**Exercise 1**

*Create and consume a REST Service using Apache CXF and JAX-RS*

**Prior Knowledge**

Basic understanding HTTP verbs, REST architecture

Some Java coding skill

**Objectives**

Understand what it takes to create REST services. Understand the REST model. Interact with a REST service using simple web clients in Chrome, on the command line, and in Java.

See how Maven and Tomcat can be used.

**Software Requirements**

(see separate document for installation of these)

* Java Development Kit 7
* Apache Maven 3.0.4 or later
* Eclipse Juno (4.2 SR1) or later – Java Development IDE
* Tomcat 7.0.30 or later
* curl
* Google Chrome plus Chrome Advanced REST extension

Step 1. **Create a new project using Maven**

Maven is a very powerful (and somewhat arcane) build tool. We are going to use Maven to create and build our project.

Maven has the ability to create new projects using “archetypes”.

a. First create a directory structure to store the exercise projects in.

(e.g. ~/oxsoa/, c:\oxsoa\, etc)

From now on this will be referred to as $oxsoa

b. Open a command line and change to that directory

c. Test that you have maven properly installed. Execute

mvn -v

You should see something similar to this (dependent on your machine, JVM, etc)

Apache Maven 3.0.4 (r1232337; 2012-01-17 08:44:56+0000)

Maven home: /Users/paul/Apps/apache-maven-3.0.4

Java version: 1.6.0\_35, vendor: Apple Inc.

Java home: /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home

Default locale: en\_US, platform encoding: MacRoman

OS name: "mac os x", version: "10.8.1", arch: "x86\_64", family: "mac"

d. Use Maven to create a sample project:

Execute

mvn archetype:generate -Dfilter=org.apache.cxf.archetype:

This will prompt you for some choices:

Choose archetype:

1: remote -> org.apache.cxf.archetype:cxf-jaxrs-service (Simple CXF JAX-RS webapp service using Spring configuration)

2: remote -> org.apache.cxf.archetype:cxf-jaxws-javafirst (Creates a project for developing a Web service starting from Java code)

Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): :

Select 1 (*type ‘1’ and hit enter!*)

Now it will ask which CXF **version** to use. The default is the latest (at the time of writing 2.6.2). *Just hit enter*.

Now it asks for a default **groupid**.

Define value for property 'groupId': :

This is a namespace, so choose something meaningful (e.g. com.mycompany.rest, uk.ac.ox.comlab.rest, etc). When creating this lab I chose *me.freo.rest,* so that is what you will see in screenshots, etc.

Define value for property 'artifactId': :

This will be the name of the WAR and the overall maven artifact created. We are going to create an Order service, so type *OrderService***.**

Define value for property 'version': 1.0-SNAPSHOT: :

Just hit *Enter* to accept this version.

Define value for property 'package': com.mycom.restservice: :

This will default to the same namespace you chose for the groupid. That should be fine, so hit *Enter* to accept.

It will then ask you to confirm these settings. Hit *Enter* and it will go and generate the code.

You should see plenty of output explaining what is happening, and also a line showing where the resulting code was placed, e.g.:

[INFO] project created from Archetype in dir: /Users/paul/oxsoa/OrderService

This will have created a set of code and a tree structure for you. If you are on Linux you can use the nice **tree** command to show this:

$ tree

.

└── OrderService

├── pom.xml

└── src

├── main

│   ├── java

│   │   └── me

│   │   └── freo

│   │   └── rest

│   │   ├── HelloWorld.java

│   │   └── JsonBean.java

│   └── webapp

│   └── WEB-INF

│   ├── beans.xml

│   └── web.xml

└── test

└── java

└── me

└── freo

└── rest

└── HelloWorldIT.java

14 directories, 6 files

e. You can now build this code:

cd ~/oxsoa/OrderService [on Linux/Mac]

cd ~\oxsoa\OrderService [on Windows]

mvn clean install

The first time this is run this will download a lot of stuff from the central maven repositories on the web. Depending how fast the network is, maybe a coffee is in order.

This will build **and test** the sample code. Its pretty cool. It actually starts an embedded Tomcat to run the service and call unit tests against it.