

# Exercise 4

*Creating a JAX-WS Service*

## **Prior Knowledge**

*Basic understanding of SOAP and WSDL*

## **Objectives**

*Understand how to create Web Services in Java*

## **Software Requirements**

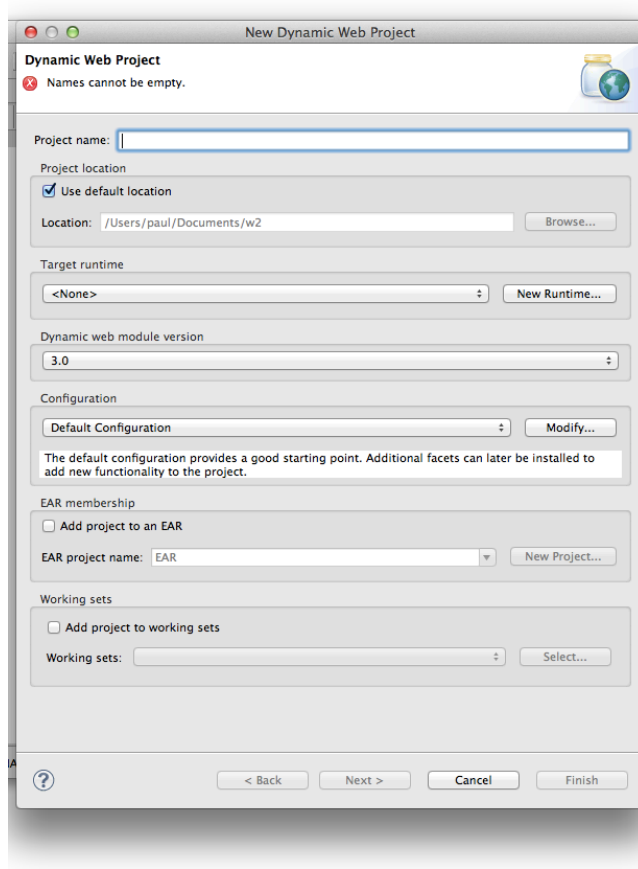
- Java Development Kit 7
- Tomcat 7.0.69
- Eclipse JEE Luna workbench
- Apache CXF 2.7.18
- SOAPUI

Steps:

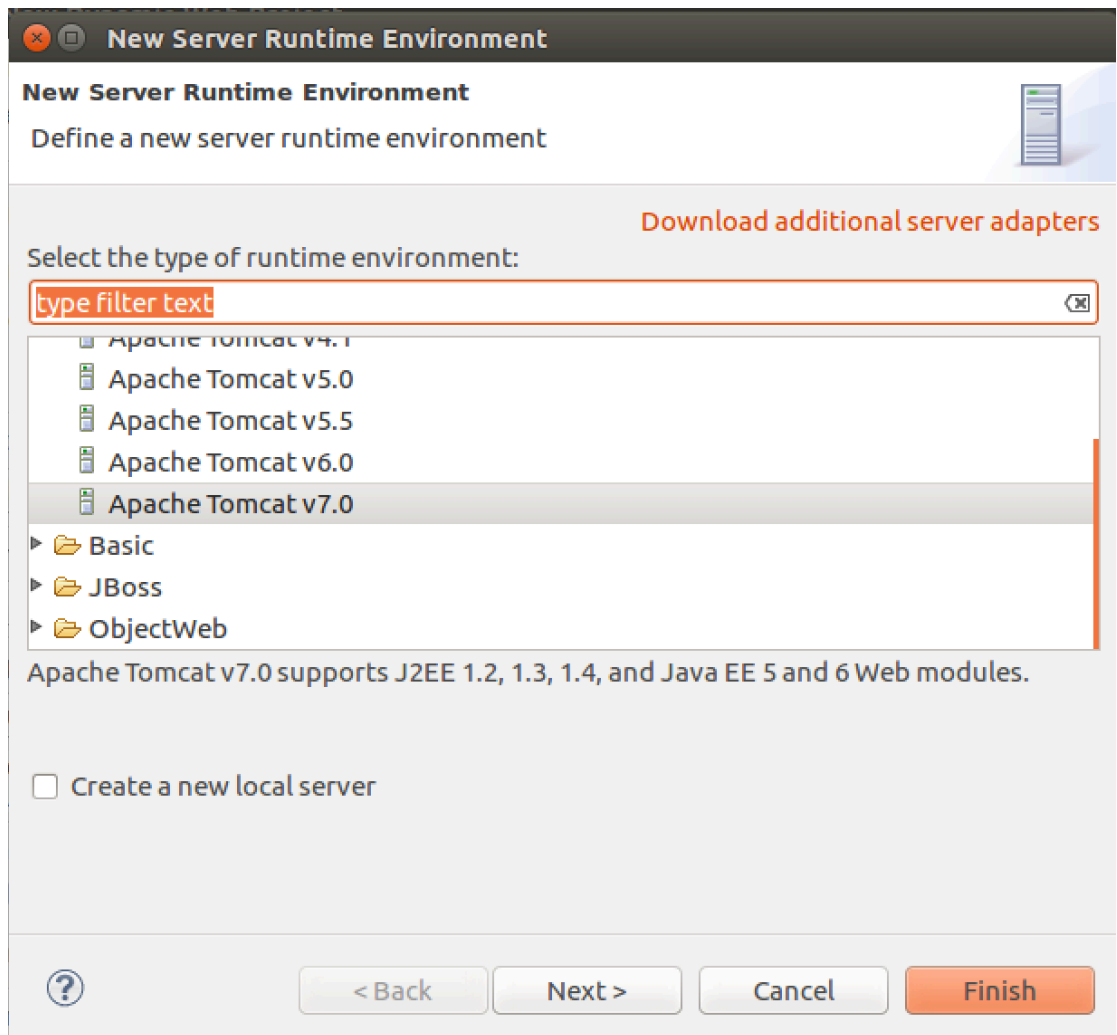
1. Make sure Tomcat is installed (e.g. in ~/servers/tomcat)
2. Check Apache CXF is installed (in ~/servers/apache-cxf-2.7.18/)
3. Open up Eclipse (from the Ubuntu Launcher)



4. Create a new Dynamic Web Project (**File -> New -> Dynamic Web Project**) (if this isn't visible, it may be you are on a different perspective – then choose Other and search for Dynamic).

