Servidor amb Java Servlets

Projecte Tecnologies d'informació

Pau Coma Ninot

Àlex Aguilera Martínez

Grupo 20

1. Objectius pràctica

L'objectiu d'aquesta pràctica és la creació d'una aplicació web senzilla de lloguer de cotxes implementant pàgines html i jsp, i fent ús d' Apache TomCat i de Java Servlets.

Les dues funcionalitats bàsiques de l' aplicació seran les següents:

- Fer una nova petició de lloguer de vehicles amb un formulari que especifica una sèrie de paràmetres.
- Obtenir una llista amb els cotxes llogats anteriorment i accedir-hi utilitzat una contrasenya que només coneix l'administrador.

Per guardar les peticions de lloguer farem servir un fitxer JSON, al qual hi accedirem amb una llibreria especialitzada.

Els fitxers a programar són CarRentalNew.java i CarRentalList.java que implementen cadascuna de les dues funcionalitats mencionades. La resta de fitxers HTML o XML han estat proporcionats pel professorat per agilitzar la feina.

2. Implementació

1- Instal·lació i configuració

Aquesta part és molt guiada i dóna poc espai a la creativitat. Hem seguit els passos indicats per la correcta instal·lació de l' entorn:

En primer lloc, ha estat necessari instal·lar Java a la màquina fent ús dels privilegis de superusuari. Amb "javac -version" vam comprovar que efectivament no estava instal·lat i seguidament vam procedir amb la instal·lació amb "sudo apt-get install default-jdk".

El pas següent va ser instal·lar el servidor Apache TomCat, obtenint-lo del gitlab de la FIB amb les següents comandes:

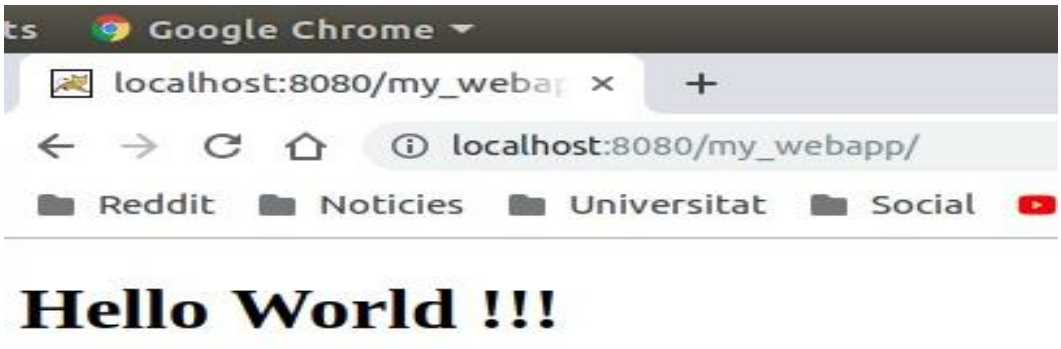
$wget https://gitlab.fib.upc.edu/pti/pti/raw/master/servlets/apache-tomcat-9.0.5.tar.gz

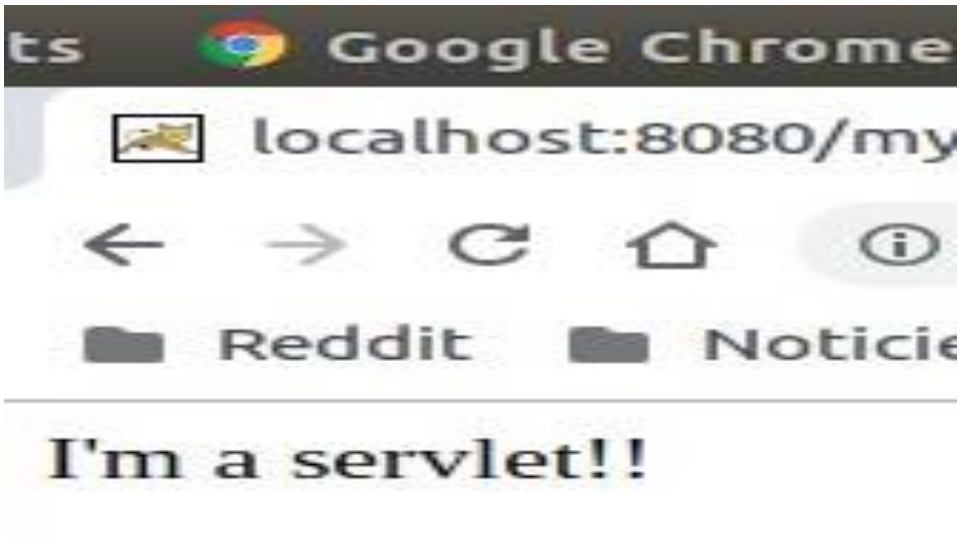
$tar -xvzf apache-tomcat-9.0.5.tar.gz

Un cop instal·lat tant Java com TomCat, ja podiem inicialitzar el servidor amb "./bin/startup.sh &" i accedir a http://localhost:8080/.

Per acabar aquesta part, en mode d'exemple, vam crear una pàgina HTML simple i un Java servlet (codi proporcionat pel professorat).

Després de reiniciar el servidor amb les seves respectives comandes, vam poder observar el següent:





2- CarRentalNew.java

Primer de tot, agafem els atributs del formulari que ens passa el fitxer html:

```
/* Obtenemos valores que introduce el usuario */
String model_vehicle = req.getParameter("model_vehicle");
String sub_model_vehicle = req.getParameter("sub_model_vehicle");
String dies_lloguer = req.getParameter("dies_lloguer");
String num_vehicles = req.getParameter("num_vehicles");
String descompte = req.getParameter("descompte");
```

Després mostrem els valors obtinguts per pantalla:

```
/* Mostramos los valores */
out.println("<html> model_vehicle:"+ model_vehicle + "<br> sub_model_vehicle:"+
            sub_model_vehicle + "<br> dies_lloguer:"+ dies_lloguer+ "<br> num_vehicles :"
            + num_vehicles + "<br> descompte:" + descompte + "</html>");
```

Llegim el fitxer JSON que tenim creat i afegim els lloguers de cotxes ja registrats en una llista. En cas que l' arxiu JSON sigui buit, no fa res.

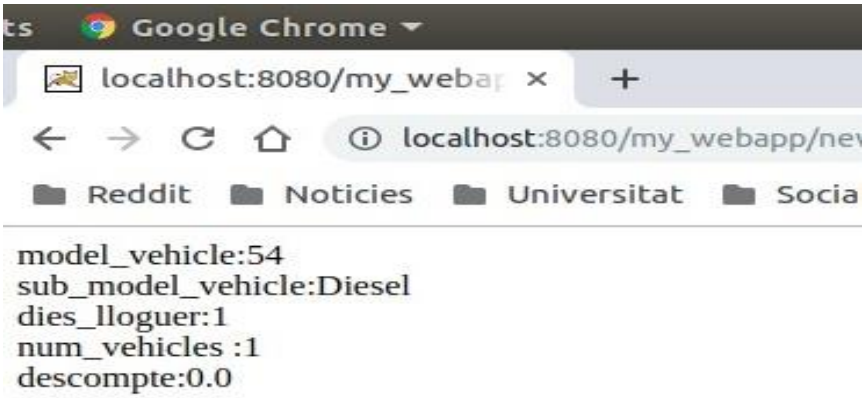
```
/* Leemos el fichero y lo ponemos en list; en caso de ser vacío, no hace nada*/
try (Reader reader = new FileReader("/home/alumne/apache-tomcat-9.0.5/webapps/my_webapp/WEB-INF/classes/mypackage/rental

    JSONObject jsonObject = (JSONObject) parser.parse(reader);
    System.out.println(jsonObject);

    // loop array
    JSONArray msg = (JSONArray) jsonObject.get("lista");
    Iterator<String> iterator = msg.iterator();
    while (iterator.hasNext()) {
        list.add(iterator.next());
    }
} catch (IOException e) {
        e.printStackTrace();
} catch (ParseException e) {
        e.printStackTrace();
}

JSONObject obj = new JSONObject();
JSONObject objfinal = new JSONObject(); //necesario para escribir
```

Afegim el nou lloguer a la llista amb la resta de lloguers:

```
obj.put("model_vehicle", model_vehicle);
obj.put("sub_model_vehicle", sub_model_vehicle);
obj.put("dies_lloguer", dies_lloguer);
obj.put("num_vehicles", num_vehicles);
obj.put("descompte", descompte);

list.add(obj);
```

Posem la llista dins el JSONObject i finalment, l' escrivim al fitxer JSON.

```
/* Ponemos list en el JSONObject para la escritura del fichero JSON*/
objfinal.put("lista",list);


try (FileWriter file = new FileWriter("/home/alumne/apache-tomcat-9.0.5/webapps/my_webapp/WEB-INF/classes/mypackage/rent
    file.write(objfinal.toJSONString());
} catch (IOException e) {
    e.printStackTrace();
}

System.out.print(obj);
```



3. CarRentalList.java

En aquest Servlet volem afegir un login, i només funcionarà correctament quan escrivim l' usuari i la contrasenya correctes.

El primer que fem és obtenir els dos paràmetres necessaris, l' usuari i la contrasenya.

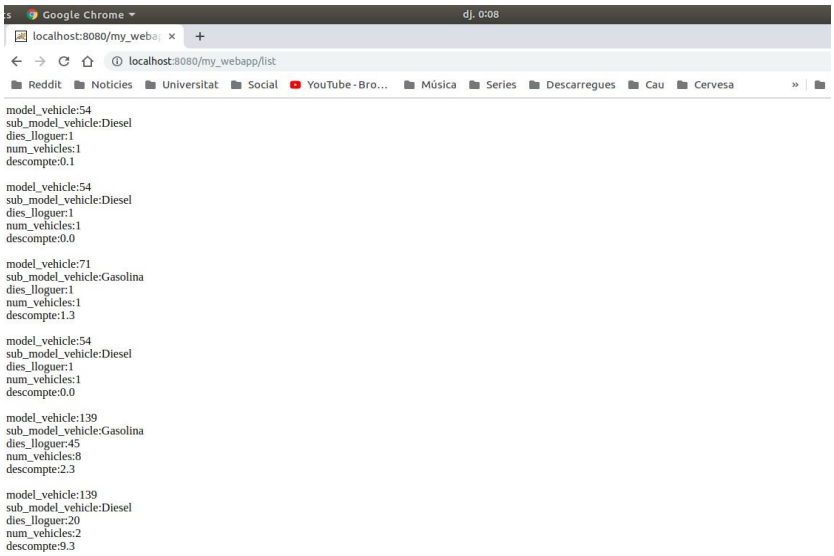
```
String user = req.getParameter("userid");
String pass = req.getParameter("password");
```

Afegim un "if" que farà la comprovació d' usuari i contrasenya.

Tot seguit, si tot és correcte, accedim al fitxer JSON i obtenim els lloguers:

```
if (user != "user" || pass != "pass") out.println("<html>Incorrecte. Vuelva a intentarlo.</html>");
else {
    try (Reader reader = new FileReader("/home/alumne/apache-tomcat-9.0.5/webapps/my_webapp/WEB-INF/classes/mypackage/re

        JSONObject jsonObject = (JSONObject) parser.parse(reader);
        System.out.println(jsonObject);

        // loop array
        JSONArray msg = (JSONArray) jsonObject.get("lista");
```

Per acabar, creem els iteradors necessaris per fer el recorregut del fitxer i anem escrivint els lloguers al fitxer html.



4. SSL/TLS Configuration

En aquest apartat havíem de crear una clau privada pel servidor i un certificat firmat per si mateix, per poder establir comunicació per https.

Primer executem la comanda següent per crear la clau privada:

```
$ keytool -genkey -alias tomcat -keyalg RSA
```

Hi afegim una paraula secreta que farem servir més tard.

Tot seguit, anem al fitxer conf/server.xml i afegim la línia següent:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol" sslProtocol="TLS" clientAuth="false" keystorePass="sistemes"
    keystoreFile="${user.home}/.keystore" SSLEnabled="true" secure="true" scheme="https" maxThreads="200"/>
```

Per acabar, reiniciem el servidor i mirem l' adreça

https://localhost:8443

Veiem que ens diu que la connexió no és privada, per tant sabem que està ben configurat.

5. Extra

Explicació dels pros i contres de l' ús de Java Servlets comparat amb altres alternatives com Node.js:

En aquesta assignatura ens estem familiaritzant amb Servlets de Java però hi ha altres alternatives que podríem utilitzar i que tenen una funcionalitat semblant, com per exemple Node.js.

Per una banda, Java Servlets és un llenguatge de programació de propòsit general orientat a objectes amb una portabilitat enorme, ja que el mateix codi pot córrer en totes les màquines independentment del sistema operatiu o de l' arquitectura.

Per l' altra banda, tenim Node.js, que és una llibreria i un entorn usats per crear aplicacions JavaScript, i que permet executar codi al costat del servidor.

Pros:

L' ús de Java Servlets té un rendiment millor que Node, en general. Pot servir respostes a una velocitat molt alta.

Java té un entorn de programació molt més senzill i familiar per programadors novells. Java pot ser més fàcil d' entendre que Node i compta amb IDEs més complexos i que fan molta feina pel programador, facilitant molt el treball respecte a Node.

Java té eines molt potents per treballar de manera concurrent. Node no ofereix opcions per fer programació de tasques que s' executin de manera concurrent, més enllà de I/O.

Java té una documentació molt més amplia i més madura que Node, ja que compta amb molts més programadors i amb moltes organitzacions que donen suport i que treballen per facilitar l' ús de l' entorn.

Contres:

La programació concurrent de Java és molt útil i potent, però no hi ha gaires programadors que la dominen i si volem fer I/O asíncrona és molt més adequat utilitzar Node respecte Java.

El codi de Node és més compacte i més curt, si volem fer programes nets i fàcils de llegir, aquesta és la nostra millor opció.

Node és molt lleuger, això vol dir que pot ser allotjat en servidors no gaire potents. Java té un requeriment més alt i si no el complim, millor fer servir Node.