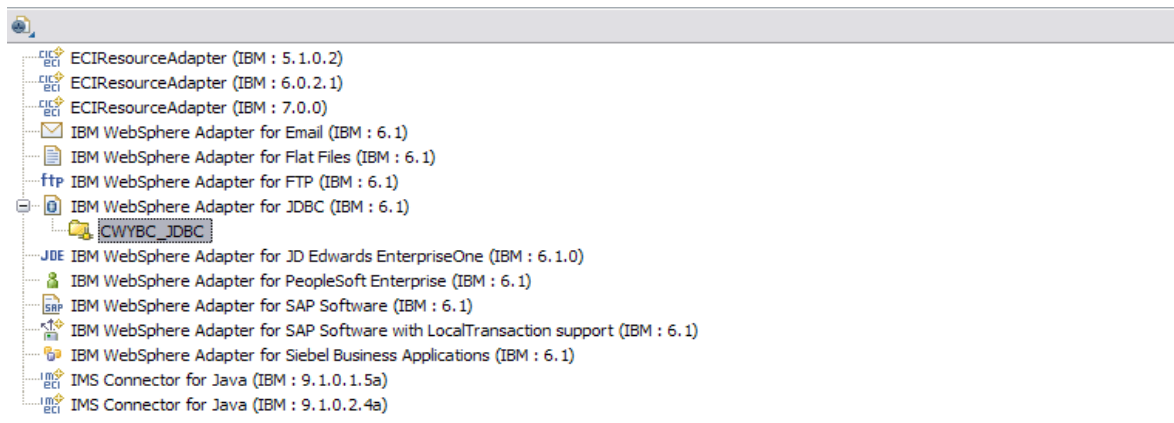
****NOTE** that there are security concerns with enabling the network server for Derby in this manner. Read the information in the `db2j.properties` file regarding these security concerns to determine if your installation will allow for the use of the network server for Derby. If not, you can continue through the lab using the integrated test client in WebSphere Integration Developer instead of your remote test environment.

-
- ___ 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**
 - ___ b. From within the Business Integration Perspective, select **File > New > External Service**
 - ___ c. Select **Adapter** and click **Next**



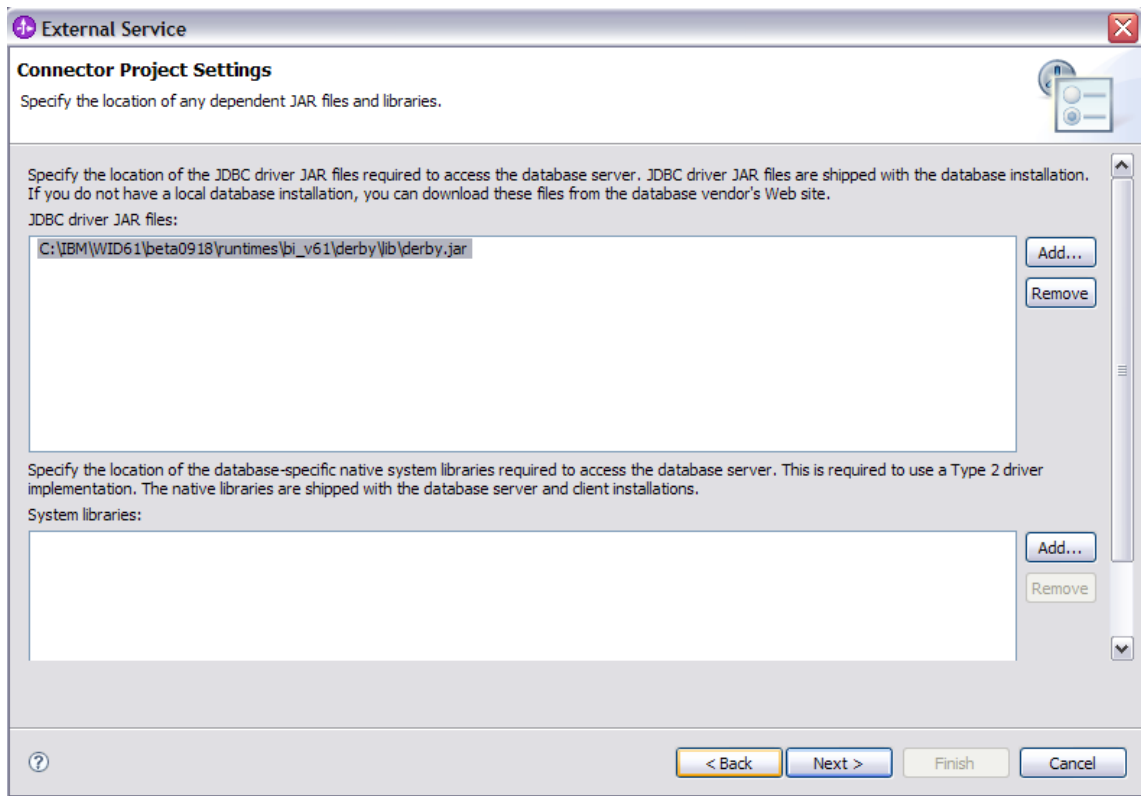
___ d. Expand **IBM WebSphere Adapter for JDBC (IBM:6.1)**

___ e. Highlight **CWYBC_JDBC** and click **Next**

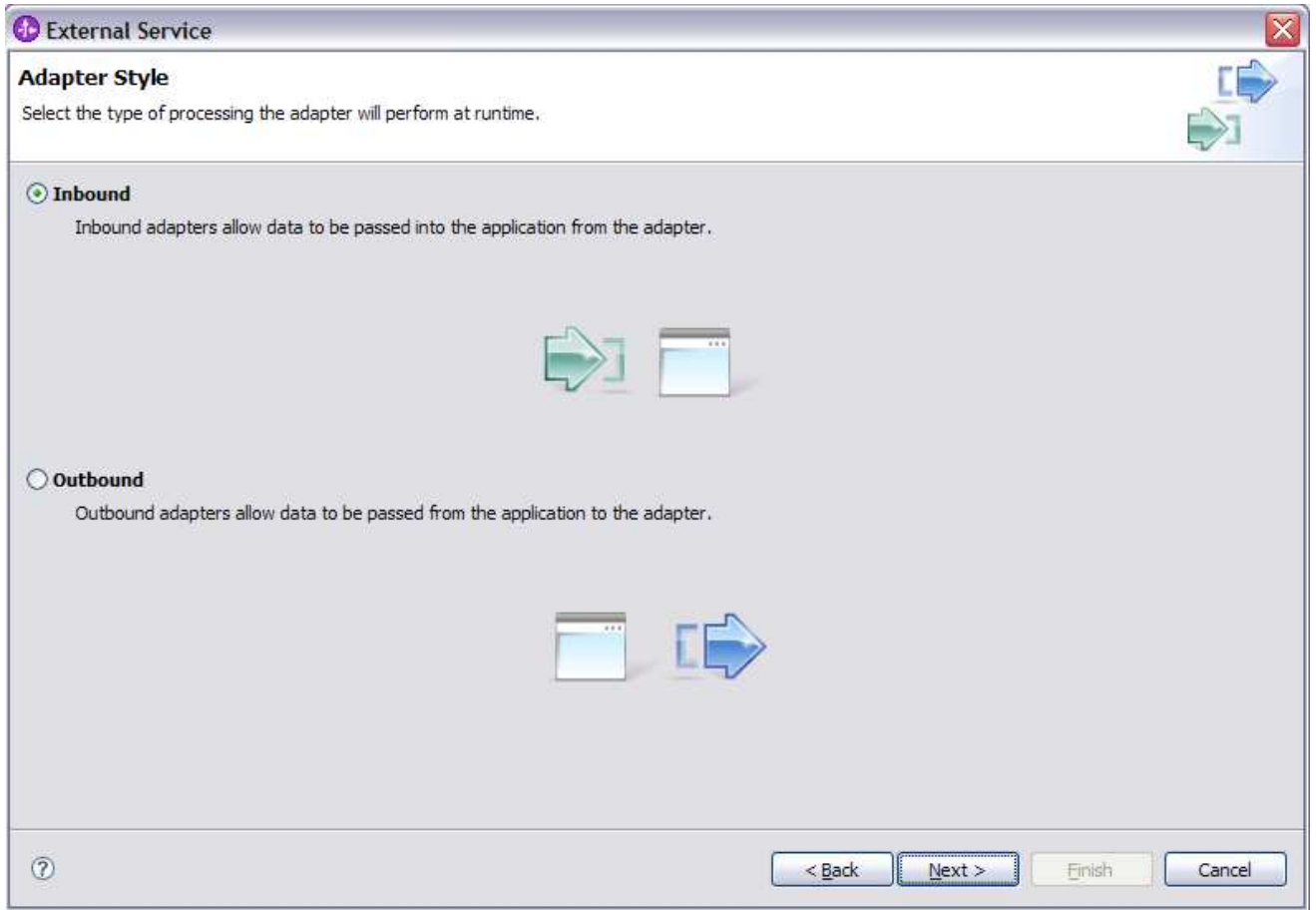


___ 2. Add any external dependencies your adapter has to the imported project. These are dependencies that the adapter can have on the JDBC applications (adapter-specific).

