

# Practica 1

Antonio Lopez Higuera

February 2019

## Introduction

Un *servlet* es una aplicación Java basada en un componente Web, administrado por un contenedor, que genera dinámicamente el contenido. Los contenedores, a veces llamados motores de servlets, son extensiones de servidores Web que proveen funcionalidad al servlet. Los servlets interactúan con el cliente por medio de un paradigma basado en peticiones y respuestas por el contenedor de servlets.

## Desarrollo

Para realizar esta práctica se requiere contar con una instalación de Java, la más reciente de la versión 8 (en este caso 8, 201); Eclipse IDE, en su versión Photon; y Jetty como *servlet container*.

## Ejercicio 1

Este ejercicio es la base para los demás ejercicios, al dirigirse a un servlet directamente por la dirección hace uso del método *doGet* y para hacer uso del método *doPost* se requiere hacer la solicitud por medio de un *form*. El método utilizado es el *doGet*, este recibe el objeto *request* y *response*, el objeto *response* es en el que se regresa la petición, se define el tipo de contenido y se envían en el *PrintWriter* las líneas de texto en formato HTML.

```
protected void doGet(HttpServletRequest request ,  
HttpServletResponse response) throws ServletException ,  
IOException {
```

```

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h2>Hello _World _Servlet!</h2>");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }

```

## Ejercicio 2

Al ingresar a una dirección web se hace una solicitud a su servidor, en este caso el contenedor de servlets, en esta petición se envía información en el encabezado, es posible hacer disposición de esta información por medio del objeto *request*.

```

protected void doGet(HttpServletRequest request ,
    HttpServletResponse response) throws
    ServletException , IOException {
    // TODO Auto-generated method stub
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("<h2>HTML _Protocol _Headers</h2>");
    out.println("<ol>");
    Enumeration<String> headerNames = request.getHeaderNames();
    while(headerNames.hasMoreElements()) {
        out.println("<li>");
        String headerName = headerNames.nextElement();
        out.println(headerName+" : "+request.getHeader(headerName));
        out.println("</li>");
    }
    out.println("</ol>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}

```

```
}
```

### Ejercicio 3

Un contenedor de servlets instancia una sola vez cada clase, por lo cual si manejas variables globales, estas mantendrán sus datos y son modificados, se mantiene su modificación, como en el caso de un contador, al recibir cada petición este se aumenta, sin embargo, la instancia se mantiene, por lo cual persisten los datos.

```
public class Ejercicio3 extends HttpServlet {
    private static final long serialVersionUID = 1L;
    int counter = 0;

    public Ejercicio3() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        counter++;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h2>" + counter + "</h2>");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
}
```

### Ejercicio 4

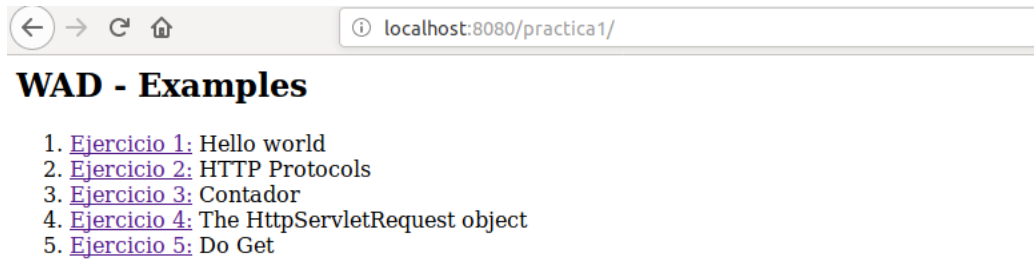
Como se explicaba en el ejemplo anterior, la variable del método no podría cambiar, debido a que esta es creada de nuevo, no como la global en una sola única instanciación.

## Ejercicio 5

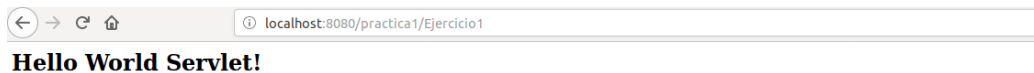
Para la utilización de la base de datos se requiere hacer una conexión por medio de un *JDBC*, para realizar la conexión se generó otro archivo *SQLConnector.java*, el cual hace la conexión con el objeto *Connection*, siendo instanciadoo con el método *getConnection* del *DriverManager*, una vez creada la comunicación se pueden hacer consultas SQL con el objeto *Statement* y el método *executeQuery* o *executeUpdate*.

