**Exercise 13**

*Creating and executing a BPMN flow*

**Prior Knowledge**

*Understand simple BPMN*

*Using Eclipse / Maven*

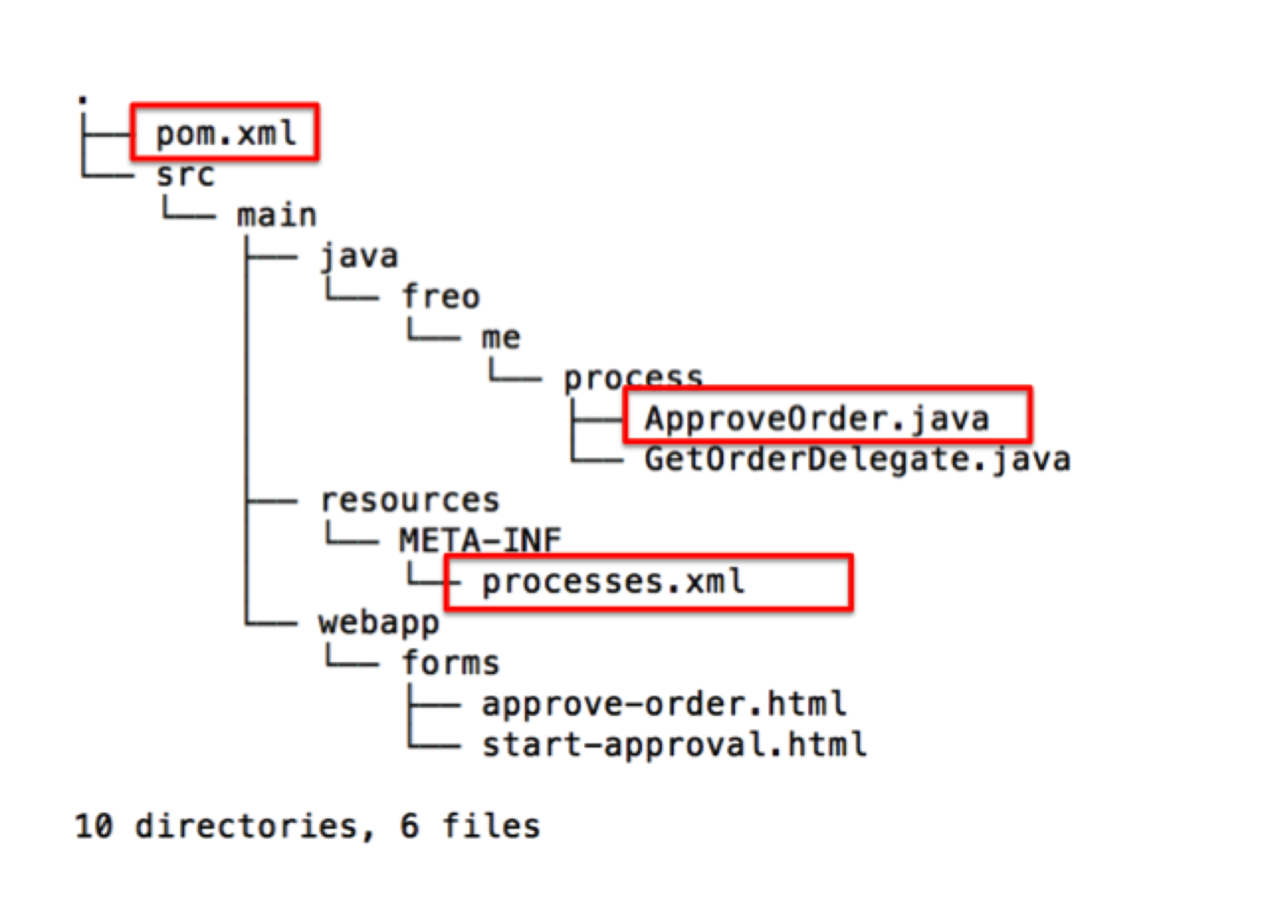
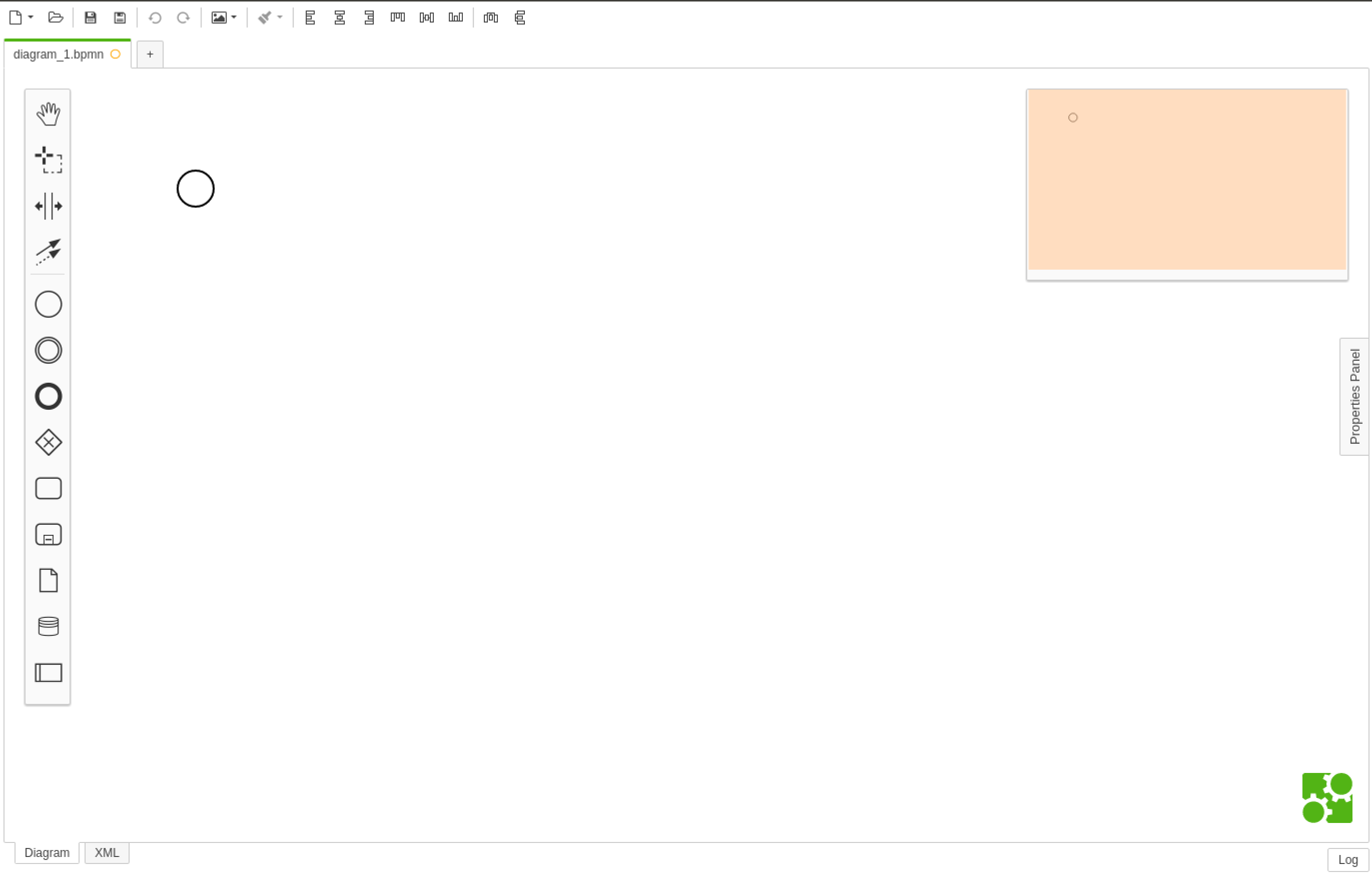
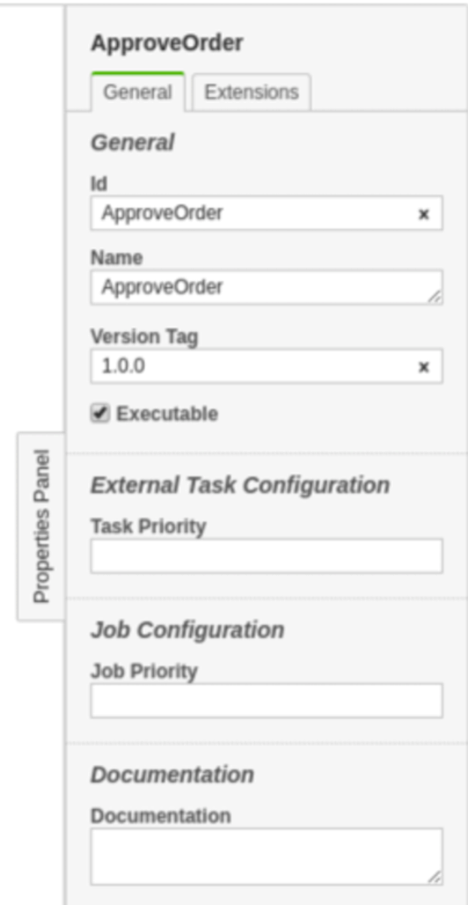
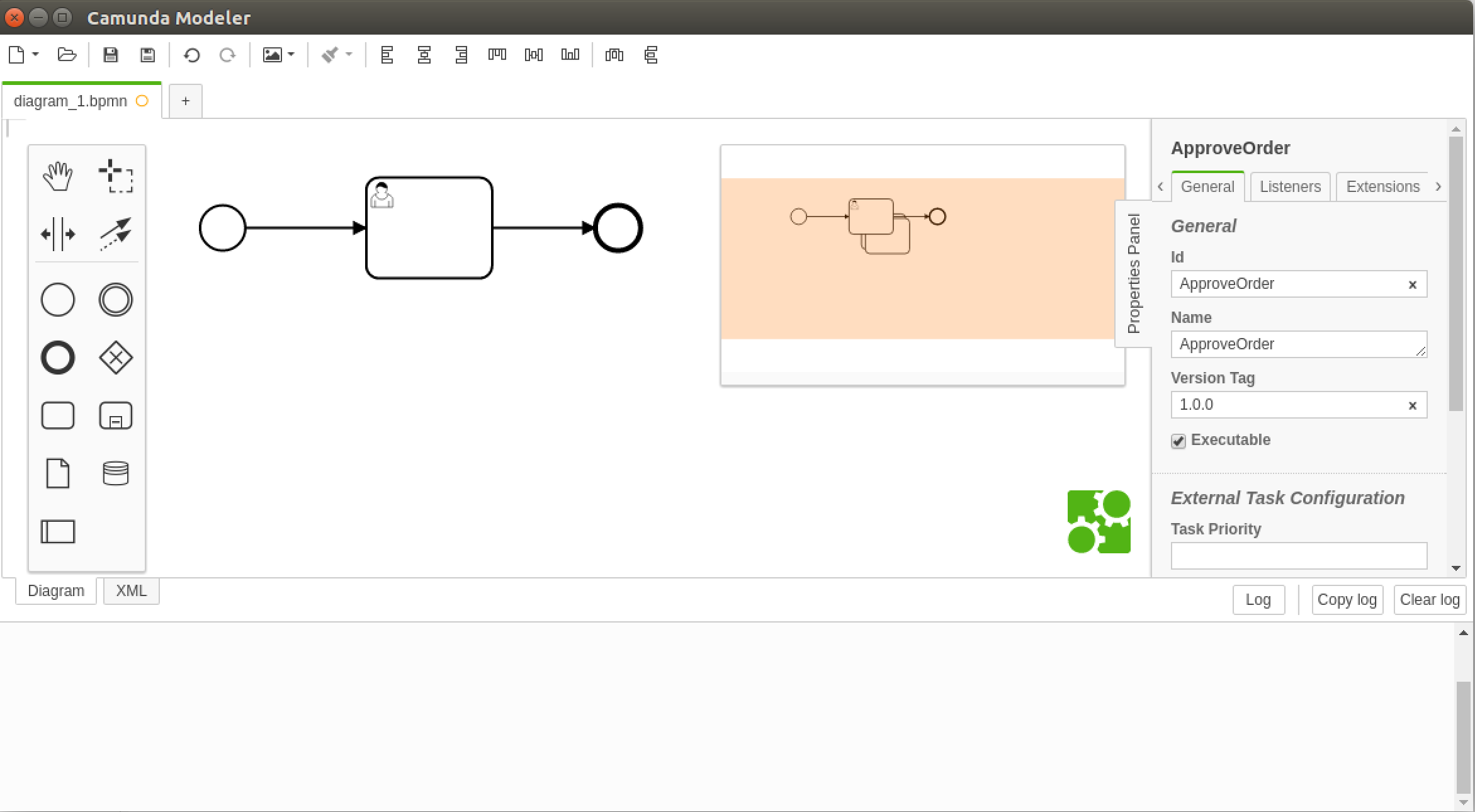
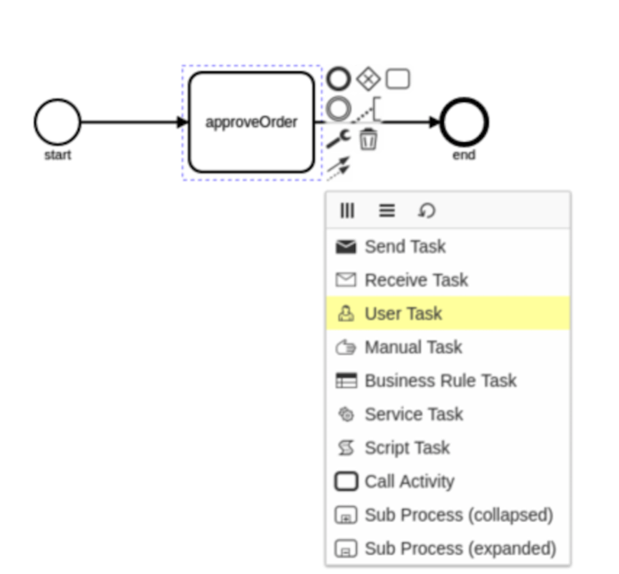
**Objectives**