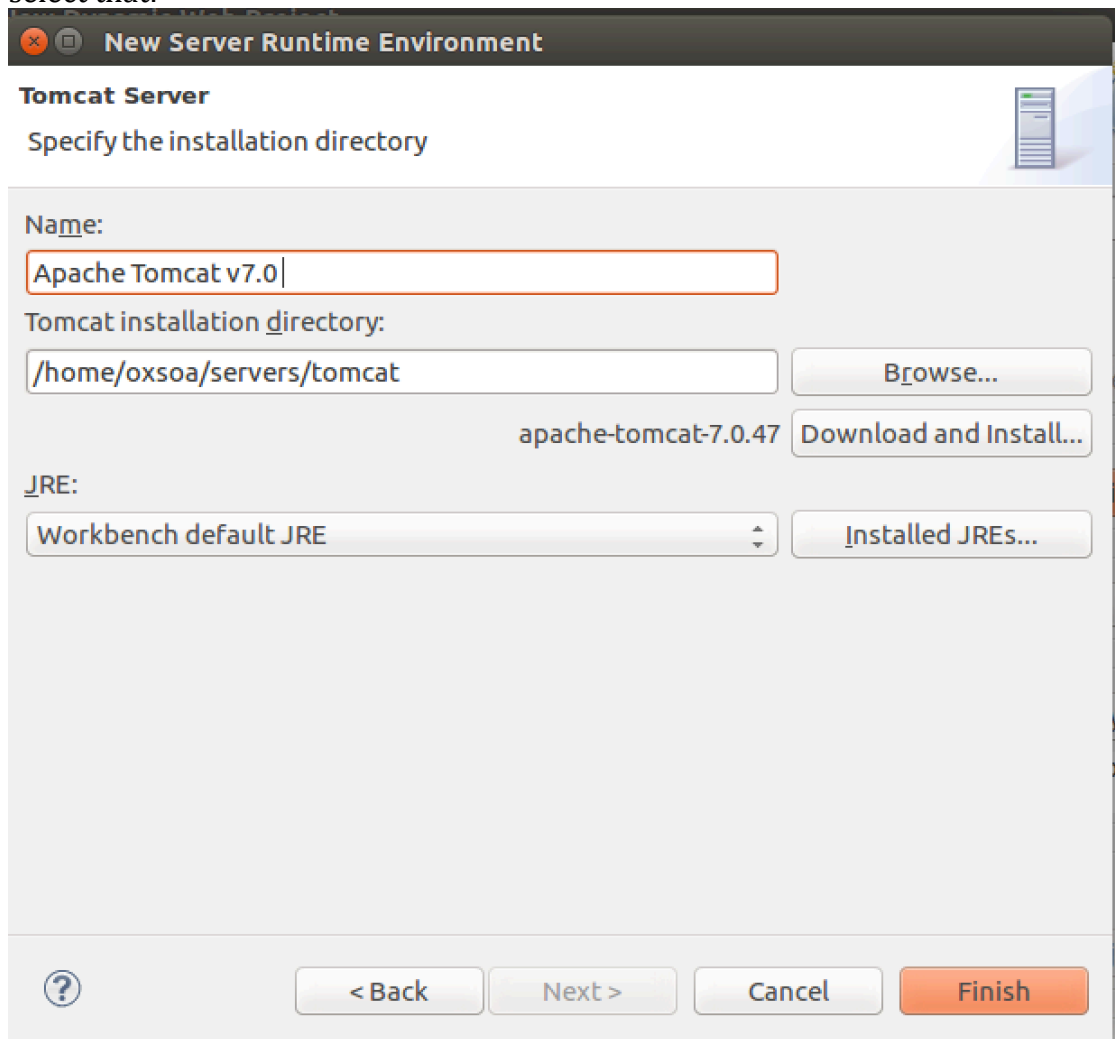


- Now click **New Runtime** and choose Tomcat 7.0



- Click **Next >**

7. Browse to the directory where you have Apache Tomcat installed and select that:

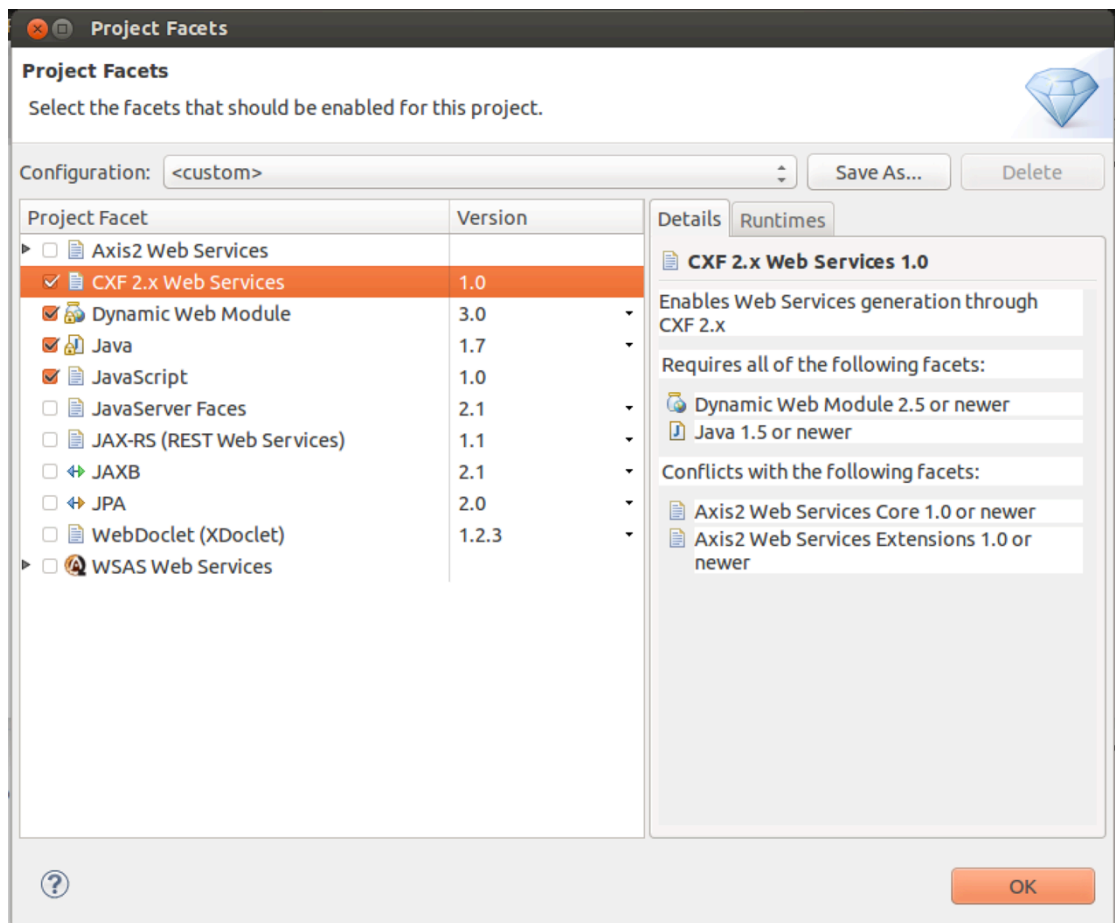


The screenshot shows a dialog box titled "New Server Runtime Environment" with a sub-header "Tomcat Server". Below the sub-header is the instruction "Specify the installation directory". The dialog contains several input fields and buttons:

- Name:** A text field containing "Apache Tomcat v7.0".
- Tomcat installation directory:** A text field containing "/home/oxsoa/servers/tomcat". To the right of this field is a "Browse..." button.
- Version:** A label "apache-tomcat-7.0.47" is displayed next to a "Download and Install..." button.
- JRE:** A dropdown menu showing "Workbench default JRE". To the right is an "Installed JREs..." button.
- Navigation:** At the bottom, there are four buttons: a help icon (?), "< Back", "Next >", "Cancel", and a prominent orange "Finish" button.

