**Exercise 3**

*Creating a JAX-WS Service*

**Prior Knowledge**

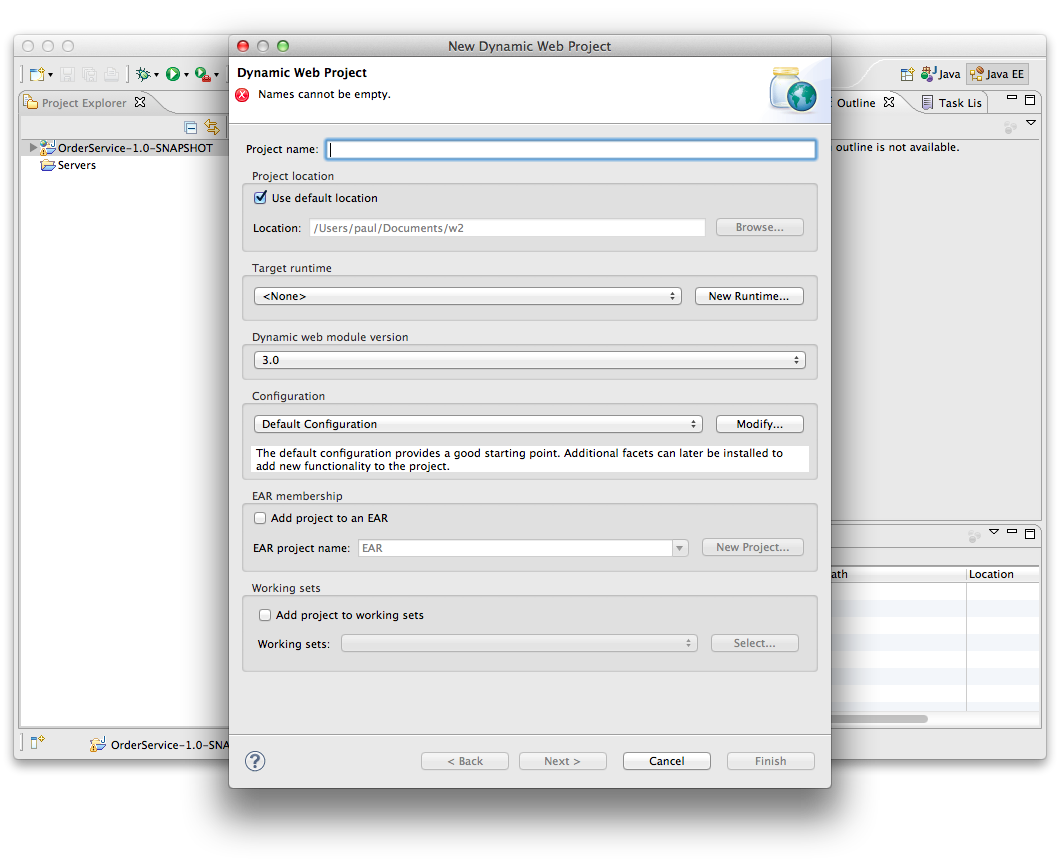