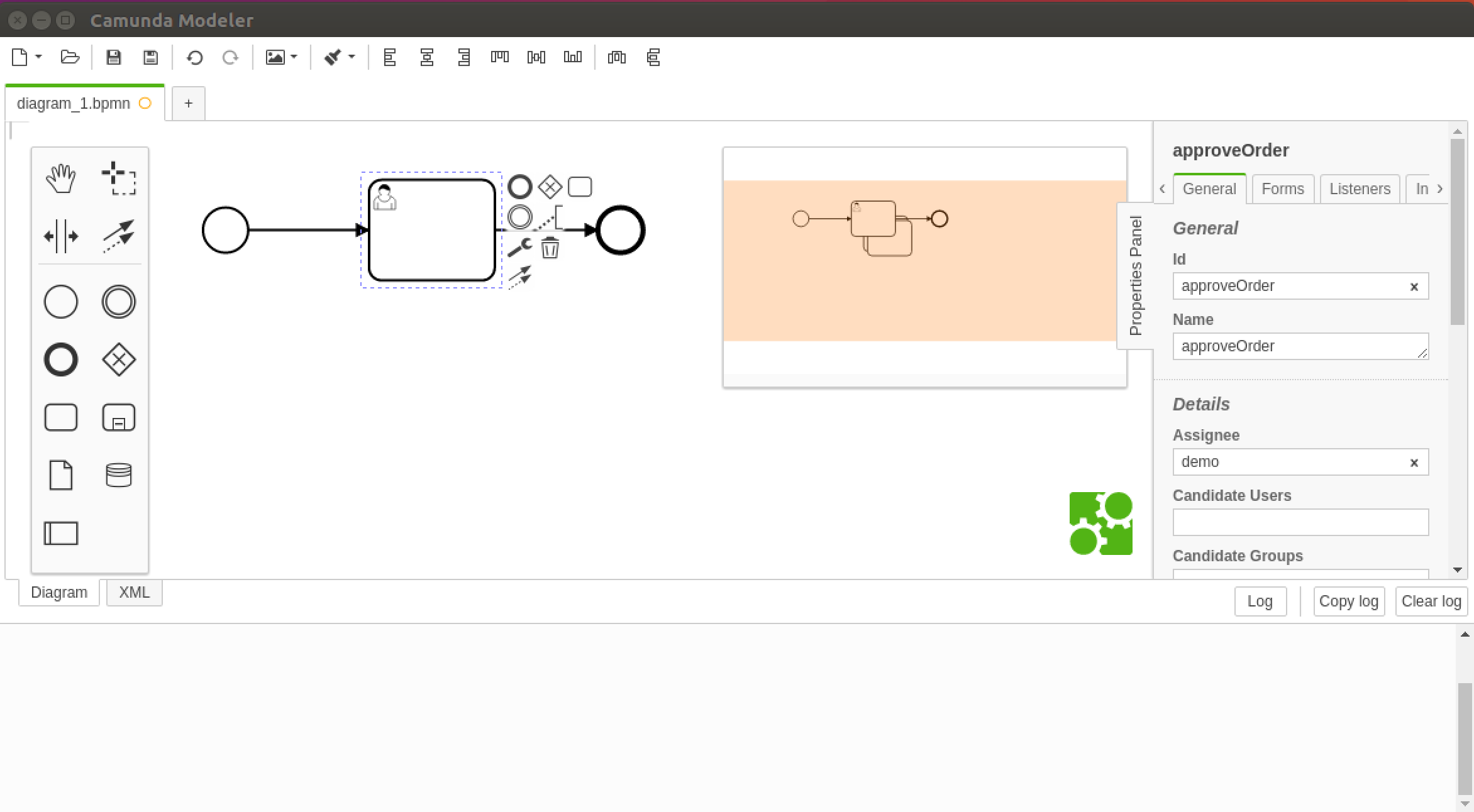
Understand the basics of the BPEL specification, and be able to create and execute a business process using the BPEL tooling in Eclipse. Deploy the BPEL into the WSO2 BPS and be able to track instances etc.

