

INFORME

PRÁCTICA 2

PTI

Eduard González Moreno

Miguel Guerrero López

08/03/2018

Entorno y configuración

En esta práctica se ha instalado un servidor tomcat sobre un sistema operativo linux. Seguidamente se ha programado dos servlets para tratar los datos de dos formularios escritos en html. En uno se recogen los datos por get y en otro se recogen los datos por post. La BBDD que se ha utilizado ha sido un fichero de tipo JSON.

Resolución

Para solucionar esta práctica se han escritos dos servlets escritos en java. Uno de ellos recoge los parámetros por get del formulario que se le enviaba desde carrentat_form_new.html para dar un nuevo alquiler de baja. Para ello se han recogido los datos desde la operación doGet que es la que se ejecuta cuando se hace una petición del tipo get.

Una vez se han recogido todos los datos, se hace una comprobación de que todo lo que nos han enviado está en el formato correcto. Para ellos también hemos programado un par de funciones que nos dicen si el dato es un número, y si es mayor o igual a 0.

A continuación, una parte del código de esta función creada por nosotros y las comprobaciones pertinentes.

```
//Funciones que comprueban formato..
private static int isNumeric(String cadena) {
    try {
        Integer.parseInt(cadena);
        if (Integer.parseInt(cadena) < 1) {
            return -1;
        }
        return 1;
    } catch (NumberFormatException nfe) {
        return -2;
    }
}

private static int isPositive(String cadena) {
    try {

        if (Double.parseDouble(cadena) < 0) {
            return -1;
        }
        return 1;
    } catch (NumberFormatException nfe) {
        return -2;
    }
}

//Comprobación del formato
if (isNumeric(numd) == -2) {
    out.println("Los dias de alquiler no contienen un numero <br>");
} else if (isNumeric(numd) == -1) {
```

```

        out.println("Los dias de alquiler no pueden ser menor o igual a 1 <br>");
    }
    if (isNumeric(numvh) == -2) {
        out.println("El numero de vehiculos no contiene un numero <br>");
    } else if (isNumeric(numvh) == -1) {
        out.println("El numero de vehiculos no puede ser menor o igual a 1 <br>");
    }
    switch (isPositive(des)) {
        case -2:
            out.println("El descuento no contiene un numero <br>");
            break;
        case -1:
            out.println("El descuento tiene que ser un numero igual o mayor que
0 <br>");
            break;
        default:
            if (submodel.equals("Diesel")) {
                aumento = 1.4;
            }
            break;
    }
}

```

Una vez se ha comprobado que todos los datos correctos nos disponemos a escribir los datos en una BBDD de tipo JSON. Para ello previamente miramos el contenido de este y en el caso de que no haya contenido se escribe un primer objeto llamado “rental” que contiene una lista con los valores del formulario. En el caso de que ya hubiese algún registro se le añaden los cinco valores que seguro recibimos del formulario a la lista.

A continuación el fragmento de código que se encarga de esto:

```

BufferedReader b;
    String aux;

try
    (FileReader f = new
FileReader("C:\\WORKSPACE\\Universidad\\PTI\\P2\\Practica2PTI\\BBDD.json")) {
    b = new BufferedReader(f);
    aux = b.readLine();
}

JSONParser parser = new JSONParser();

try {
    JSONArray raiz;
    if (aux != null) {
        Object obj = parser.parse(aux.replace("\\", ""));
        JSONObject jsonObject = (JSONObject) obj;
        raiz = (JSONArray) jsonObject.get("rentals");
    } else {
        raiz = new JSONArray();
    }

    ///////////////////////////////////
    JSONObject obj2 = new JSONObject();
    JSONArray list = new JSONArray();
    raiz.add(model);
}

