|  |
| --- |
|  |

Oracle Complex Event Processing:

Tutorial: Custom Adapter

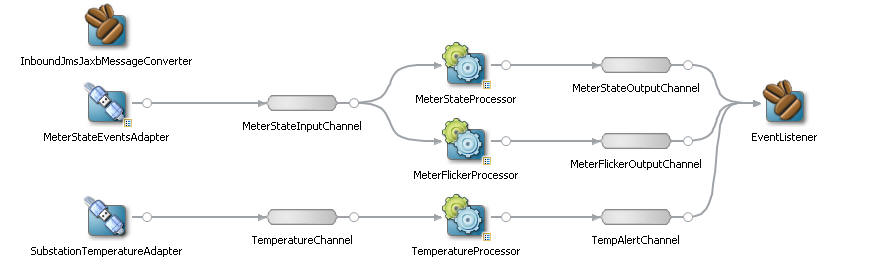
An Oracle Tutorial

Updated December 2013

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

**Objectives:**

This tutorial illustrates how to create your own custom adapter for your Oracle Event Processing application. If this example, you will create a custom socket adapter that connects to a TCP/IP socket and receives data.



***Table of Contents***

[Set-up: 4](#_Toc236636371)

[Part 1: Adding a New Custom Adapter 4](#_Toc236636372)

[Part 2: Creating a Channel 8](#_Toc236636373)

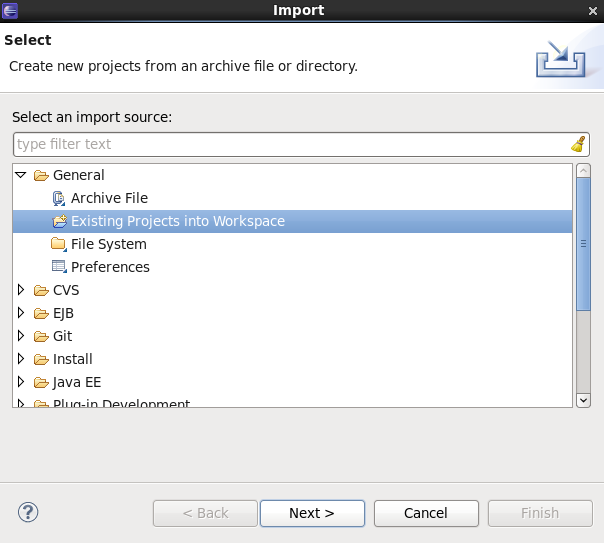
[Part 3: Add a Temperature Processor 10](#_Toc236636374)

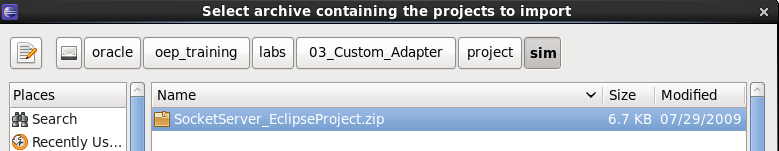
[Part 4: Deploying the Application 12](#_Toc236636375)

[Part 5: Testing 13](#_Toc236636376)

##### Set-up:

1. This tutorial is a continuation of tutorial 02\_CQL\_Processor. You must have an environment created to do that tutorial and complete the tutorial or import the solution.
2. From the File menu, use the “Import” option to import a TCP/IP socket server java project that will be used in this tutorial. Be sure to select “General”, “Existing Projects into Workspace”. The project that you want to import can be found in “<OEP\_TRAINING>\labs\03\_Custom\_Adapter\project\sim\SocketServer\_EclipseProject.zip”. This is the simulator that will open a server socket TCP/IP connection and send substation temperature events to clients that connect to it. The OEP application will be the TCP/IP client application.