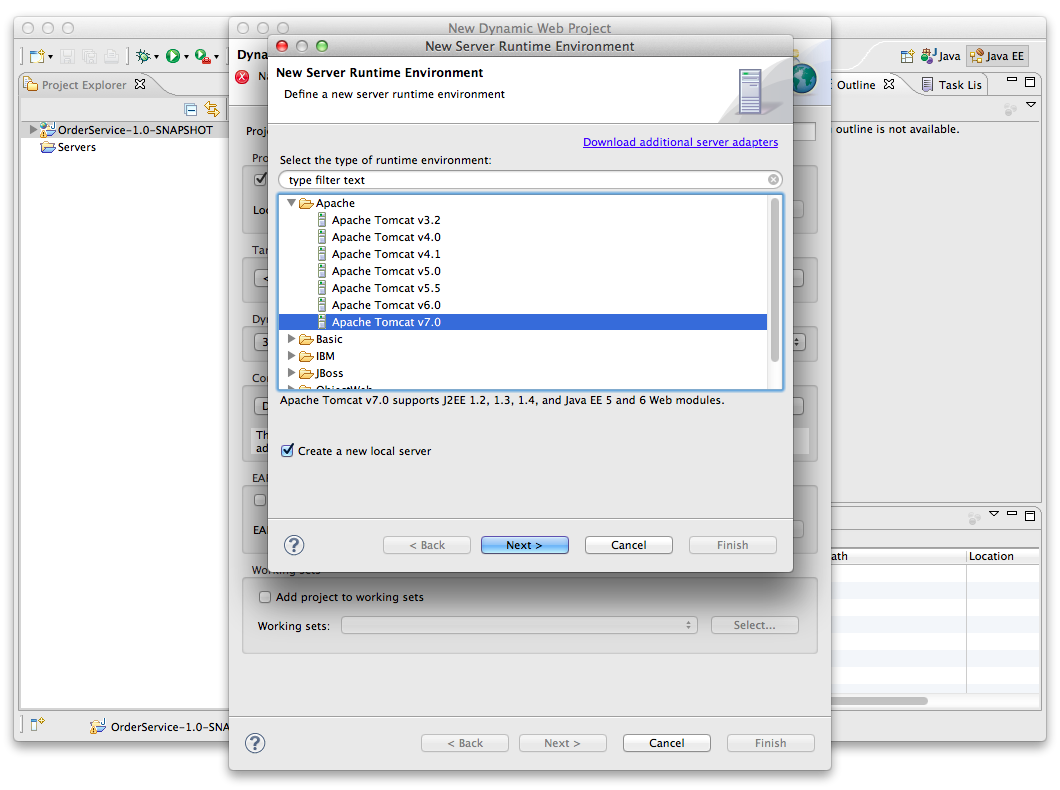
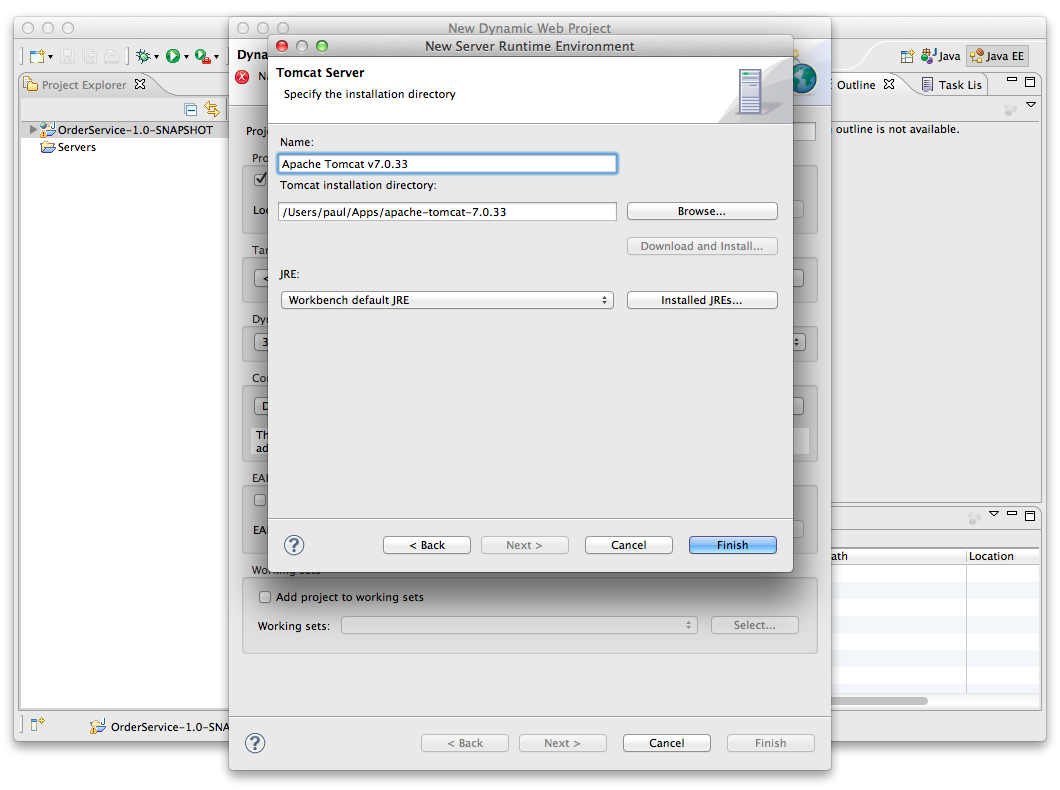
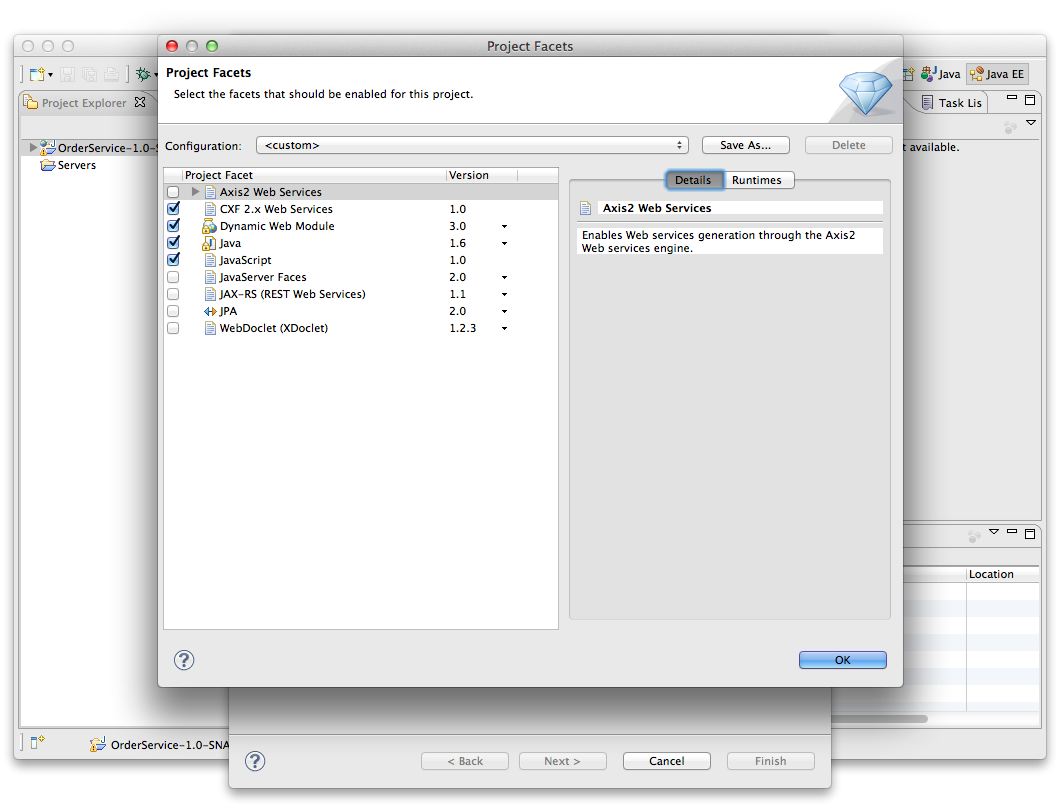
