

TP Web Services

Daniel Hagimont (daniel.hagimont@irit.fr)

The objective of this lab is to introduce the use of Web Services (WS). We will be working with WS in REST mode.

0) Installs

I assume that vscode and java (JDK 17) are properly installed.

To install Tomcat:

```
cd
```

```
mkdir ws
```

```
cd ws
```

```
wget https://sd-160040.dedibox.fr/hagimont/software/apache-tomcat-11.0.1.tar.gz
```

```
tar xzf apache-tomcat-11.0.1.tar.gz
```

```
# edit your .bashrc and add
```

```
export TOMCAT_HOME=$HOME/ws/apache-tomcat-11.0.1
```

```
export PATH=$TOMCAT_HOME/bin:$PATH
```

Create a new terminal to take into account the modifications in your .bashrc

To start or stop tomcat from a terminal:

```
startup.sh
```

```
shutdown.sh
```

Under vscode, to use Spring-boot :

install this extension: spring initializr java support

1) Creating a WS with Spring-boot

A REST WS has been implemented. It is deployed on a machine in the cloud and is available at the following address:

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

It provides two interfaces:

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

Call example:

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

Call example:

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

```
{
```

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

```
}
```

Test this online WS with your web browser.

Recreate this project with spring-boot in vscode. The sources of the classes are given in the students-server-src folder.

Create Spring-boot project:

View -> Command Palette

-> Spring initializ : Create a Maven Project

-> 3.4.1 -> Java -> n7 -> students-server -> War -> 17

-> Selected 0 dependencies

Select the location where the project is created. Then, open the created project.

Copy classes Student, Record and StudentsController in src/main/java/n7/students_server/

Execute and test this project:

- as a standalone application
 - in the project: ./mvnw spring-boot:run
 - it starts your application in a tomcat server
 - URL: localhost:8080/getstudent....
 - NB : in this mode, you don't have the project name in the URL
- by exporting a war in a running tomcat
 - in the project: ./mvnw package
 - the generated war is located in the target folder
 - start your tomcat (startup.sh)
 - copy the war with the students-server.war name in the webapp folder of your tomcat
 - URL: localhost:8080/students-server/getstudent....
 - NB : in this mode, you have the project name in the URL

You can rely on the slides from the WS lecture.

2) Calling a WS with RestEasy

Implement in a Java project a client program that invokes the previously mentioned service (using reateasy). You need to:

- download reateasy :

<https://sd-160040.dedibox.fr/hagimont/software/reteasy-jaxrs-3.0.9.Final-all.zip>

- create a Java project (students-client) in vscode
- include in your java project (referenced libraries) the reteasy jars (in lib)
- create the Java beans associated with the used Json (Student, Record)
- describe the service interface with annotations @GET, @Path, @Produces ...
- create and use the reteasy proxy
- an exemple is detailed in the slides from the WS lecture.
- we provide a script (run.sh) to execute the reteasy client (update it if necessary), or you can run the application in vscode

3) Develop a new WS

Create a REST WS (named students-marks) which returns, from the firstname/lastname of a student and the name of a lecture, the mark that this student obtained in that discipline. This WS is both client (from the previous WS, students-server) and server as it provides an interface:

Method **getmark**, parameters **firstname**, **lastname** and **lecture**, returns an **Integer**.

Call example:

http://localhost:8080/students-marks/getmark?
firstname=Alain&lastname=Tchana&lecture=systems

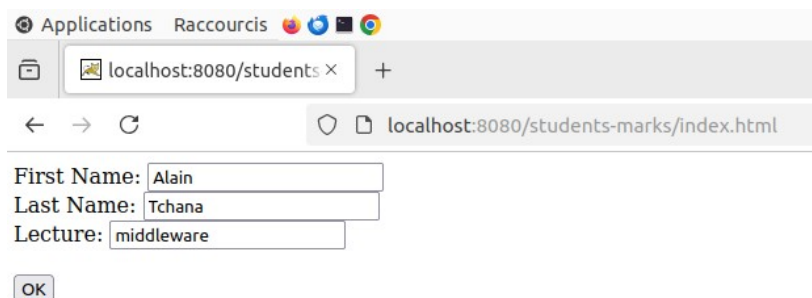
5

To implement this WS, you have to create a new Spring-boot project. You can refer to the project provided in 1) for guidance

To invoke service students-server from service students-marks, we will use the RestClient API from Spring. An example is detailed in the slides from the WS lecture.

Implement this WS and test it with a web browser.

Another way to test your WS is to use a little web page (that we provides, index.html) which includes a JavaScript code sequence which invokes this REST API. Install this web page in the students-marks project (in src/main/resources/static) and test it with this page.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/students-marks/index.html'. Below the address bar, there is a form with three input fields: 'First Name:' with the value 'Alain', 'Last Name:' with the value 'Tchana', and 'Lecture:' with the value 'middleware'. An 'OK' button is located below the form fields.