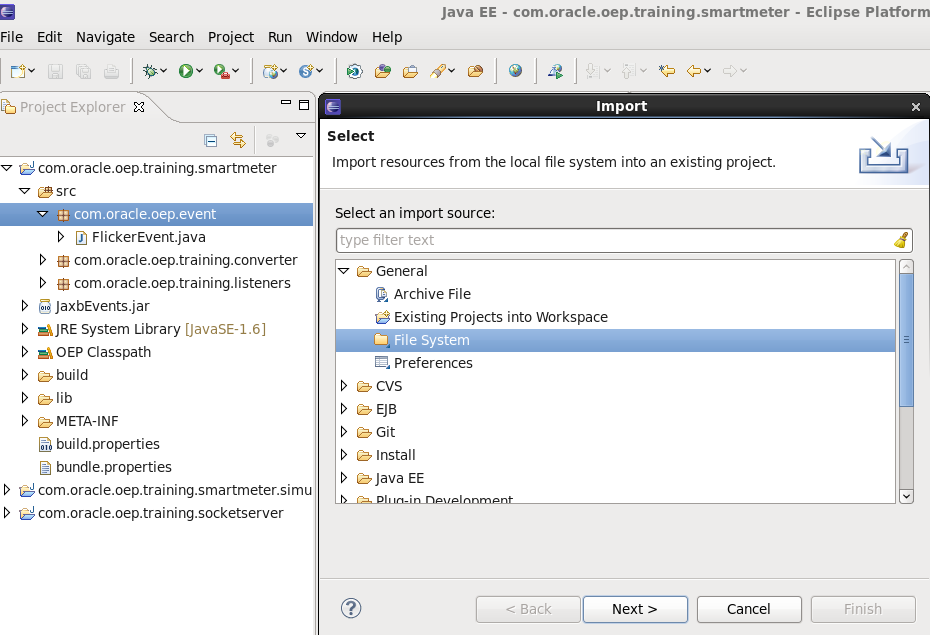


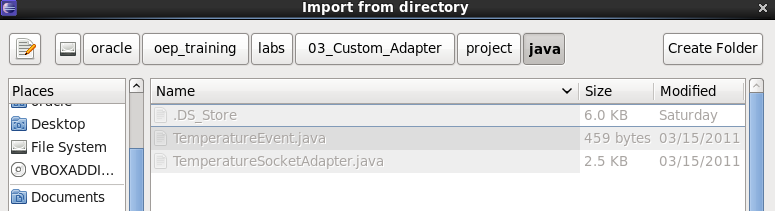


##### Part 1: Adding a New Custom Adapter

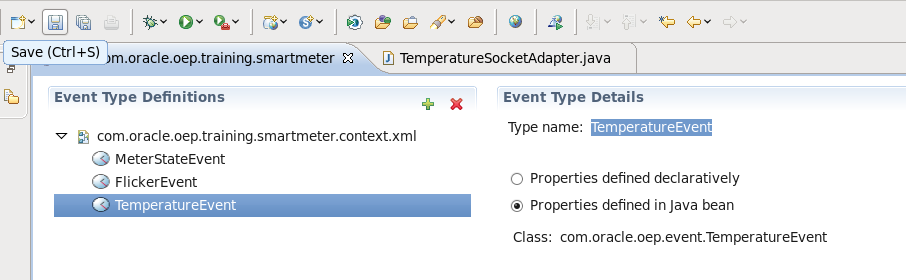
In this exercise, you will add a custom adapter that uses a java class instead of an out-of-the-box provider.

1. Click on the “com.oracle.oep.event” package under “src”. Import a new event called “TemperatureEvent” by using File, Import, File System and selecting the file “<OEP\_TRAINING>\03\_Custom\_Adapter\project\java\TemperatureEvent.java”. Click Finish.