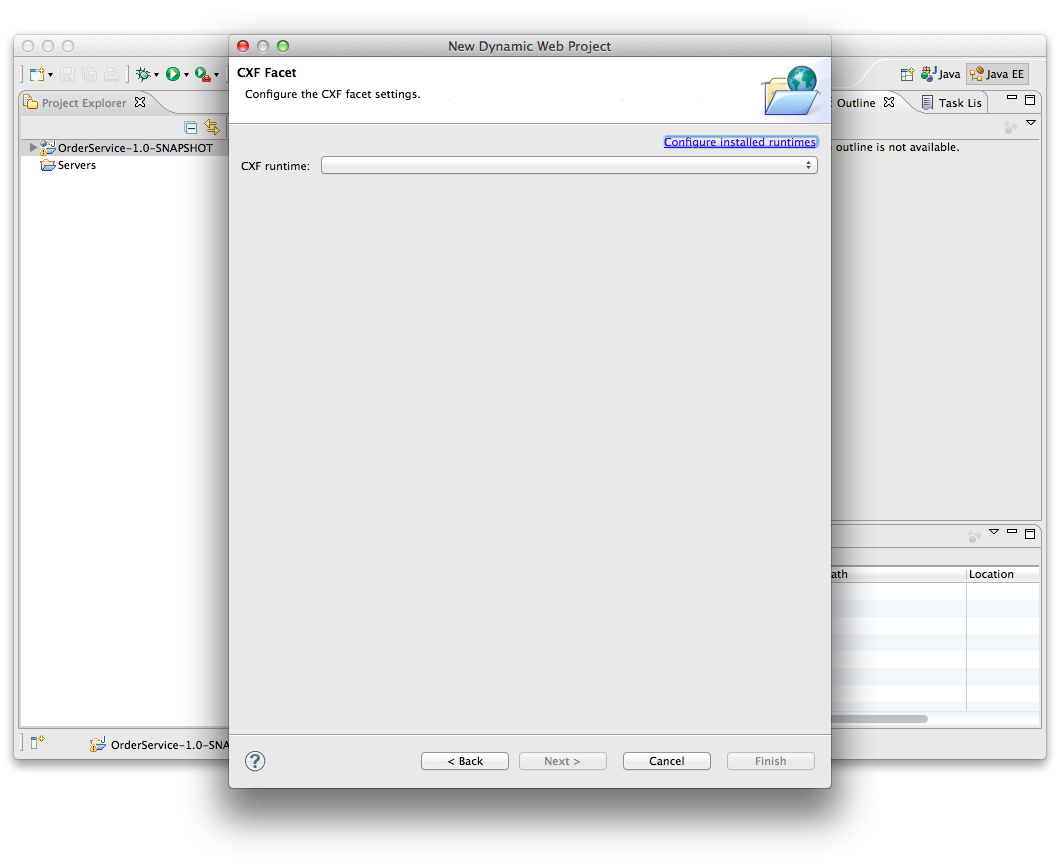
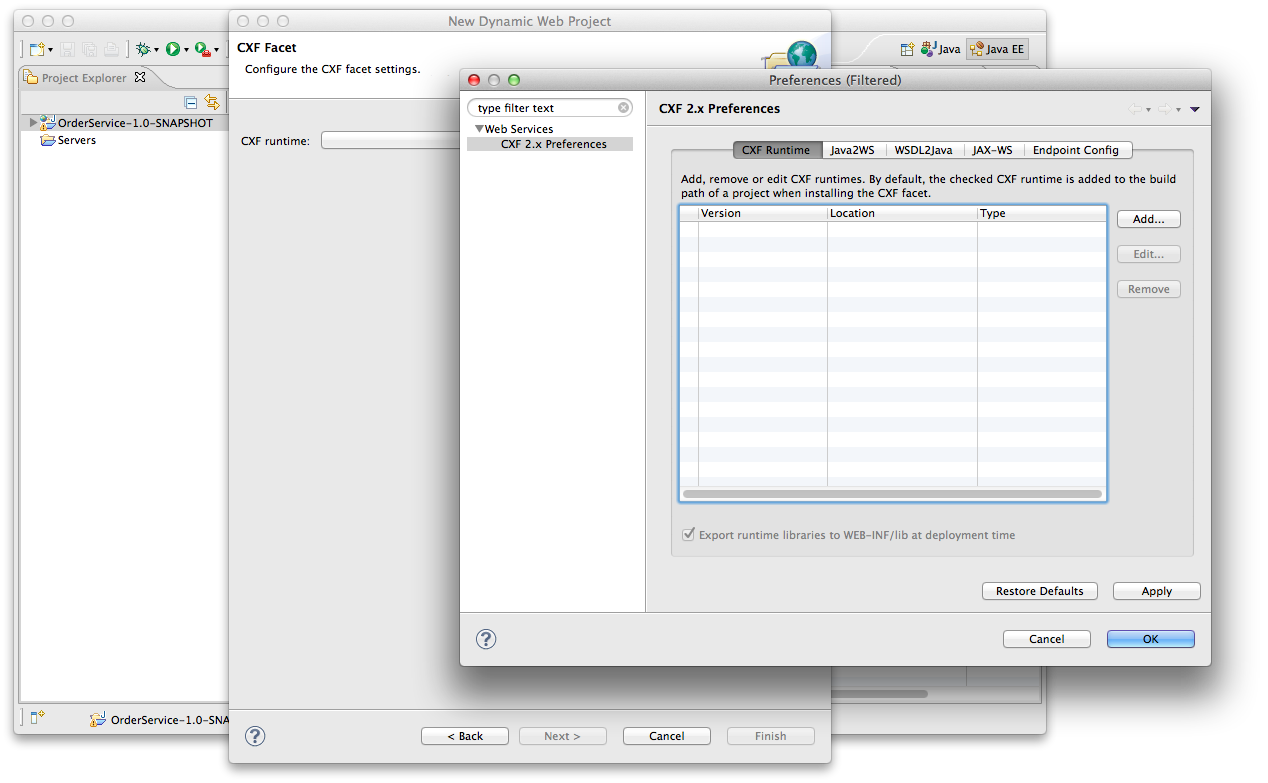
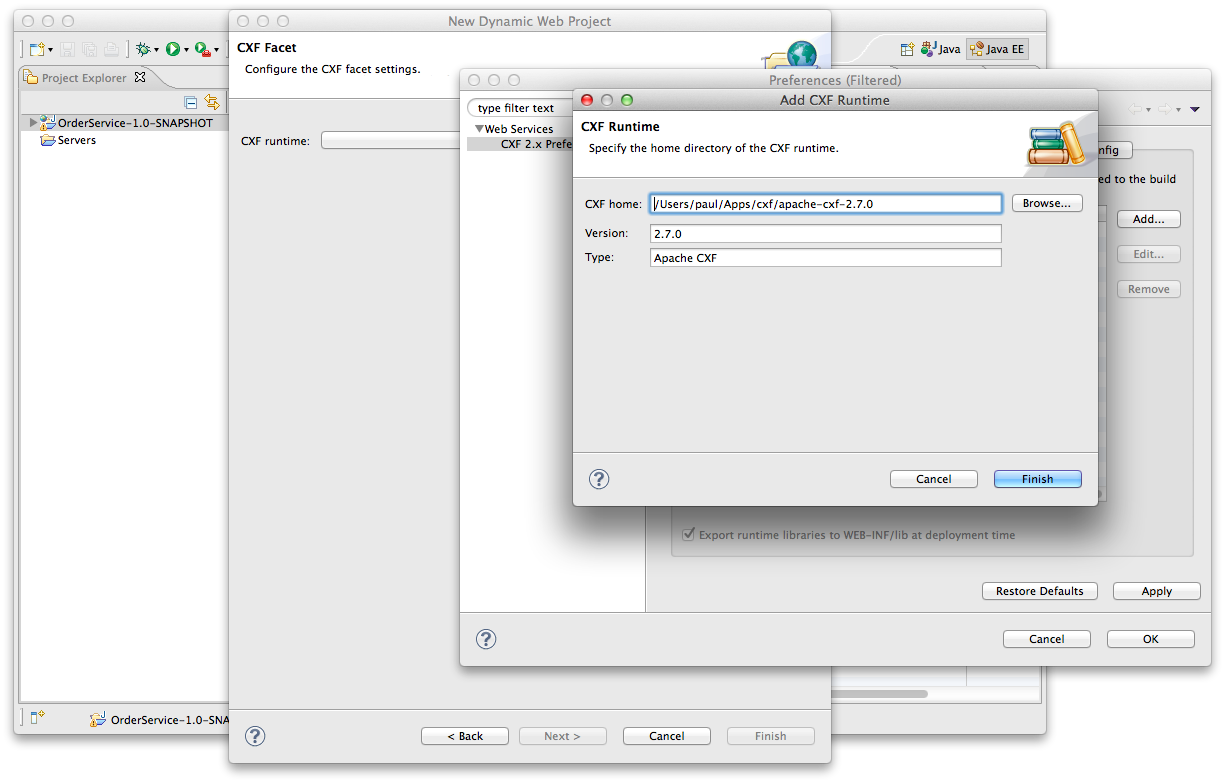
