JAVA SERVLETS – LAB 1

1. Entorn i configuració de la pràctica

Per començar, instal·lem tant java:

- `sudo apt-get install default-jdk`

Com Tomcat 7:

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

2. Solució

La aplicació web està composada de 2 funcionalitats, com podem veure a la figura 1:

- Demanar un " New Rental "
- Demanar el " List of all Rentals "

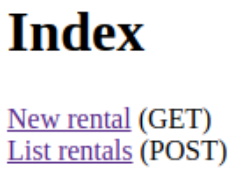

Figura 1. carrental_home.html

Implementem la primera de les funcionalitats per tal de demanar un "New Rental". Per tal de poder demanar els diferents camps d'informació per tal de fer el request utilitzarem un tag <form> de HTML com al Codi 1.

```
<form action="/my_webapp/new" method="GET">
<table summary="">
<tr>
<td>Car Model:</td>
<td><select name=model_vehicle>
<option selected VALUE=54>Econ?mic
<option VALUE=71>Semi-Luxe
<option VALUE=82>Luxe
<option VALUE=139>Limusina
</select>
</td>
<td>Engine:</td>
<td><select name=sub_model_vehicle>
<option selected VALUE=Diesel>Diesel
<option VALUE=Gasolina>Gasolina
</select>
</td>
</tr>

<tr>
<td>Number of days:</td>
<td><input name=dies_lloguer size=3 maxlength=3 value=1></td>
<td>Number of units:</td>
<td><input name=num_vehicles size=3 maxlength=3 value=1></td>
</tr>
<tr>
<td>Descompte(%):</td>
<td><input name=descompte size=3 maxlength=3 value=0.0></td>
</tr>

</table>
  <br>
<input name=llogar type=submit value="Submit (GET)">
<br>
</form>
```

Codi 1.

Que al final quedarà com a la Figura 2. Quan clickem al botó SUBMIT les dades de la form s'enviaran al mètode "doGet" del servlet "CarRentalNew.java" tal com indiquem al atribut "action" del <form>.

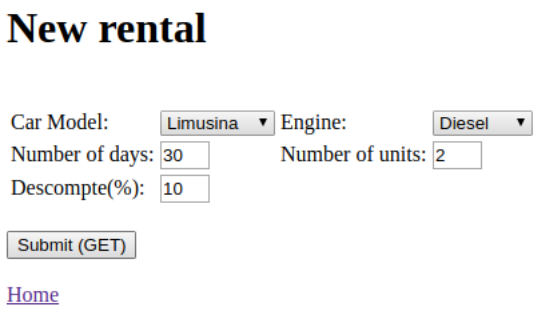

Figura 2. carrental_new.html

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
                  throws ServletException, IOException {
  res.setContentType("text/html");
  PrintWriter out = res.getWriter();
  String model_vehicle = req.getParameter("model_vehicle");
  String model_vehicle_string = "Empty";
  switch(model_vehicle) {
    case "54":
      model_vehicle_string = "Econòmic";
      break;
    case "71":
      model_vehicle_string = "Semi-Luxe";
      break;
    case "82":
      model_vehicle_string = "Luxe";
      break;
    case "139":
      model_vehicle_string = "Limusina";
      break;
  }
  String sub_model_vehicle = req.getParameter("sub_model_vehicle");
  String num_vehicles = req.getParameter("num_vehicles");
  String dies_lloguer = req.getParameter("dies_lloguer");
  String descompte = req.getParameter("descompte");
```

Codi 2.

Dins d'aquest mètode, com podem veure al Codi 2, recollim les dades amb el mètode "getParameter" i una de les primeres decisions que prenem és guardar directament el "model_vehicle" en la seva versió paraula per tal de després poder fer el display directament.

```
    Object obj2 = parser.parse(new FileReader("./test.json"));
    obj = (JSONObject) obj2;
    JSONArray list = (JSONArray) obj.get("messages");
    list.add(model_vehicle_string);
    list.add(sub_model_vehicle);
    list.add(num_vehicles);
    list.add(dies_lloguer);
    list.add(descompte);

    obj.put("messages", list);

} catch (FileNotFoundException e) {
  e.printStackTrace();
}
catch (ParseException e) {
        e.printStackTrace();
    }
if (obj == null) {
      obj.put("name", "Exemple1");