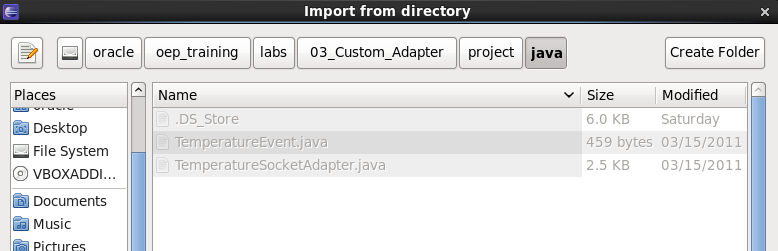


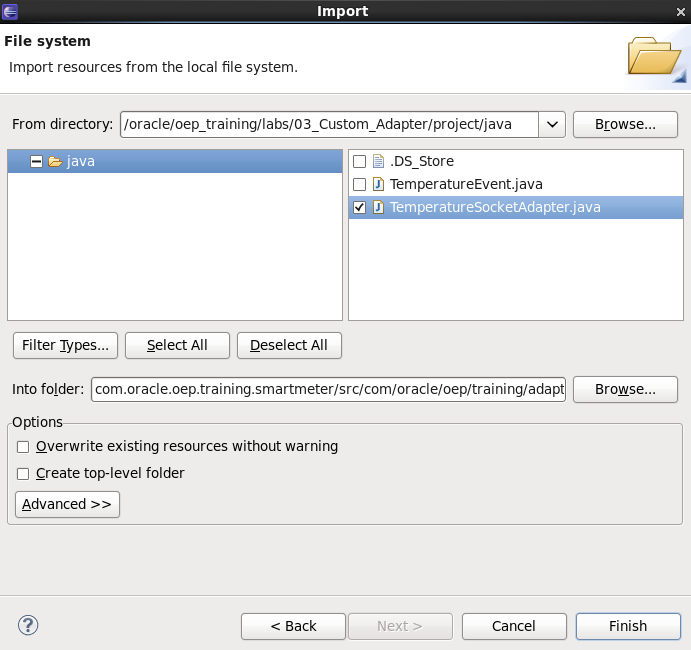


1. Add the event to the event-type repository. From the EPN editor click on the “EventTypes” lower tab.
2. Click on the green plus sign to add a new event.
3. Supply the Event Type name “TemperatureEvent” on the right-hand side under “Event Type Details” where it says “Type Name:” (It doesn’t look like you can type there, but you can).
4. Select the radio button corresponding to “Properties supplied in a Java Bean”. Supply the class name “com.oracle.oep.event.TemperatureEvent”.
5. When you save the changes, the IDE will update the event name in the list on the left-hand side.

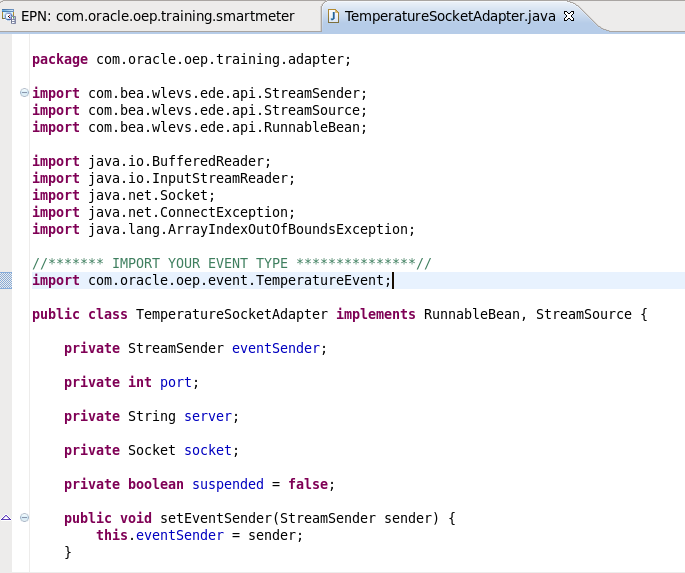


1. Under the “src” folder create a new package called: “com.oracle.oep.training.adapter”. Click Finish.
2. Import the source code for the adapter by clicking on the newly created package and using File, Import, ‘General’, ‘File System’ and select the file “<CEP\_TRAINING>\104-Custom-Adapter\java\TemperatureSocketAdapter.java”. Click Finish.

