Session 1. Java Servlets

Introduction

In this session we will learn to develop a web application with Java servlets, a convenient mechanism of programming the back end of a web application (or any other kind of client-server distributed application) in Java without the programmer having to deal with the HTTP protocol (or any other protocol) directly.

The objectives of this assignment are to:

- Understand the context for servlets by setting up Tomcat, a simple servlet-capable web server.

- Understand what a servlet is.

- Learn the basic servlet APIs.

- Implement a simple servlet-based web application.

Each group will have to:

- Tutorial: Follow a brief tutorial about how to develop the back end (server side) of a web applications with servlets and Tomcat 9.

- Assignment (basic part): Complete the lab assignment consisting on developing a simple car rental web application.

- Extensions: Optionally complete one of the suggested extensions.

- Write a .pdf report describing the steps taken to complete the assignment, including screenshots of the application output.

1. Java Servlets with Tomcat 9, a quick tutorial

1.1 Booting the machine

Select the latest Ubuntu imatge (e.g. Ubuntu 14)

user: alumne

pwd: sistemes

1.2 Install Tomcat 9

1) Open a terminal (CTRL+ALT+T).
2) Let's start by updating the Ubuntu's Package Index:

   sudo apt-get update
3) Check if a Java SDK is installed:

   javac -version
4) If not installed do the following to install OpenJDK:

   sudo apt-get install default-jdk
5) Install Tomcat 9:

   wget https://gitlab.fib.upc.edu/pti/pti/raw/master/p1_servlets/apache-tomcat-9.0.5.tar.gz

   tar -xvzf apache-tomcat-9.0.5.tar.gz
6) Enter the Tomcat 9 directory (we will work from here from now on):

   cd apache-tomcat-9.0.5
7) Launch Tomcat 9:

   ./bin/startup.sh &
8) Check if it's running (with the browser): *http://localhost:8080/*

1.3 Create and display a simple HTML page

1) Type the following commands:

mkdir webapps/my_webapp

vi webapps/my_webapp/index.html

```
<html>
  <h1>Hello World!</h1>
</html>
```

2) Check: *http://localhost:8080/my_webapp*

1.3 Create and simple servlet

1) Type the following commands:

mkdir webapps/my_webapp/WEB-INF

vi webapps/my_webapp/WEB-INF/web.xml

```
<web-app>
  <servlet>
    <servlet-name>my_servlet</servlet-name>
    <servlet-class>mypackage.MyServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>my_servlet</servlet-name>
    <url-pattern>/my_servlet</url-pattern>
  </servlet-mapping>
</web-app>
```

2) Type the following commands:

mkdir webapps/my_webapp/WEB-INF/classes

mkdir webapps/my_webapp/WEB-INF/classes/mypackage

vi webapps/my_webapp/WEB-INF/classes/mypackage/MyServlet.java

```
package mypackage;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<html><big>I'm a servlet!!</big></html>");
  }
}
```

3) Type the following commands:

javac -cp lib/servlet-api.jar webapps/my_webapp/WEB-INF/classes/mypackage/.java*

4) After changing the .class files, it may be necessary to restart Tomcat. You can do it this way:

./bin/shutdown.sh

./bin/startup.sh &

5) Check browser:

http://localhost:8080/my_webapp/my_servlet

6) Check errors (replace XX-XX by the current date):

tail -n 200 logs/localhost.2018-XX-XX.log

tail -n 200 logs/catalina.out

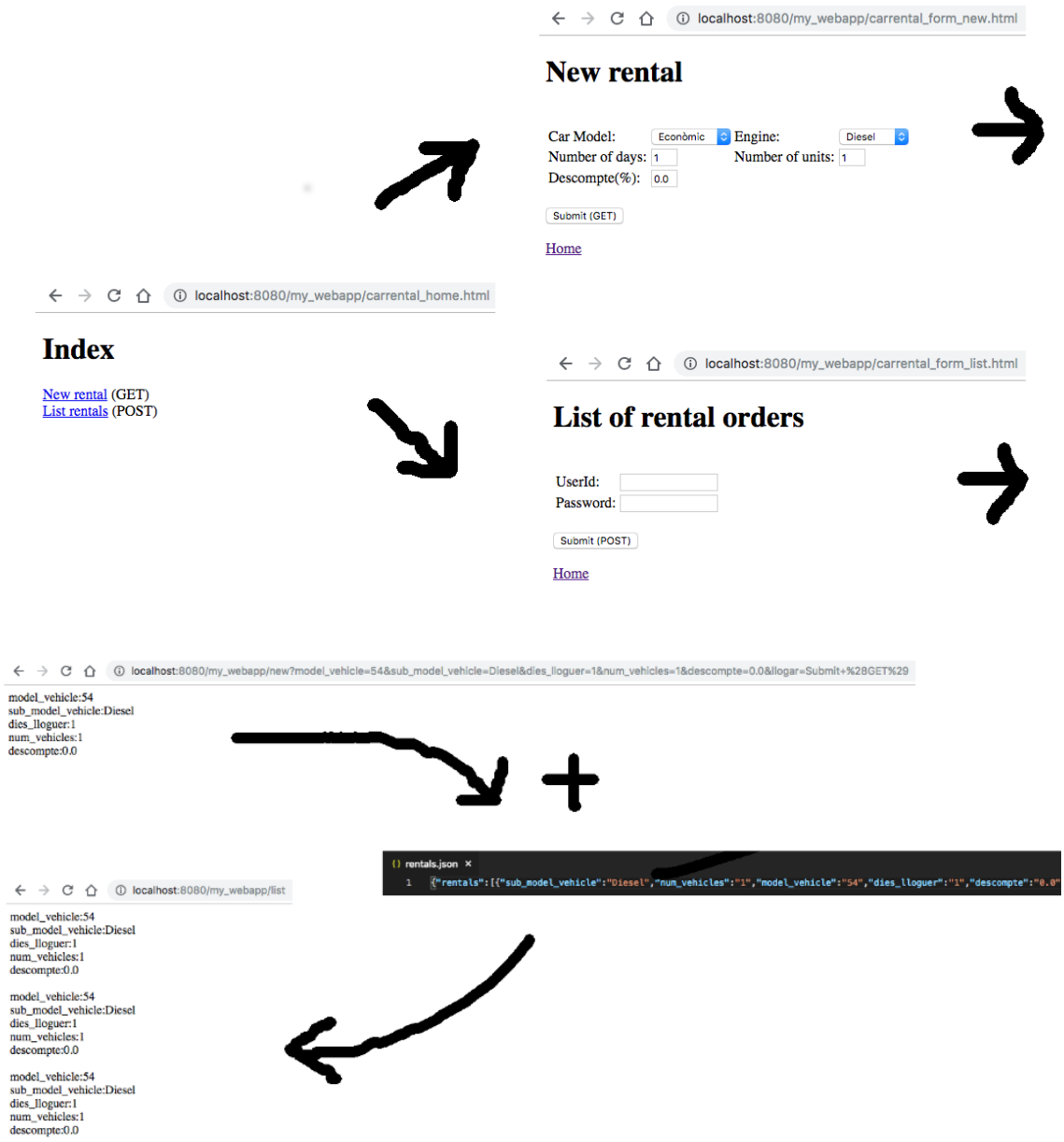
7) It's useful to open a dedicated terminal and check errors continuously:

tail -f 200 logs/localhost.2018-XX-XX.log

2. Lab assignment (basic part): Creating your own car rental web page

You have to program a simple car rental web application using Tomcat and servlets. It will consist in two functionalities:

- **Request a new rental:** A form to enter a new rental order. Input fields will include the car maker, car model, number of days and number of units. If all data is correct the total price of the rental will be returned to the user along with the data of the requested rental.
- **Request the list of all rentals:** A form asking a password (as only the administrator can see this information) that will return the list of all saved rental orders.



Both functionalities will consist in a request form plus a response page. To make it simple it's recommended that the request forms are static HTML pages and the responses are HTML dynamically generated from the servlets. For simplicity, in case of invalid input we will not show the request form again (though you can do it if you want).

In order to keep the rentals data (to be able to list them) you will need to save the data to the disk. A single text file where each line represents a rental will be enough (though not in a real scenario). We recommend you using JSON for writing/reading rental orders to disk. We have included json-simple-1.1.1.jar (http://www.mkyong.com/java/json-simple-example-read-and-write-json/).

2.1 Install the provided sources

In order to help you, some files are provided:

- An HTML index file: carrental_home.html
- A rental request HTML form: carrental_form_new.html (it calls CarRentalNew.java)
- A rentals list request HTML form: carrental_form_list.html (it calls CarRentalList.java)
- Two servlets (partially programmed): CarRentalNew.java and CarRentalList.java.
- A JSON library: json-simple-1.1.1.jar.

NOTE: It's not compulsory to use these files within the solution (you may, for instance, prefer to generate the forms dynamically from the servlets).

In order to install the provided files perform the following steps:

1) Download the files:

   wget https://gitlab.fib.upc.edu/pti/pti/raw/master/p1_servlets/carrental.tar.gz

   tar -xvzf carrental.tar.gz

2) Move the files to Tomcat and compile the servlets:

   *mv *.html webapps/my_webapp/*

   *mv *.java webapps/my_webapp/WEB-INF/classes/mypackage*

   mkdir webapps/my_webapp/WEB-INF/lib

   mv json-simple-1.1.1.jar webapps/my_webapp/WEB-INF/lib

   javac -cp lib/servlet-api.jar:webapps/my_webapp/WEB-INF/lib/json-simple-1.1.1.jar webapps/my_webapp/WEB-INF/classes/mypackage/.java*

3) Add two new servlet definitions to web.xml:

vi webapps/my_webapp/WEB-INF/web.xml

```
<web-app>
  <servlet>
    <servlet-name>new</servlet-name>
    <servlet-class>mypackage.CarRentalNew</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>new</servlet-name>
    <url-pattern>/new</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>list</servlet-name>
    <servlet-class>mypackage.CarRentalList</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>list</servlet-name>
    <url-pattern>/list</url-pattern>
  </servlet-mapping>
</web-app>
```

4) Check the following link and its sublinks:

http://localhost:8080/my_webapp/carrental_home.html

Now add the necessary code to CarRentalNew.java and CarRentalList.java to make the application work properly.

3. Submission

You need to upload the following files to your BSCW's lab group folder before the next lab session:

- A tarball containing the source files.
- A .pdf with a report describing the steps taken to complete the assignment, including screenshots of the application output.

4. Grading. Basic part and extensions.

Completing the basic part of the assigment will let you obtain a maximum of 8 points over 10. In order to obtain the maximum grade you can complete any of the following extensions:

- Configure HTTPS in Tomcat.
- Dockerize your web application.
- Save the data within a database (e.g. MySQL) instead of a file.

5. Further reading

Java Servlet Technology section of the "Java Platform, Enterprise Edition (Java EE) 8. The Java EE Tutorial".