Java Servlets

Alumnos:

- Jordi Solé García (Grupo 11)
- Víctor Gallego Izquierdo (Grupo 11)

Objetivos de la práctica

El objetivo de la práctica es la creación de una aplicación web con servlets Java, en el backend, y con html simple en el frontend. En cuanto al servidor, usaremos Apache Tomcat.

Funcionalidades básicas de las dos páginas:

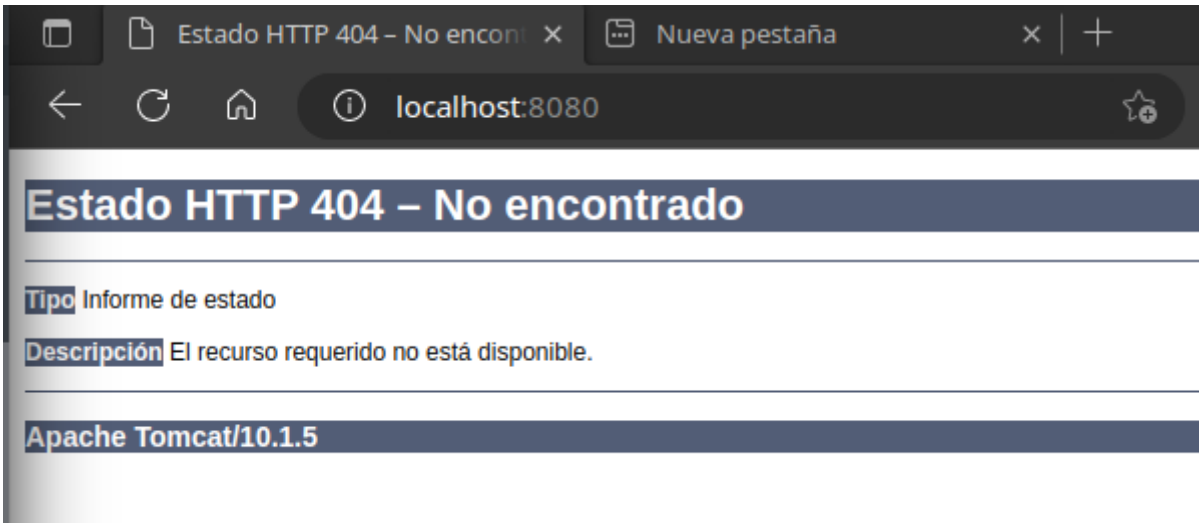
- Realizar una petición de alquiler de coche mediante un formulario simple en html, en las que se especifican diferentes parámetros como: el modelo de coche, el número de días que se va a alquilar, el descuento que se aplica, el tipo de motor y la cantidad de coches a alquilar con estas características.
- Obtener una lista de los coches alquilados anteriormente. Para acceder a estos, solo puede hacerlo el usuario administrador con su respectiva contraseña.

Implementación de la práctica

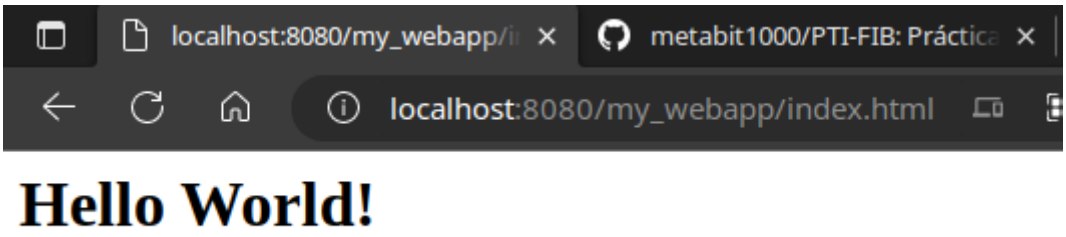
Instalación del servidor

Antes de nada, necesitamos comprobar que Java SDK estuviera instalado, y para ello usamos el comando `javac -version` y comprobamos que ya estaba instalado. Además, decidimos clonar el repositorio completo en nuestra máquina para poder tener todo disponible más fácilmente.