Click **Finish**

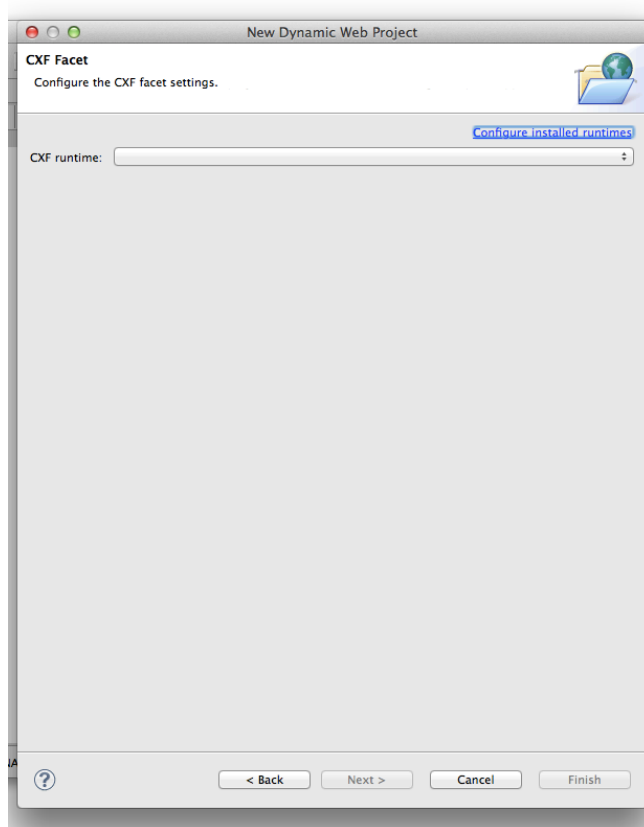
8. Now we need to modify the Tomcat configuration. Click on **Modify** next to Default Tomcat Configuration:



Enable the **CXF 2.x Web Services** section. Click **OK**

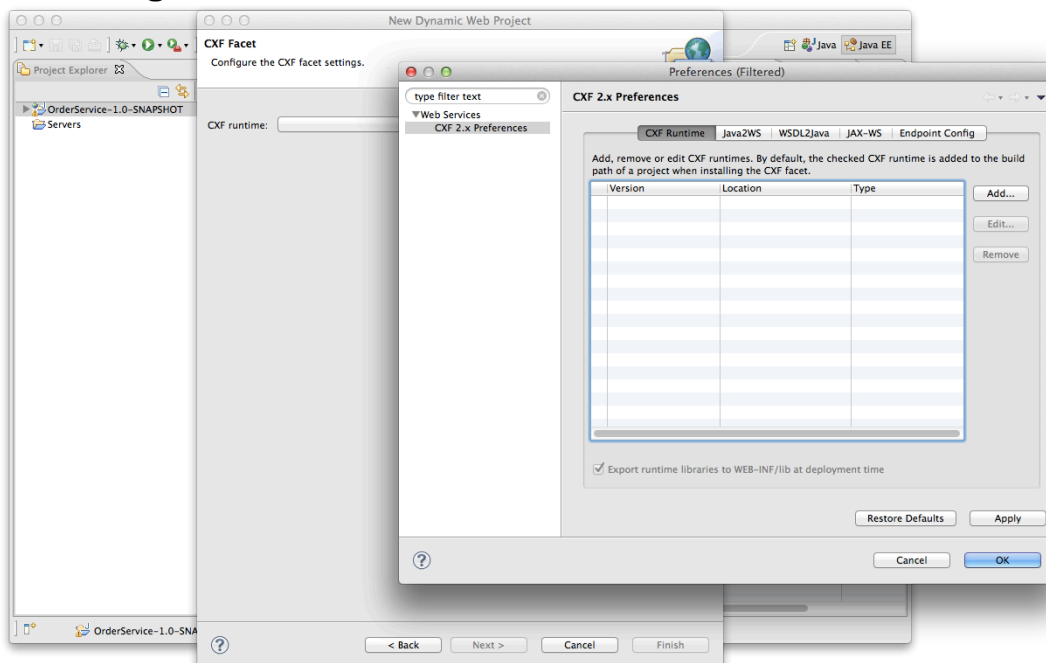
9. Now you should be back in the initial Dialog, so give the project a name, e.g. **JAXWSSample**

10. Click **Next, Next, Next** until you are at the CXF Facet configuration dialog

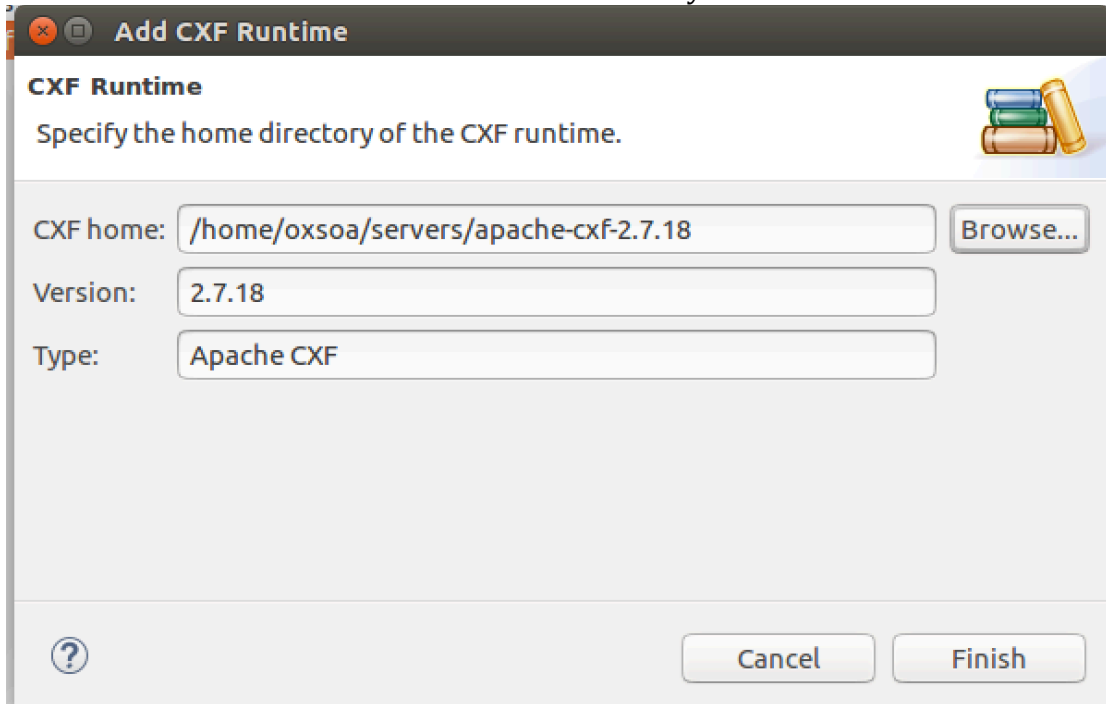


11. *If the CXF 2.7.18 runtime is already configured you can skip the next two steps.*