**Software Requirements**

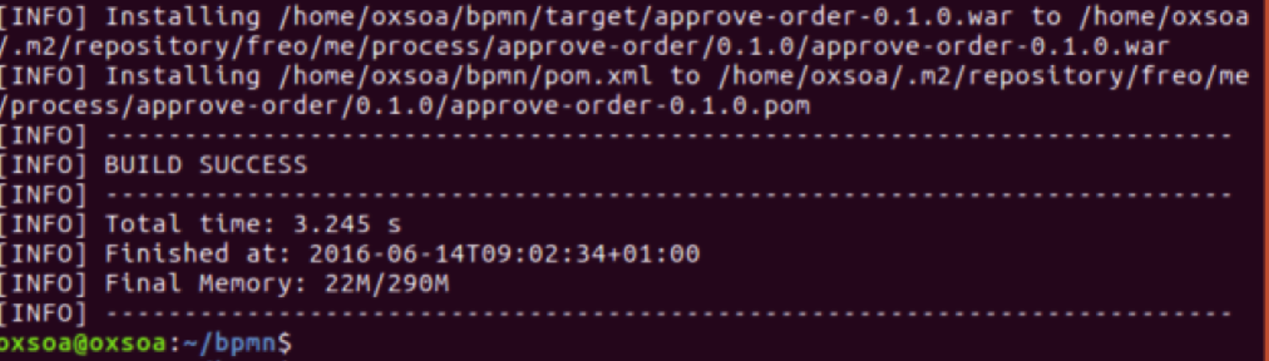
* Camunda BPMN Modeler 1.8.0
* Camunda BPMN runtime 7.6.0