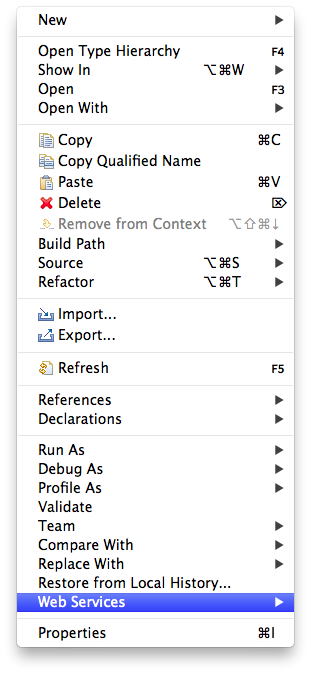
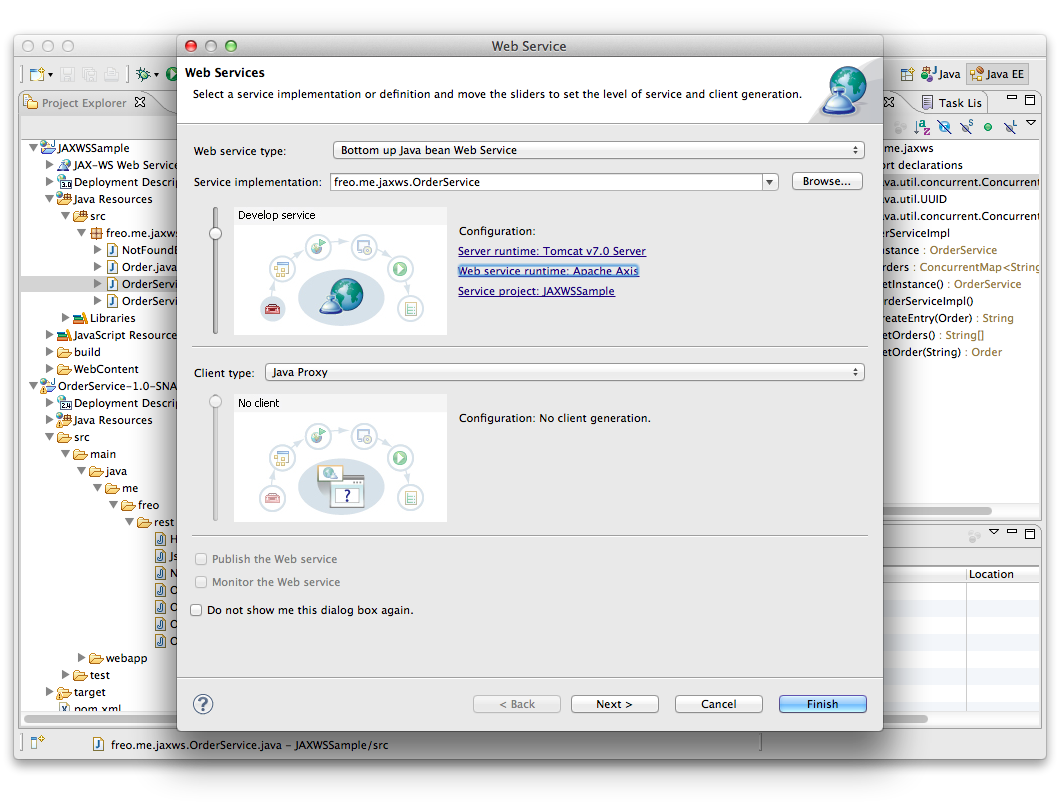