Primero necesitamos instalar el servidor para poder ejecutar la aplicación. Para ello, descomprimimos el archivo del apache con el comando `tar -xvzf apache-tomcat-10.0.10.tar.gz` y entramos al directorio. Para ponerlo en marcha, ejecutamos `./bin/startup.sh &` y entramos en la dirección localhost:8080/ para poder comprobar que se ejecutaba correctamente. Se muestra la ventana de error al no encontrar la página web:



Para comprobar que realmente funciona, hacemos un fichero en html con un simple "Hello World!". El resultado al acceder a la URL localhost:8080/my_webapp/index.html es el siguiente:



Por último, para comprobar la ejecución de Servlets, creamos un archivo `web.xml` donde añadimos la información necesaria para poder ejecutar correctamente el servlet y después creamos el archivo `.java` donde estará el código de este. Para comprobar que funciona correctamente, es necesario compilar el código, apagar el servidor y volverlo a encender y acceder a la URL localhost:8080/my_webapp/my_servlet .

Práctica

CarRentalNew.java

Tenemos que crear el servlet que es capaz de recoger los datos del formulario y esriba en un fichero para después ser mostrados por parte del administrador. En el código está comentado que hace cada parte de este.

```
package mypackage;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import java.io.FileWriter;
import java.io.IOException;

import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import java.util.Iterator;

public class CarRentalNew extends HttpServlet {



  public void doGet(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

      // Coger los parámetros del formulario
    String co2 = req.getParameter("co2_rating");
    String engine = req.getParameter("sub_model_vehicle");
    String rentDays = req.getParameter("dies_lloguer");
    String numUnits = req.getParameter("num_vehicles");
    String discount =req.getParameter("descompte");

    //Comprobamos que no haya alguno que esté vacío
    if (!co2.isEmpty() && !engine.isEmpty() && !rentDays.isEmpty() &&
!numUnits.isEmpty() && !discount.isEmpty()) {
```

```
        //Imprimimos los parámetros en la página web
        if (co2.equals("54")) co2 = "Extralow";
        else if (co2.equals("71")) co2 = "Low";
        else if (co2.equals("82")) co2 = "Medium";
        else  if (co2.equals("139")) co2 = "High";
        out.println("<html><big>CO2 rating:"+ co2 +"</big><br></html>");
        out.println("<html><big>Engine:"+engine +"</big><br></html>");
        out.println("<html><big>Number of days:" +rentDays + "</big><br>
</html>");
        out.println("<html><big>Number of units:" + numUnits+ "</big><br>
</html>");
        out.println("<html><big>Discount(%):" + discount +  "</big><br>
</html>");

       //Abrimos el archivo donde se guardan los datos
       File f = new File("orders.json");
       JSONArray list = new JSONArray();

       //En caso de existir, leemos los datos para no sobreescribirlos.
       if (f.exists()) {
        try (Reader reader = new FileReader("orders.json")) {
            JSONParser parser = new JSONParser();
            JSONObject jsonObject = (JSONObject) parser.parse(reader);
            JSONArray orders = (JSONArray) jsonObject.get("lista");
            Iterator<JSONObject> iterator = orders.iterator();
             while (iterator.hasNext()) {
                 list.add(iterator.next());
                  }
        } catch (ParseException e) {
            e.printStackTrace();
        }
       }

       //Añadimos el objeto JSON con los datos al archivo final donde se
van a guardar.
       JSONObject obj = new JSONObject();
       JSONObject objfinal = new JSONObject();
       obj.put("CO2_rating", co2);
       obj.put("Engine", engine);
       obj.put("Number_of_days", rentDays);
       obj.put("Number_of_units", numUnits);
       obj.put("Discount", discount);
       list.add(obj);
       objfinal.put("lista",list);
       try (FileWriter file = new FileWriter("orders.json")) {
            file.write(objfinal.toJSONString());
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        else out.println("Faltan campos por rellenar");
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
      doGet(req, res);
    }
}
```

Una vez hecho esto, se puede comprobar que funciona correctamente al listar los datos desde el otro formulario, una vez esté completado el código del servlet que permite el login.