- ___ a. Click on **Add** and browse to the location of c:\<WPS_HOME>\derby\lib and select the **derby.jar**. If you are using a remote test environment on z/OS, you will also need to select **db2jcc.jar** and **db2jcc.license.jar**. The db2jcc.jar file is sometimes located in directory \derby\lib\otherJars.



- ___ b. Click **Next**
- ___ 3. Select the type of processing the adapter will perform at runtime
 - ___ a. Select **Inbound** and click **Next**



- ___ 4. 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 **Generic JDBC** and select **1.0**
 - ___ b. From the right panel, enter/select these following values
 - 1) JDBC Driver type: **Other**
 - 2) JDBC Driver classname: **org.apache.derby.jdbc.EmbeddedDriver**
 - 3) Database URL: **jdbc:derby:<WPS_HOME>derby\databases\JDBCTEST**
 - ___ c. Enter valid user ID and password values, for example
 - Username: **Wbidemo**
 - Password: **Wbi15Demo1**

If you are using a remote test environment, use a valid user ID and password for the remote system.
 - ___ d. Enter these values for DatabaseURL and JdbcDriverClass

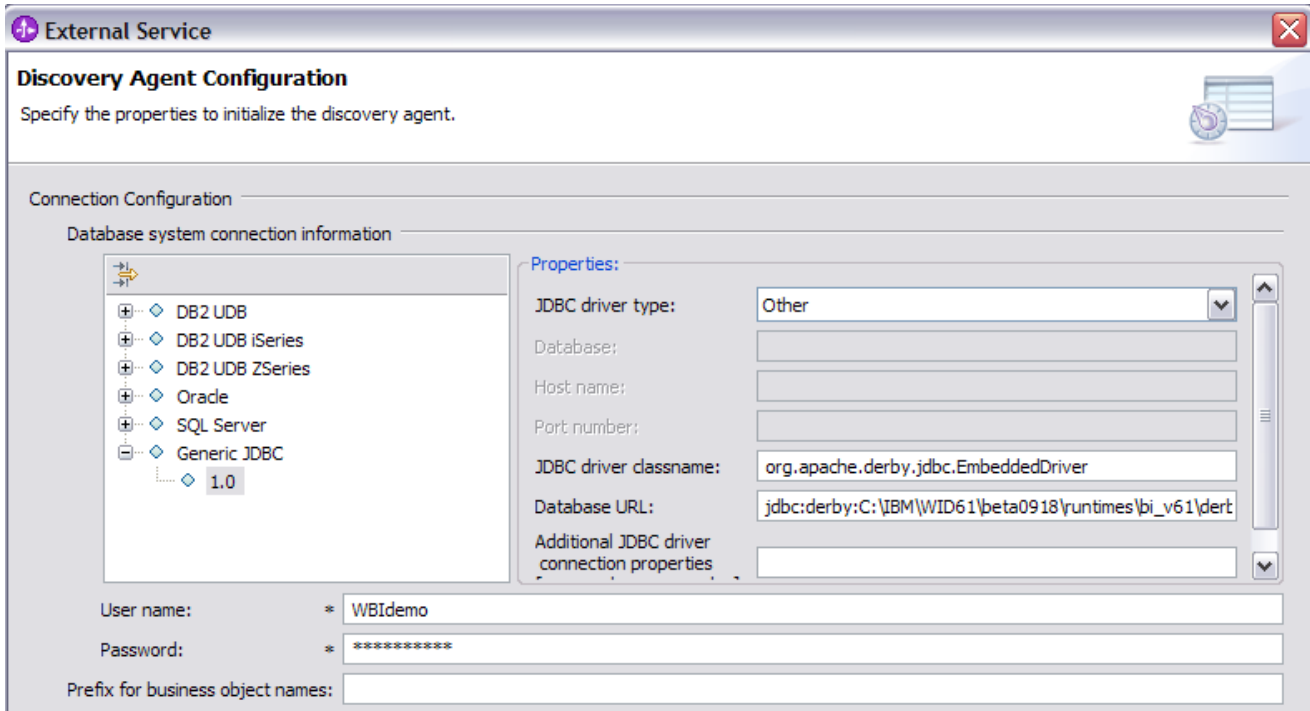
DatabaseURL: **jdbc:db2j:<WPS_HOME>\runtimes\bi_v6\derby\databases\JDBCTEST**

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

NOTE: If using a remote test environment on z/OS use these values:

DatabaseURL: **jdbc:db2j:net://<HOST_NAME>:1527//<remote_derbydb_path>/JDBCTEST**