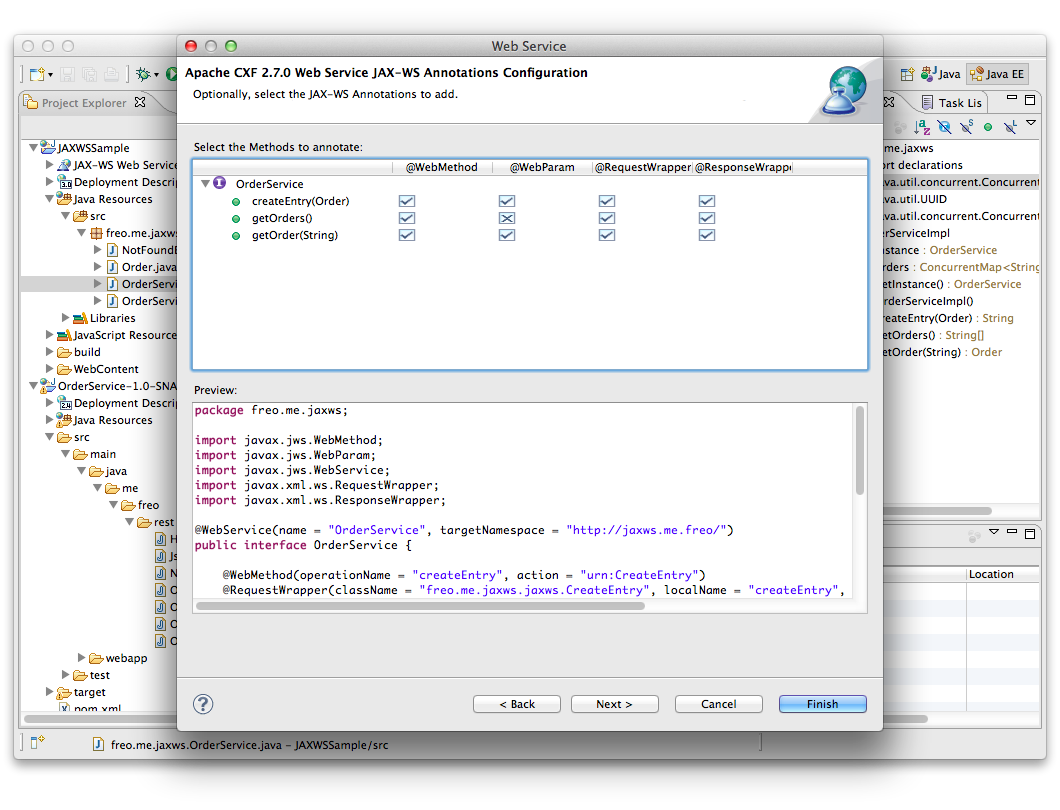
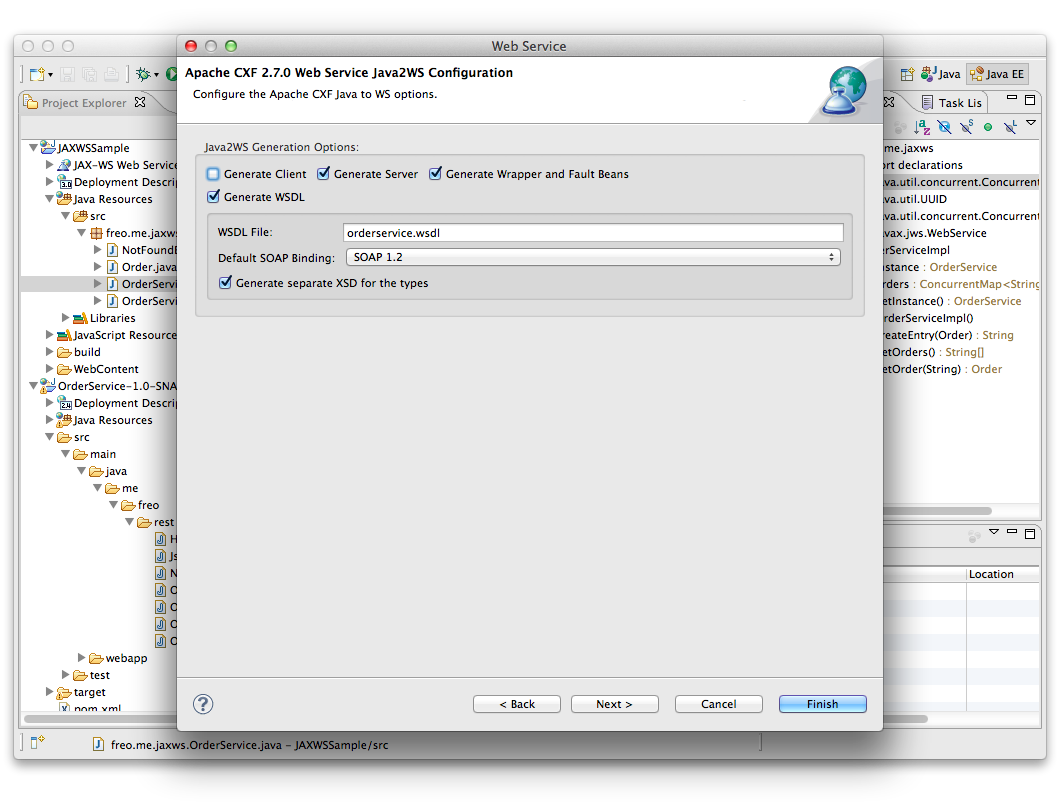
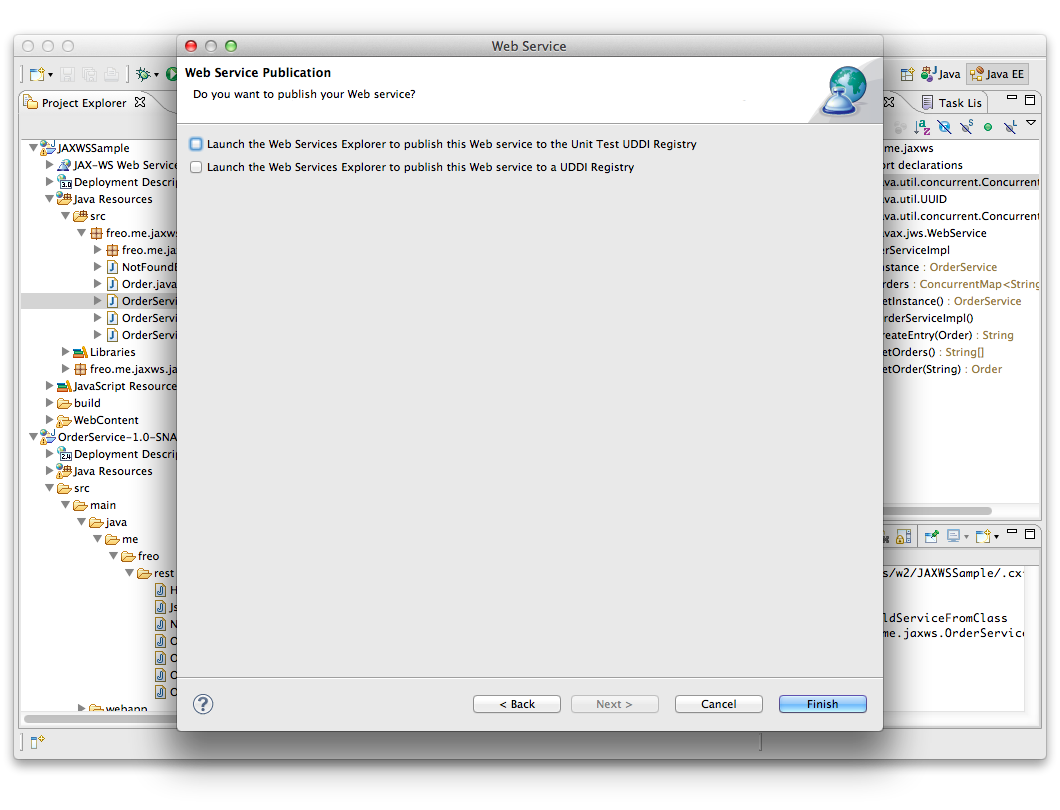