   JSONArray list = new JSONArray();
   list.add(model_vehicle_string);
   list.add(sub_model_vehicle);
   list.add(num_vehicles);
   list.add(dies_lloguer);
   list.add(descompte);

  obj.put("messages", list);
}

try (FileWriter file = new FileWriter("./test.json")) {

    file.write(obj.toJSONString());
    file.flush();
```

Codi 3.

Per tal de mantenir les dades de les rentals utilitzarem un fitxer JSON (json-simple-1.1.1.jar) per tal d'escriure i llegir les dades de disc.

Com podem veure al Codi 3, primerament llegim el file test.json per tal de veure si hi havia dades anteriors i afegir les noves a les anteriors. Una nova decisió que prenem és afegir totes les dades al mateix parametre del JSON. I per finalitzar reescrivim de nou el fitxer amb les dades actualitzades.

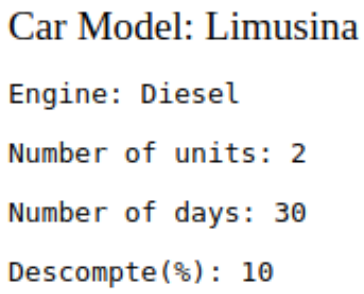

Figura 3.

Quan introduïm el “New Rental” ens retornarà la pàgina les dades que hem introduit anteriorment, com podem veure a la figura 3.

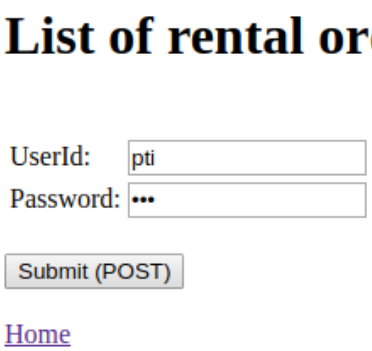

Figura 4. carrental_list.html

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
                  throws ServletException, IOException {
  res.setContentType("text/html");
  PrintWriter out = res.getWriter();
  String nombre = req.getParameter("userid");
  String password = req.getParameter("password");
  out.println("<html>");

  JSONParser parser = new JSONParser();
  JSONObject obj = null;
  if (nombre.equals("pti")) and password.equals("pti")){
    try {
      Object obj2 = parser.parse(new FileReader("./test.json"));
         obj = (JSONObject) obj2;
         JSONArray list = (JSONArray) obj.get("messages");
      Iterator<String> iterator = list.iterator();
      while(iterator.hasNext()){
        if (cont % 5 == 0) {
          out.println("<br><br>");
        }
              out.println("<br> <big>" + iterator.next() + "</big>");
        cont++;
          }
      out.println("</html>");
    }
    catch (ParseException e) {
          e.printStackTrace();
      }
  }
}
```

Codi 4.

Després de rebre la request per mostrar la llista de rentals el mètode doGet s’encarrega de comprovar que el usuari i la contrasenya coincideixin amb les que hi ha al codi (en aquest cas està hardcodejat però es podria fer per

exemple amb una base de dades). Després llegim el fitxer test.json i procedim a iterar sobre els elements que hi ha al .json i cada 5 (camps que té una rental) sabem que és una rental nova. I es veuria tal i com mostrem a la figura 5.

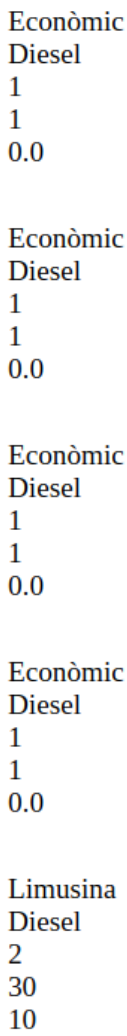

Figura 5

3. Aspectes positius i negatius de la tecnologia

Hem de tenir en compte alguns aspectes en comparació a altres tecnologies més modernes com ara Node, que porten molt menys temps però tenen moltes avantatges.

Es podrien llistar algunes avantatges de fer servir NodeJS envers Java Servlets:

Avantatges de fer servir Node envers JavaServlets

- Els processos de node son més lleugers en quan a carrega del servidor.
- NPM es més intuïtiu que Maven i Gradle
- Benchmarking més ràpid que gairebé tots els llenguatges de scripting dinàmics populars de la part de servidor.
- Es barat d'allotjar. JVM consumeix massa memòria
- El codi de NodeJS es molt menys detallat que el codi de Java

Avantatges de fer servir Java Servlets envers Node

- El rendiment absolut es millor en JVM
- La depuració es molt més eficient a Java que no pas a Node
- L'ecosistema de JVM es realment bó. Bons projectes de la fundació Apache, bones bases de dades i biblioteques.
- Fàcil trobar experts en Java i no en NodeJS ja que es una tecnologia molt més extensa i popular.