JdbcDriverClass: **COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver**




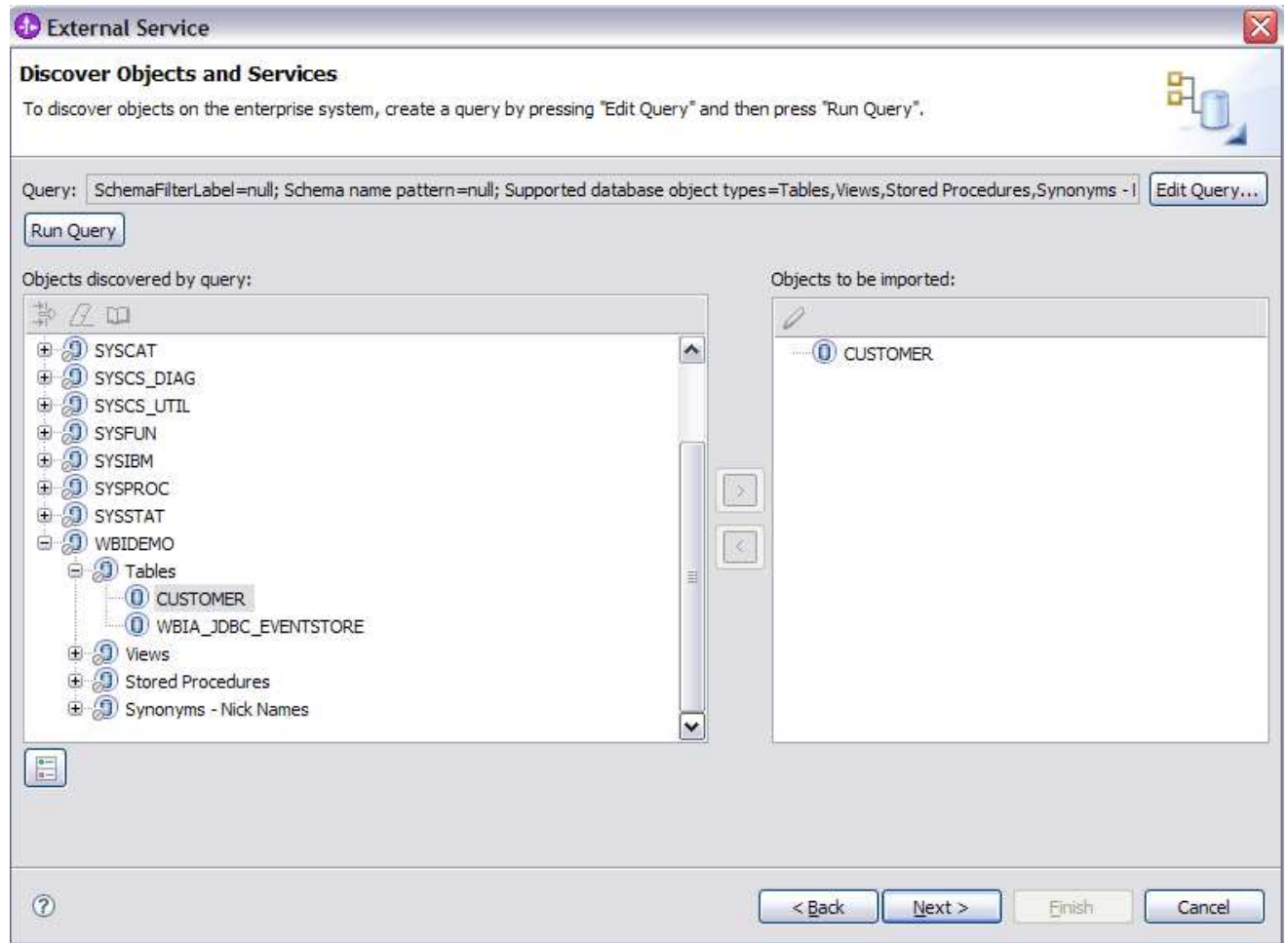
___c. Click **NEXT**

___5. Complete the **Discover Objects and Services** panel

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

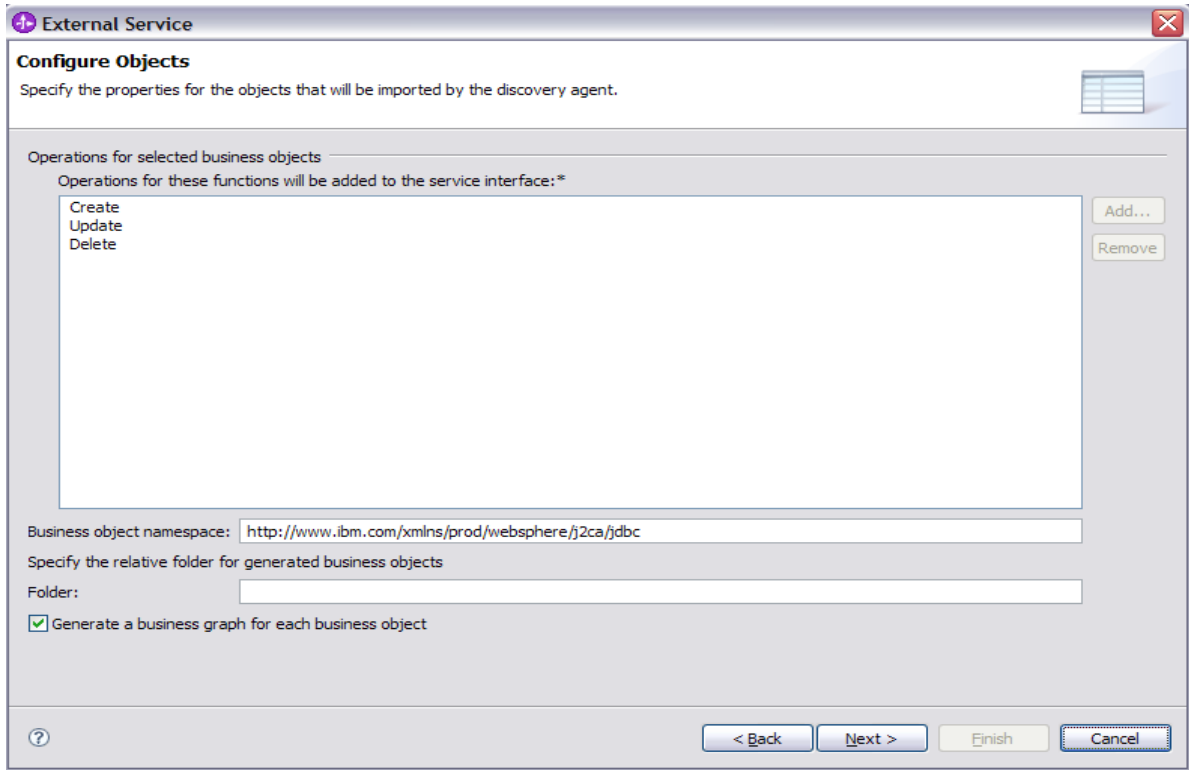
Note: Select the **Edit Query ...** button to see the available options, however, do not change any of the defaults for this lab exercise.

- ___ b. Expand the schema named **WBIDEMO**, expand **Tables**, highlight **CUSTOMER**, click on  button. **CUSTOMER** now appears in the **Objects to be imported** window. Click **Next**



___ 6. Complete the **Configure Objects** panel

- ___ a. Leave the default value for Namespace, leave blank for **BOLocation**. Note the Operations available.
- ___ b. Note that Business Graph is optional - It will determine if you want to generate business graph or not. If the property is checked then business graphs are generated. By default, the property is checked.



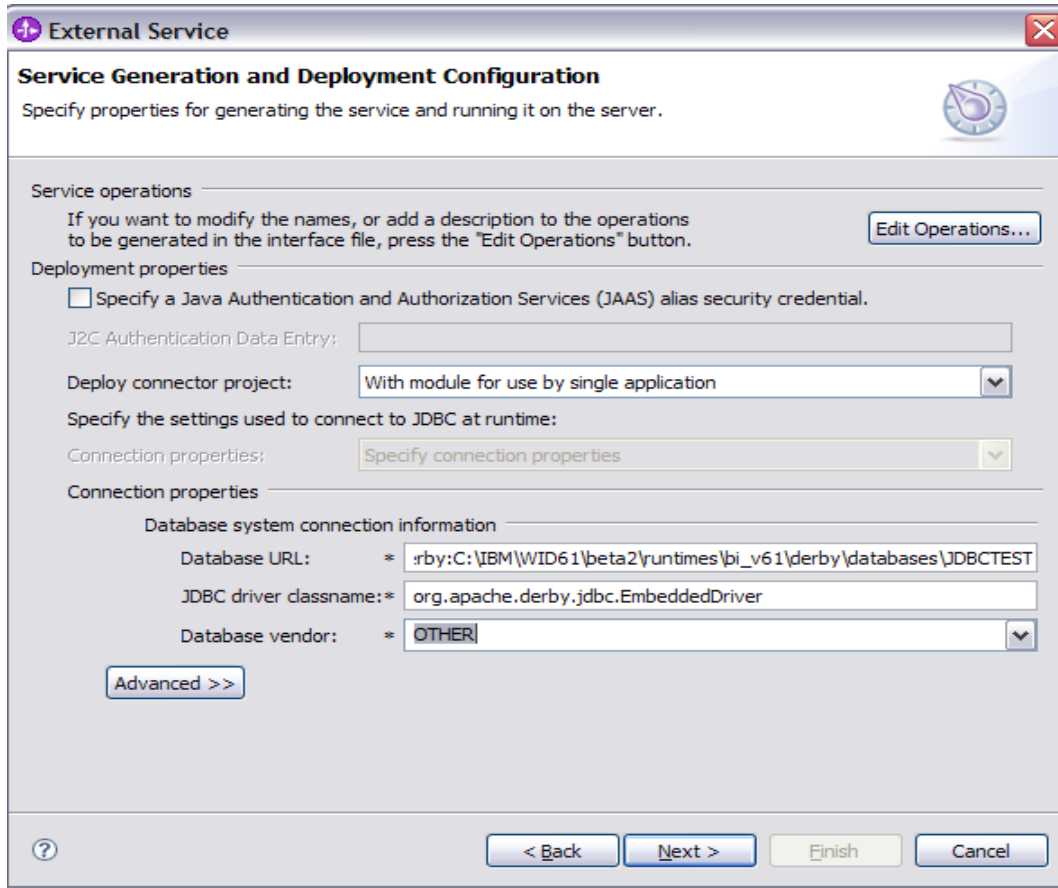
__ c. Click **Next**.

___ 7. Complete the **Publishing Object Configuration Properties**

__ a. Leave the default **“With module for use by single application”** option in this exercise.

Note: All IBM WebSphere Adapters are now supported deploying RAR separately. To do so, select “With module for use by single application” from the Deploy connector project drop down list. If you want the adapter to be deployed with the module which is packaged as an Enterprise Archive file (EAR file), then the “Deploy connector with module” check box should be checked. Finally, the “Use discovered connection properties” check box should always be checked.

- ___ b. Uncheck the **JAAS Alias security credential** as you are going to specify the JNDI data source name instead.
- ___ c. Select the radio button to the left of **Use discovered connection properties**
 - 1) Database Vendor is pre-populated with value **Other** (If you were using DB2, Oracle, or MSSQLServer, you see those values instead as specific adapter processing is available with those specific databases.)



- ___ d. Click on **Advanced** button for additional properties options
- ___ e. Expand **Event delivery configuration**
 - 1) Verify the **Ensure once-only event delivery** is checked
- ___ f. Expand **Advanced connection configuration** and enter this following value
Data Source JNDI Name: **jdbc/Derby XA for JDBC**
- ___ g. Expand Event Configuration
 - 1) Event Order By: **event_time, event_priority**
 - 2) Event Table Name: **WBIA_JDBC_EventStore**

Note if you are using a remote test environment, the values for **databaseURL** and **JdbcDriverClass** should be what you filled in earlier for the External Service wizard.