```

```

        raiz.add(submodel);
        raiz.add(numvh);
        raiz.add(numd);
        raiz.add(des);
        if (aux != null) {

            obj2.put("rentals", raiz);
        } else {
            obj2.put("rentals", raiz);
        }

        try
            (FileWriter file = new
FileWriter("C:\\WORKSPACE\\Universidad\\PTI\\P2\\Practica2PTI\\BBDD.JSON")) {

            file.write(obj2.toJSONString());
            file.flush();
            file.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

Para finalizar una vez guardados los datos en la BBDD mostramos por pantalla el resumen de factura y el precio total. A continuación, el fragmento de código:

```

out.println("<h2>Su peticion de alquiler ha sido aceptada</h2><br><hr><br>");
out.println("<h3>Factura simplificada:</h3>");
out.println("Precio del modelo= " + model + "<br>");
out.println("Suplemento por submodelo= " + aumento + "<br>");
out.println("Dias de alquiler= " + numd + "<br>");
out.println("Numero de vehiculos= " + numvh + "<br>");
out.println("Descuento= " + des + "<br>");
out.println("<hr>");
double total = (Integer.parseInt(model) * (aumento) *
Integer.parseInt(numd) * Integer.parseInt(numvh)) * ((100 - Double.parseDouble(des))
/ 100);

out.println("<b>Total :</b>" + total + "<br><br>");

```

En el otro archivo nos encargamos de recoger los parámetros de autenticación vía post. Para ello hemos utilizado la función de doPost que es la que se ejecuta automáticamente para estos casos.

Una vez hemos recibido los datos comprobamos que son correctos. En nuestro caso al no tener una tabla de usuarios se comprueba directamente en el código:

```

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

if (user.equals("admin") && pass.equals("admin")) {

```

Seguidamente leemos nuestra BBDD y cargamos en memoria para después mostrar por pantalla los datos que queremos en una tabla.

```

Object obj = parser.parse(new
FileReader("C:\\WORKSPACE\\Universidad\\PTI\\P2\\Practica2PTI\\BBDD.json"));
JSONObject jsonObject = (JSONObject) obj;
JSONArray rentals = (JSONArray) jsonObject.get("rentals");
Iterator<String> iterator = rentals.iterator();

```

Para movernos por la lista que hemos leído utilizamos un iterador que nos devolverá el siguiente valor cada vez que invoquemos la función “iterator.next()”. Por lo tanto, montamos una pequeña tabla en html y mostramos el valor con los iteradores. Y como sabemos que cada cinco valores es una nueva línea estamos tranquilos que podemos utilizar la función cinco veces seguidas sin consultar que hay un siguiente.

```

out.println( "<table>");
out.println( "<tr>");
out.println( "<th> Modelo </th>");
out.println( "<th> Submodelo </th>");
out.println( "<th> Dias de alquiler </th>");
out.println( "<th> Numero de vehiculos </th>");
out.println( "<th> Descuento </th>");
out.println( "</tr>");

while (iterator.hasNext()){
    out.println( "<tr>");
    String modelo = iterator.next();
    switch (modelo) {
        case "54":
            out.println( "<td>Economic</td>");
            break;
        case "71":
            out.println( "<td>Semi Luxe</td>");
            break;
        case "82":
            out.println( "<td>Luxe</td>");
            break;
        case "139":
            out.println( "<td>Limusina</td>");
            break;
        default:
            break;
    }
    out.println( "<td>" + iterator.next() + "</td>");
    out.println( "<td>" + iterator.next() + "</td>");
    out.println( "<td>" + iterator.next() + "</td>");
    out.println( "<td>" + iterator.next() + "</td>");
    out.println( "</tr>");
}
out.println("</table>");

```

Y en el caso de que los datos de autenticación sea incorrecto sale un mensaje por pantalla informando de esta situación.

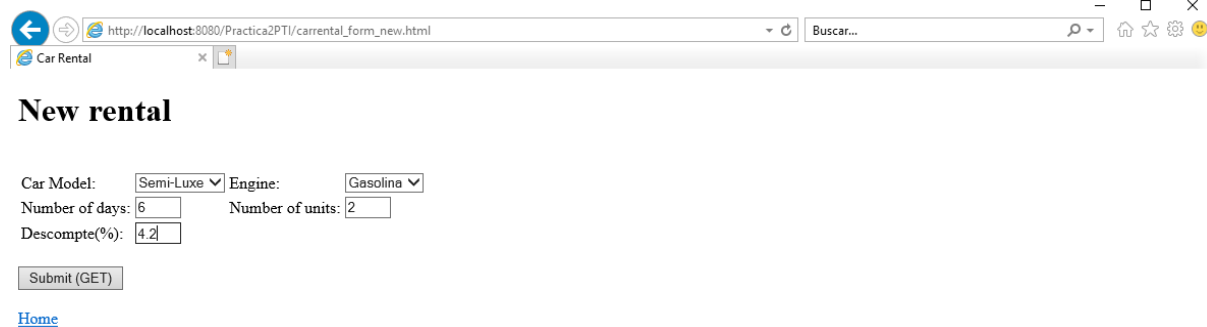
```