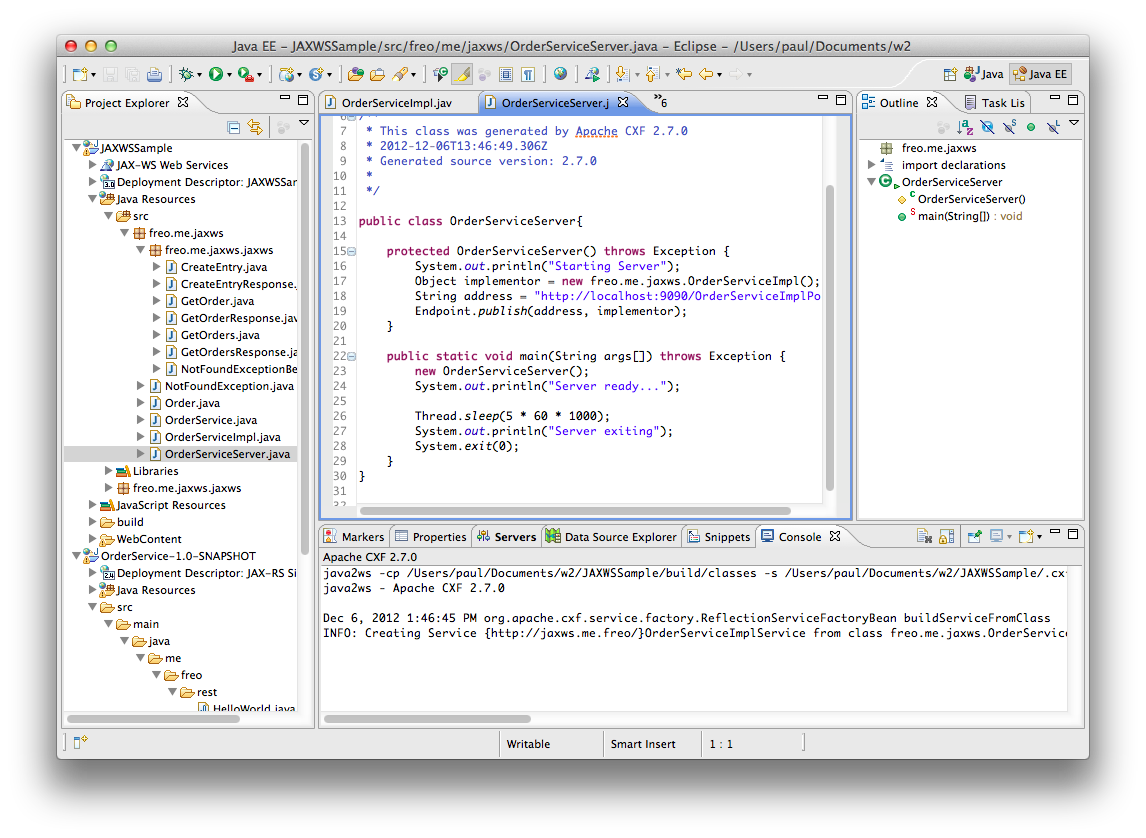
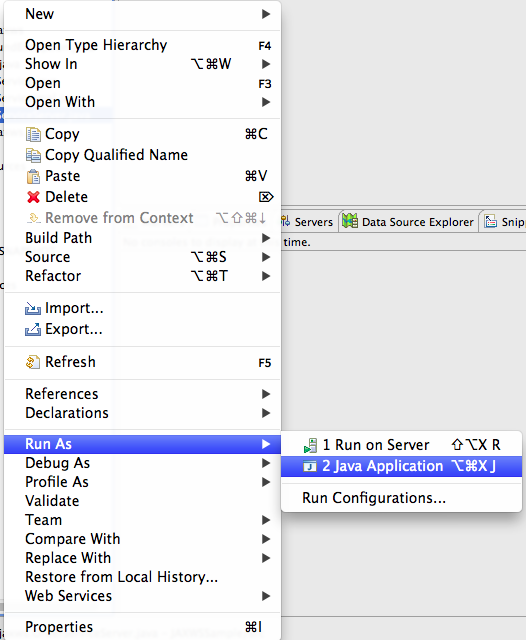
*Basic understanding of SOAP and WSDL*