The screenshot shows three configuration panels in a web application:

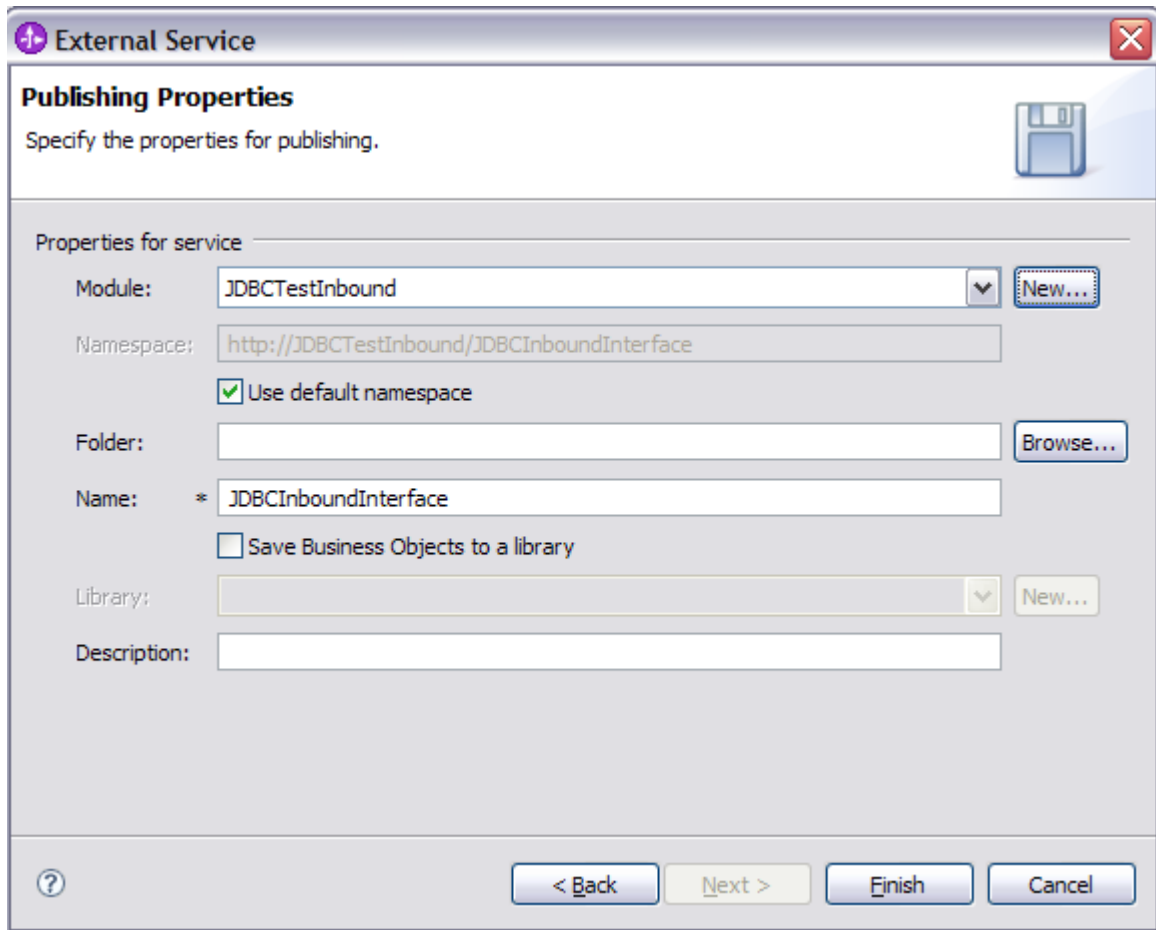
- Event delivery configuration:**
 - Type of delivery:
 - Ensure once-only event delivery (may reduce performance)
 - Do not process events that have a timestamp in the future
 - Event types to process:
 - Number of connections for event delivery:
 - Minimum:
 - Maximum:
- Advanced connection configuration:**
 - Datasource JNDI name:
 - Additional JDBC driver connection properties [name:value;name:value]:
 - SQL query to verify the connection:
 - Query timeout (seconds):
 - Return business object even when the stored procedure result set is empty:
- Event configuration:**
 - Event order by:
 - Event table name: *
 - Stored procedure to run before poll:
 - Stored procedure to run after poll:
 - Event query type for processing events:
 - User-defined event query:
 - User-defined update query:
 - User-defined delete query:

___ h. Click **Next**

___ 8. Complete **Publishing Properties** 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.

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



___ c. Click **Finish**

___ 9. Use the Assembly Diagram to wire the **JDBCTestInbound** Interface to a Java Component.

___ a. 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.



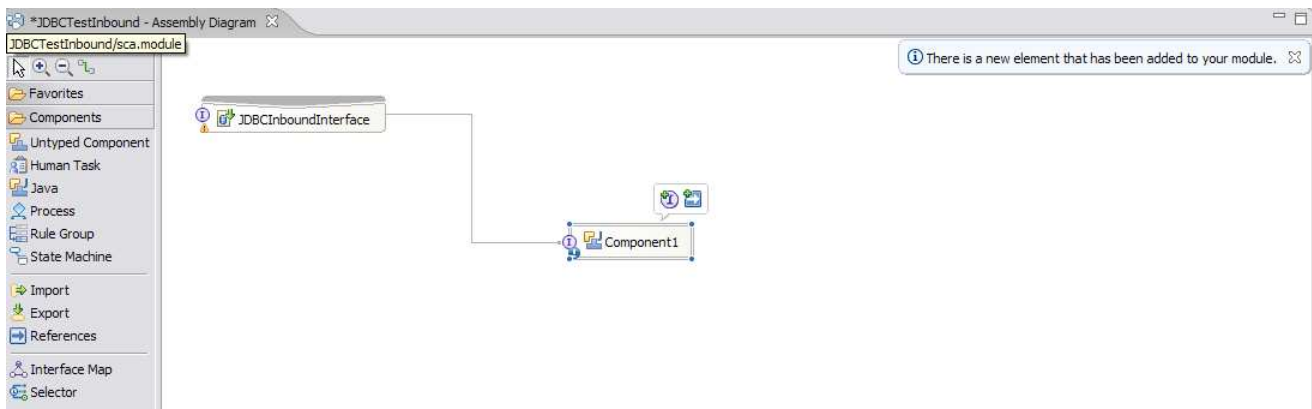
- c. From the palette, expand **Component**, then select the **Java Component** and drop it on the Assembly Diagram.



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

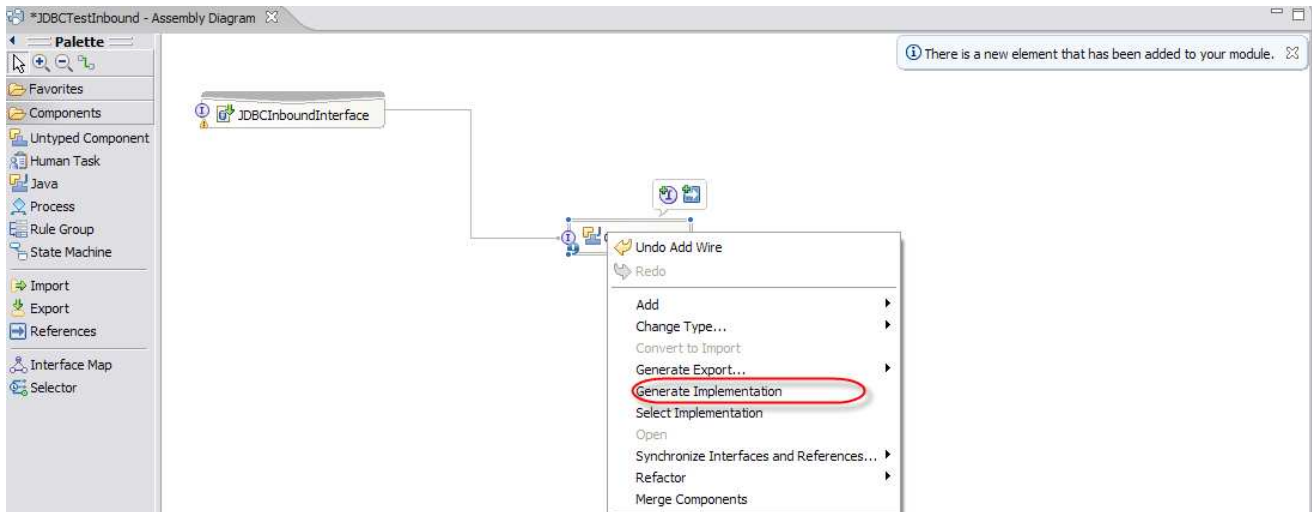


- e. The Assembly Diagram now looks as follows:



- 10. Generate the implementation for the Java Component

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



___ b. Highlight the **com.test.data** package and select **OK** (Click New Package and create the com.test.data package if it does not already exist.)