**Steps.**

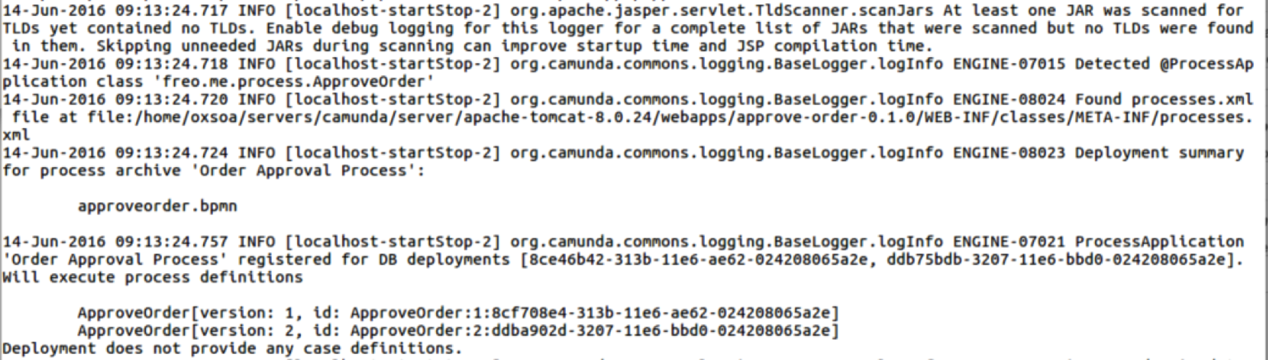
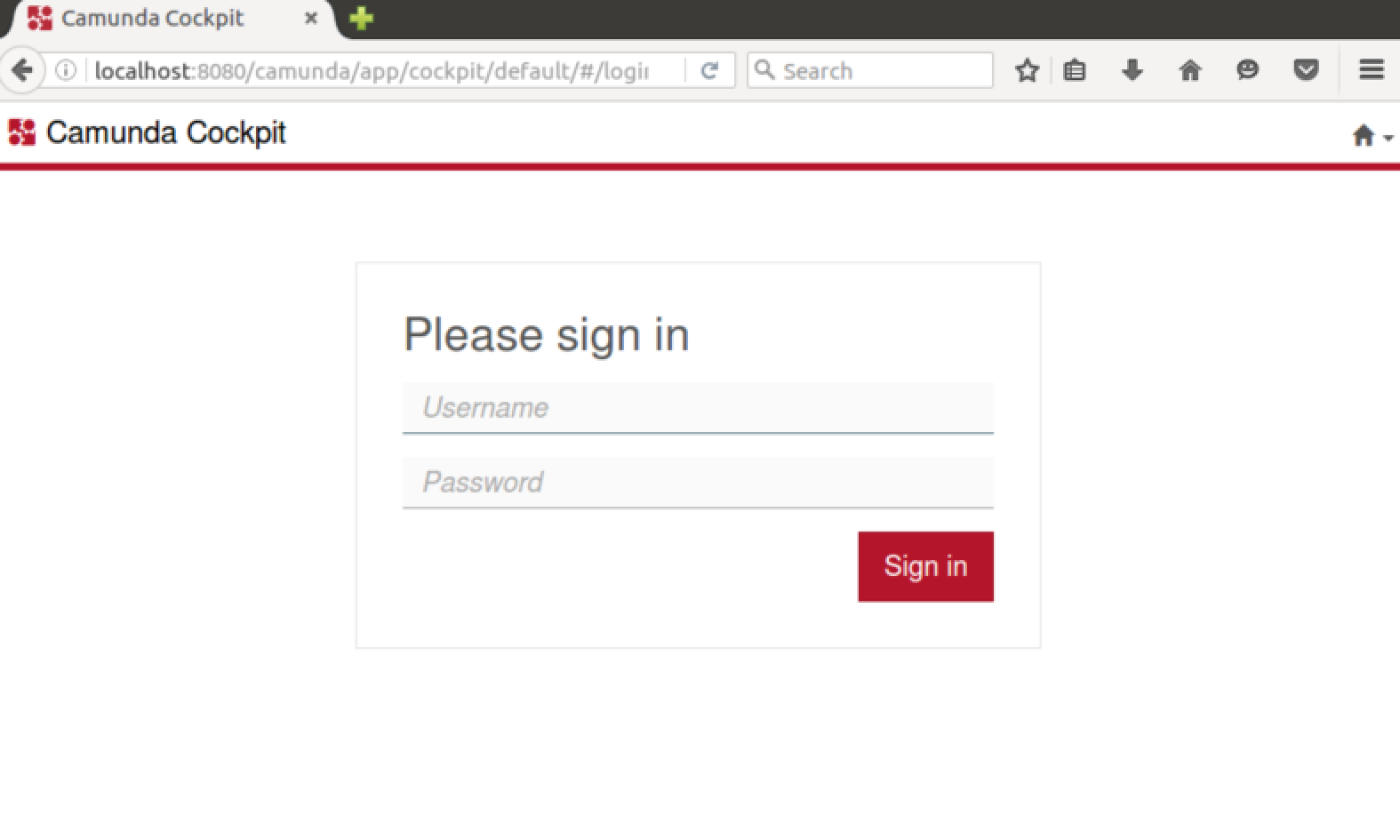
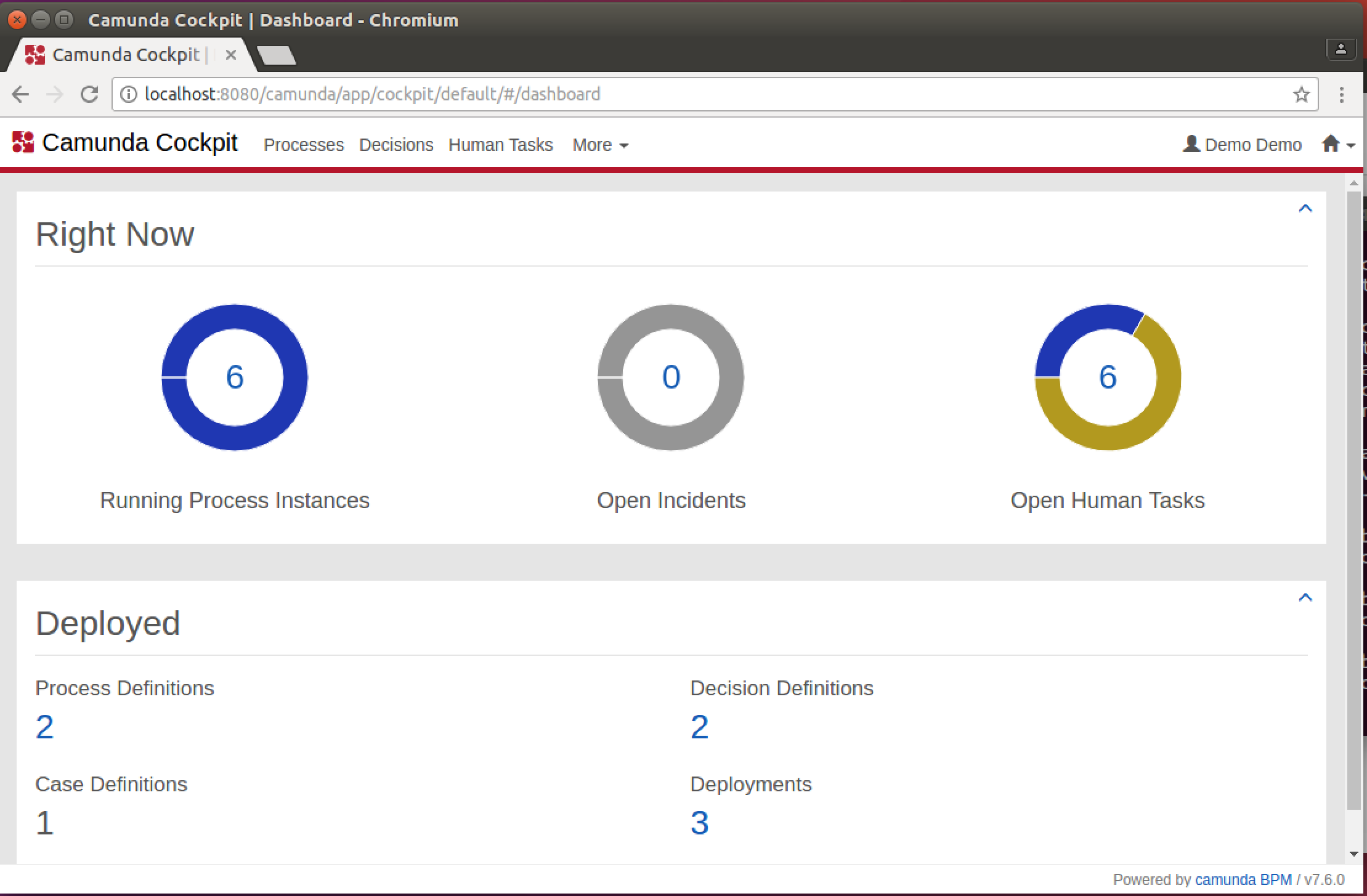
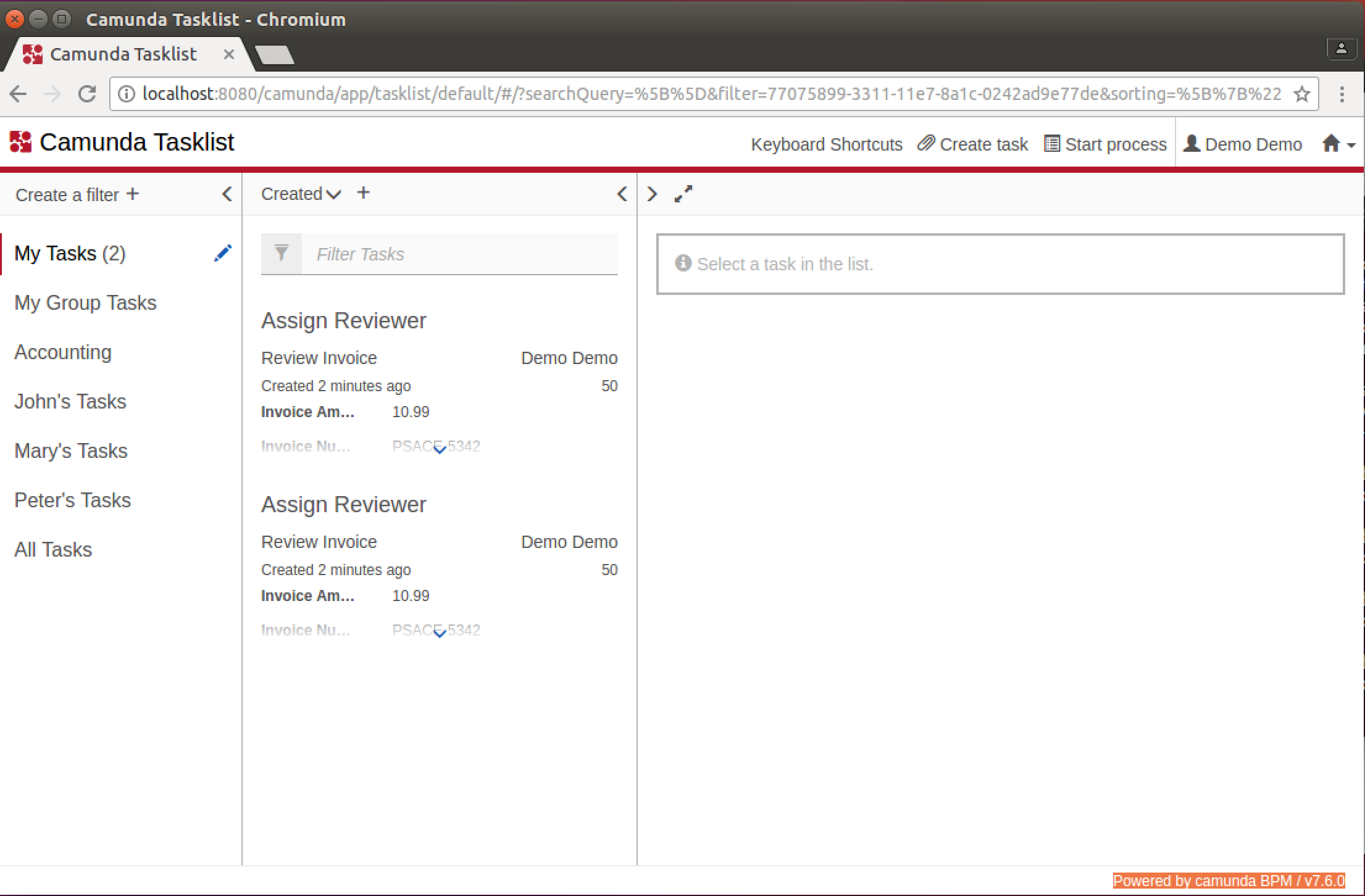
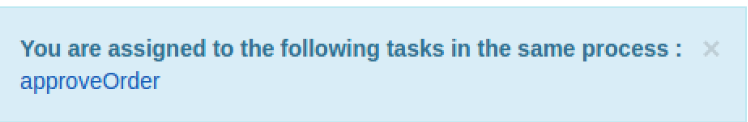
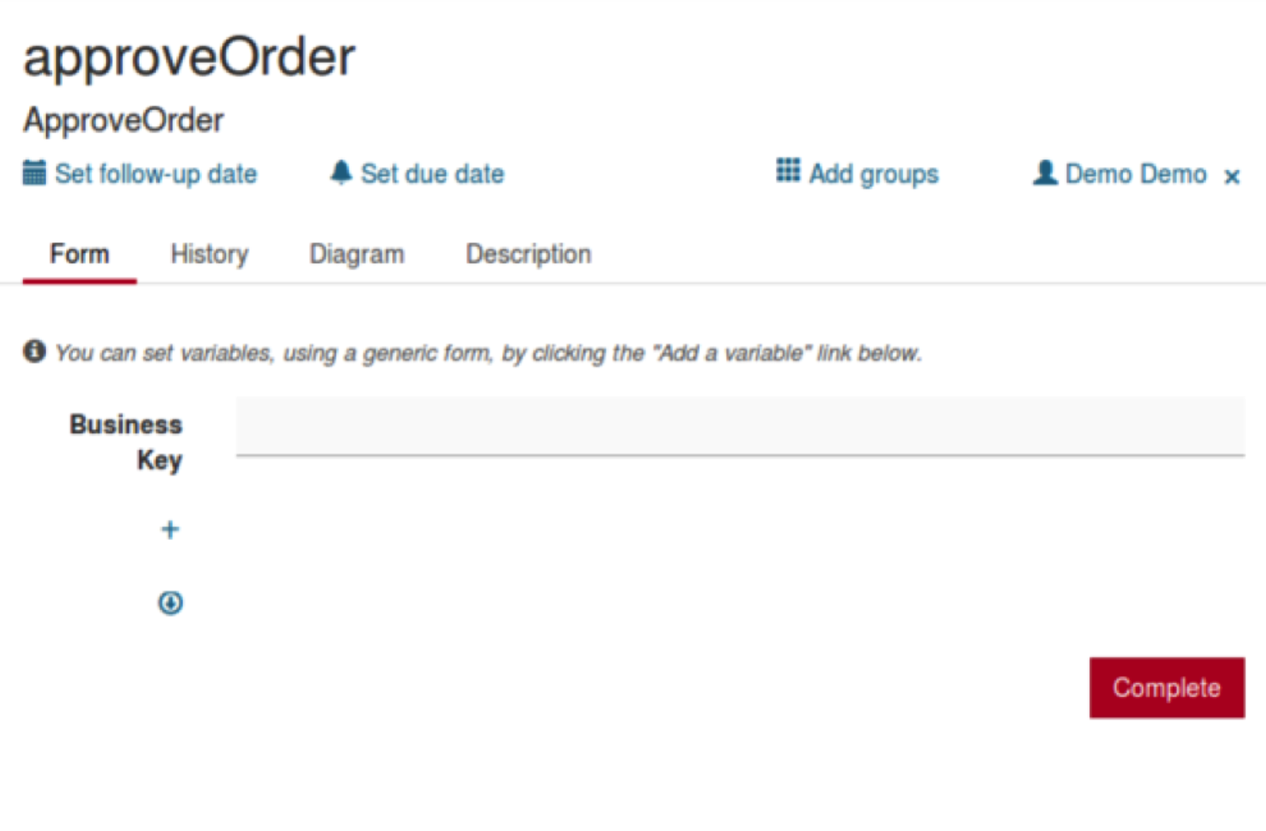
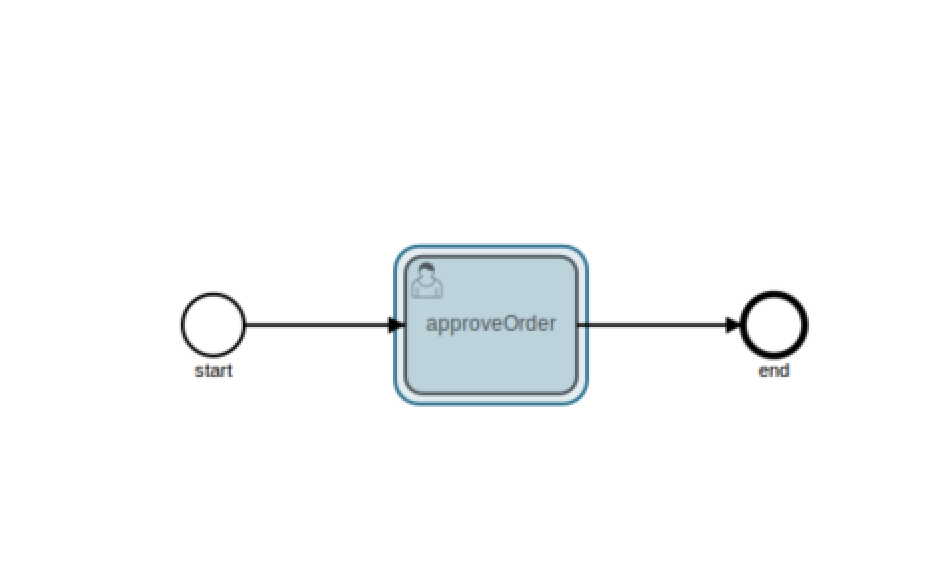
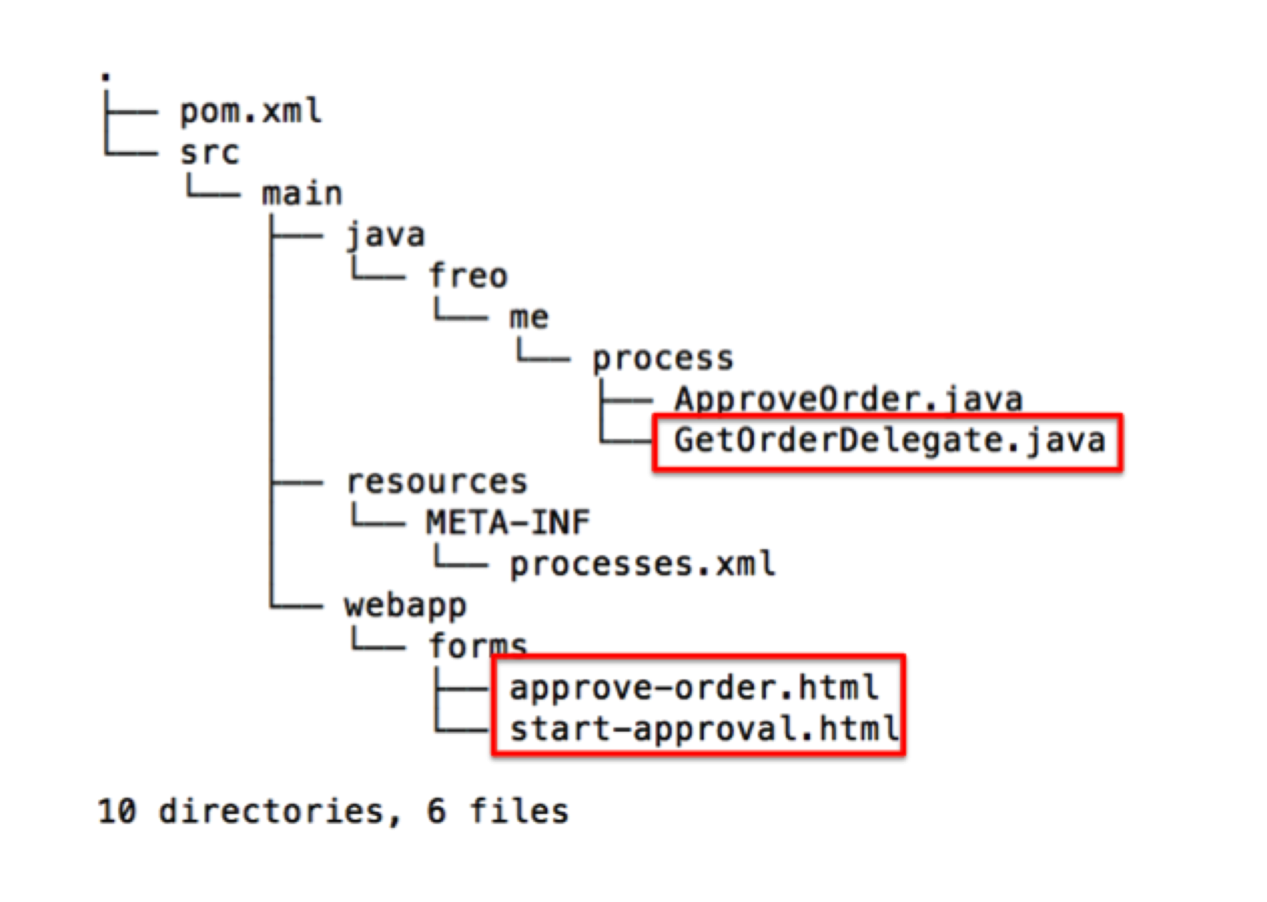
1. Let’s get our project initiated. Firstly let’s make a directory structure:  
   mkdir ~/bpmn/
2. Now lets grab a simple default project layout:  
   cd ~/bpmn  
   wget <http://freo.me/bpmn-kit> -O process-archive.zip  
   unzip process-archive.zip
3. The main files here are the build (pom.xml), a generic process descriptor XML (see <https://docs.camunda.org/manual/latest/user-guide/process-applications/the-processes-xml-deployment-descriptor/#empty-processes-xml>) and a basic Application class (ApproveOrder). We’ll ignore the other files for the moment. Take a look at those three.  
     