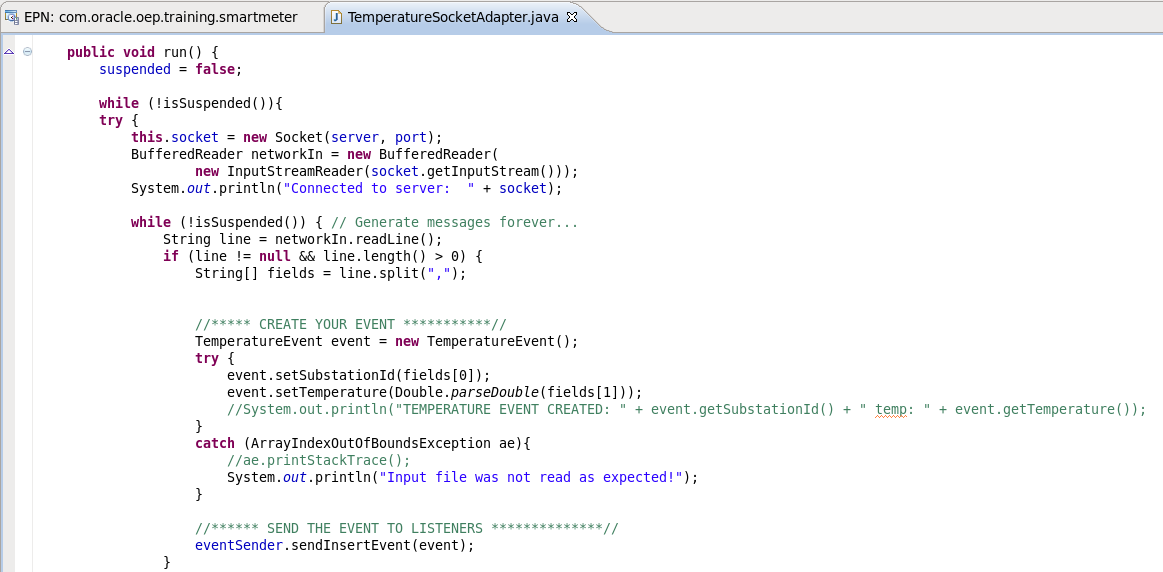




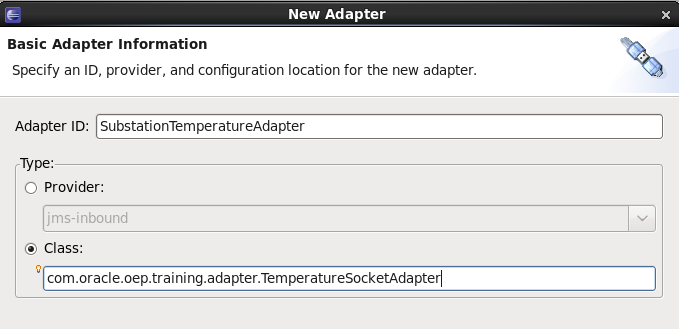
1. Notice in the “TemperatureSocketAdapter” that we import the new event at the top in the import section (you may need to use the + sign to expand the list).



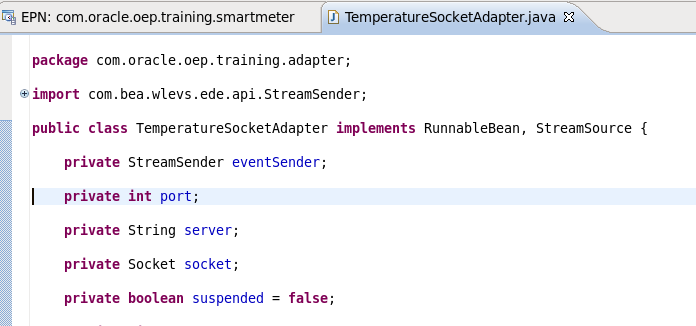
1. Also, notice in the run() method how the temperature event is instantiated and how it is populated. Since this is a custom adapter that we are specifically developing for our own input, we know the layout of the data and how it is delineated with a comma “,”.



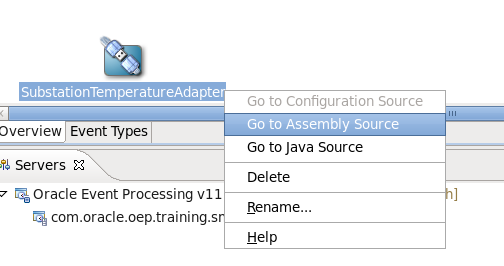
1. Another important thing to notice is the API to send the events to all of the listeners defined in the EPN: *eventSender.sendInsertEvent(event)*
2. On the EPN canvas, right-click and create a new adapter called “SubstationTemperatureAdapter”.
3. This time instead of using the provider radio button, choose Class and enter the class name of your custom adapter and click “Finish”.