___ 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 **createWbidemoCustomerBG(Object createWbidemoCustomerBGInput)** method that needs to be implemented. Paste this code into the method so the complete method looks as follows:

```
public void createWbidemoCustomerBG(Object createWbidemoCustomerBGInput) {
    System.out.println("CREATE customer");
    DataObject bg=(DataObject)createWbidemoCustomerBGInput;
    DataObject bo=bg.getDataObject("WbidemoCustomer");
    System.out.println("CUSTOMER KEY is: "+bo.getString("pkey"));
    System.out.println("CUSTOMER LAST NAME is: "+bo.getString("lname"));
    System.out.println("CUSTOMER FIRST NAME is: "+bo.getString("fname"));
    System.out.println("CUSTOMER CODE is: "+bo.getString("ccode"));
    System.out.println("CREATE end");
}
```

- ___ e. Scroll down and locate the **updateWbidemoCustomerBG (Object updateWbidemoCustomerBGInput)** method that needs to be implemented. Paste this code into the method so the complete method looks as follows:

```
public void updateWbidemoCustomerBG(Object updateWbidemoCustomerBGInput) {
    System.out.println("UPDATE customer");
    DataObject bg=(DataObject)updateWbidemoCustomerBGInput;
    DataObject bo=bg.getDataObject("WbidemoCustomer");
    System.out.println("CUSTOMER KEY is: "+bo.getString("pkey"));
    System.out.println("CUSTOMER LAST NAME is: "+bo.getString("lname"));
    System.out.println("CUSTOMER FIRST NAME is: "+bo.getString("fname"));
    System.out.println("CUSTOMER CODE is: "+bo.getString("ccode"));
    System.out.println("UPDATE end");
}
```

- ___ f. Scroll down and locate the **deleteWbidemoCustomerBG (Object deleteWbidemoCustomerBGInput)** method that needs to be implemented. Paste this code into the method so the complete method looks as follows:

```
public void deleteWbidemoCustomerBG(Object deleteWbidemoCustomerBGInput) {
    System.out.println("DELETE customer");
    DataObject bg=(DataObject)deleteWbidemoCustomerBGInput;
    DataObject bo=bg.getDataObject("WbidemoCustomer");
    System.out.println("CUSTOMER KEY is: "+bo.getString("pkey"));
    System.out.println("DELETE end");
}
```

- ___ 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.

- ___ 11. Release the connection to the Derby database by using **Switch Workspace**

Note: Switch Workspace is a way to release the existing connection to the Derby database from the External Service process. In the next part, you will start the WebSphere Process Server and it will need a connection to the database to retrieve records. This step is necessary only because you are using the Derby embedded driver in this exercise which supports a connection from a single JVM.

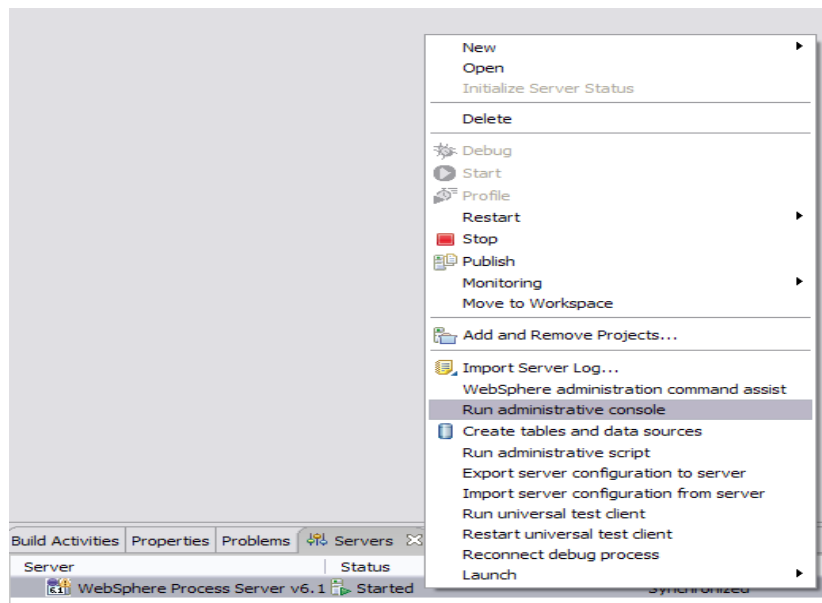
- ___ a. In server view, stop the server by right click on **WebSphere Process Server V6.1** and select **Stop**
- ___ b. From the top menu bar, select **File > Switch Workspace** and select the same workspace from which you have been working.

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

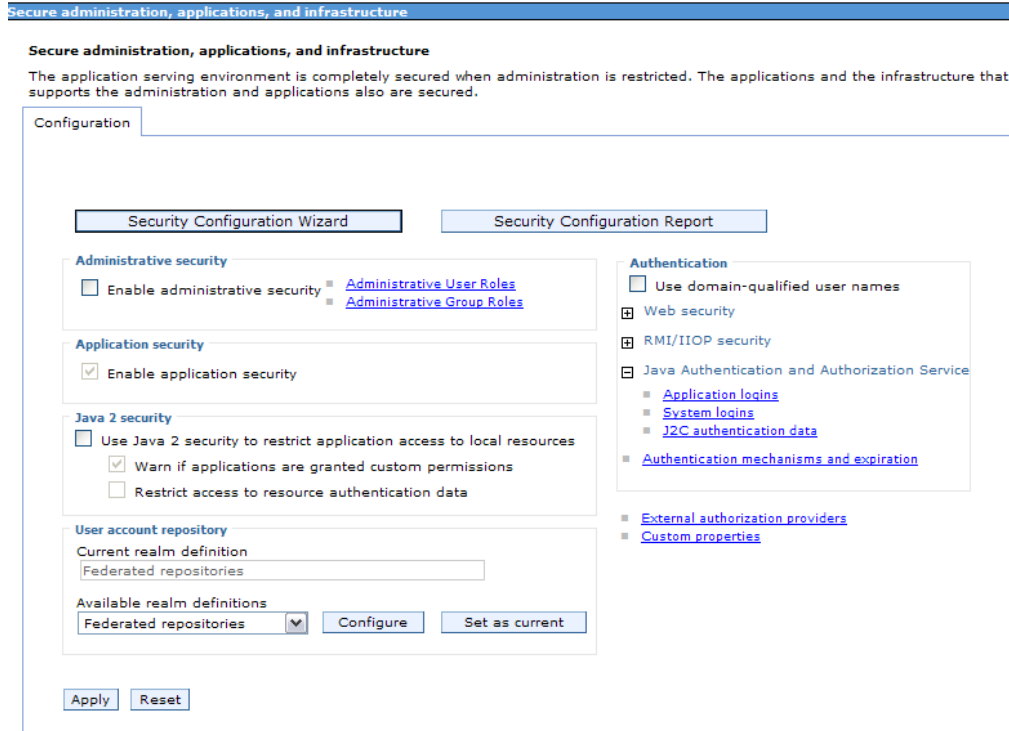
Note: If you've already completed this section in any of JDBC Outbound labs, you can skip this and proceed to Part 5.

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 on the **Server** and then select **Start**
 - ___ b. When the server status is **Started**, right click on the **Servers**, and then select **Run administrative console**.



- ___ c. Log in to the administrative console by clicking the “Log in” button
- ___ d. Click on **Security → Secure administration, application, and infrastructure**
- ___ e. On the right, expand **Java Authentication and Authorization Service** under the Authentication heading.

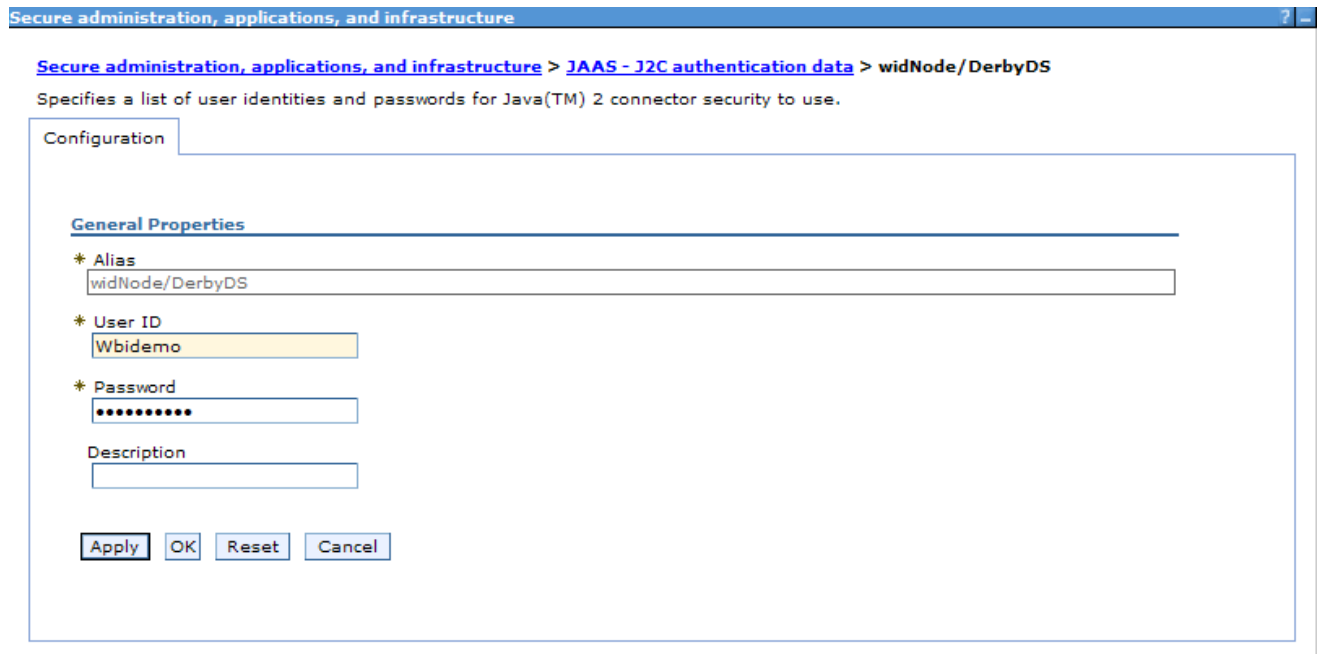


__ f. Click on **J2C authentication alias**. It gives the list of existing aliases.

1) Click **New** and enter these following values

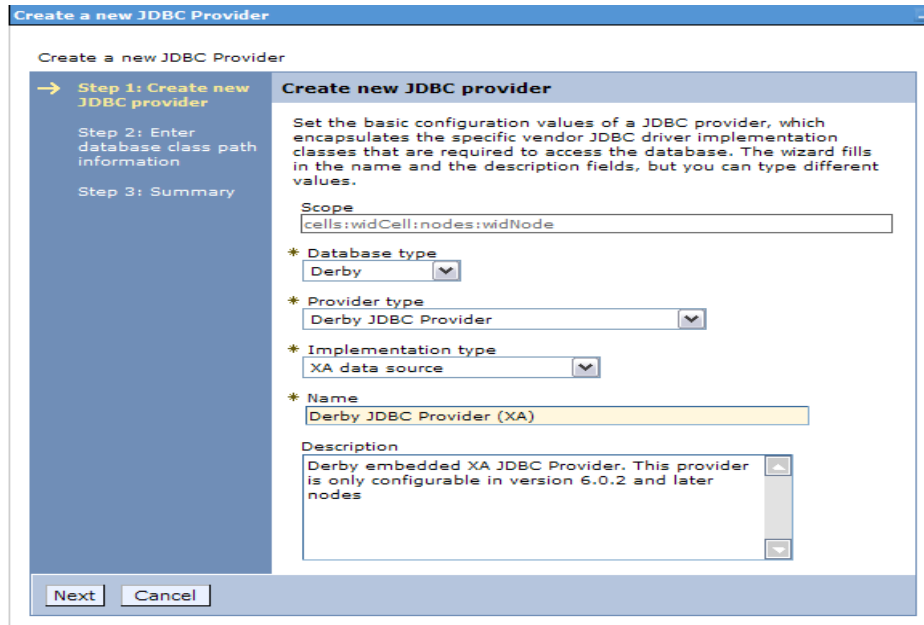
a) User ID: **Wbidemo**

b) Password: **Wbi15Demo1**

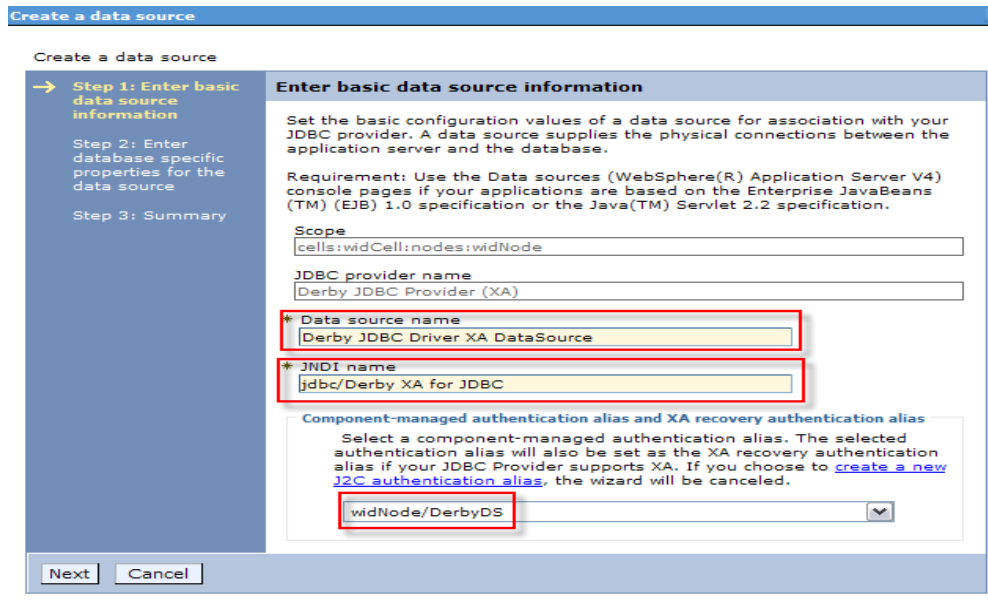


__ g. Click **Ok** and save the changes

- ___ 3. Create a JDBC Provider to which is used to create data source
 - ___ a. From administrative console, click on **Resources** → **JDBC** → **JDBC Providers**
 - ___ b. On the right, click on **New** and choose these following values
 - 1) Database type: **Derby**
 - 2) Provider type: **Derby JDBC Provider**
 - 3) Implementation type: **XA data source**



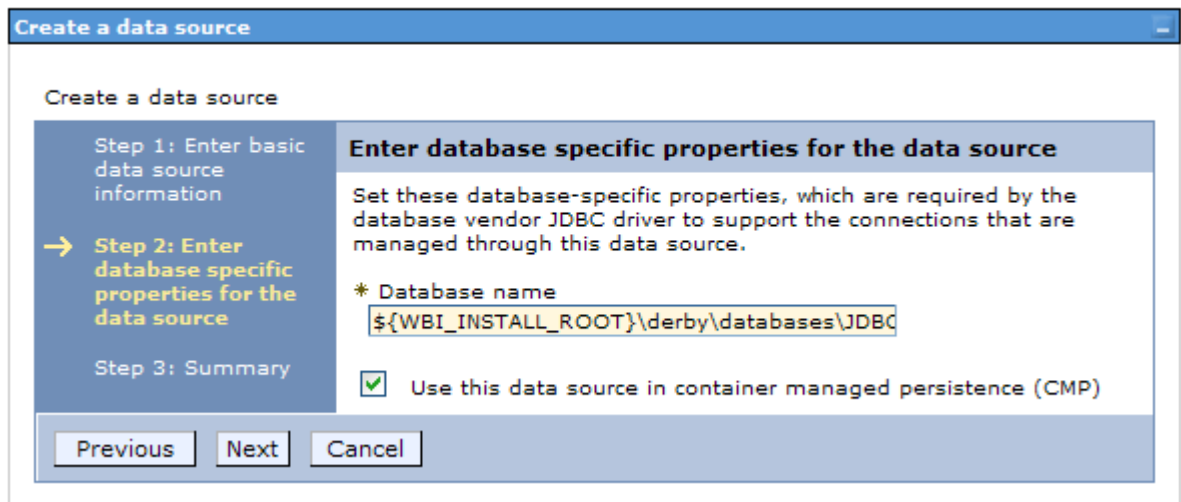
- ___ c. Click **Next** and review the summary of changes.
 - ___ d. Click **Finish** and save the changes.
- ___ 4. Create JDBC data source
 - ___ a. On the right, click on **Data Sources** under **Additional Properties** heading
 - ___ b. Click **New** to set a data source and enter these following values
 - 1) Data source name: **Derby JDBC Driver XA DataSource**
 - 2) JNDI name: **jdbc/Derby XA for JDBC**
 - ___ c. Set the Component-managed authentication alias to the one created in the earlier section.