12. Click **Configure Installed Runtimes**



13. Now click **Add**. Browse to the CXF install directory.



Now click **Finish**

14. Now **select the tick box** next to the CXF 2.7.18 version.

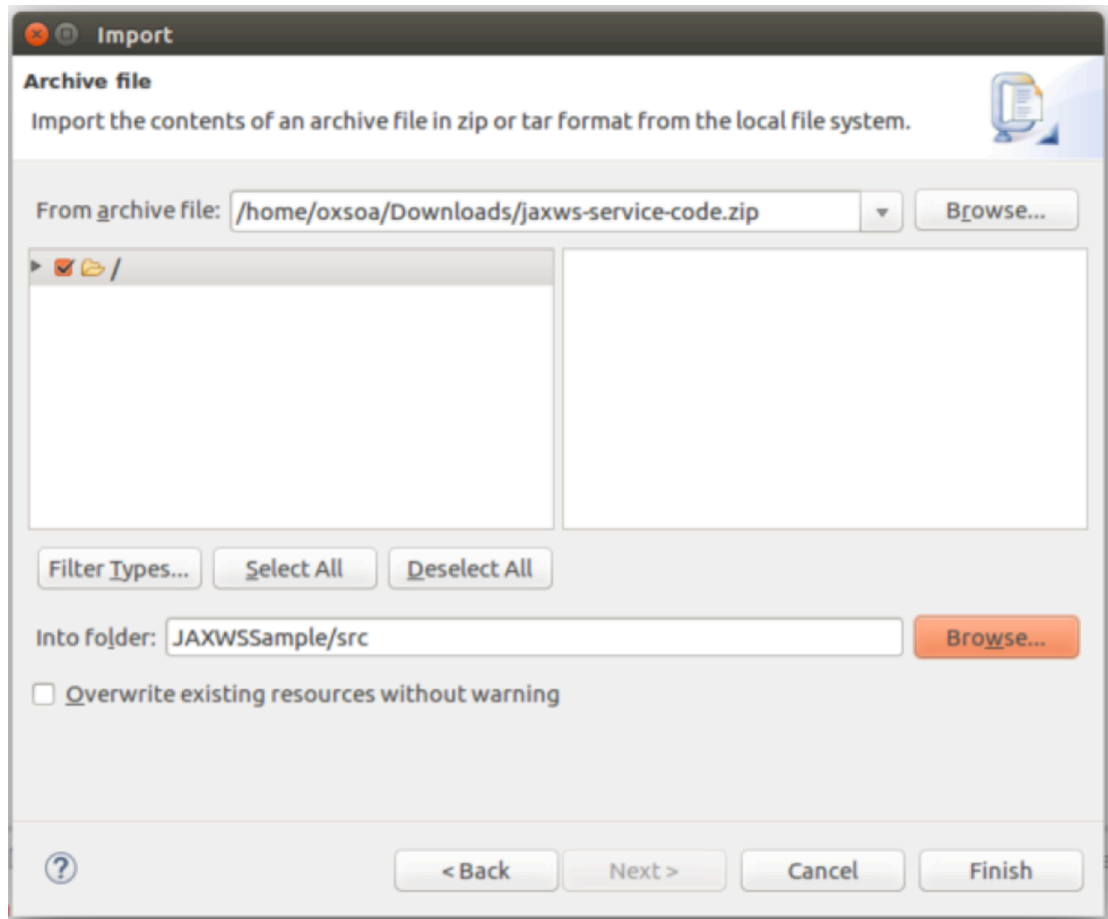
15. Click **OK** and **Finish**. You should now have a project.

16. Import some existing code I have written:

**File->Import / General -> Archive File**

**Browse** to /home/oxsoa/Downloads/jaxws-service-code.zip

Change the Into Folder to be: **JWSSample/src**



17. The package includes the following files:

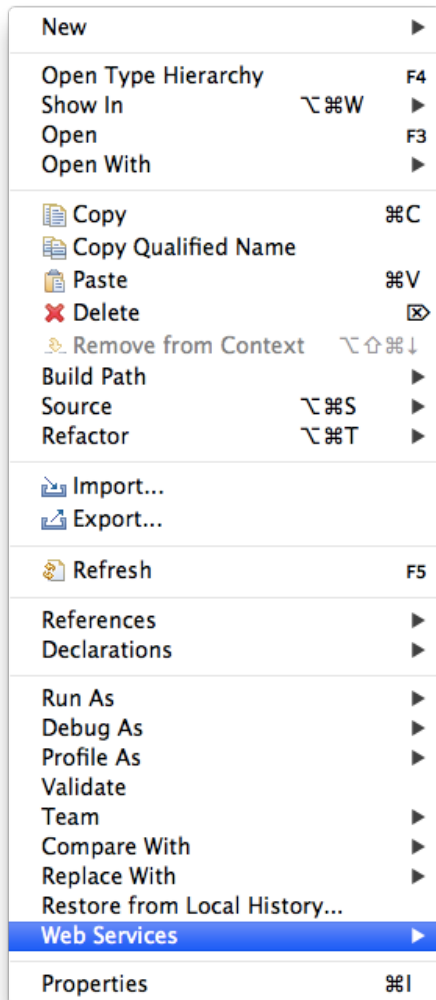
freo.me.jaxws.Order  
freo.me.jaxws.OrderService  
freo.me.jaxws.OrderServiceImpl  
freo.me.jaxws.NotFoundException

*The code is pure Java and therefore you should be able to take a good look at the code and understand it.*

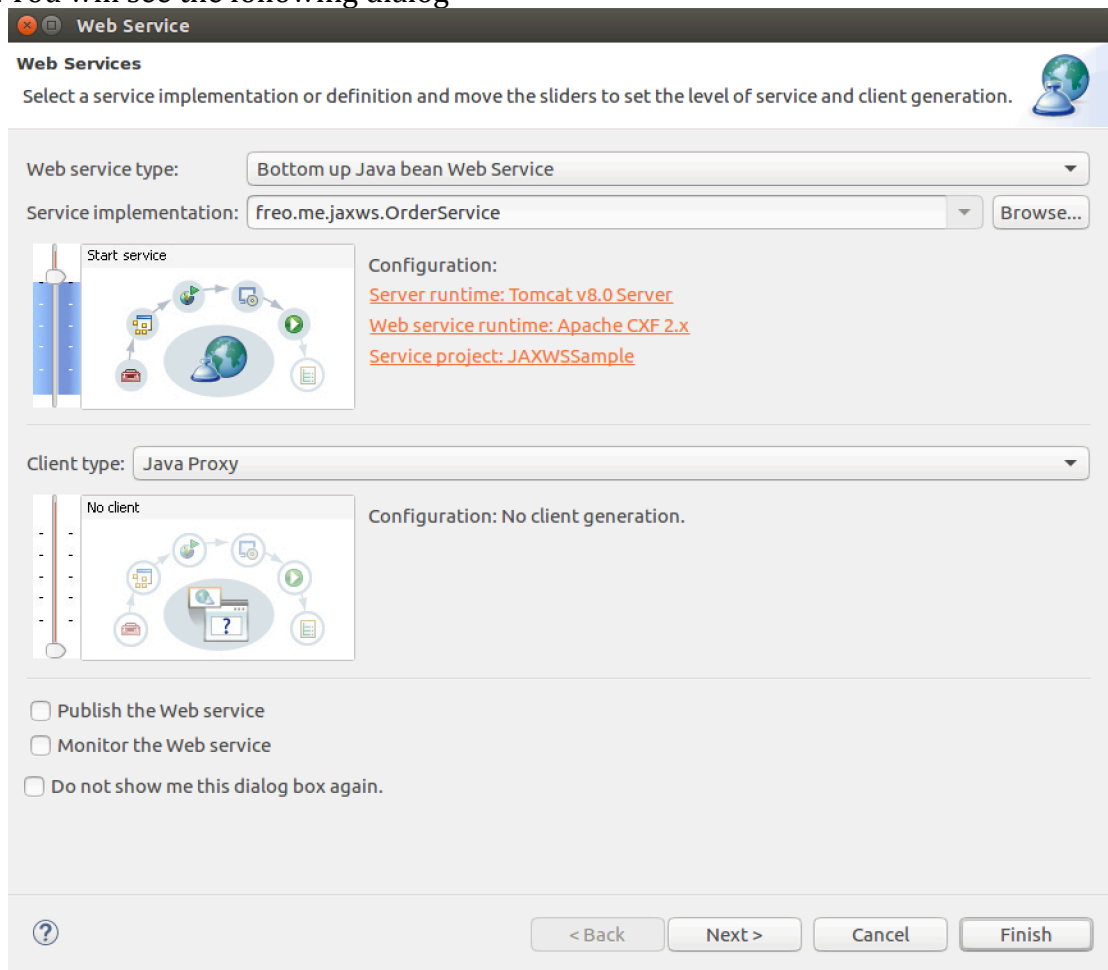
Now we can expose this code as a Web Service using the JAXWS wizard built into Eclipse.