## Pruebas

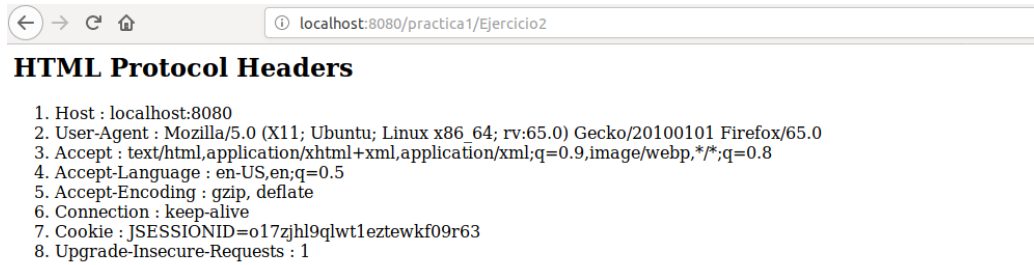
### Pantalla Principal



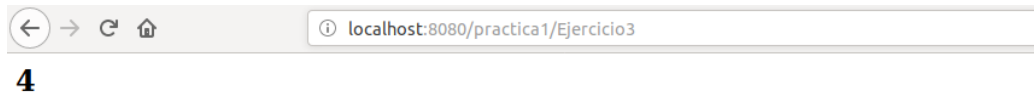
## Ejercicio 1



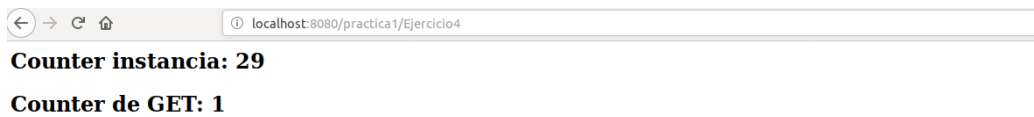
## Ejercicio 2



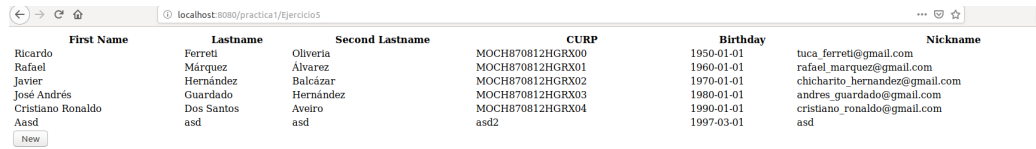
## Ejercicio 3



## Ejercicio 4



## Ejercicio 5



A screenshot of a web browser window displaying a table of player data. The browser's address bar shows 'localhost:8080/practica1/Ejercicio5'. The table has six columns: First Name, Lastname, Second Lastname, CURP, Birthday, and Nickname. The data rows include Ricardo Ferretti Oliveria, Rafael Márquez Álvarez, Javier Hernández Balcázar, José Andrés Guardado Hernández, Cristiano Ronaldo Dos Santos Aveiro, and Aasd asd. A 'New' button is located at the bottom left of the table.

First Name	Lastname	Second Lastname	CURP	Birthday	Nickname
Ricardo	Ferretti	Oliveria	MOCH870812HGRX00	1950-01-01	tuca_ferretti@gmail.com
Rafael	Márquez	Álvarez	MOCH870812HGRX01	1960-01-01	rafael_marquez@gmail.com
Javier	Hernández	Balcázar	MOCH870812HGRX02	1970-01-01	chicharito_hernandez@gmail.com
José Andrés	Guardado	Hernández	MOCH870812HGRX03	1980-01-01	andres_guardado@gmail.com
Cristiano Ronaldo	Dos Santos	Aveiro	MOCH870812HGRX04	1990-01-01	cristiano_ronaldo@gmail.com
Aasd	asd	asd	asd2	1997-03-01	asd

New

## Conclusiones

Utilizar servlets es más fácil de lo que creí. (: