

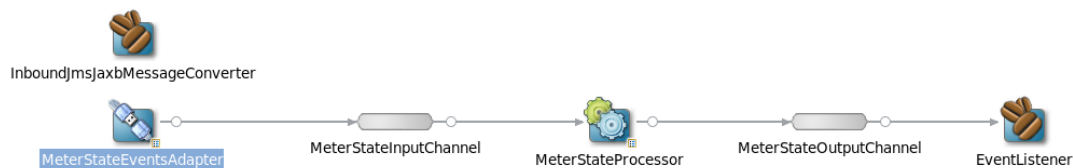
Oracle Event Processing:  
Tutorial: EPN Editor  
(JMS and JAXB Application)

*An Oracle Tutorial*  
*Updated December 2013*

**Supported Version:** Oracle Event Processing 11g (11.1.1.7)

**Objectives:**

This document describes how to create up a basic OEP application using the EPN Editor in the Eclipse IDE tooling. The CEP application uses the built-in JMS adapter to receive JMS text messages containing XML that is then converted to a java objects using JAXB. It also illustrates how to create a channel, a processor and a listener that will write events to the console.



***Table of Contents***

<b>Set-up:</b> .....	<b>4</b>
<b>Part 1: Creating a New Application</b> .....	<b>6</b>
<b>Part 2: Adding an Existing Java Library to an OEP Application</b> .....	<b>6</b>
<b>Part 2: Defining an Event in the OEP Application</b> .....	<b>9</b>
<b>Part 3: Creating a JMS Adapter</b> .....	<b>10</b>
<b>Part 4: Defining a Channel</b> .....	<b>13</b>
<b>Part 5: Creating a CQL Processor</b> .....	<b>14</b>
<b>Part 5: Creating an Event Listener Bean</b> .....	<b>17</b>
<b>Part 6: JMS Adapter Converter Bean</b> .....	<b>19</b>
<b>Part 7: Deploying the Application</b> .....	<b>23</b>
<b>Part 8: Testing</b> .....	<b>25</b>

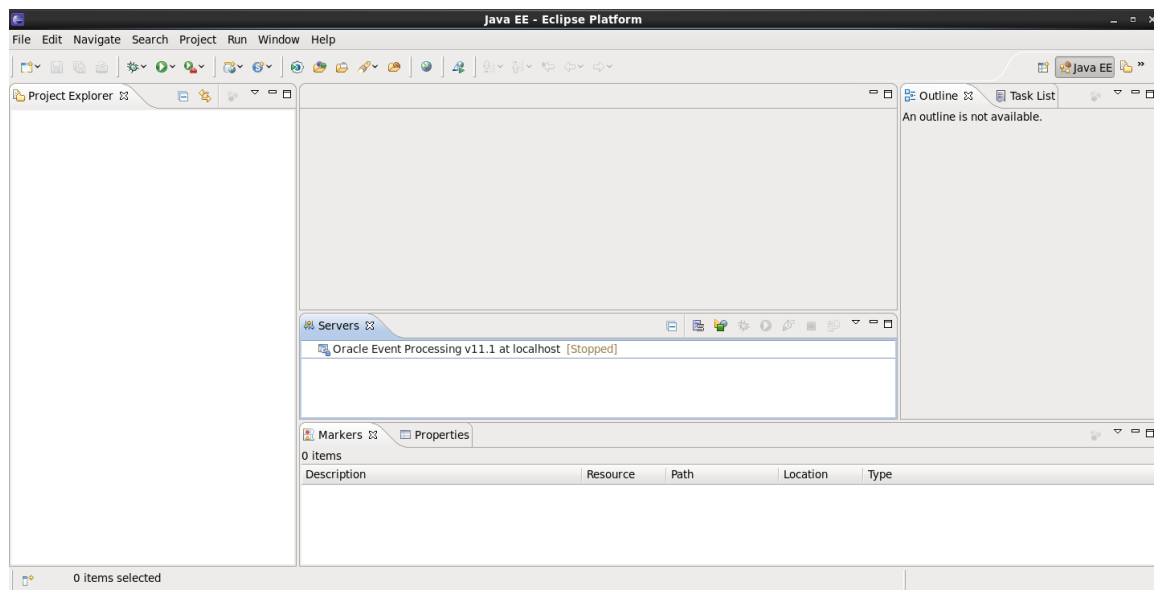
## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

---

### Set-up:

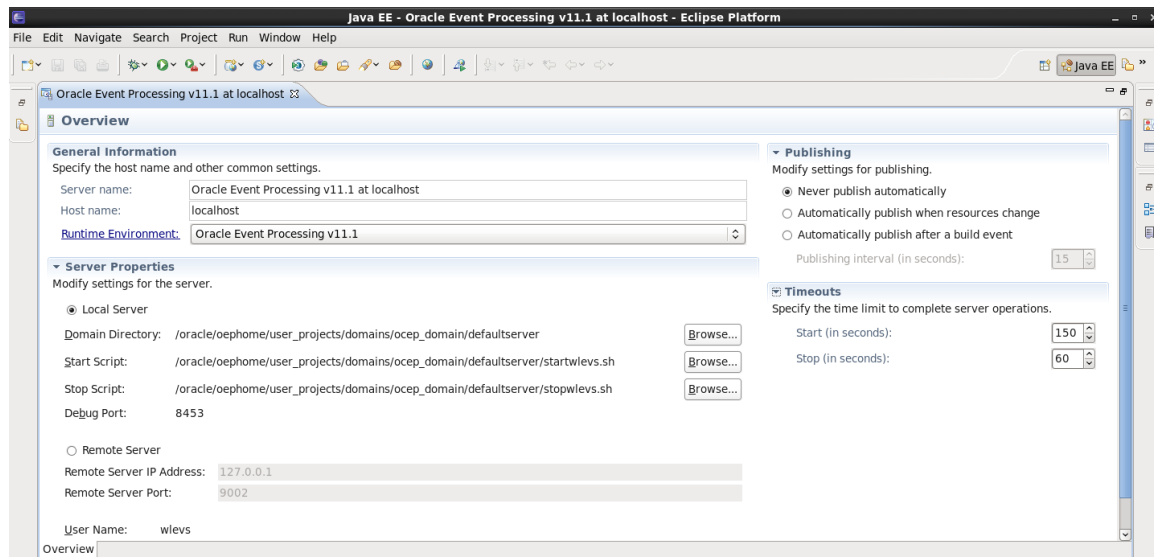
On the hosted environment, the files needed for these labs are in the **/oracle/oep\_training** directory. This will be referred to in this document as: **<OEP\_TRAINING>**

1. You should already have your Eclipse environment set-up with the OEP plug-in installed.
2. Double-Click on the icon labeled “Indigio” to open the Eclipse IDE for OEP 11g.
3. It should open the workspace “/oracle/oep\_training/oep\_workspace” in the Java EE perspective.
4. There should already be an OEP server created.

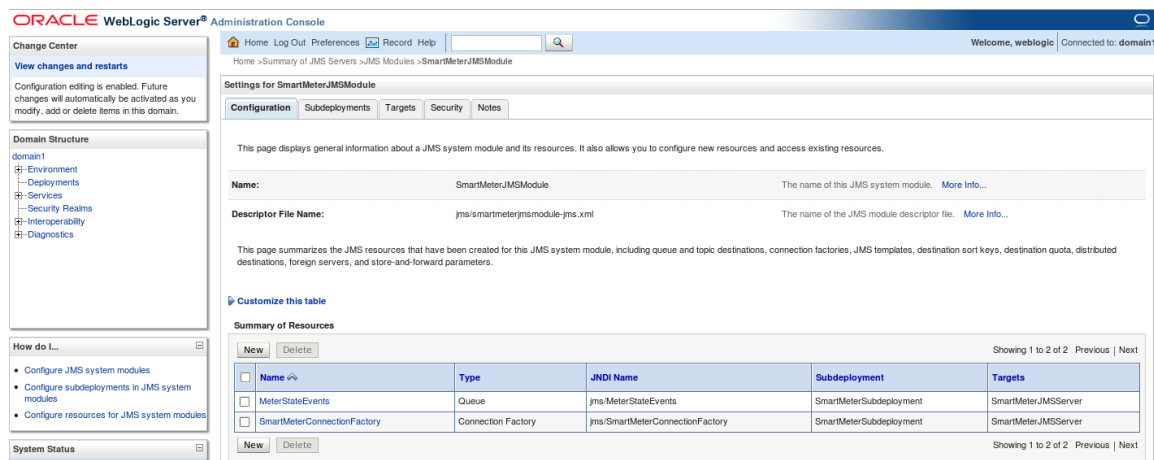


5. Right-click on the server and choose “Open”. Notice the location of the domain directory.

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)



6. You should already have the “com.oracle.oep.training.smartmeter.simulator” project imported into your workspace. This is just a simple Java project for generating events.
  - a. If it is not in your workspace or you are setting up your own local environment, you can import it using File, Import, “Existing Projects into Workspace”, select the “Archive File” radio button.
  - b. Browse to find the file “<OEP\_TRAINING>\labs\01\_EPN\_Editor\project\simulator\01-Simulator-EclipseProject.zip”. Click “Open” and “Finish”.
7. There are already JMS queues for this lab on the hosted environment.
  - a. If you are using the hosted environment, verify that the JMS queues exist using the WebLogic Server Admin Console.

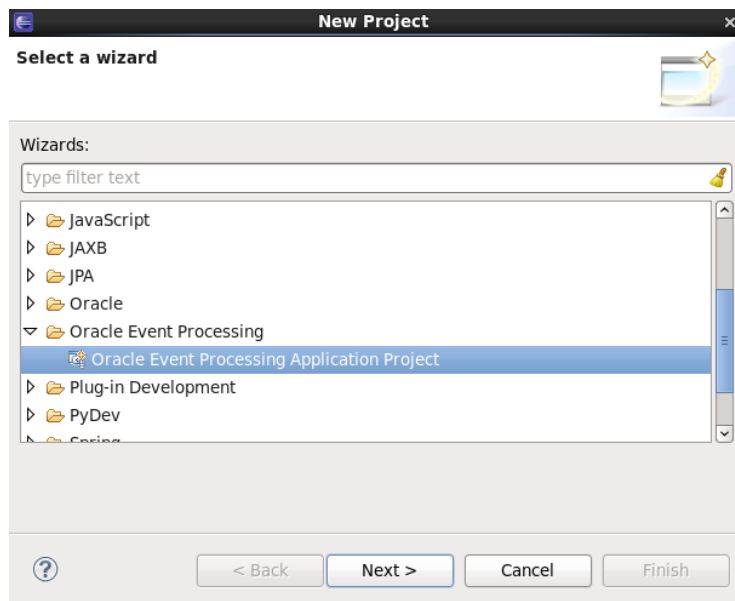


- b. If you are NOT using a pre-configured environment, you will need to create a simple weblogic server domain using port 7001 and execute the “configJMS” script to create the JMS queues on the WebLogic Server. The WLST files necessary can be found in the under the “wlst” folder of the simulator project.

## **Part 1: Creating a New Application**

In this exercise, you will create a new application from scratch in the Eclipse development environment.

1. Select File, New, Project. Open the Oracle Event Processing folder and choose “Oracle Event Processing Application Project”. Click Next.



2. Name the project: com.oracle.oep.training.smartmeter. Click Next.
3. The default “Bundle Properties” are fine. Click Next. Note that you can create a project based upon one of the sample templates. In this case, we’ll start from scratch, so click “Finish”. Even though we haven’t selected a template, we are still given the basic structure of an OEP application in the Eclipse workspace under Project Explorer.

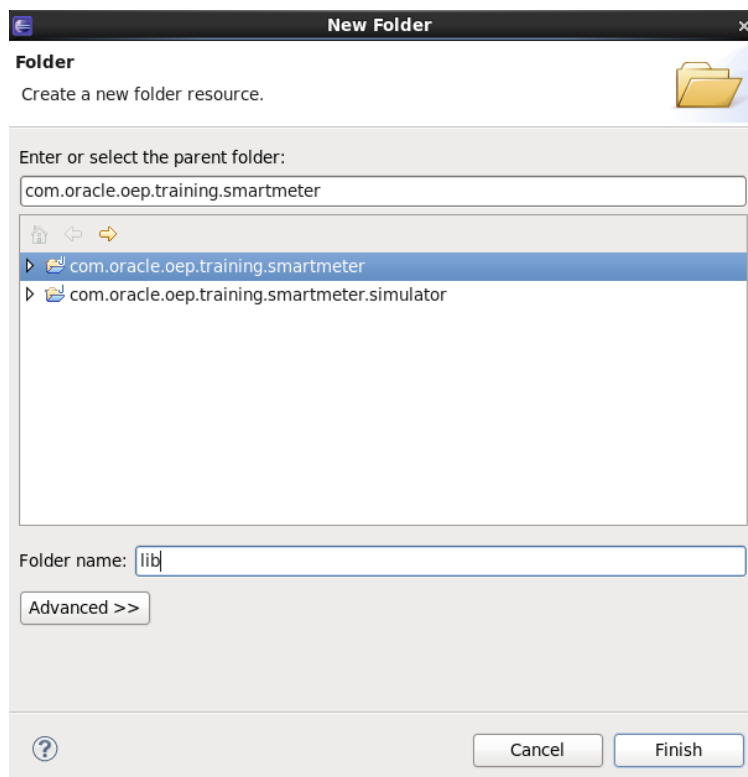
## **Part 2: Adding an Existing Java Library to an OEP Application**

## Tutorial: Oracle Event Processing – EPN Editor

(01\_OEP\_EPN\_Editor.docxx)

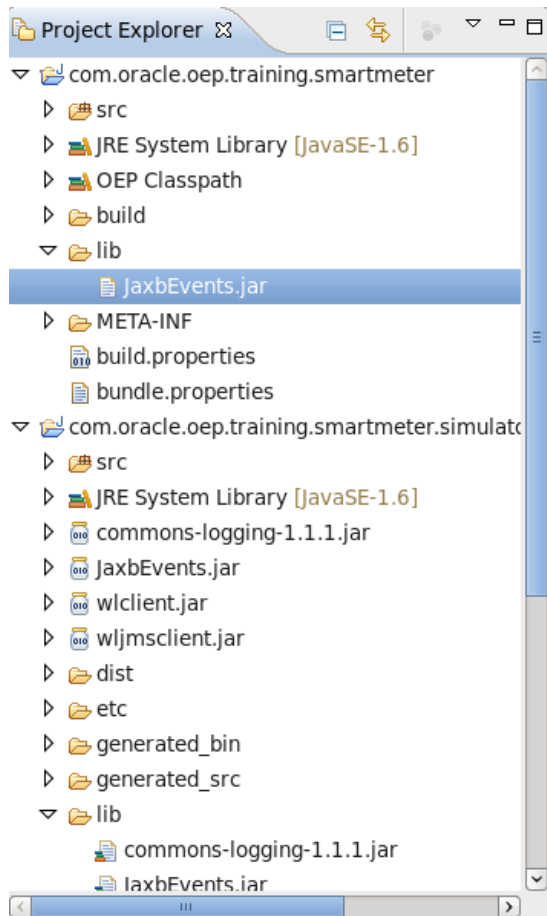
---

1. We need to create events for this project. We are going to use events that were created by compiling a schema using JAXB. The schema is located in the “com.oracle.oep.training.smartmeter.simulator” java project. The schema is called events.xsd in the schemas folder. The jaxb\_build.xml was created to compile the “events.xsd” schema and create a jar file called “JaxbEvents.jar” in the lib directory. This has already been done for you.
2. You should create a lib folder in your OEP application project, by right-clicking on the project name and selecting “New”, “Folder” and call it lib. By right-clicking, you can copy and paste the “JaxbEvents.jar” from the simulator project to your new lib folder in the OEP application.

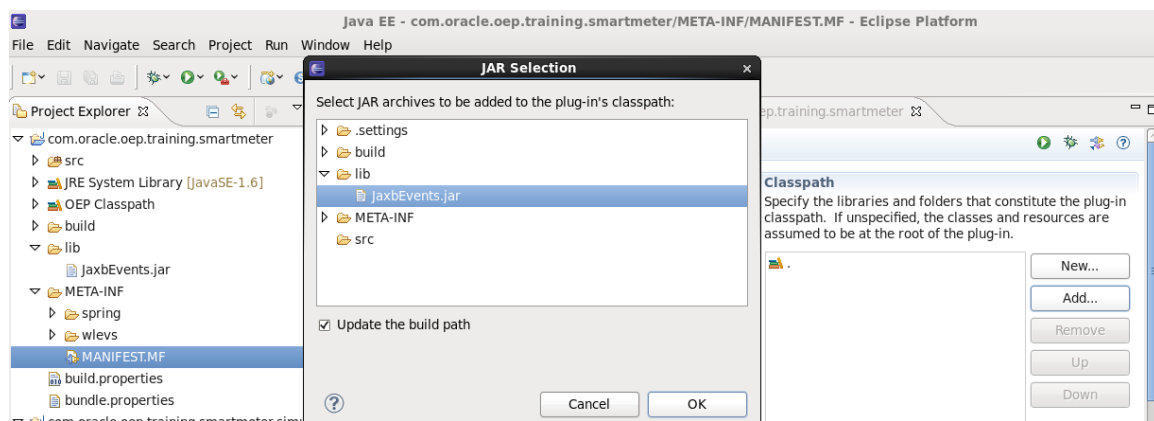


## Tutorial: Oracle Event Processing – EPN Editor

### (01\_OEP\_EPN\_Editor.docxx)



3. This jar is not yet part of the OEP application. To do that, we must add it to the project's MANIFEST.MF file under the META-INF folder. Eclipse provides some helpful tabs to make it easier for you to edit this file. Click on the “Runtime” tab and you’ll find a “Classpath” section on the top right-hand side. Use the “Add” button to add the jar to the application’s classpath. Make sure that you save the changes (CTRL+S). The \* in the name of the tab will go away when the changes have been saved.

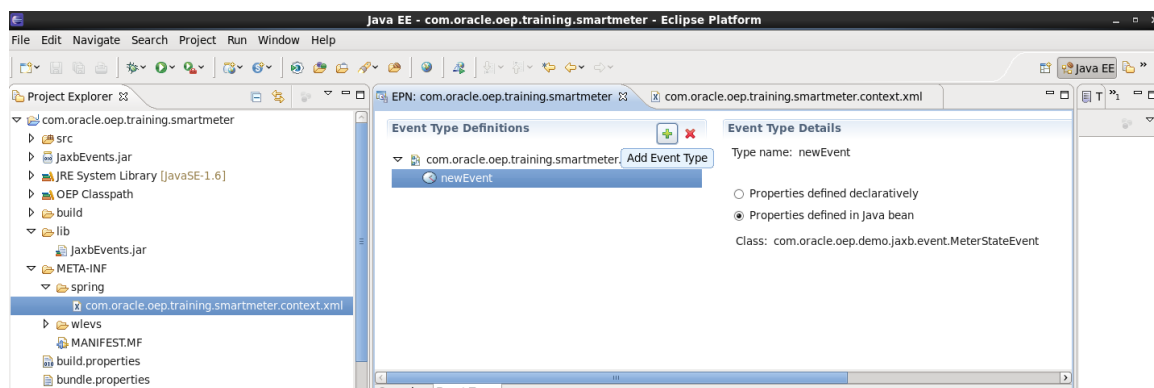




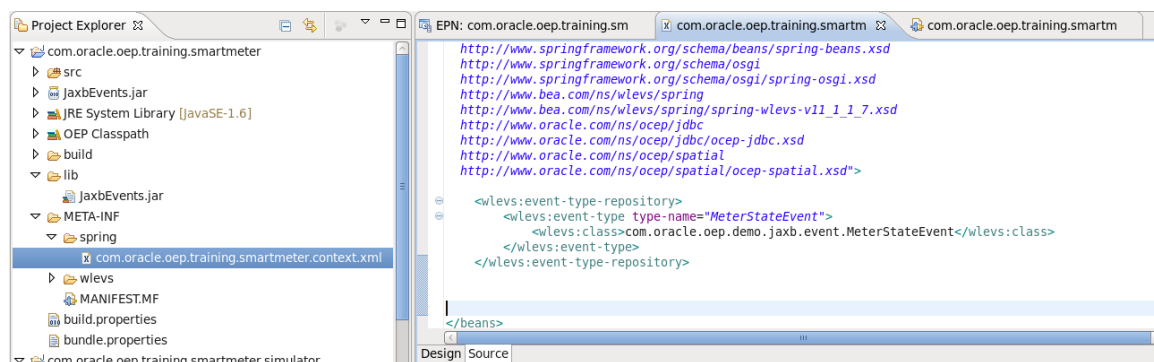
### Part 2: Defining an Event in the OEP Application

There are 2 ways to define an event in an OEP application. Either use a Java class or define the event declaratively in the OEP Spring XML Assembly File. In this example, we will define the event using a Java class. You don't always need to import an existing lib file as we have done in Part 1. You can easily create a simple POJO in the OEP application itself.

1. Go to the EPN Editor and look for the “Event Types” tab **at the bottom of the editor window**.
2. Click on the green “+” sign to add an event type.



3. Choose the radio button “Properties defined in Java bean”. Start typing “com.oracle.oep.demo.jaxb.event.MeterStateEvent”. It should start to appear in a list and you can simply choose it.
4. Change the name of the event to “MeterStateEvent” manually in the XML file in the “META-INF” spring folder. The XML looks as follows:

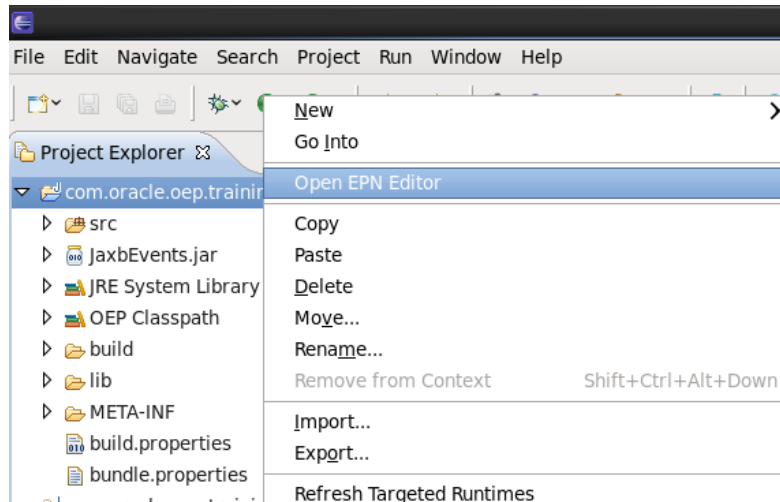


NOTE: The server will read every XML file in the META-INF “spring” and “wlevs” folder, if they have the .xml extension. You can group your application artifacts in any manner you wish.

### **Part 3: Creating a JMS Adapter**

Next, you will create a JMS adapter to receive messages from a JMS queue.

1. Switch back to the “Overview” tab of the EPN editor.



2. An empty EPN canvas should be open in the main window.
3. Right-click anywhere in the EPN Editor and select New, Adapter.
4. Give the Adapter the ID: MeterStateEventsAdapter
5. Set the provider as: jms-inbound
6. Select the “create a new file” radio button and allow the default name to be used.

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

---

**New Adapter**

**Basic Adapter information**  
Specify an ID, provider, and configuration location for the new adapter.

Adapter ID:

Type:

☒ Provider:

☐ Class:

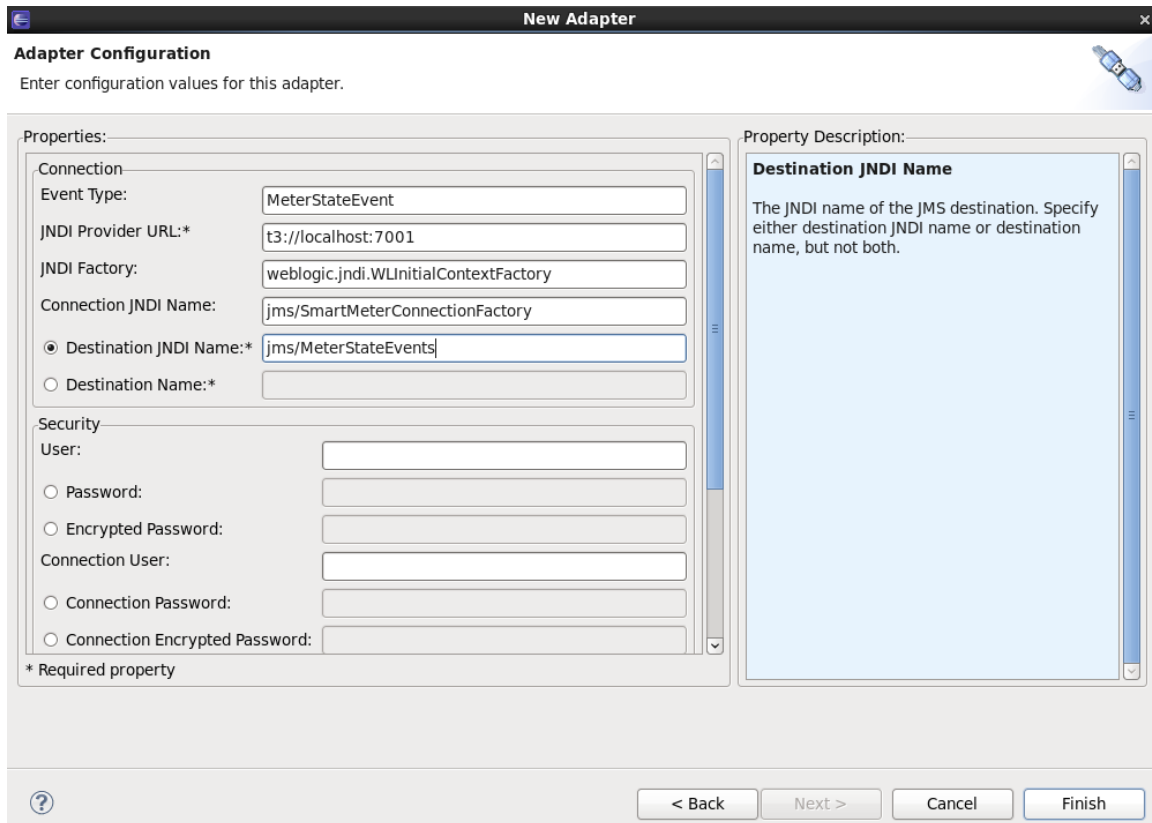
Configuration location:

☒ Create a new file:

☐ Use an existing configuration file:

7. Click Next.
8. Enter “MeterStateEvent” as the Event Type.
9. Enter “t3://localhost:7001” as the JNDI Provider URL.
10. Accept the default JNDI Factory.
11. Change the Connection JNDI Name to: “jms/SmartMeterConnectionFactory”
12. Select the “Destination JNDI Name” radio button and enter “jms/MeterStateEvents”.

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)



The 'New Adapter' dialog box is titled 'Adapter Configuration'. It contains two main sections: 'Properties' and 'Property Description'. The 'Properties' section is divided into 'Connection' and 'Security' sub-sections. The 'Connection' section includes fields for 'Event Type' (MeterStateEvent), 'JNDI Provider URL' (t3://localhost:7001), 'JNDI Factory' (weblogic.jndi.WLInitialContextFactory), 'Connection JNDI Name' (jms/SmartMeterConnectionFactory), and 'Destination JNDI Name' (jms/MeterStateEvents). The 'Security' section includes fields for 'User', 'Password', 'Encrypted Password', 'Connection User', 'Connection Password', and 'Connection Encrypted Password'. The 'Property Description' section provides a detailed explanation of the 'Destination JNDI Name' property, stating that it is the JNDI name of the JMS destination and should specify either the destination JNDI name or the destination name, but not both. The dialog box has a 'Back' button, a 'Next >' button, a 'Cancel' button, and a 'Finish' button.

**New Adapter**

**Adapter Configuration**  
Enter configuration values for this adapter.

**Properties:**

**Connection:**

Event Type: MeterStateEvent

JNDI Provider URL: t3://localhost:7001

JNDI Factory: weblogic.jndi.WLInitialContextFactory

Connection JNDI Name: jms/SmartMeterConnectionFactory

☒ Destination JNDI Name: jms/MeterStateEvents

☐ Destination Name:

**Security:**

User:

☐ Password:

☐ Encrypted Password:

Connection User:

☐ Connection Password:

☐ Connection Encrypted Password:

\* Required property

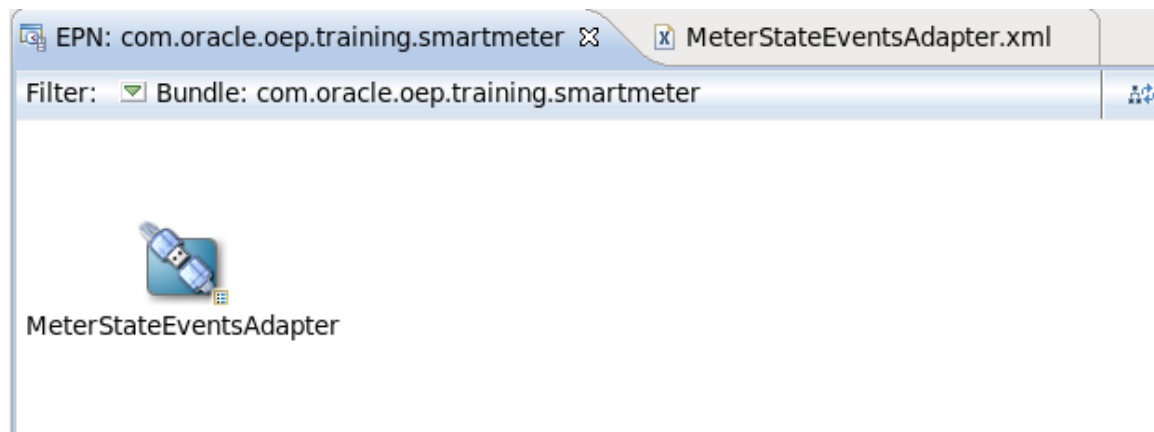
**Property Description:**

**Destination JNDI Name**

The JNDI name of the JMS destination. Specify either destination JNDI name or destination name, but not both.

< Back Next > Cancel Finish

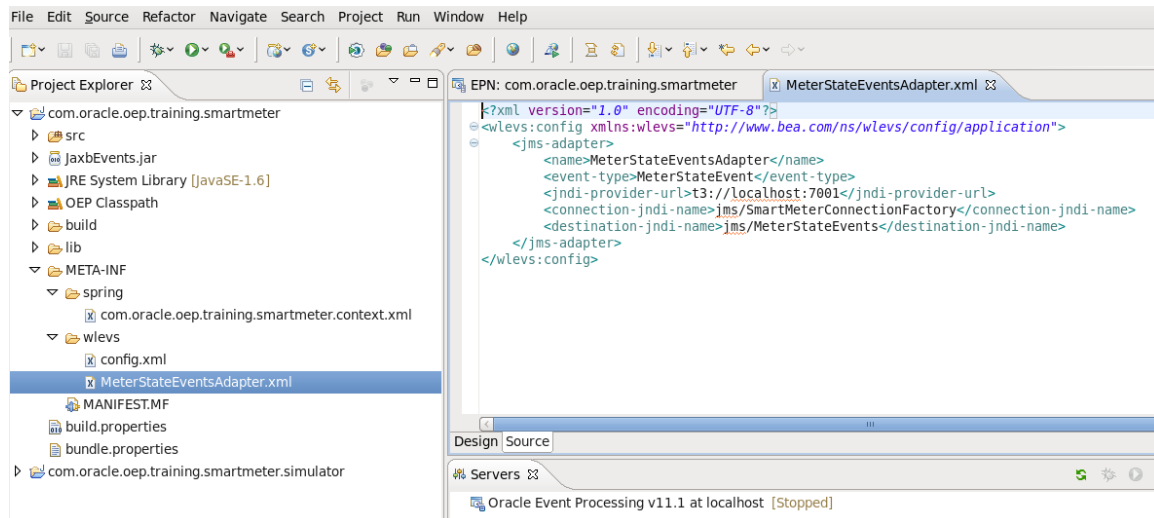
13. There is no need to enter any further information. Make sure that what you have entered looks like the picture above.
14. Click Finish.
15. Notice that an icon appears in the EPN Editor window to represent the newly created adapter.



## Tutorial: Oracle Event Processing – EPN Editor

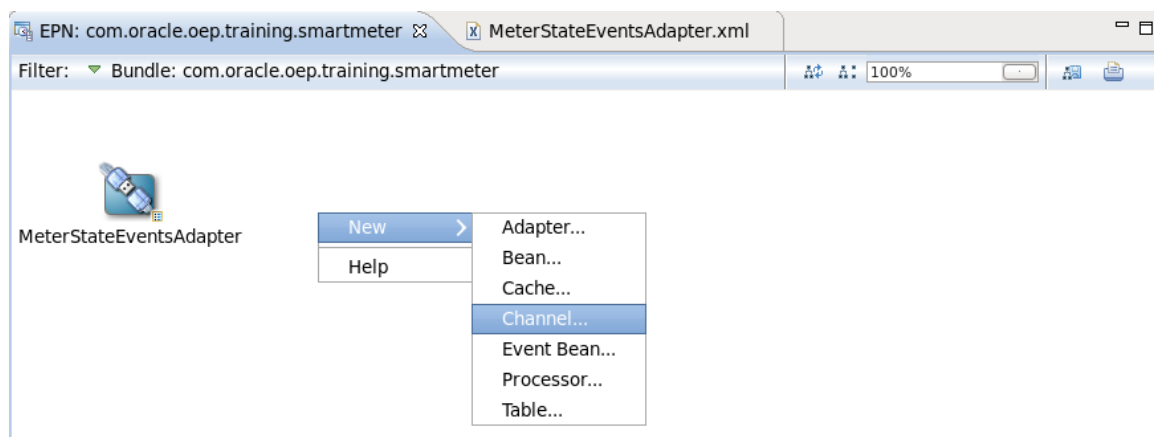
### (01\_OEP\_EPN\_Editor.docxx)

16. You should also notice that in addition to an entry that was made the XML file in the META-INF\spring directory, a configuration file was written to the META-INF\wlevs folder containing the configuration information that you entered on the second wizard window. If you made a mistake entering the configuration, you can manually edit this file.



## Part 4: Defining a Channel

1. Go back to the Overview Diagram of the EPN.
2. Right-click again in the EPN Editor and create a Channel. Rename it “MeterStateInputChannel”.

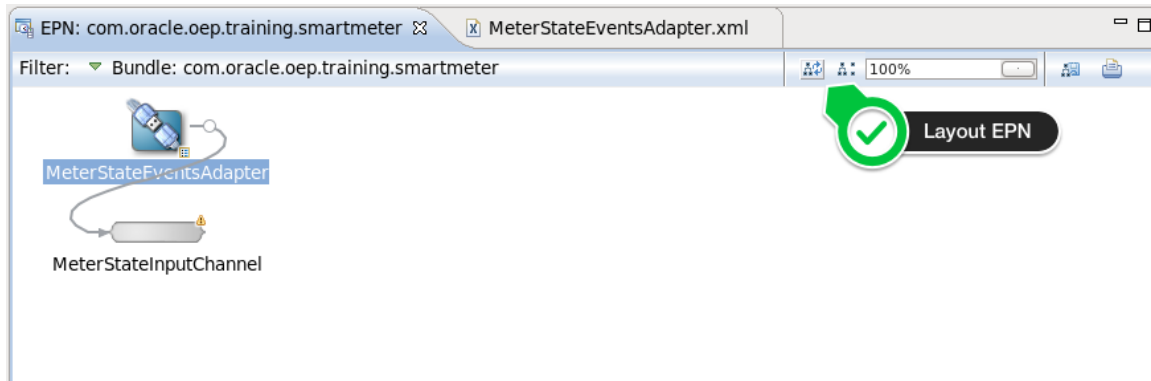


3. Click on the “MeterStateEventsAdapter” and hold down the mouse button and drag a line to the MeterStateInputChannel.

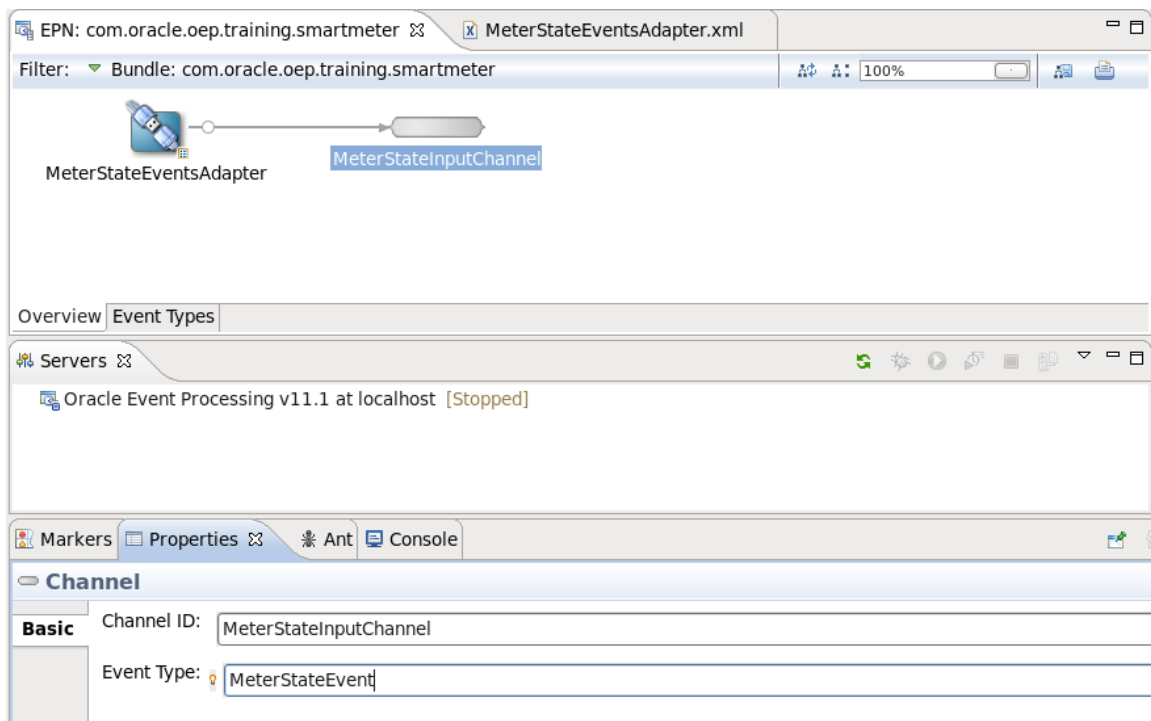
## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

---

4. Use the layout EPN button to adjust the layout of the EPN.



5. Click on the “MeterStateInputChannel” and find the “Properties” tab at the bottom middle section of the Eclipse IDE. You should see the Channel ID and an empty Event Type. Click on the Event Type input box and use CTRL+SPACE to see a list of events and select “MeterStateEvent”. You can also simply type “MeterStateEvent” in the box.



Be sure to save your changes.

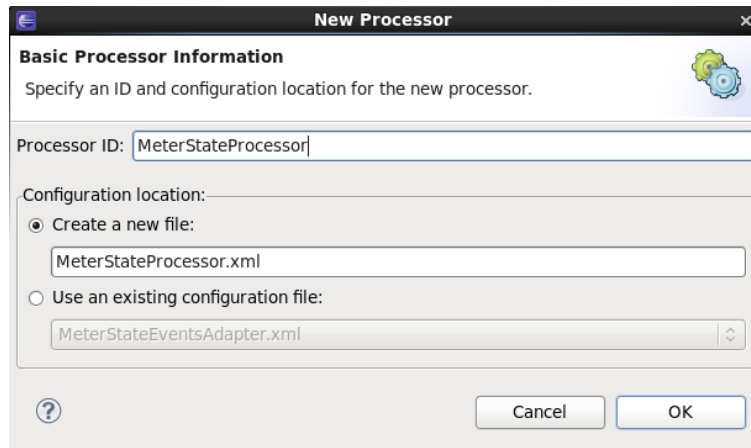
## Part 5: Creating a CQL Processor

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

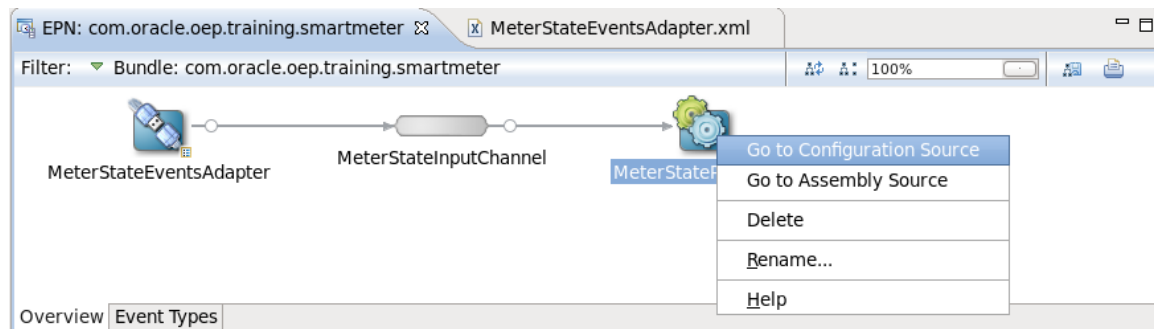
---

Next, we want a CQL processor so that we can use to execute complex event processing queries against the “MeterStateInputChannel”.

1. Go back to the “EPN Editor” canvas and right-click to create a new “Processor” and call it “MeterStateProcessor”. Click OK.



2. Join the input channel to the processor. Remember to hold down the mouse button on the channel and drag the mouse to the processor.
3. Click the icon to adjust the layout of the EPN.
4. Right-click the processor icon and select “Go to Configuration Source”.



5. By default, you are provided with a sample XML file. NOTE: You can find this file in the project directory structure under “META-INF\wlevs”.
6. Replace the sample <rules> section with the following rules:  

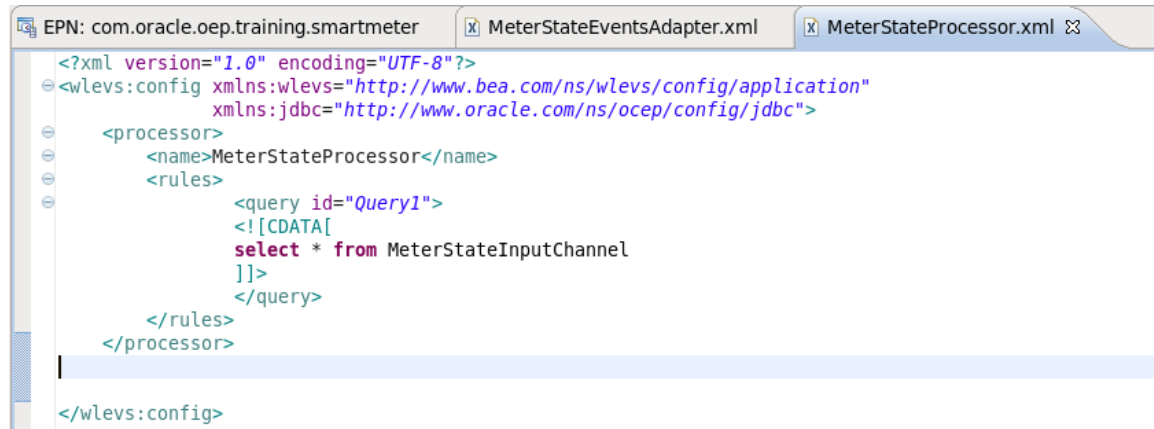
```
<rules>
<query id="Query1">
<![CDATA[ select * from MeterStateInputChannel
]]>
</query>
```

## Tutorial: Oracle Event Processing – EPN Editor

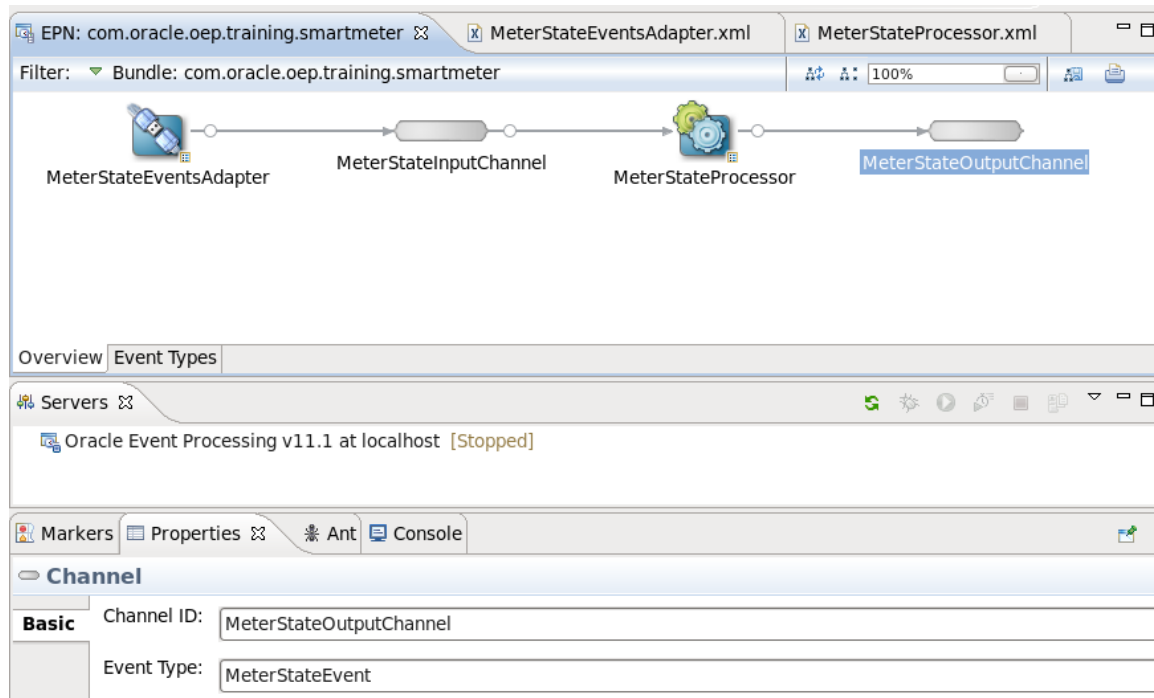
(01\_OEP\_EPN\_Editor.docxx)

```
</rules>
```

- This query will simply select all events that are received on the MeterStateInputChannel. You will learn how to create more complex queries in the CQL tutorial.



- Next in the EPN Editor create another channel called "MeterStateOutputChannel".
- Join the processor to the channel visually in the EPN Editor.
- Just like you did for the input channel, specify an event-type="MeterStateEvent" for the output channel.

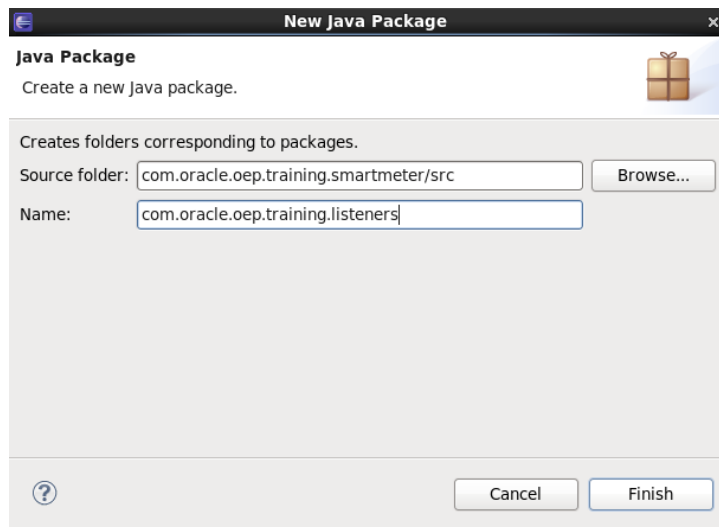




## **Part 5: Creating an Event Listener Bean**

Next, let's create an event bean to see the output of our query.

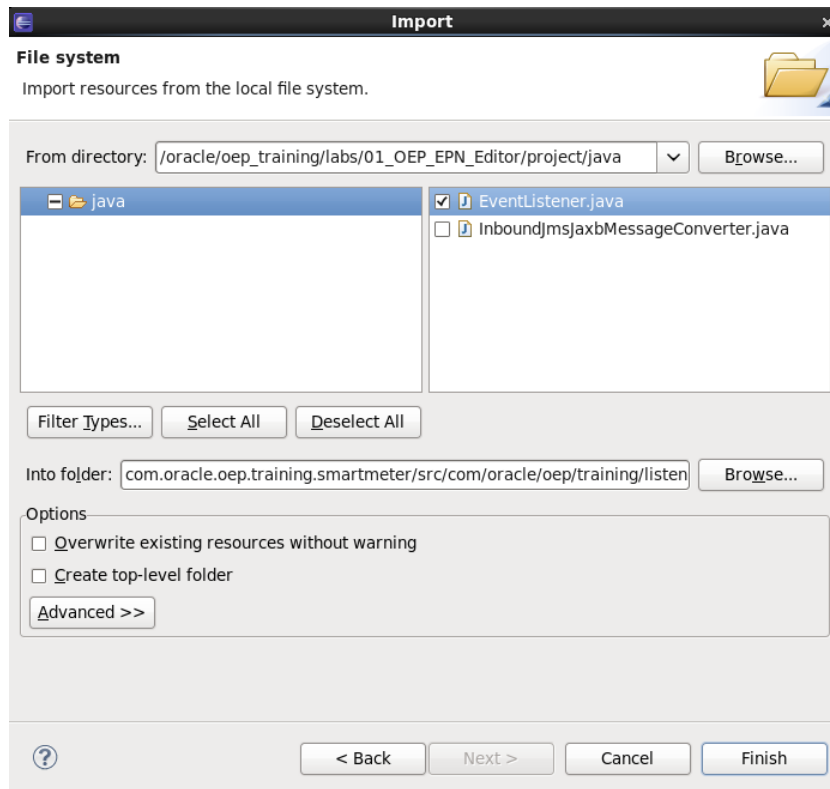
1. In the Project Explorer, locate the “src” folder. Right-click and create a new “Package” called: com.oracle.oep.training.listeners



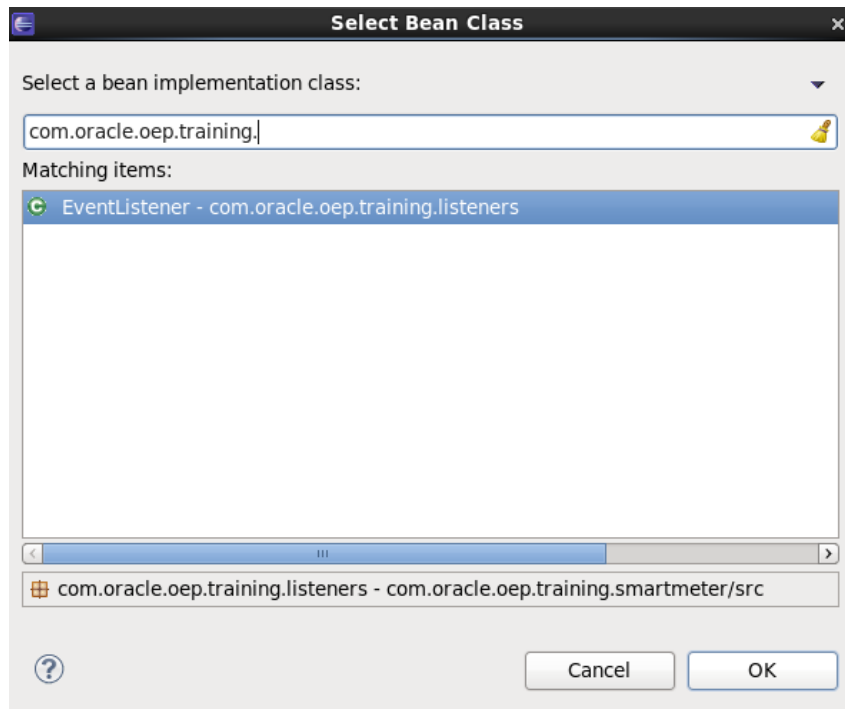
2. Click Finish.
3. Click on the name of the package in the “Project Explorer”. Right-click and select “Import”. Choose “General”, “File System” and select “Next”.
4. Browse to the location “<OEP\_TRAINING>\labs\01\_OEP\_EPN\_Editor\project\java” folder and Click OK.
5. Select: “Event Listener.java”. Make sure that the only the checkbox for this file is checked. Click “Finish”.

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

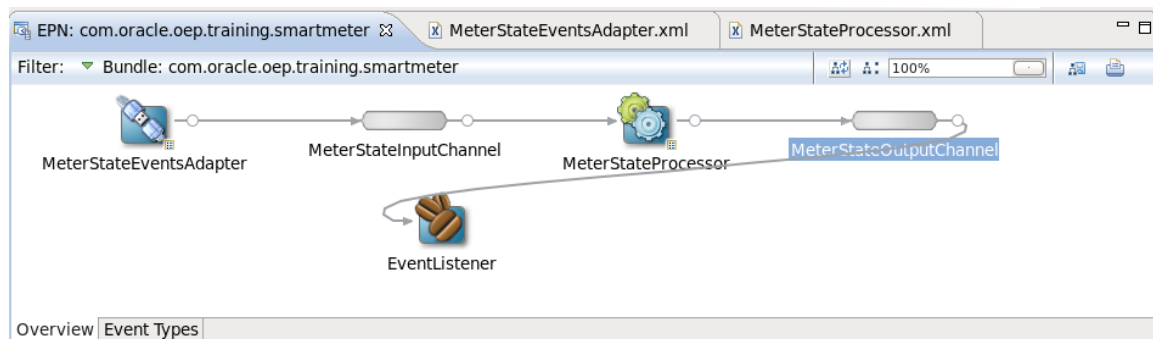
---



6. Go back to the “EPN Editor” canvas and right-click to create a new “Bean”.
7. Enter “com.oracle.cep.training.” as the bean implementation class. You’ll be able to select “EventListener” after typing part of the name of the package. Select it and click OK.



6. Connect the “MeterStateOutputChannel” to the “EventListener”.



7. Use the “Layout EPN” icon, to line everything up again.

NOTE: Sometimes it’s helpful to double-click on the EPN Editor tab to see a larger view of the window. Double-click again to put the view back to the original size.

## **Part 6: JMS Adapter Converter Bean**

We have a complete end-to-end EPN Assembly, but we need to add a converter to convert incoming messages from XML to POJOs (plain-old-java-objects).

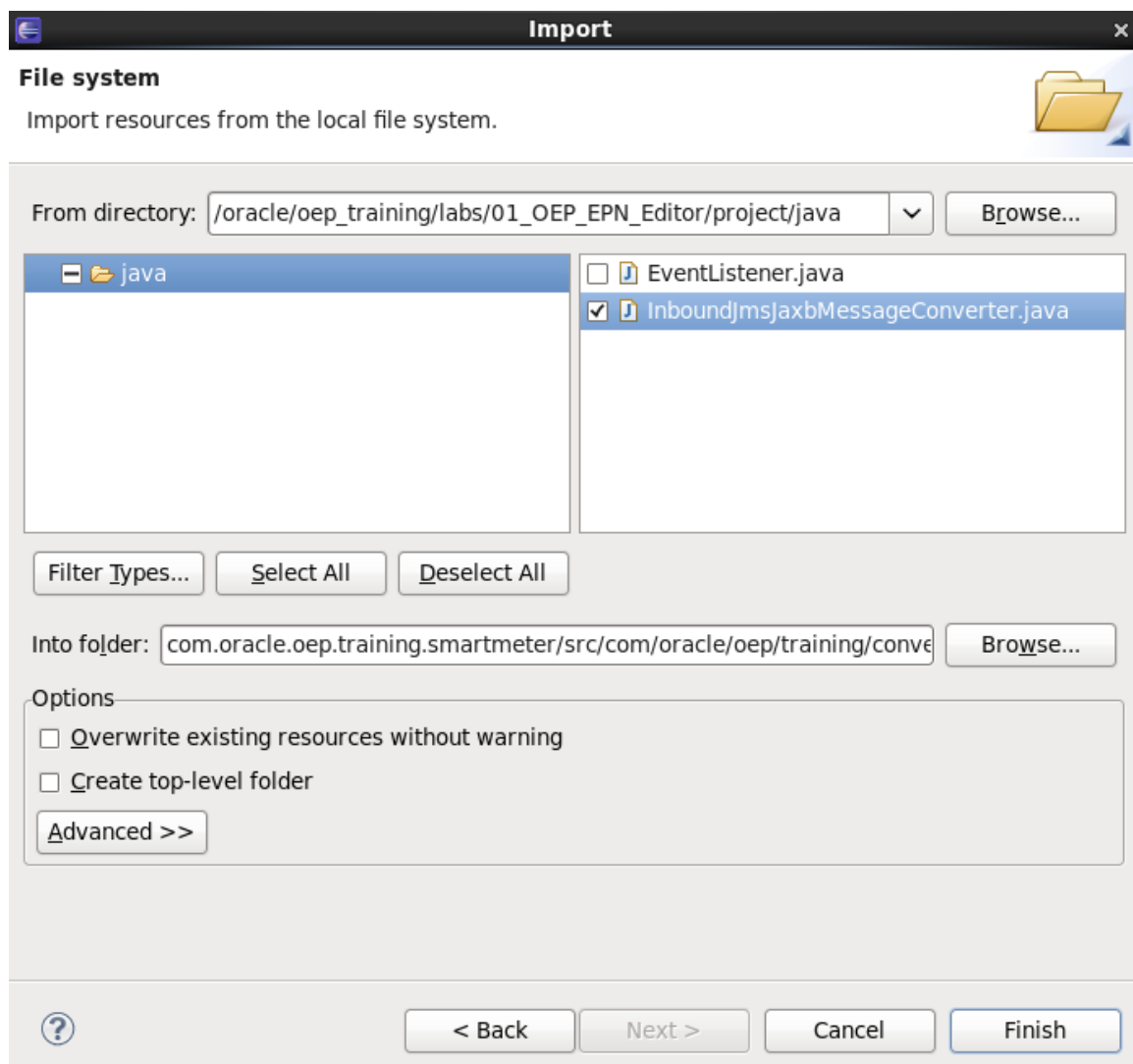
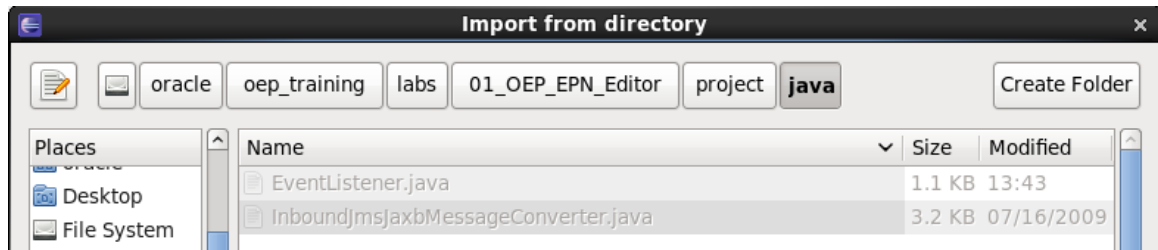
1. In the Project Explorer, locate the “src” folder again. Right-click and create a new “Package” called: com.oracle.oep.training.converter.

## Tutorial: Oracle Event Processing – EPN Editor

(01\_OEP\_EPN\_Editor.docxx)

---

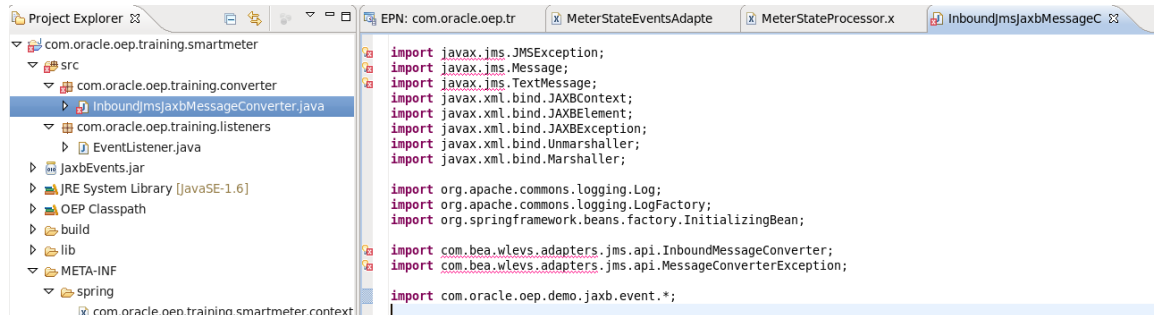
- Just like you did in above for the event sink, import “InboundJmsJaxbMessageConverter.java” from “<OEP\_TRAINING>\labs\01\_EPN\_Editor\project\java” into the new package. Make sure you only select the one file.



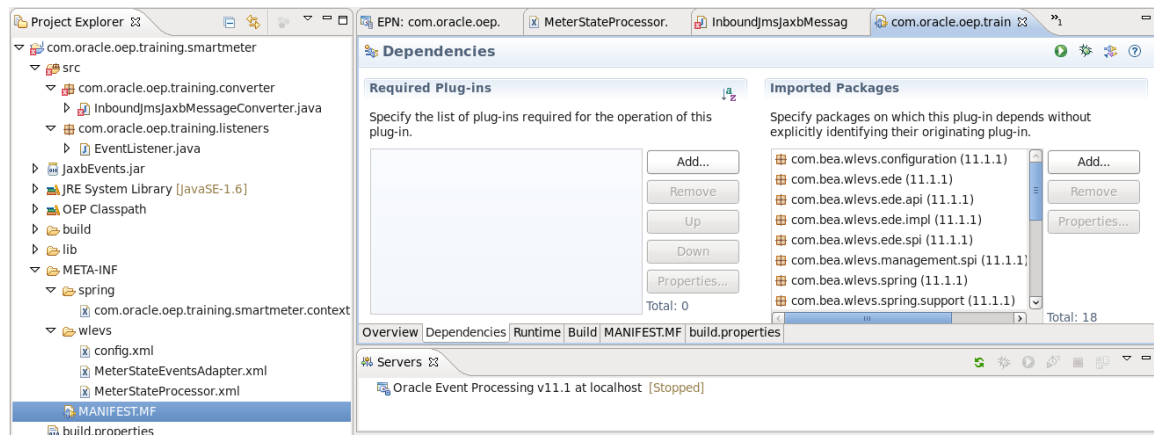
## Tutorial: Oracle Event Processing – EPN Editor

### (01\_OEP\_EPN\_Editor.docxx)

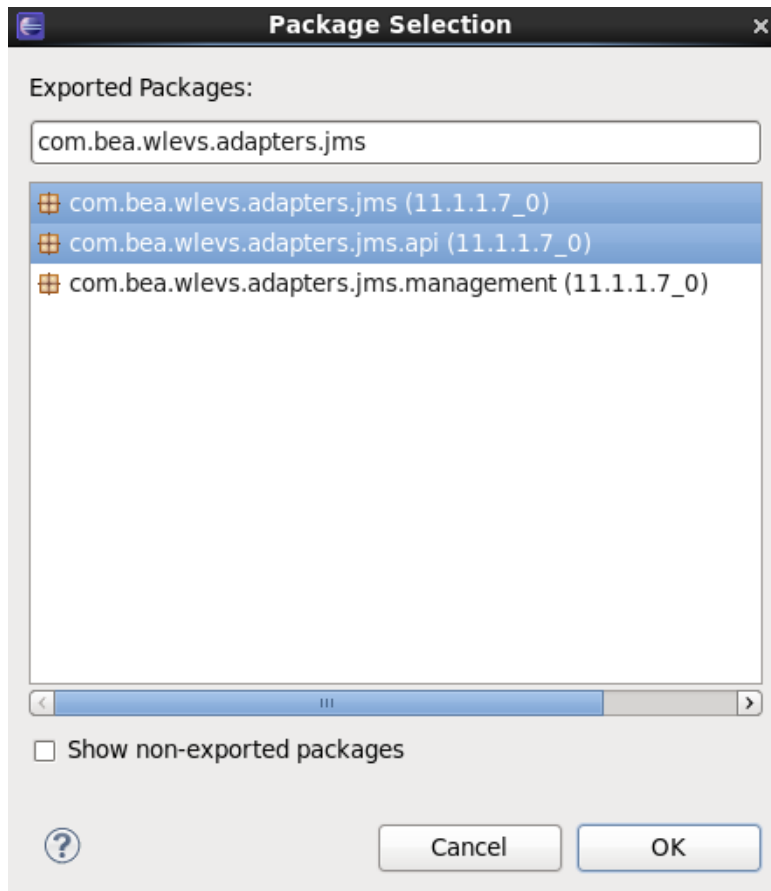
3. You will get errors when you import this file. This is because this class uses other classes that are not yet a part of the OEP application. In this case, they are a part of the server's classpath. You simply need to modify the “MANIFEST.MF” file to add them to your application's classpath.



4. Double-click on “MANIFEST.MF” under “META-INF”. This time click on the “Dependencies” tab. You will see “Imported Packages” in the top right-hand side.



5. Click the “Add” button”.
6. Start entering “com.bea.wlevs.adapters.jms”. The list should get shorter.
7. Select both “com.bea.wlevs.adapters.jms” and “com.bea.wlevs.adapters.jms.api”. You may need to hold down the “control” key to select more than 1 entry at a time. Click OK.



8. We need some more packages. Use the same procedure to add the following:

```
com.sun.xml.bind.v2
javax.jms
javax.xml.bind
javax.xml.bind.annotation
javax.xml.bind.annotation.adapters
javax.xml.bind.attachment
javax.xml.bind.helpers
javax.xml.bind.util
javax.xml.stream
```

9. After you save the changes the compilation problems should disappear.
10. We need to add this converter to the JMS configuration in the EPN assembly file. Click on the icon for the “MeterStateEventsAdapter” and select “Go to Assembly Source”.
11. Above the “<wlevs:adapter>” entry add the following “<bean>” declaration. Cut and Paste or Use CTRL+Space to help you create it manually.

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

---

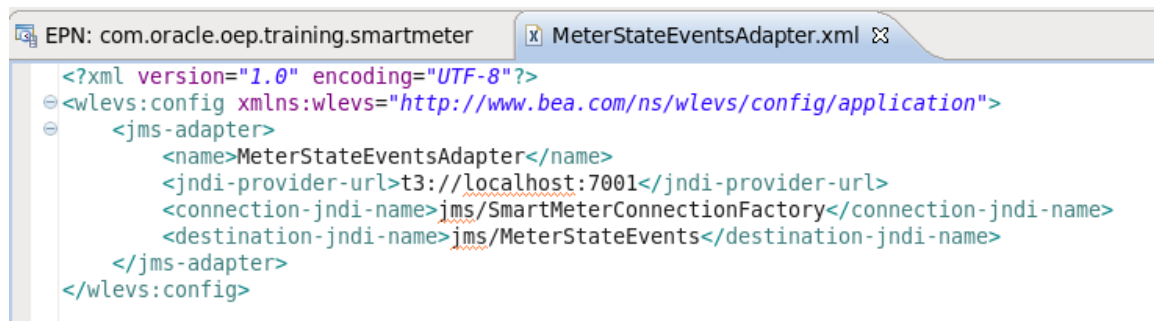
```
<bean id="JmsJaxbMessageConverter"  
class="com.oracle.oep.training.converter.InboundJmsJaxbMessageConverter"  
/>
```

12. Add the following “instance-property” to the adapter configuraton.

```
<wlevs:instance-property name="converterBean"  
ref="JmsJaxbMessageConverter"/>
```



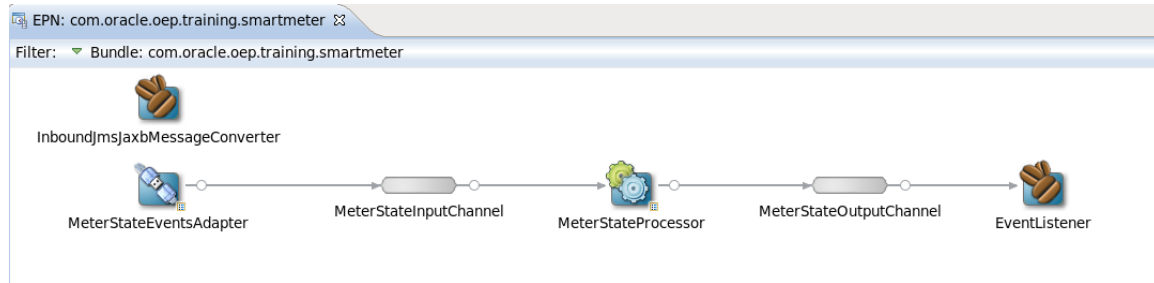
13. Since we are using a converter bean for the JMS Adapter, we do not need to specify an event type in the adapter configuration. Right-click on the adapter and select “Go to Configuration Source”. Remove the line showing the event type. The adapter’s configuration XML should look like this:



## Part 7: Deploying the Application

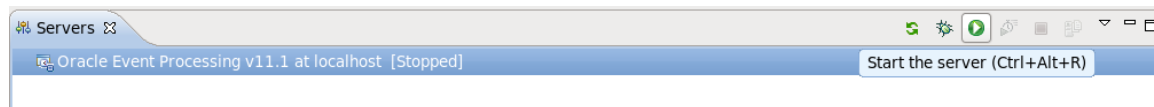
Finally, we are ready to deploy the application to the server and test it by running the JMS simulator. Make sure that you have saved all your work. Your EPN should look like this:

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

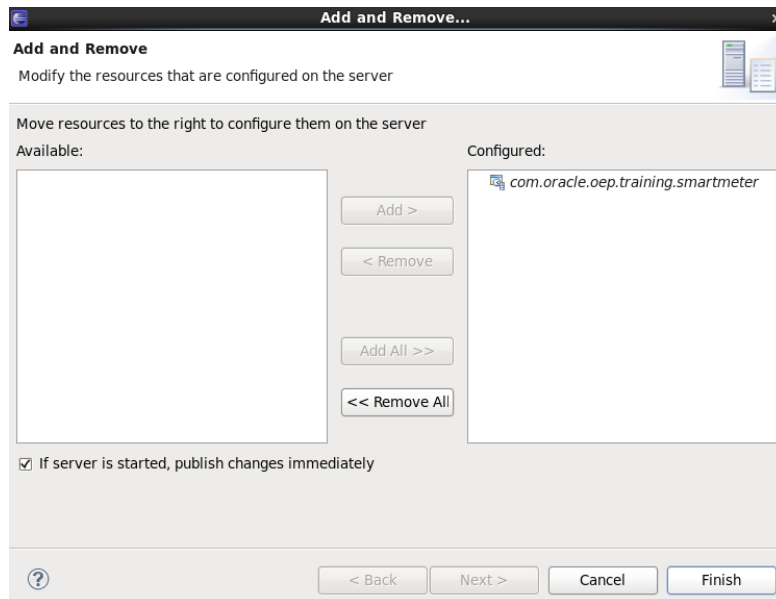


Since this application is using JMS queues that are running on WebLogic Server, start the WebLogic server that has the queues. On the virtual machine, there is a “startWebLogic.sh” script on the desktop. You can double-click it and choose “Run In Terminal”. You’ll know that it has finished starting when you see the message: <Server started in RUNNING mode>.

1. Click on “Oracle Event Processing v11.1” in the “Servers” tab.



2. Push the green start arrow OR right-click and select “Start”.  
You should eventually see a “server started” message in the console:  
<Server STARTED>
3. Right-click and select “Add and Remove”.





4. Use the arrow to add the “com.oracle.oep.training.smartmeter” project to the server and click “Finish”.
5. If you have done everything successfully, you will see a messages indicating that the application bundle was “deployed successfully” and another that it was started successfully.

```
<Notice> <Deployment> <BEA-2045000> <The application bundle  
"com.oracle.oep.training.smartmeter" was deployed successfully to  
file:/oracle/oephome/user_projects/domains/ocep_domain/defaultserver/applica  
tions/com.oracle.oep.training.smartmeter/com.oracle.oep.training.smartmeter.  
jar with version 1388788690643>
```

```
<Notice> <Spring> <BEA-2047000> <The application context for  
"com.oracle.oep.training.smartmeter" was started successfully>
```

6. If there is a problem with your project, the application will automatically un-deploy.
7. You will get an error if you try to remove it from the server stating that it does not exist on the server. This is OK.
8. Fix any errors and re-deploy the application again.
9. If you get an error saying the destination is unreachable, the WebLogic server hosting your JMS queues is not started or there is a problem with your JMS configuration. The JMS server must be started before the OEP application can be deployed successfully.

```
at weblogic.wen.WenExceptionImpl.<init>(WebExceptionImpl.java:415)  
Caused by: java.net.ConnectException: t3://localhost:7001: Destination unreachable; nested exception is:  
java.net.ConnectException: Connection refused: connect; No available router to destination  
at weblogic.rjvm.RJVMFinder.findOrCreateInternal(RJVMFinder.java:216)
```

10. When you are ready to try again with a corrected project, you either removing the project and adding it again or using the “force publish” feature. You may get an error message removing the project, stating that the project is not on the server. This is OK. This only happened because the application could not be deployed on the server, but it was not removed from the IDE.

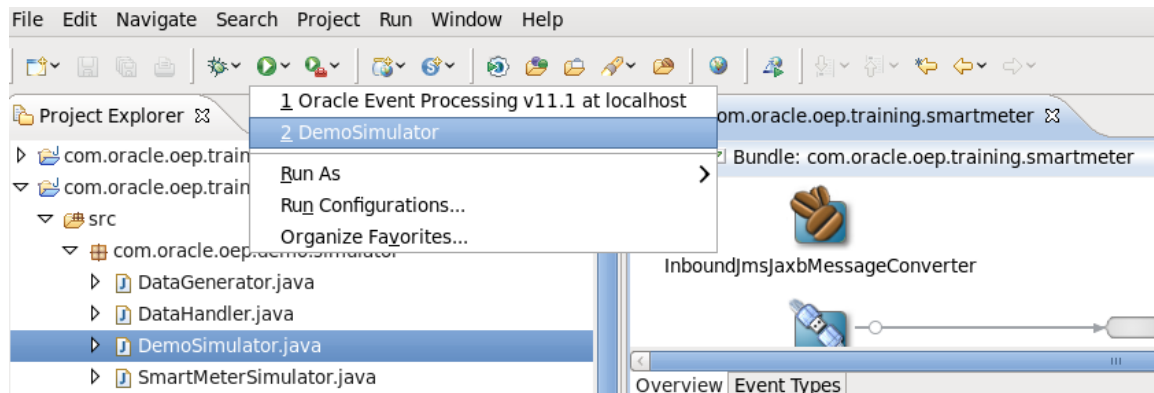
## **Part 8: Testing**

To test, we simply run the JMS simulator.

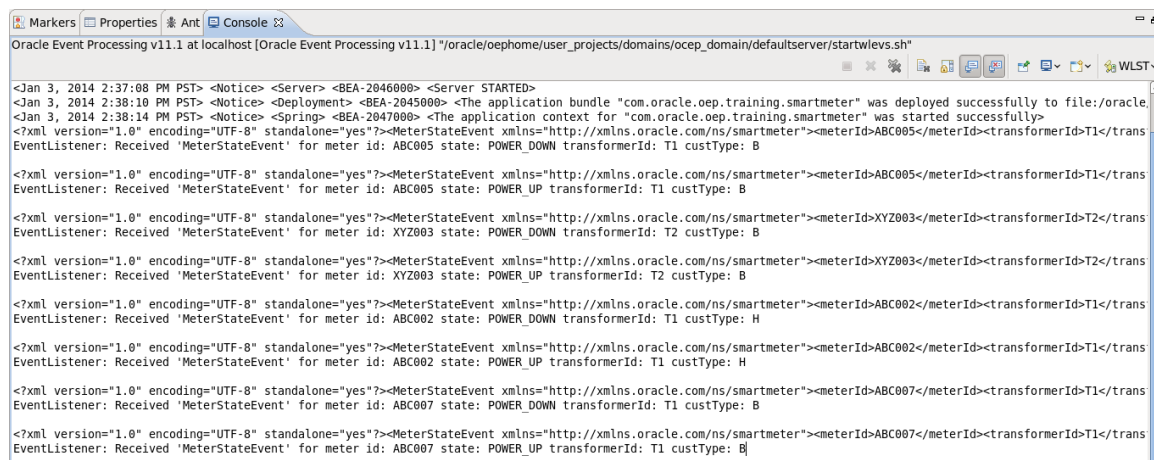
1. The **DemoSimulator.java** program from  
“com.oracle.oep.training.smartmeter.simulator” project in the  
“com.oracle.oep.demo.simulator” package generates sample events.
2. It has been configured to run in Eclipse.
- 3.

## Tutorial: Oracle Event Processing – EPN Editor (01\_OEP\_EPN\_Editor.docxx)

4. Choose “Demo Simulator” from the run configurations menu.



5. You should see some messages that are showing the receipt of the JMS XML Message and the output of the message written to the console in the Event Bean.



### Summary:

Now that you have completed these exercises, you understand how to:

- Create an OEP application using the OEP tools for Eclipse
- Configure a JMS Adapter with a converter bean
- Transform JMS Text Messages using JAXB
- Create a CQL Processor
- Create an Event Bean and use it as an event listener