**Exercise 6b**

*Evolving our Java REST service into something good*

**Prior Knowledge**

Basic understanding HTTP verbs, REST architecture

Some Java coding skill  
Exercise 6a

**Objectives**

Develop a good understanding of JAX RS assertions.  
Understand the Richardson Maturity Model

**Software Requirements**

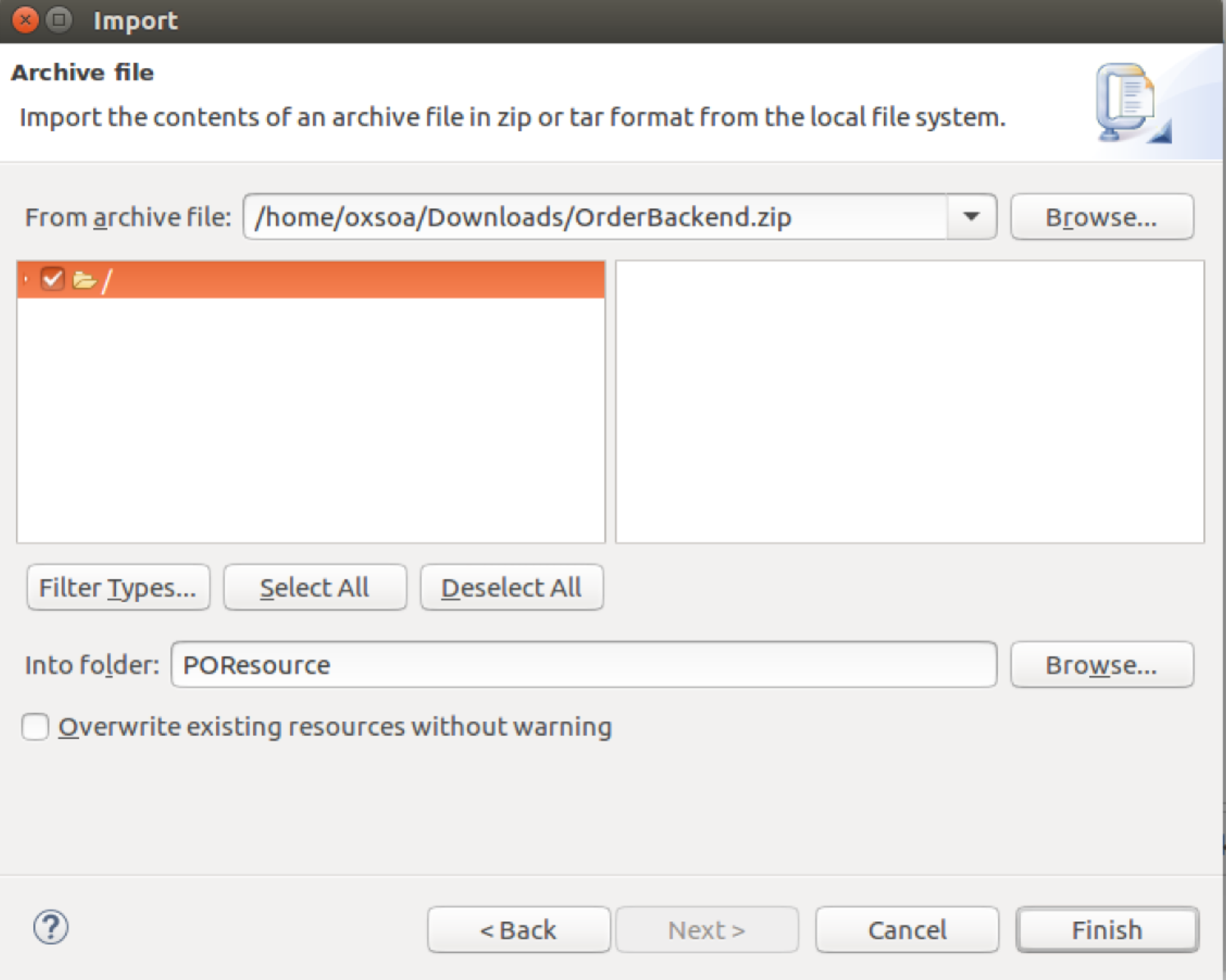
(see separate document for installation of these)

* Java Development Kit 8
* Gradle build system
* Jetty and Jersey
* Eclipse Neon
* curl
* Google Chrome/Chromium plus Chrome Advanced REST extension

**Overview**

*The service we developed in Exercise 6 was frankly pretty useless. It was designed to show a simple framework for building RESTful services in Java and to introduce the build system. Now we need to turn this into something useful.*

**Steps**

1. Import the zip file **~/Downloads/OrderBackend.zip**:  
     
   From project **POResource**  
   Right-click, **Import.. General->Archive File->Next**
2. Browse to the zip file and make sure the import destination is POResource:  
   
3. You will see some project errors. This is because the Eclipse project has not got the right dependencies defined.
4. In the existing POResource project, go to the build.gradle and add the following dependency (in the obvious place!):  
   compile 'org.json:json:20160212'
5. Now right-click on the project POResource and choose **Gradle->Refresh Gradle Project**. This should now look clean, without any compilation problems.  
     
   *By this stage of the exercises, I’m going to assume you are comfortable with coding in Eclipse, and be less directive in the instructions. If you need help, just ask.*
6. Take a look at the new code. There are three main things to note:
   1. I have chosen to use my own serialization into and out of JSON. Jersey, JAXRS and Java have many ways of serializing into and out of JSON, XML and other formats. I recommend you look up Moxy and Jackson as two approaches. However, this approach is perfectly simple.
   2. There is a result which is my “bean” depends on org.json.JSONObject, et al. This is a trade-off.
   3. There is a class OrderInMemory which implements 5 main methods (in pseudo-code)
      1. uuid = createOrder(JSON)
      2. updated = updateOrder(uuid, JSON)
      3. JSON\_array = getOrders()
      4. JSON = getOrder(uuid) throws NotFoundException
      5. Deleted = deleteOrder(uuid)
7. This class is designed to be used as a singleton. This is not a serious implementation model but it saves us needing a real backend database at this stage.   
     
   You can instantiate this class in POResource with the following code   
   (which you should add to POResource).  
   *POX level*

OrderInMemory orderSingleton = **null**;

**public** POResource() {

orderSingleton = OrderInMemory.*getInstance*();  
}

1. Let’s first support creating an Order.  
   In the Richardson Maturity Model, the first stage is to allow POX (Plain Old XML) or in our case, Plain Old JSON. The simplest example of this will be that we post a JSON to the endpoint, and we get back a JSON with some kind of response in it.
2. I have created a Test case for this. The test cases should have loaded alongside the other files you installed but they are currently all commented out. I’d like you to:  
   1. Uncomment out the POST\_JSON.java test case
   2. Create a method inside POResource that handles POST calls and correctly passes the test case.
   3. You might want to review the JAX-RS presentation once again.

*Resource level – improving POST to create a resource*

1. Blah
2. Blah

*HTTP Verbs (Get / Put / Delete)*

1. Blah
2. Blah

*Hypermedia (and Get all orders)*

1. Blah
2. Blah