   With the three mentioned files, this will create a WAR that will run *whatever* BPMN process we deploy into the **src/main/resources** folder.
4. Let’s create our first BPMN.
5. Start the Camunda BPMN Modeler from the launcher.  
    
6. Click on Create a BPMN diagram.
7. You should see:
8.  Make sure the properties panel is expanded.
9. Make sure the **Executable** tag is ticked, and change the process id and name to be **ApproveOrder.** Change the version number to be 1.0.0:  
   
10. Use the tool to draw a simple process like this. Make the **id** of each object match the name (e.g start/start).  
    
11. In order to make the task into a User Task (with the little “man” icon), click on the spanner/wrench and choose User Task. 

1. Edit the properties for the Approve Order Task:  
   Id: **ApproveOrder**Name: **ApproveOrder**Assignee: **demo**  
   
2. Save the file as ~/bpmn/src/main/resources/approveorder.bpmn  
   *Hint: Sometimes there is a bug with the BPMN editor and the Save dialog box appears behind the main window. To solve it, click on the new icon that appears in the Launcher.*
3. Now you can test your process.
4. In the BPMN directory, type:

mvn clean install

1. You should see some text in the console like:   
   
2. The Camunda server is running on 8080, so make sure none of your other servers is still running on 8080.
3. Start the server:  
   cd ~/servers/camunda/server/apache-tomcat-8.0.24  
     
   bin/catalina.sh run
4. We haven’t yet copied our process across, but you should see the server deploy a default process:
5. Now we can copy over our process WAR. In a new command window:  
     
   cp ~/bpmn/target/approve-order-0.1.0.war ~/servers/camunda/server/apache-tomcat-8.0.24/webapps

*All on one line!*

1. If everything is going well, you can go back to the other terminal window and see the log, and you will see the process be deployed:  
   
2. Let’s test the process now.
3. Start up a browser (please use Chromium as there seems to be a problem with Firefox) and browse to <http://localhost:8080/camunda/app/cockpit/>
4. You should see: 
5. Sign in with *demo/demo*
6. You should see something similar to this: 
7. Select the little Home icon in the corner and choose **Tasklist**
8. Now you should see something like this:   
     
   Those tasks are from the default process that comes with Camunda. Ignore those.
9. Click on **Start Process**
10. Choose **ApproveOrder  
    **
11. Enter anything you like in Business Key and then click **Start**
12. Do you remember where you set the assignee for the User Task in the BPMN process to **demo**? Well this has just happened and you are logged in as demo and hence the portal has popped up a message saying you’ve been assigned a task to work on:   
    