18. Select the **OrderService** class and Right-Click on it. Now Select **Web Services**. Then **Create Web Service**.



19. You will see the following dialog

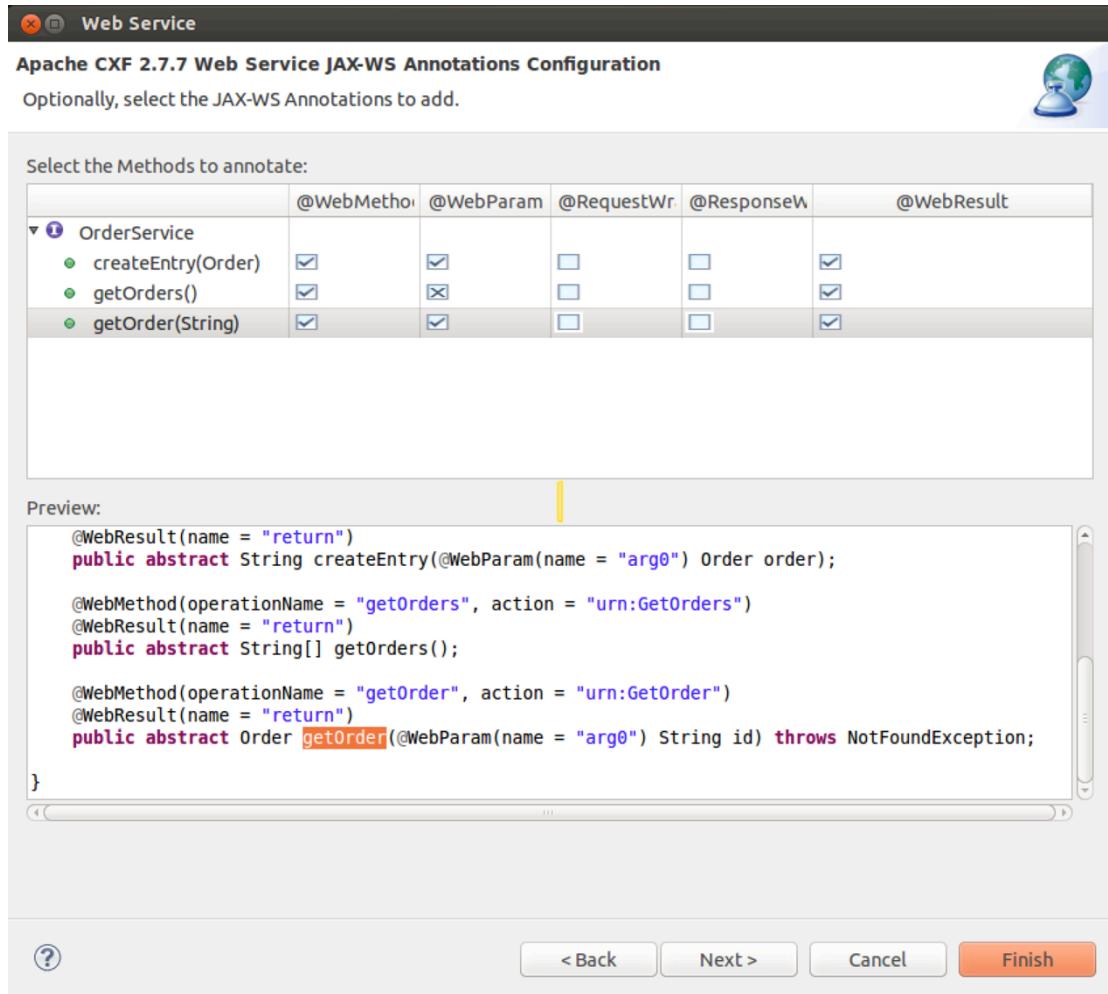


20. Now Click on the **Web Service Runtime** and change it to **Apache CXF v2.x**.  
Click **Next>**

21. Choose **freo.me.jaxws.OrderServiceImpl** as the Service Implementation.  
Click **Next>**

Click **Next>**

You will see a screen like this:



Web Service

Apache CXF 2.7.7 Web Service JAX-WS Annotations Configuration

Optionally, select the JAX-WS Annotations to add.

Select the Methods to annotate:

	@WebMethod	@WebParam	@RequestWr	@ResponseW	@WebResult
OrderService					
createEntry(Order)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getOrders()	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
getOrder(String)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Preview:

```
@WebResult(name = "return")
public abstract String createEntry(@WebParam(name = "arg0") Order order);

@WebMethod(operationName = "getOrders", action = "urn:GetOrders")
@WebResult(name = "return")
public abstract String[] getOrders();

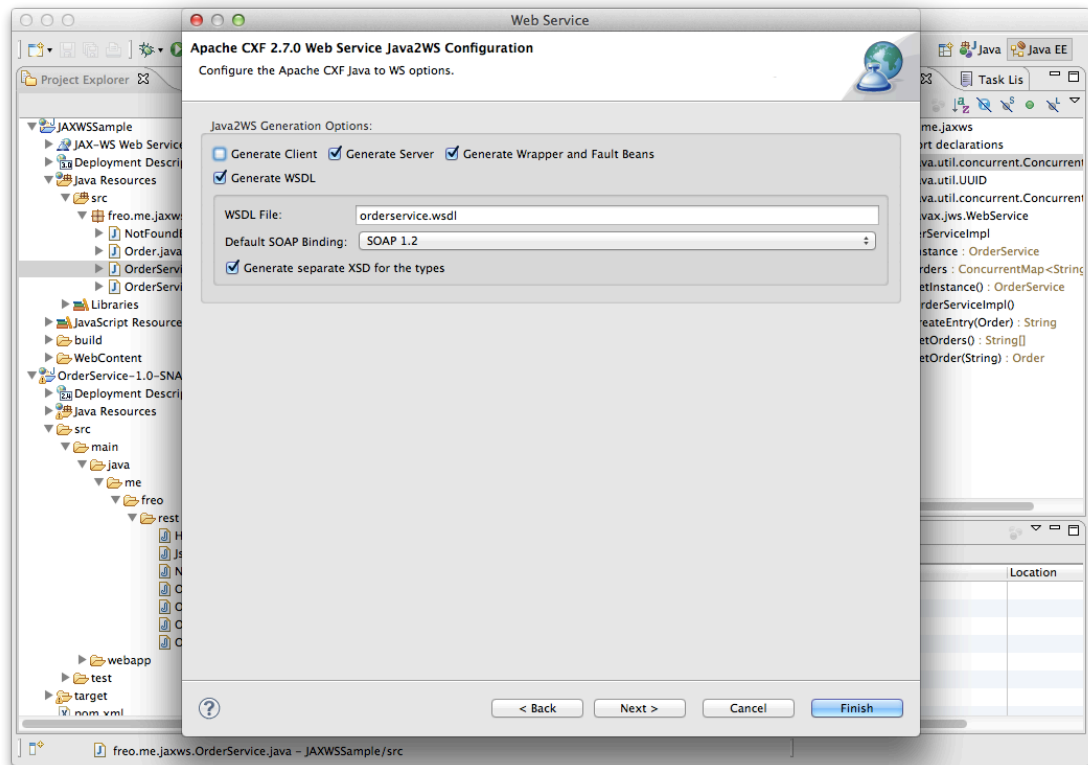
@WebMethod(operationName = "getOrder", action = "urn:GetOrder")
@WebResult(name = "return")
public abstract Order getOrder(@WebParam(name = "arg0") String id) throws NotFoundException;
}
```

< Back Next > Cancel Finish

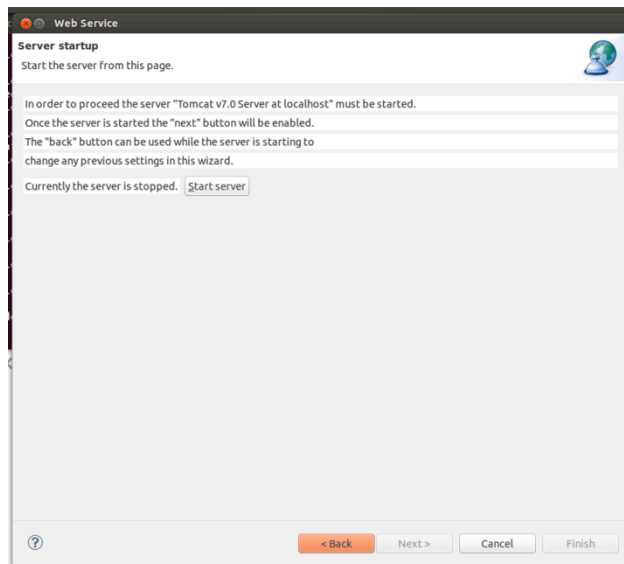
For each method, select the **WebMethod**, **WebParam** and **WebResult** boxes and remove the **Request Wrapper** and **Response Wrapper** checkmarks as shown.

Then click **Next>**.

## 22. You should see:



Now make sure Generate Server is clicked. Click **Next>**

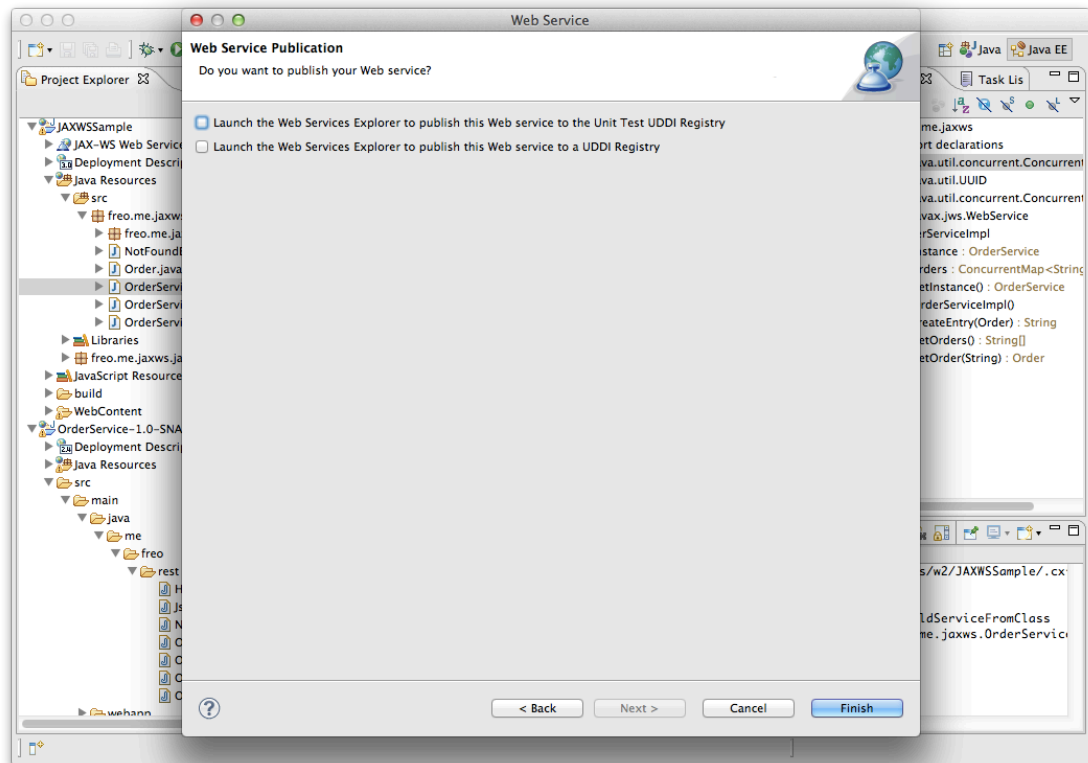


Click **Start Server**

*Hint: if the server fails to start, you may still be running another server (e.g. Tomcat or Node) on port 8080. Close those down.*

