Exercise with Web Services

Daniel Hagimont daniel.hagimont@irit.fr

The objective of this exercise is to experiment with Web Services (WS). We will use WS in the 2 modes: SOAP and REST.

We will use 3 software:

- Apache Tomcat which is a Web server allowing the execution of servlets written in Java. It allows to make available WS.

- RestEasy which is a development chain for simplifying the development of REST WS.

- eclipse for develoing Java code. It also incluses Axis, a development chain for developing SOAP WS.

1) Using a WS REST + JSON

We are using ReastEasy to simplify the development of REST WS (client and server sides).

A REST WS (students-server) has been implemented. It is available at the following URL:

http://sd-160040.dedibox.fr:8080/students-server/rest/

This WS proposes 2 interfaces :

Method **getstudent**, parameters **firstname** and **lastname** , returns a **JSON**.

Example of invocation :

http://sd-160040.dedibox.fr:8080/students-server/rest/getstudent?firstname=Alain&lastname=Tchana

```
{
        "firstname":"Alain",
        "lastname":"Tchana",
        "birthdate":"18/12/1984",
        "sex":"male",
        "address":"3 rue Jeff Rouchon",
        "city":"Toulouse",
        "zip":"31000",
        "country":"France",
        "phone":"0102030405",
        "email":"alain.tchana@enseeiht.fr",
        "ine":"1111111111"
}
```

Method **getrecord** parameter **ine,** returns a **JSON**.

Example of invocation :

http://sd-160040.dedibox.fr:8080/students-server/rest/getrecord?ine=1111111111

```
{
        "mathematics":"12",
        "middleware":"14",
        "networks":"11",
        "systems":"5",
        "architecture":"16",
        "programming":"18",
        "ine":"1111111111"
}
```

First, you can test the availability of the WS with a web browser and the previous URLs.

Then, implement in Java a client program which invokes this WS (using RestEasy). An exemple is given in the slides of the lecture.

- Create a Java project and add in the buildpath the jars from EasyRest (in lib)
- Create the Java Beans which correspond to the JSON objects. You will have 2 beans : Student and Record. You can use this online service to generate these beans from json examples :

  https://www.site24x7.com/tools/json-to-java.html
- Describe the interface (StudentsInterface) of the WS with annotations @GET, @Path, @Produces …

```java
@Path("/rest")
public interface StudentsInterface {

      @GET
      @Path("/getstudent")
      @Produces({ "application/json" })
      public Student getStudent(
            @QueryParam("firstname") String firstname,
            @QueryParam("lastname") String lastname);
```

- Program a Test class (notice, in the URL, do not include "rest" at the end)

```java
public static void main(String[] args) {

   final String path = "http:·….:8080/students-server";

   ResteasyClient client = new ResteasyClientBuilder().build();
   ResteasyWebTarget target = client.target(UriBuilder.fromPath(path));
   StudentInterface proxy = target.proxy(StudentInterface.class);

   // try invoking one method
```

2) Creation of a REST WS + JSON

Create a REST WS which allows to get from the firstname and lastname of a student the record of his marks (returned as a JSON). Therefore this WS must use the previous WS to first get the student object from the firstname/lastname and then use the previous WS to second get the mark record from the INE of the student.

The interface of this REST WS is :

Method **getmarks**, parameter **firstname, lastname**, returns a **JSON** (record of marks).

The returned JSON looks like :

```
{"mathematics":"12","middleware":"14","networks":"11","systems":"5","architectur
e":"16","programming":"18","ine":"1111111111"}
```

You should be able to test this REST WS with a web browser with the following URL :

http://localhost:8080/marks-server/rest/getmarks?firstname=Alain&lastname=Tchana

- Create a Dynamic Web Project and add in the buildpath the jars from EasyRest (in lib)
- Create a package for your application
- As this WS is client of the previous WS (students-server), you will need the interface of this WS (StudentsInterface) and the classes Record and Student from the previous client.
- You have to implement a class (Marks) for this new REST WS as follows

```
@Path("/rest")
public class Marks {

      final String path = "http:·../students-server";

@GET
@Path("/getmarks")
@Produces({ "application/json" })
public Record getMark(@QueryParam("firstname") String firstname,
                     @QueryParam("lastname") String lastname) {

// invoke getstudent and getmark and return the record
```

- You need to add a class RestApp

```
public class RestApp extends Application {
      private Set<Object> singletons = new HashSet<Object>();
      public RestApp() {
            singletons.add(new Marks());
      }
      public Set<Object> getSingletons() {
            return singletons;
      }
}
```

- You need to a file web.xml in the WebContent/WEB-INF folder

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" version="3.1">
  <display-name>essai-server</display-name>
  <servlet>
    <servlet-name>resteasy-servlet</servlet-name>
    <servlet-class>
            org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
        </servlet-class>
    <init-param>
      <param-name>javax.ws.rs.Application</param-name>
      <param-value>pack.RestApp</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>resteasy-servlet</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
</web-app>
```

In order to use Tomcat :

- Uncompress the archive of Tomcat
- Use the install-tc.sh script to install the jars from RestEasy in Tomcat (you have to adapt the script).
- Go in the bin directory of Tomcat and launch Tomcat : ./startup.sh
- Export the war file from you project and store it in the webapp directory of Tomcat

You can then test your REST WS with the URL :

http://localhost:8080/marks-server/rest/getmarks?firstname=Alain&lastname=Tchana

3) Creation and test of a SOAP WS

You can test the sequence of creation and test of a SOAP WS as seen in the lecture.