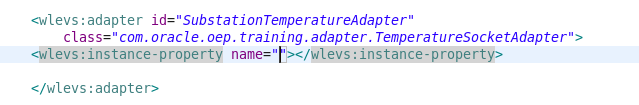


1. Notice that this adapter defines the properties “port” and “server”.

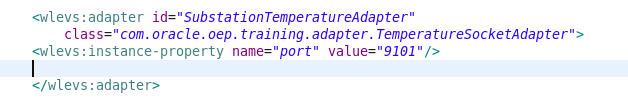


1. You set these in the EPN Assembly file where you define the adapter (using the Spring framework). Click between the starting and ending XML tags for the adapter and use CTRL+SPACE and select “instance-property”.

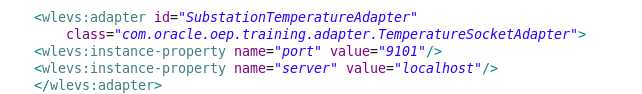




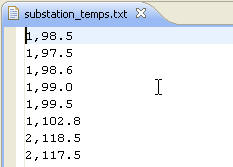
1. Set the name to: **port** and use CTRL+SPACE after that and choose “value” and set that to: **9101** (9101 corresponds to the port number that our demo socket server will open to accept client connections). If this port number is in use on your machine, you can choose an available port number. Make sure that you change the port number in the “demo.properties” file of the socket server.



1. Follow the same procedure to create an instance-property called: **server**
2. Set the value of server to: **localhost**
3. If you’d like, adjust the formatting of the XML. It should look similar to this:



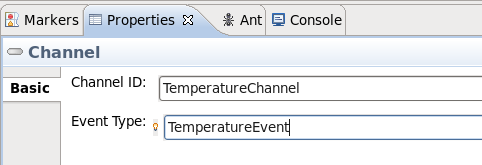
1. Look in the “com.oracle.oep.training.socketserver” project that you imported. The “data” folder has the data file that you will be reading and supplying to your application.



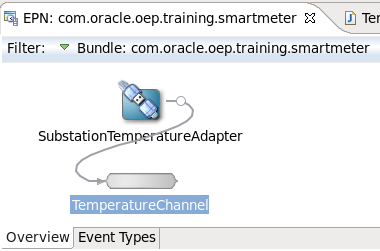
##### Part 2: Creating a Channel

Next, you will need to create a channel.

1. Go back to the EPN Editor and right-click to select new “Channel”. Name the channel “TemperatureChannel” and supply the event type “Temperature Event” in the “Properties” tab.



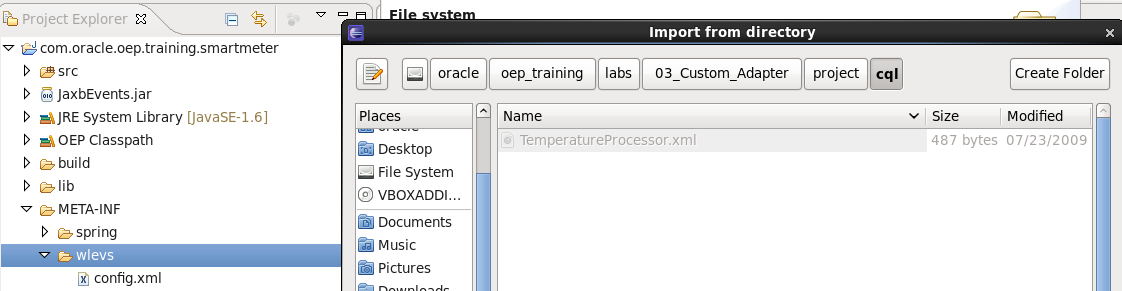
1. Connect the adapter that you just created to the new “TemperatureChannel”.

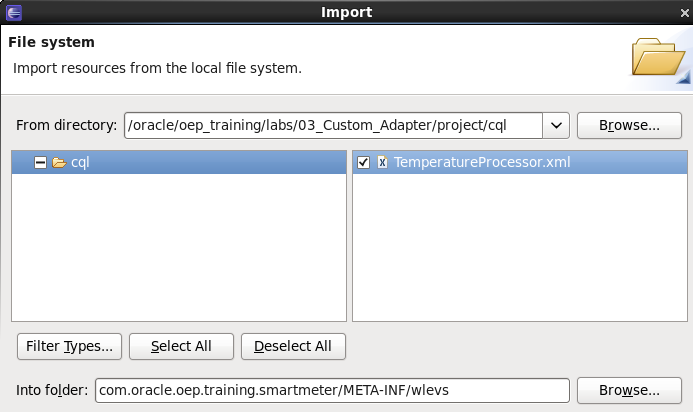


##### Part 3: Add a Temperature Processor

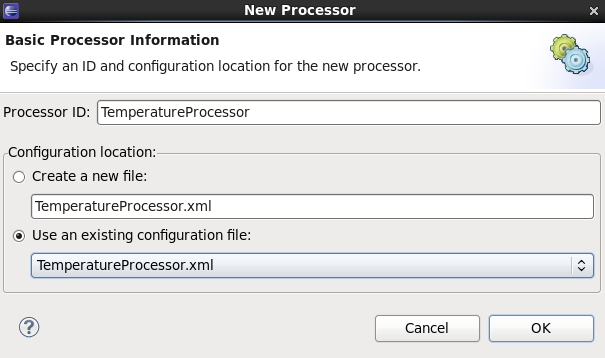
Next, we want a processor that filters out temperatures that are < 100.0

1. Click on the “META-INF\wlevs” folder. Import “TemperatureProcessor.xml” from the “<OEP\_TRAINING>\labs\03\_Custom\_Adapter\project\cql” folder.





1. In the EPN Editor, right-click to create a new processor called “TemperatureProcessor”. Instead of creating a new file, use the existing file that you just imported and click OK.



1. Join the “TemperatureChannel” to the “Temperature Processor”.
2. Create a “TempAlertChannel”.
3. Join the “TemperatureProcessor” to the “TempAlertChannel”.
4. Right-click and “Go to Assembly Source” and assign the event-type “TemperatureEvent” to the “TempAlertChannel” or use the “Property” tab to set it.
5. Don’t forget to save the changes to the file.
6. Connect the “TempAlertChannel” to the existing “EventListener”.
7. Add some code to the Event Listener (click on it in the EPN Editor and select “Go to Java Source”) to get some information about the Temperature events received.

**if**(event **instanceof** TemperatureEvent) {

TemperatureEvent theEvent = (TemperatureEvent) event;

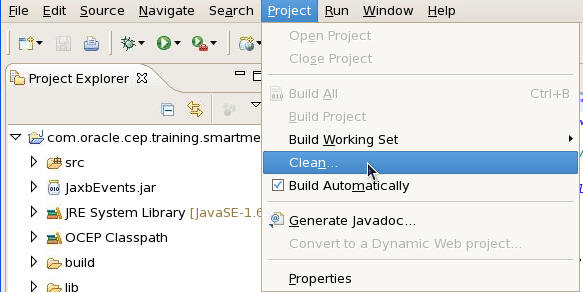
String id = theEvent.getSubstationId();

**double** temp = theEvent.getTemperature();

System.*out*.println("EventListener: Received \*\*\*Temperature Alert\*\*\* for substation id: " + id + " temp: " + temp + "\n");

}

1. Make sure that you import the “TemperatureEvent”.
2. Clean the project to make sure there are no errors, especially since we imported some code.

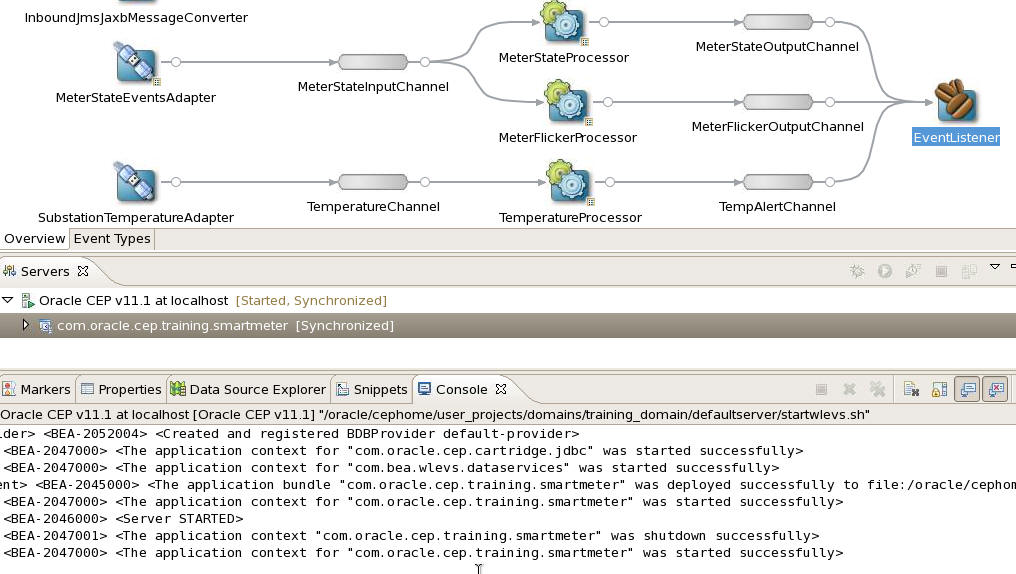


##### Part 4: Deploying the Application

Finally, you are ready to deploy the application to the server.

Start the server and add the project, if necessary.

1. If the application is already on a running server, you simply need to “publish” the new version. If there are no errors, you should see a “deployed successfully” and “started successfully” messages when the application is added to the server and the publish task is complete (with no more error messages):

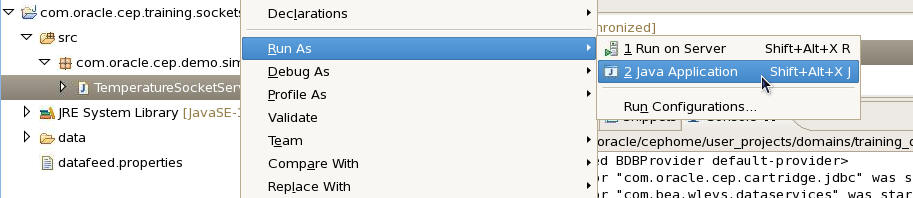


##### Part 5: Testing

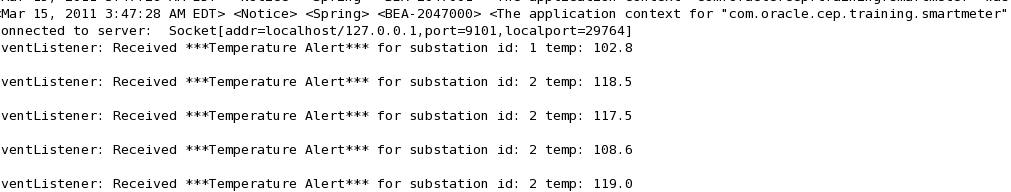
To test, we start the socket server java program that we imported.

NOTE: You’ll still need to have the WebLogic Server domain hosting the JMS queue running since the application defines a JMS Adapter and will attempt to connect to the queue.

1. Find the “com.oracle.cep.training.socketserver” project in the Project Explorer.
2. Open the “src” folder and the “com.oracle.cep.demo.simulator” package.
3. Click on “**TemperatureSocketServer**.java”.
4. Right-click and select: “Run As”, “Java Application”.



1. You should see some messages that are showing the temperature alerts written to the console.
2. Notice that they all indicate a temperature > 100.0



To stop the SocketServer, you may need to switch the console view to the TemperatureSocketServer and then push the red stop button.



##### 

**Summary:**

At the end of these exercises the participants understand how to:

* Create a custom TCP/IP socket adapter for a CEP Application
* Assign property values in the EPN for custom adapters using Spring
* Create a CQL query that can filter events