**Objectives**

*Understand how to create Web Services in Java*

**Software Requirements**

* Java Development Kit 7
* Tomcat 7.0.33 or later
* Eclipse JEE workbench
* Apache CXF 2.7.0 or later
* SOAPUI

1. Make sure Tomcat is installed (e.g. in ~/apache-tomcat-7.0.33)
2. Make sure Apache CXF is installed (unzip in ~ to create ~/apache-cxf-2.7.0)
3. Open up Eclipse
4. Create a new Dynamic Web Project (**File -> New -> Dynamic Web Project**)  
     
   
5. Now click **New Runtime** and choose Tomcat 7.0  
     
   
6. Click **Next >**
7. Browse to the directory where you have Apache Tomcat installed and select that:  
     
   
8. Now we need to modify the Tomcat configuration. Click on **Modify** next to Default Tomcat Configuration:  
     
     
     
   Enable the **CXF 2.x Web Services** section. Click **OK**
9. Now you should be back in the initial Dialog, so give the project a name, e.g. **JAXWSSample**
10. Click **Next, Next, Next** until you are at the CXF Facet configuration dialog
11. Click **Configure Installed Runtimes**
12. Now click **Add.** Browse to the CXF install directory. (Yours should be different to my screenshot)  
      
      
    Now click **Finish**
13. Now select the tick box next to the CXF 2.7.0 version. Click **Finish** and **Finish** again. You should now have a project. Import the following four files into the project.   
      
    freo.me.jaxws.Order  
    freo.me.jaxws.OrderService  
    freo.me.jaxws.OrderServiceImpl  
    freo.me.jaxws.NotFoundException  
      
    From the following file  
    <https://raw.github.com/pzfreo/ox-soa/master/lab-exercises/source/jaxws-service-code.zip>  
      
    You can get the file using curl:  
      
    curl [https://raw.github.com/pzfreo/ox-soa/master/lab-exercises/source/jaxws-service-code.zip -o jaxws-service-code.zip](https://raw.github.com/pzfreo/ox-soa/master/lab-exercises/source/jaxws-service-code.zip%20-o%20jaxws-service-code.zip)
14. Take a good look at the code and understand it.  
      
    Now we can expose this code as a Web Service using the JAXWS wizard built into Eclipse.
15. Select the **OrderService** class and Right-Click on it. Now Select **Web Services**. Then **Create Web Service**.   
    
16. You will see the following dialo
17. Now Click on the **Web Service Runtime** and change it to **Apache CXF v2.x.** Click **Next>**
18. Choose **freo.me.jaxws.OrderServiceImpl** as the Service Implementation.
19. Click **Next>**You will see a screen like this:  
      
      
    Keep everything the same and click Next.
20. You should see:  
    Now make sure Generate Server is clicked. Click **Next>**
21. Do **not** select either UDDI option:  
    
22. Click **Finish**
23. This will generate a number of additional classes.  
    Firstly, it will create a set of classes to map XML types. Secondly it will create a set of annotations in your existing code. Finally it will create a Server class that can be used to run the service standalone.
24. Look through the code and identify all three parts. Take a look at the annotations.
25. You can now run this in one of three ways:  
    \* Using the generated server class   
    \* Deploying in Tomcat using Eclipse  
    \* Deploying in Tomcat standalone
26. To run the Generated Server class, find the class called OrderServiceServer, and right-click on it. Now choose **Run As -> Java Application**  
      
      
    

In the console window you should see output similar to:  
  
Starting Server

Dec 8, 2012 6:46:04 AM org.apache.cxf.service.factory.ReflectionServiceFactoryBean buildServiceFromClass

INFO: Creating Service {http://jaxws.me.freo/}OrderServiceImplService from class freo.me.jaxws.OrderService

Dec 8, 2012 6:46:06 AM org.apache.cxf.endpoint.ServerImpl initDestination

INFO: Setting the server's publish address to be ***http://localhost:9090/OrderServiceImplPort***

Dec 8, 2012 6:46:06 AM org.eclipse.jetty.server.Server doStart

INFO: jetty-8.1.7.v20120910

Dec 8, 2012 6:46:06 AM org.eclipse.jetty.server.AbstractConnector doStart

INFO: Started SelectChannelConnector@localhost:9090

Dec 8, 2012 6:46:06 AM org.apache.cxf.service.factory.ReflectionServiceFactoryBean buildServiceFromWSDL

INFO: Creating Service {http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01}Discovery from WSDL: classpath:/org/apache/cxf/ws/discovery/wsdl/wsdd-discovery-1.1-wsdl-os.wsdl