else {
    out.println("User or password incorrect");
}

```

Evaluación de la solución

Primero para evaluar la solución del new seleccionamos los parámetros que queramos en el formulario para realizar un nuevo alquiler igual que en la práctica 1. En la siguiente imagen ponemos un ejemplo. A continuación pulsamos submit y comprobamos que el alquiler se ha realizado con éxito.



New rental

Car Model: Engine:

Number of days: Number of units:

Descompte(%):

[Home](#)

Como podemos ver en la siguiente imagen, el alquiler se ha realizado sin problemas. Hemos usado el mismo formato que con la práctica 1.



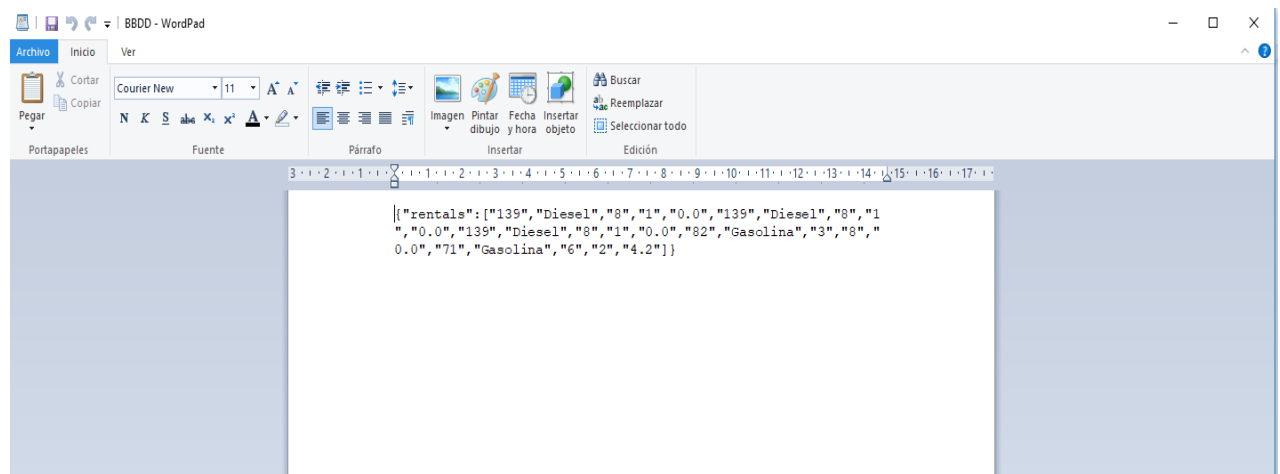
Su petición de alquiler ha sido aceptada

Factura simplificada:

Precio del modelo= 71
Suplemento por submodelo= 1.0
Dias de alquiler= 2
Numero de vehiculos= 6
Descuento= 4.2

Total :816.216

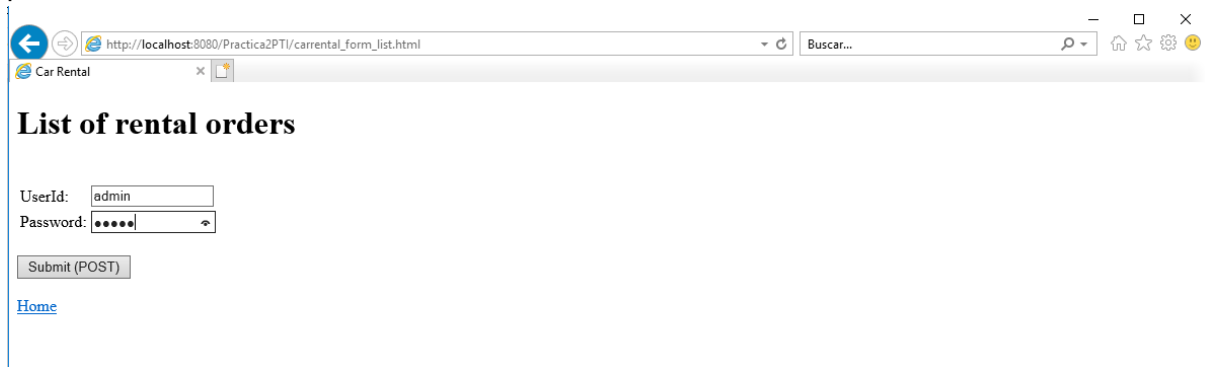
En esta solución en vez de guardar los alquileres en un CSV, lo hemos hecho en un fichero JSON. Para comprobar que se guarda correctamente abrimos el JSON y vemos el siguiente resultado.