13. Click on **Approve Order**
14. You should see something like this:  
    
15. Click on the **Diagram** tab  
    You will see the process you designed, now with the current step highlighted:  
    
16. Go back to the **Form** tab and click Complete. This will “complete” this instance of the process.
17. This process works, but lets be honest, it is almost pointless.
18. To make the process more interesting we need to tie it in with our existing order processing system. A simple way to do that would be to take an order ID as input, and then gather information about that order to put in front of the approver.
19. If you look at the directory, you should see that there are three other files that came with process-archive.zip that will help:  
    
20. There are two forms already deployed in the src/main/webapp/forms directory. Take a look at these.
21. These are standard HTML with some extensions for Camunda that enable the developer to specify fields that will be displayed or collected and to map those fields to process instance variables.
22. There is also a “Delegate” Java class. Here is the code listing:

public class GetOrderDelegate implements JavaDelegate {

private final static Logger LOGGER =   
 Logger.getLogger("APPROVAL-REQUESTS");

public void execute(DelegateExecution de) throws Exception {

String id = (String) de.getVariable("id");

LOGGER.info("Processing request by '" + de.getVariable("id"));

Client client = ClientBuilder.newClient();

WebTarget target = client.target("http://localhost:80")

.path("purchase").path(id);

Response response =   
 target.request(MediaType.APPLICATION\_JSON).get();

if (response.getStatus() == 200) {

JSONObject json =   
 new JSONObject(response.readEntity(String.class));

de.setVariable("lineItem", json.get("lineItem"));

de.setVariable("date", json.get("date"));

de.setVariable("quantity", json.get("quantity"));

de.setVariable("customerNumber",   
 json.get("customerNumber"));

de.setVariable("poNumber", json.get("poNumber"));

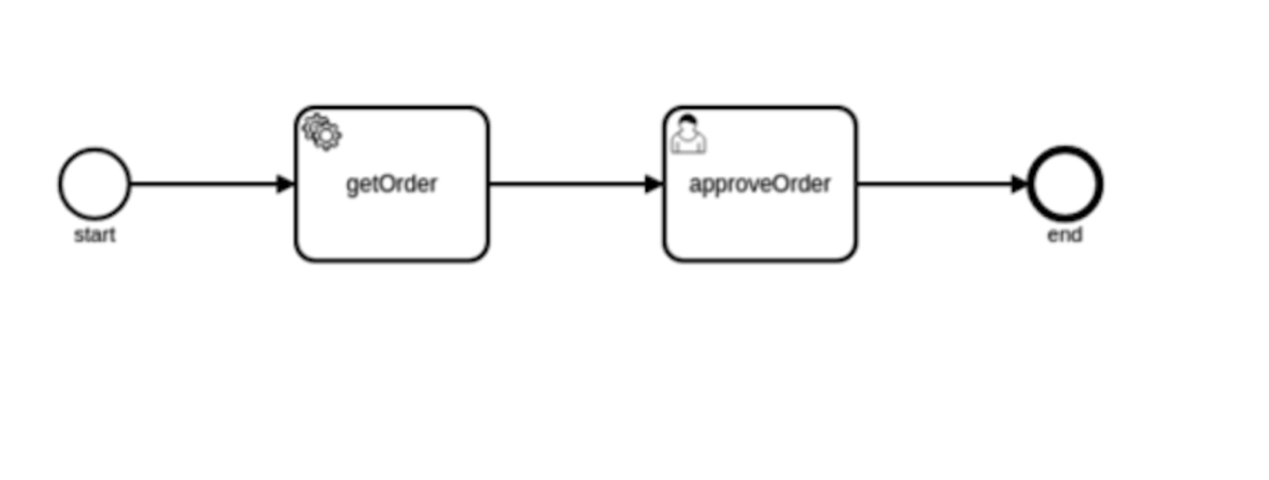
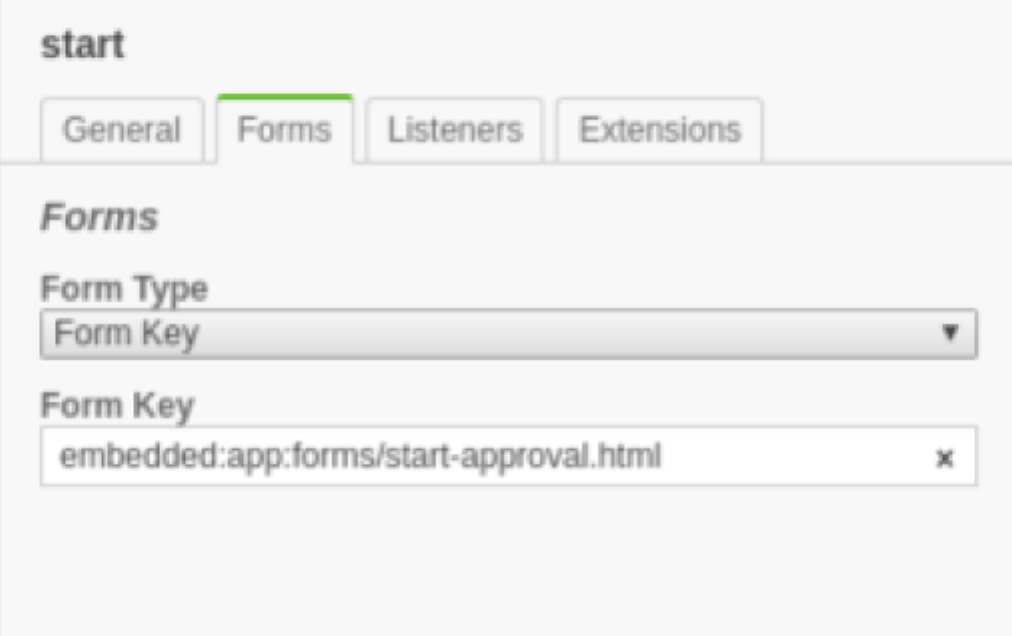
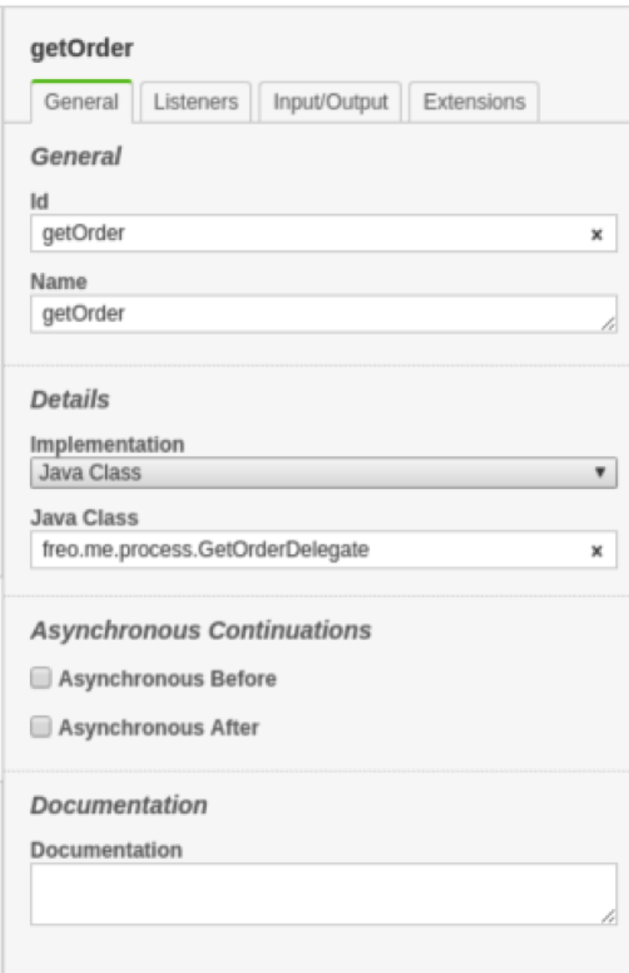
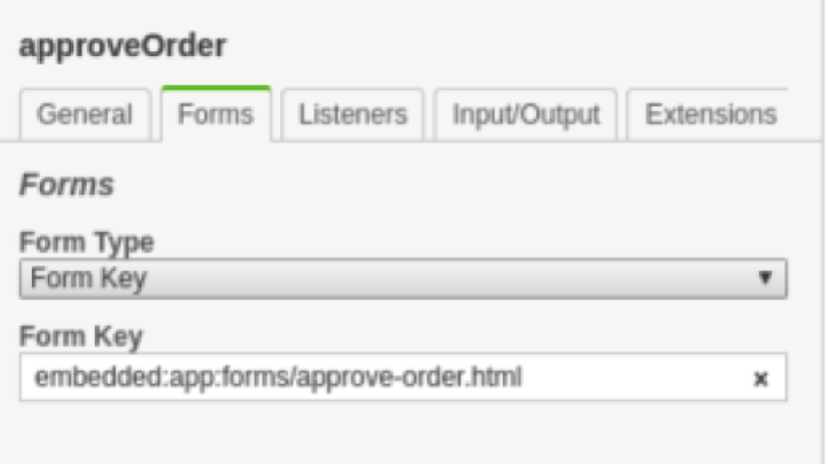
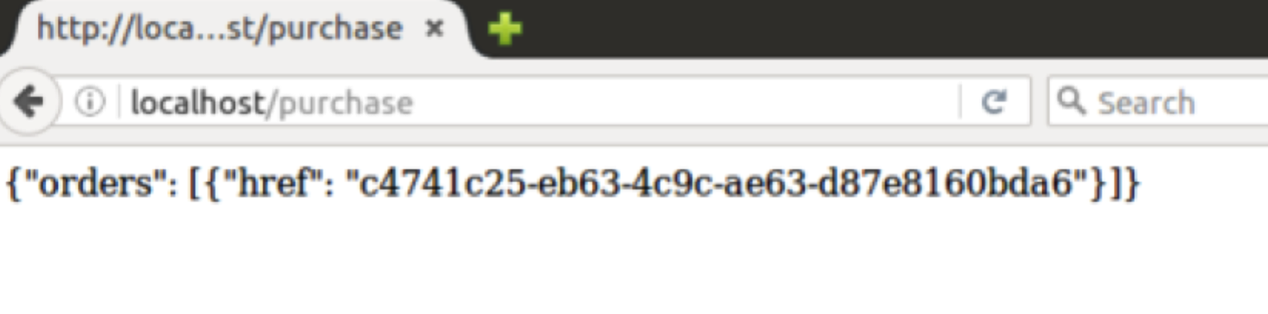
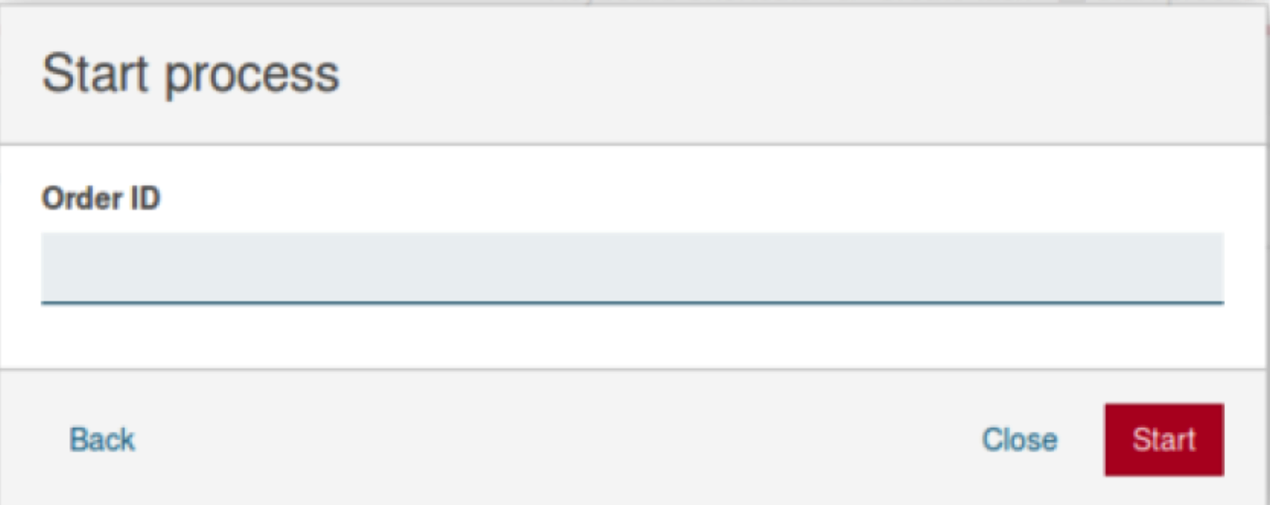
} else {