___ d. Click **Next**.

___ e. Enter the full path to the location of database name **JDBCTEST**.

JDBC providers



___ f. Click **Next** and view the summary

___ g. Click **Finish** and save the changes.

___ 5. Test the JDBC Data Source connection

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

___ b. You should see this success message on the top of the screen

The screenshot shows the 'JDBC providers' management console. At the top, a message box indicates: 'The test connection operation for data source Derby JDBC Driver XA DataSource on server server1 at node widNode was successful.' Below this, the breadcrumb path is 'JDBC providers > Derby JDBC Provider (XA) > Data sources > Data sources'. A paragraph explains that this page is used to edit settings for a data source associated with a selected JDBC provider. A 'Preferences' section contains buttons for 'New', 'Delete', 'Test connection' (highlighted with a red box), and 'Manage state...'. Below the buttons is a table with columns: 'Select', 'Name', 'JNDI name', 'Scope', 'Provider', 'Description', and 'Category'. One row is visible, representing the 'Derby JDBC Driver XA DataSource' with JNDI name 'jdbc/Derby XA for JDBC', Scope 'Node=widNode', and Provider 'Derby JDBC Provider (XA)'. The description notes it is only configurable in version 6.0.2 and later nodes. A 'Total 1' summary is shown at the bottom of the table.

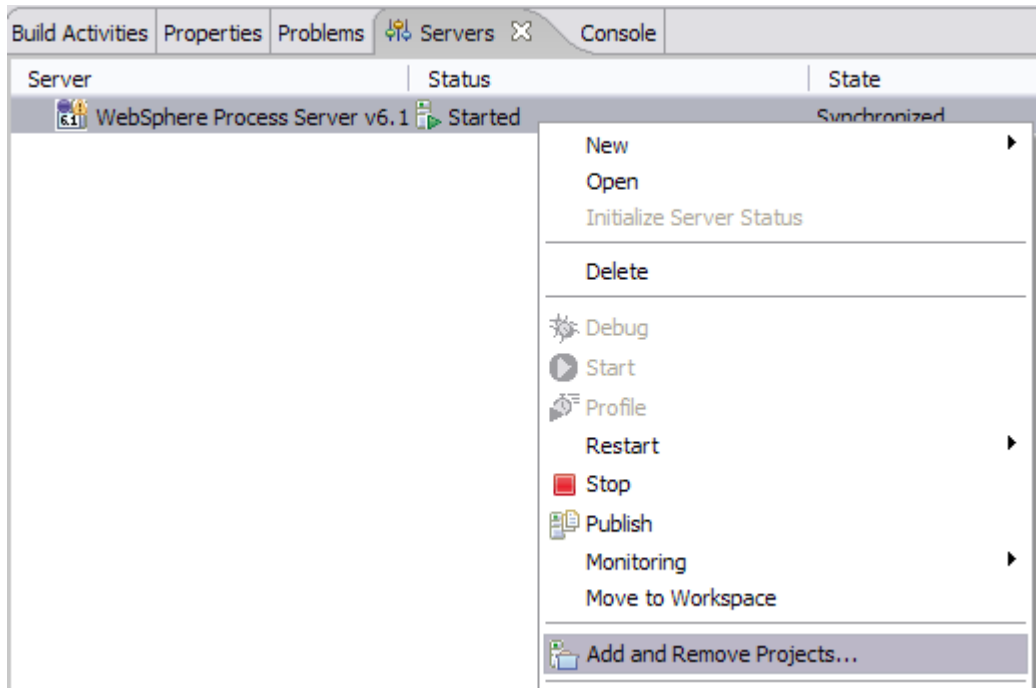
Select	Name	JNDI name	Scope	Provider	Description	Category
<input checked="" type="checkbox"/>	Derby JDBC Driver XA DataSource	jdbc/Derby XA for JDBC	Node=widNode	Derby JDBC Provider (XA)	New JDBC Datasource. This Datasource type is only configurable in version 6.0.2 and later nodes	

- ___ 6. Release the connection to the Derby database by using **Switch Workspace** again
 - ___ a. From the top menu bar, select **File > Switch Workspace** and select the same workspace from which you have been working.

Part 5: Test the application using the WebSphere test environment

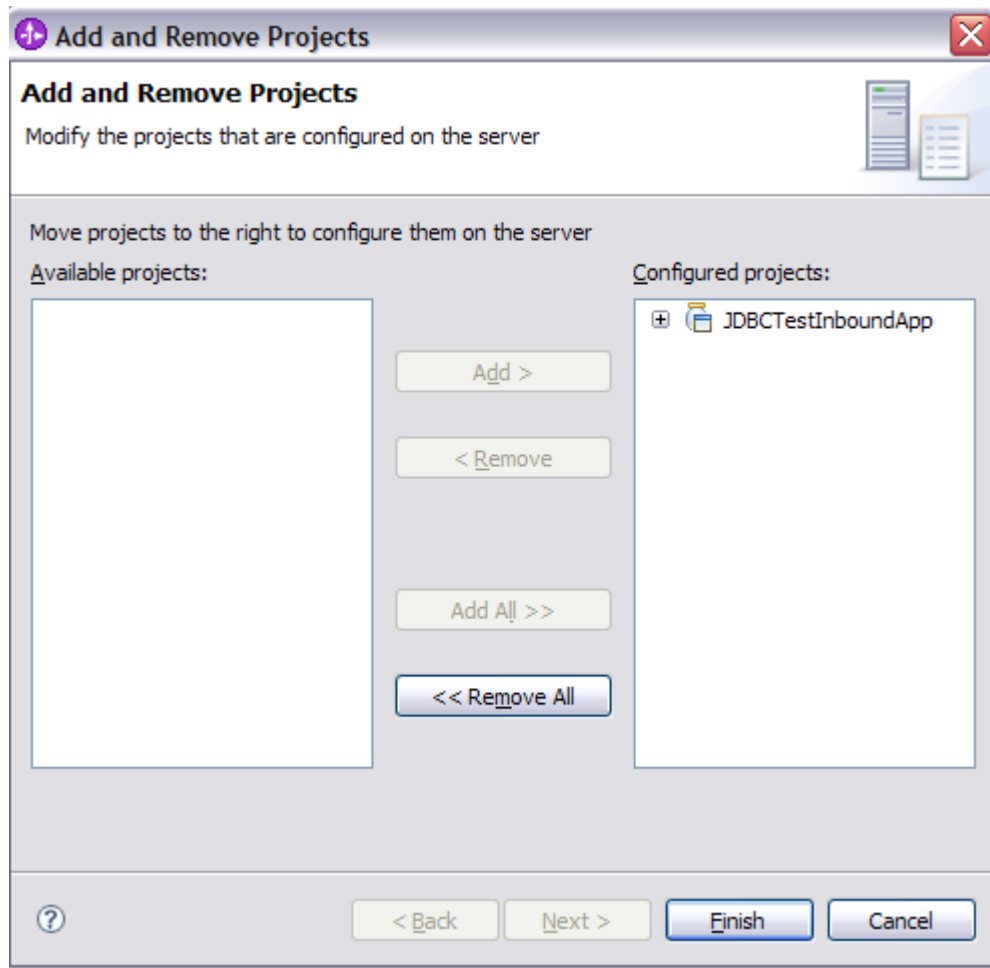
In this part you will use the WebSphere Test Environment to test the SCA application event processing by starting the application. Verifying that it polls for and picks up the Create event already waiting in the WBI_A_JDBC_EVENTSTORE Table.