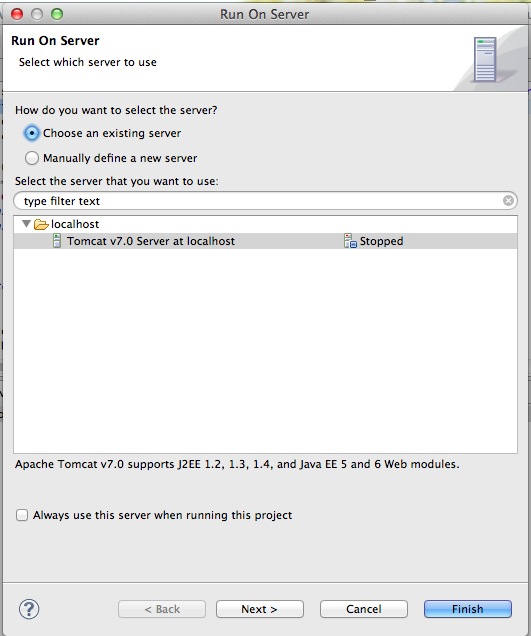
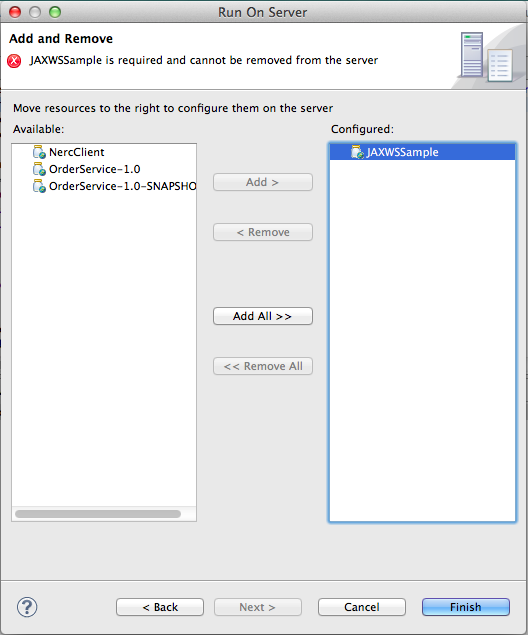
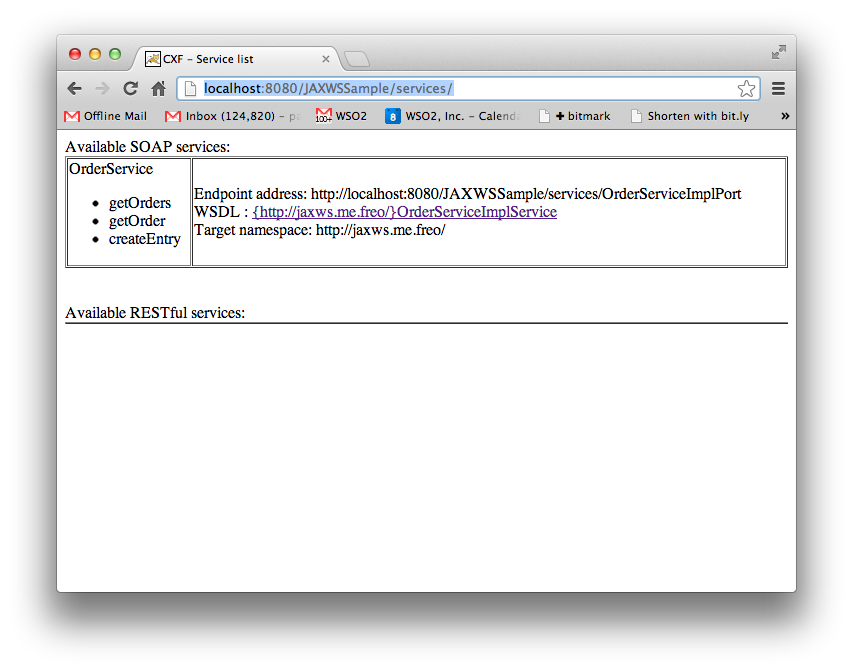
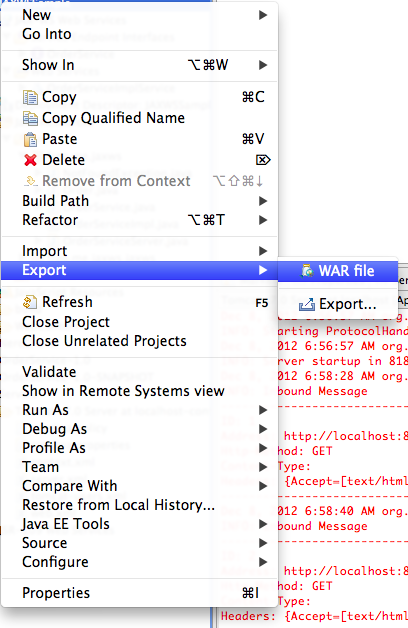
Dec 8, 2012 6:46:06 AM org.apache.cxf.endpoint.ServerImpl initDestination

INFO: Setting the server's publish address to be soap.udp://239.255.255.250:3702

Dec 8, 2012 6:46:06 AM org.apache.cxf.service.factory.ReflectionServiceFactoryBean buildServiceFromClass

INFO: Creating Service {http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01}DiscoveryProxy from class org.apache.cxf.jaxws.support.DummyImpl

Server ready...

1. Notice the line I have bold/italicized. Find the same line in your console and cut/paste it into your browser address. Don’t hit enter (well you can – it won’t explode). Instead add a **?wsdl** to the end. Now hit enter. You should see a WSDL.
2. Try your service out with SOAPUI as before.
3. To run Tomcat locally “within” Eclipse, choose the overall JAXWSSample app in the left hand Project Explorer, right-click and select **Run As->Run on Server**. You will see a dialog like this:  
   ****
4. Choose Tomcat 7.0 and click **Next>**You should see your JAXWSSample app in the Configured column on the right:  
   
5. Click Finish  
   The server will start up and Eclipse will try to start up a browser against your app. There will be a 404 Not Found error because our WAR file has no “Welcome Page”.
6. Browse <http://localhost:8080/JAXWSSample/services/> and you should see a CXF generated page:  
     
   
7. Click on the link and it should take you the URL. Once again make sure everything is working using SOAPUI.
8. Finally, Export your WAR file (Right Click on the Project, Export->WAR File)  
   
9. Stop the Server inside Eclipse, and copy your exported WAR file to Tomcat’s **webapp** directory. Start Tomcat from the command line as before, and test again.
10. Congratulations if you got this far!
11. Extension Exercise.   
      
    Refactor the code between the SOAP and REST services so there is a single model which is exposed as both SOAP and REST.