```
{
  "rentals": [
    "139", "Diesel", "8", "1", "0.0", "139", "Diesel", "8", "1", "0.0", "139", "Diesel", "8", "1", "0.0", "82", "Gasolina", "3", "8", "0.0", "71", "Gasolina", "6", "2", "4.2"
  ]
}
```

Como podemos ver, los datos se han guardado correctamente, las últimas cinco palabras coinciden con los datos introducidos en el formulario anterior. Con esto verificamos que guardamos correctamente los datos.

Para comprobar el `carrenal_list`, rellenamos el siguiente formulario con `user:admin`, y `password:admin`.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Practica2PTI/carrenal_form_list.html`. The page title is "Car Rental". Below the title, there is a heading "List of rental orders". Underneath, there is a login form with two input fields: "UserId:" containing the text "admin" and "Password:" containing five dots. Below these fields is a button labeled "Submit (POST)". At the bottom left of the form area, there is a blue link labeled "Home".

A continuación, pulsamos el botón de submit. Como podemos ver en la siguiente imagen, nos aparecen todos los alquileres que se habían guardado en el fichero JSON. Con esto podemos comprobar que los alquileres se guardan correctamente, y también que podemos leerlos del fichero sin problema.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Practica2PTI/list`. The page title is "localhost". Below the title, there is a table with the following data:

Modelo	Submodelo	Días de alquiler	Numero de vehiculos	Descuento
Limusina	Diesel	8	1	0.0
Limusina	Diesel	8	1	0.0
Limusina	Diesel	8	1	0.0
Luxe	Gasolina	3	8	0.0
Semi Luxe	Gasolina	6	2	4.2

Ventajas tecnológicas

La gran ventaja de utilizar un lenguaje como java es que es un lenguaje muy maduro y hay muchos manuales, aparte de ya estar muy optimizado para muchos ámbitos y tiene una gran capacidad de librerías. Además, es un lenguaje muy optimizado y muy claro de utilizar con lo que para otros programadores es muy entendible. También tiene muchas bibliotecas ya creadas y fáciles de importar, que gracias a ella hace que sea más fácil trabajar con otras tecnologías como JSON.

A la hora de utilizar JSON ha sido fácil ya que tiene muy pocas normas para ser utilizado todo lo contrario de PHP, y a la hora de recuperar la información ha sido fácil