# Lab 4 – JAX-RS Book Server

## JAX-RS Server Configuration (as before)

* Choose JEE perspective.
* New – Dynamic Web Project.
  + Target Runtime – accept defaul
  + Configuration - accept default
  + **Generate web.xml deployment descriptor - Y**
* Copy jersey-bundle-1.15 into lib folder and add to your BuildPath.
* Copy into WEB-INF/lib folder
  + jersey-bundle-1.15
  + asm-3.3.1.jar
* Modify web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* id=*"WebApp\_ID"* version=*"3.0"*>

<!-- more stuff here –>

<servlet>

<servlet-name>Jersey REST Service</servlet-name>

<servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>com.sun.jersey.config.property.packages</param-name>

**<param-value>myApp</param-value>**

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>Jersey REST Service</servlet-name>

**<url-pattern>/rest/\*</url-pattern>**

</servlet-mapping>

</web-app>

## Q1. (JAX-RS Books Server)

Create a Dynamic Web Project called j2BookServer.

1. Copy the web.xml from the first project.
2. Create a package called myApp containing the following classes.
3. Create a class called Book with attributes id, title, author, year. Generate getters and setters for the attributes. Add JAXB annotations.
4. Create a class called BookDAO as given in the notes. Create a 3rd Book object there.
5. Create a class called BookResource.
6. Run the application on the apache tomcat server.
7. In the browser send a request for
   * http://localhost:8080/j2bookServer/rest/books

## Q2.

1. Update the server application by adding the method getBook() as outlined in the notes.
2. Test with a browser

## Q3. Module Server Example

1. Write classes Module, ModuleResource and ModuleDAO
2. <http://localhost:8080/j3moduleServer/rest/modules> should produce the following output.

<?xml version="1.0"?>

<modules>

<module>

<id>1</id>

<name>Distributed Systems</name>

<lecturer>EG</lecturer>

<hoursPerWeek>4</hoursPerWeek>

</module>

<module>

<id>2</id>

<name>WebDev</name>

<lecturer>EG</lecturer>

<hoursPerWeek>2</hoursPerWeek>

</module>

</modules>

## Q4.

1. Update the server application by adding the method getModule() to get individual Modules as outlined in the notes for the Books Example.
2. Test with a browser