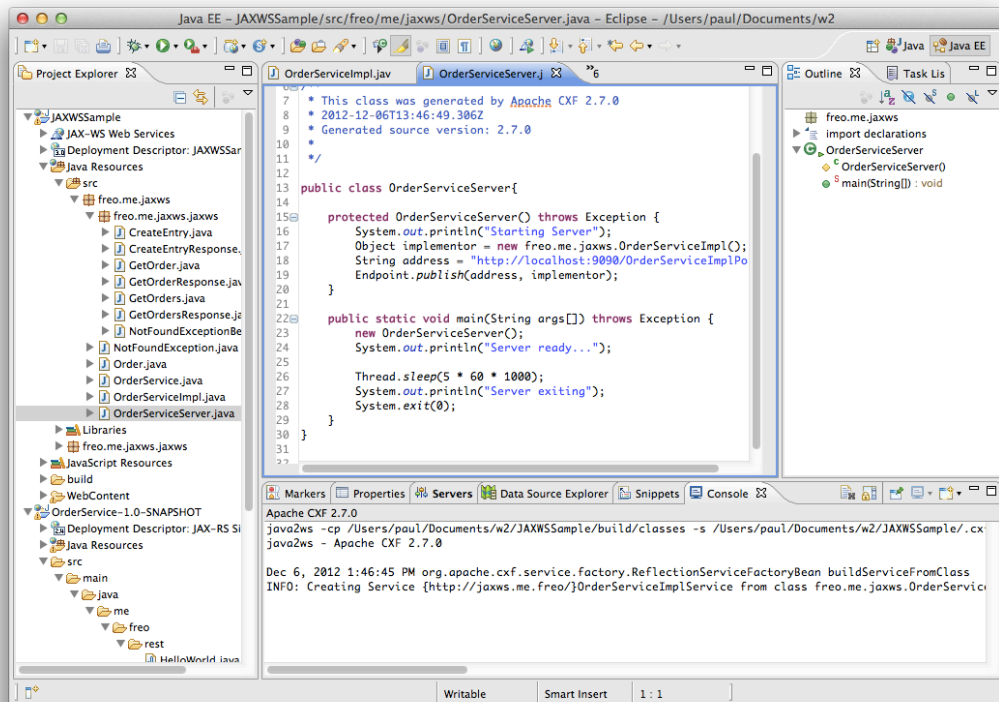
Then **Next**

23. Do **not** select either UDDI option:



24. Click **Finish**

25. 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.

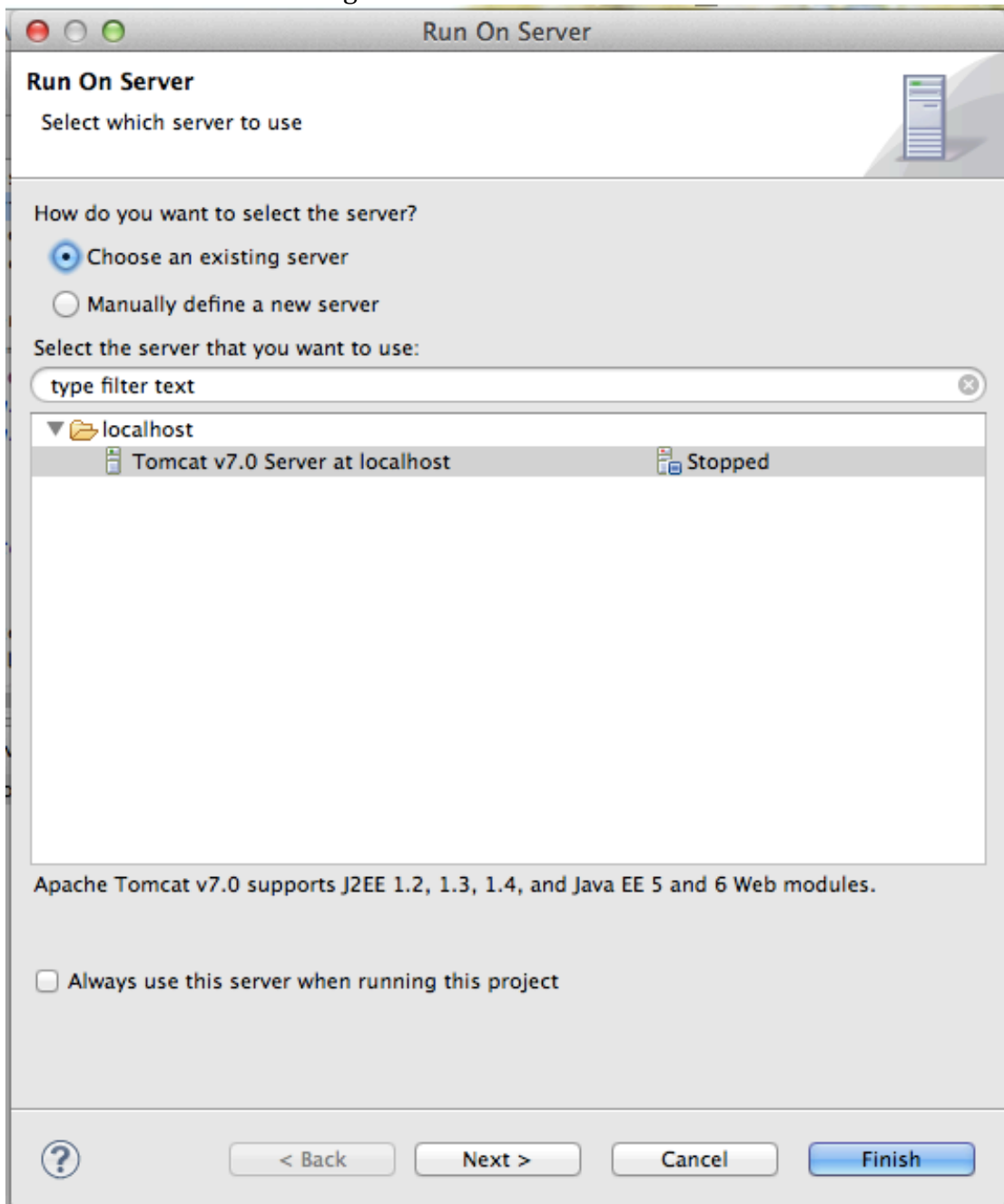
26. Look through the code and identify all three parts. Take a look at the annotations.

27. You can now run this in one of three ways:

- \* Using the generated server class
- \* Deploying in Tomcat using Eclipse **[This should already be running!]**
- \* Deploying in Tomcat standalone

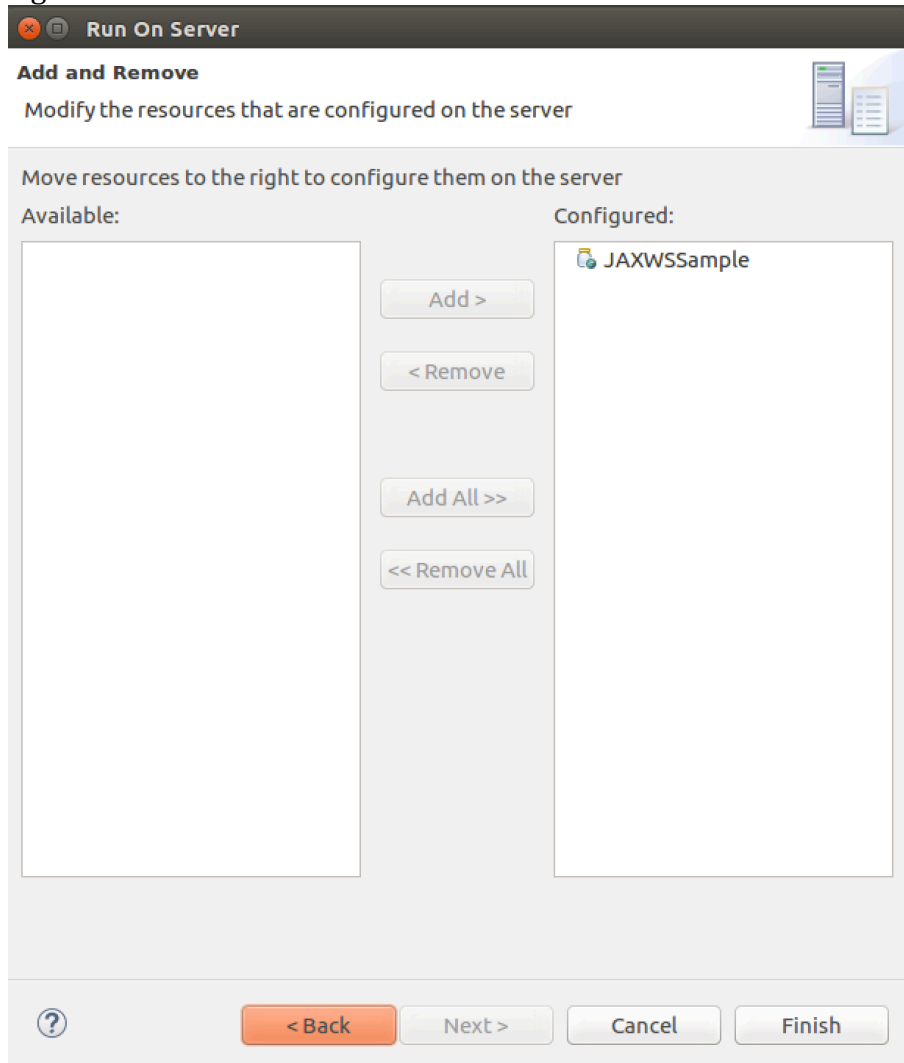
28. We are going to focus on the second option.