CarRentalList.java

Este servlet muestra en la página web de la aplicación los datos de los coches que han alquilado los distintos usuarios. Solo el administrador tiene acceso a estos.

```
package mypackage;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.Iterator;

public class CarRentalList extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    //Recogemos los datos del formulario
    String nombre = req.getParameter("userid");
    String password = req.getParameter("password");

    //Comprobamos que es el usuario admin y su contraseña
    if (nombre != null && nombre.equals("admin") && password != null &&
 password.equals("1234")) {
        //Abrimos el archivo donde se encuentran los datos del alquiler de
los coches
        JSONParser parser = new JSONParser();
        try (Reader reader = new FileReader("orders.json")) {
            //Una vez abierto, creamos los iteradores de los objetos del
JSON para poder mostrarlos en la página.
            JSONObject jsonObject = (JSONObject) parser.parse(reader);
            JSONArray orders = (JSONArray) jsonObject.get("lista");
            Iterator<JSONObject> CO2 = orders.iterator();
            Iterator<JSONObject> ENGINE = orders.iterator();
            Iterator<JSONObject> DAYS = orders.iterator();
            Iterator<JSONObject> UNITS = orders.iterator();
```

```
            Iterator<JSONObject> DISCOUNT = orders.iterator();
            while (CO2.hasNext()) {
               String co2 = (String) CO2.next().get("CO2_rating");
               String engine = (String)  ENGINE.next().get("Engine");
               String rentDays = (String)
DAYS.next().get("Number_of_days");
               String numUnits = (String)
UNITS.next().get("Number_of_units");
               String discount = (String) DISCOUNT.next().get("Discount");
               out.println("<html><big>CO2 rating:"+ co2 +"</big><br>
</html>");
               out.println("<html><big>Engine:"+engine +"</big><br>
</html>");
               out.println("<html><big>Number of days:" +rentDays + "</big>
<br></html>");
               out.println("<html><big>Number of units:" + numUnits+ "
</big><br></html>");
               out.println("<html><big>Discount(%):" + discount +  "</big>
<br></html>");
               out.println("<html><br></html>");
            }

        } catch (IOException e) {
            e.printStackTrace();
        } catch (ParseException e) {
            e.printStackTrace();
      }
    }
    else {
        out.println("<html>" +
            "<h1>User or Password incorrect</h1>" +
        "</html>");
    }
  }

  public void doPost(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {
    doGet(req, res);
  }
}
```

Configuración SSL/TLS

El propósito de este apartado es la configuración del certificado que permite utilizar HTTPS para cifrar la conexión y los datos que se envían. Para ello, primero ejecutamos el comando `keytool -genkey -alias tomcat -keyalg RSA` donde nos pedirá que introduzcamos el password **changeit** . Seguido, nos pide distintos datos a rellenar que no es necesario. Ahora solo queda añadir el siguiente trozo de código al archivo que se encuentra en `conf/server.xml`.

```
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"

 sslImplementationName="org.apache.tomcat.util.net.jsse.JSSEImplementation"
    port="8443"
    maxThreads="150"
    SSLEnabled="true">
    <SSLHostConfig>
        <Certificate
        certificateKeystoreFile="${user.home}/.keystore"
        type="RSA"
      />
    </SSLHostConfig>
</Connector>
```

Reiniciamos el servidor con los comandos y podremos acceder al enlace localhost:8443 donde para comprobar que funciona correctamente, nos mostrará un mensaje de error indicando que no es un lugar privado, ya que el certificado está autofirmado. Simplemente damos a avanzaod y continuar y ya podremos acceder a la aplicación.

```
./bin/shutdown.sh
./bin/startup.sh &
```

Your connection isn't private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages, or credit cards).

NET::ERR_CERT_AUTHORITY_INVALID

Advanced

Go back

Dockerizar nuestra práctica

Para dockerizar nuestra práctica, es necesario encontrar el docker por el que empezar, en este caso al ser Tomcat, ya existe una imagen oficial en DockerHub -> Tomcat.

```
FROM tomcat:10 #Partimos de la imagen oficial
COPY . /my_webapp #Copiamos la carpeta con la aplicación
WORKDIR /
RUN cp -r my_webapp /usr/local/tomcat/webapps #Dentro del contendor,
ponemos la carpeta en la ruta donde va a ejecutar el servidor la aplicación
```

Extra

Como extra, hemos decidido crear el docker-compose para que sea más fácil realizar la creación y ejecución del contenedor.

```
version: "3.9"
services:
  my_webapp: #Nombre del servicio.
    build: . #Hacer el buil del contenedor para obtener la imagen.
    ports: #Abrimos el puerto para tener acceso a este desde el navegador
de la aplicación.
      - "8080:8080"
```