- ___ 1. In the **Servers** tab in the lower-right corner pane, right click on 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 on Server and select **Start**.
 - ___ b. Right click on 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. 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 events. (These messages are also reflected in the <WPS_HOME>\profileName\logs\server1\Systemout.log).

```
[10/14/07 21:00:44:843 CDT] 00000044 SystemOut      O CREATE customer
[10/14/07 21:00:44:843 CDT] 00000044 SystemOut      O Customer Key is: 10
[10/14/07 21:00:44:843 CDT] 00000044 SystemOut      O Customer LAST NAME is:
Johnson
[10/14/07 21:00:44:843 CDT] 00000044 SystemOut      O Customer FIRST NAME
is: John
[10/14/07 21:00:44:843 CDT] 00000044 SystemOut      O Customer CODE is:
Office
[10/14/07 21:00:44:843 CDT] 00000044 SystemOut      O CREATE end
. . .
[10/14/07 21:00:44:859 CDT] 00000042 SystemOut      O Customer Key is: 30
[10/14/07 21:00:44:859 CDT] 00000042 SystemOut      O Customer LAST NAME is:
Walker
[10/14/07 21:00:44:859 CDT] 00000042 SystemOut      O Customer FIRST NAME
is: Jane
[10/14/07 21:00:44:859 CDT] 00000042 SystemOut      O Customer CODE is:
Office
[10/14/07 21:00:44:859 CDT] 00000042 SystemOut      O CREATE end
```

If you like to try other events, such as an update or delete, you will need to first stop the server to release its connection to the JDBCTEST database. Then, start the ij.bat GUI to add, delete or update records to the database. Once completed, exit the IJ 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.

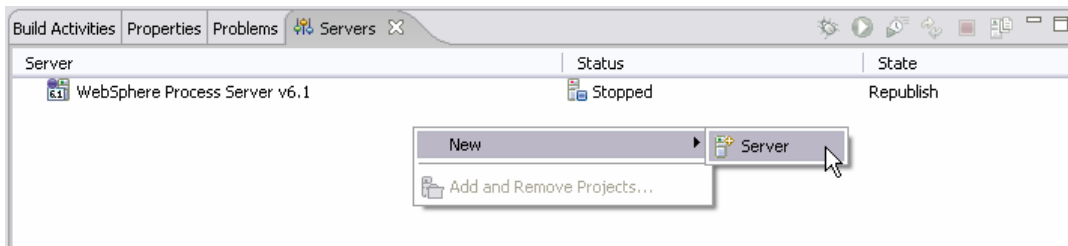
What you did in this exercise

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

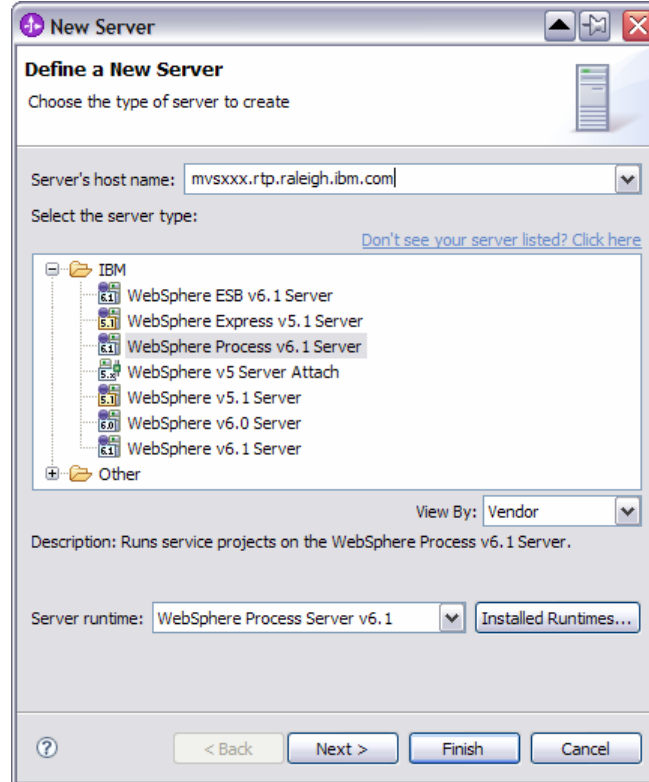
Task: Adding remote server to the WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer test environment. This example uses a z/OS machine.

- ___ 1. Define a new remote server to WebSphere Integration Developer.
 - ___ a. Right click on the background of the **Servers** view to access the pop-up menu.
 - ___ b. Select **New → Server**.

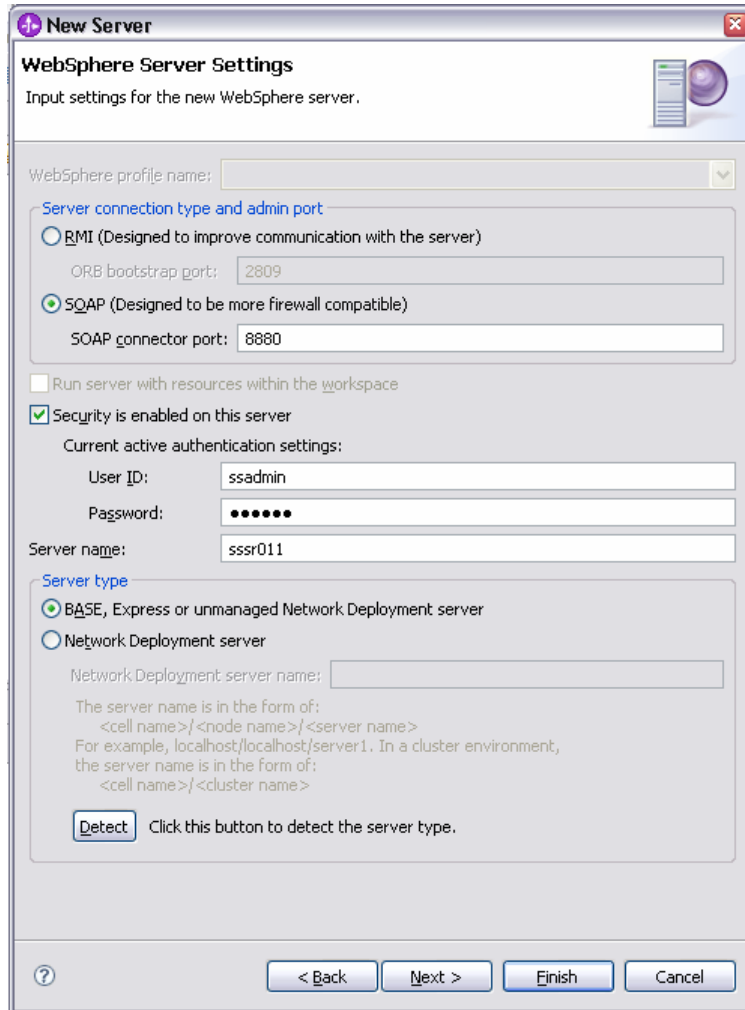


- ___ c. In the New Server dialog, specify the remote server's host name, **<HOSTNAME>**.
- ___ d. Ensure that the appropriate server type, **'WebSphere Process v6.1 Server'** or **'WebSphere ESB v6.1 Server'**, is highlighted in the server type list

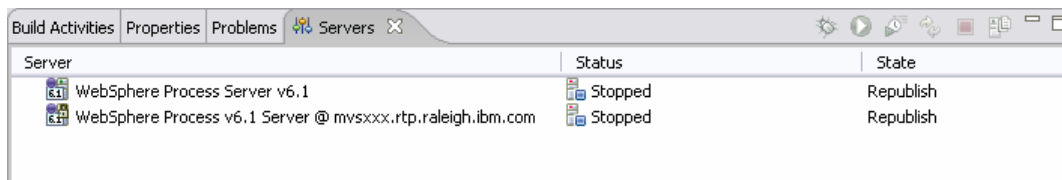


- ___ e. Click **Next**.

- ___ f. On the WebSphere Server Settings page, leave the radio button for **SOAP** selected, changing the **SOAP connector port** to the correct setting (<SOAP_PORT>). If security is on in your server, check the box for 'Security is enabled on this server' and input <USERID> for the user ID and <PASSWORD> for the password.



- ___ g. Click **Finish**.
- ___ h. The new server should be seen in the Server view.



- ___ 2. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.
- ___ a. From a command prompt, telnet to the remote system if needed:

'telnet <HOSTNAME> <TELNET_PORT>'

User ID : **<USERID>**

Password : **<PASSWORD>**

__ b. Navigate to the bin directory for the profile being used:

cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin

__ c. Run the command file to start the server: **./startServer.sh <SERVER_NAME>**

__ d. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status
```

```
ADMU3000I: Server sssr01 open for e-business; process id is 0000012000000002
```