throw new BpmnError("ID NOT FOUND");

}

}

}

1. This is a class specific to Camunda that can be called from a “ServiceTask”. As you can see, it takes a process variable (id) and uses it to look up a PO from our purchase server, using JAX RS REST client code.
2. Let’s use these extra files.
3. Firstly expand the BPMN flow to look like this:  
   ****
4. The getOrder task is a Service Task.
5. Go to the properties of the Start event and add a Form, with the Form Type is **Form Key** and the value is:  
   embedded:app:forms/start-approval.html  
   ****
6. Edit the properties of the Service Task getOrder:  
   Implementation: Java Class  
   Java Class: freo.me.process.GetOrderDelegate
7. Add a form to the approveOrder User Task and reference the other form:  
   embedded:app:forms/approve-order.html  
   
8. Now we should be ready to run this. **Save the BPMN.**
9. Rebuild the WAR file:  
   cd ~/bpmn  
   mvn clean install
10. Recopy the WAR to the Camunda directory:  
    cp ~/bpmn/target/approve-order-0.1.0.war ~/servers/camunda/server/apache-tomcat-8.0.24/webapps
11. Make sure your backend purchase service is up and running. This is the docker-compose service from Exercise 9[[1]](#footnote-1).  
    You can check that the service is up and running by browsing:   
    <http://localhost/purchase>  
      
    You should see something like 
12. Keep this window open as you are going to need that order id.
13. If Camunda’s Tomcat is already running it should have redeployed the process, otherwise start up the Tomcat again.
14. Go back to the Camunda tasklist
15. Now when you start the ApproveOrder process, you should see a new form asking for the id:  
    
16. Copy and paste the id from the previous browser window and click **Start**
17. Once again you should be notified that you are the approver for this process instance. Now you can click to approve that.
18. You should see a form that has data collected from the REST service. You can click the **Approve** tickbox and then Complete.
19. Check out the Cockpit and the Admin windows (from the little House icon). There are a lot of features in the package that you would need for a real process management scenario (e.g. adding users, creating tenants, adding new approvers to existing processes, checking on the state of processes, etc).
20. That’s all.
21. **Extension**

If you did the node.js/MongoDB Bonus exercise you will also have a catalogue to look up prices and a customer database. You could extend this process to create an invoice for the order.

1. If you didn’t get this working, you can run the service using the following command lines:  
   cd ~  
   git clone <https://github.com/pzfreo/POResourceComplete.git>  
   cd POResourceComplete  
   gradle clean shadowJar  
   sudo docker-compose up [↑](#footnote-ref-1)