29. 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:



30. Choose Tomcat 7.0 and click **Next>**

You should see your JAXWSSample app in the Configured column on the right:

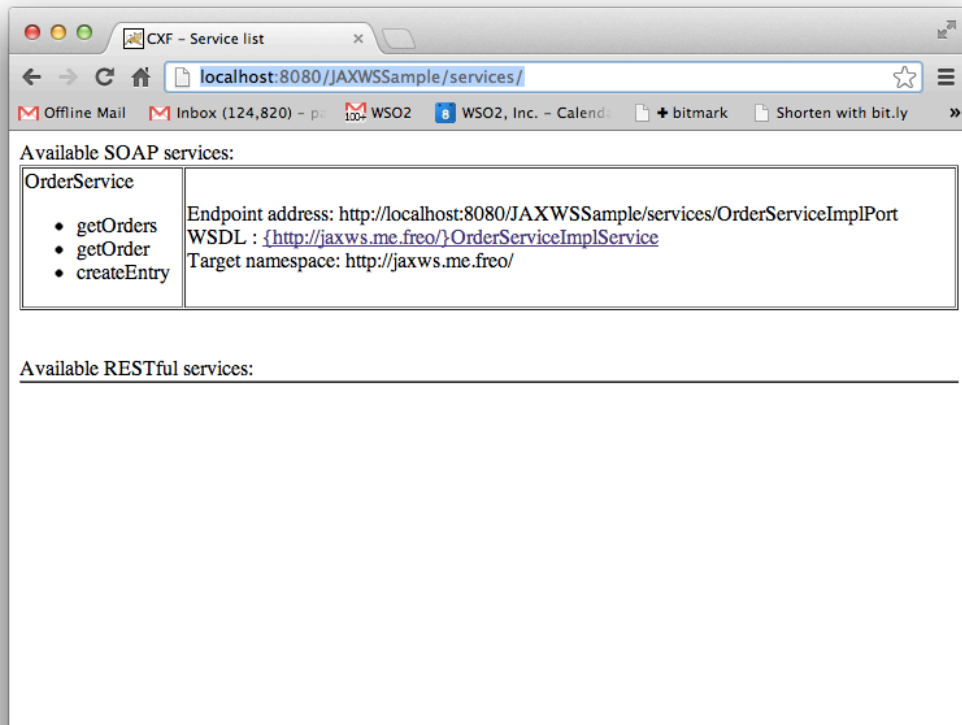


31. Click **Finish**

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".

32. Browse <http://localhost:8080/JAXWSSample/services/> and you should see a CXF generated page:

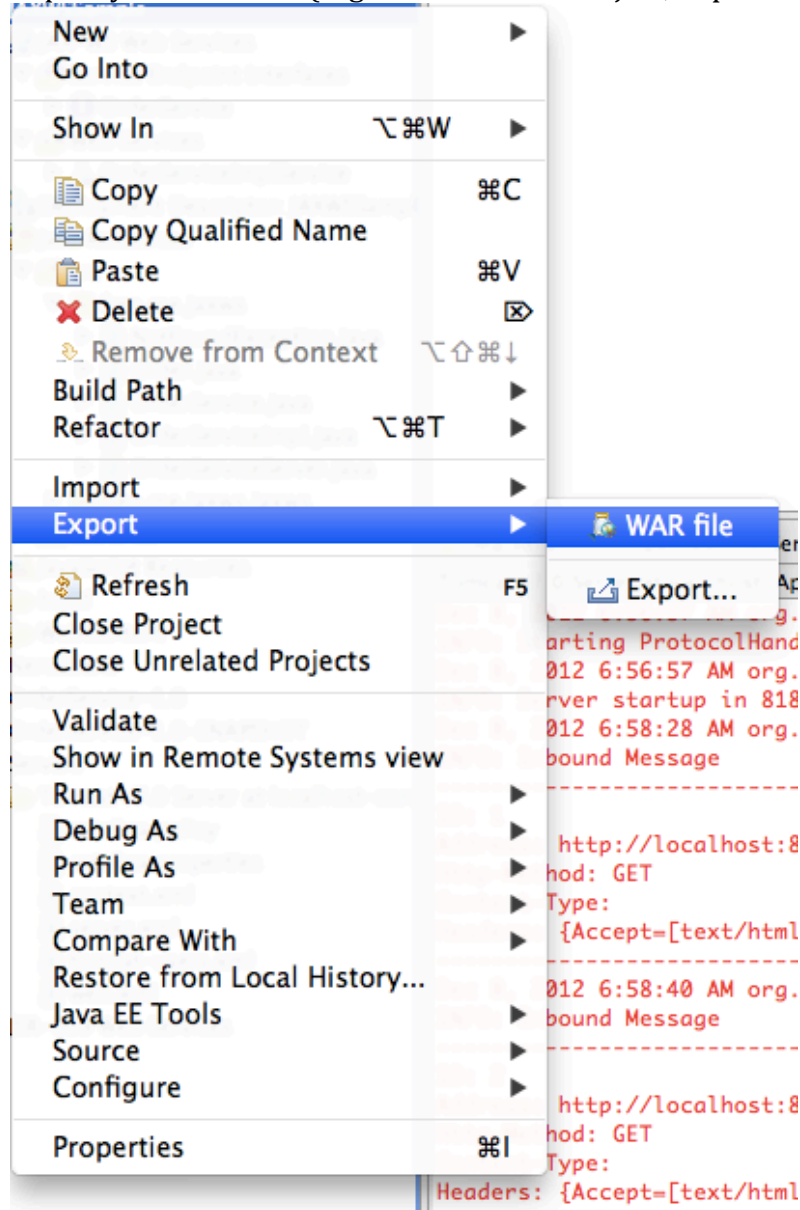




33. Click on the link and it should take you the URL.
34. Import this URL into SOAPUI and test the service.
35. That's it! (unless you fancy an extension).

## Extension

Export your WAR file (Right Click on the Project, Export->WAR File)



36. 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.

37. Congratulations