Práctica 2

Macedo Borbolla Eduardo Sai

Servlets

Web Application Development

11 de septiembre de 2018

Resumen

Reporte del desarrollo de Servlets

19-1

Teoría: Los Servlets son programas escritos en Java que se utilizan en un servidor, que puede ser o no ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java.

Los Servlets son para los servidores lo que los applets para los navegadores, aunque los servlets no tienen una interfaz gráfica.

Extienden de la superclase HttpServlet para la fácil gestión con el protocolo HTTP y responden de acuerdo al método de transferencia de datos que se establezca en la comunicación con el mismo con los objetos **HttpRequest** y **HttpResponse.**

Desarrollo:

```
package puntos;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

1.-Programar un servlet que muestre en pantalla el mensaje "Hola mundo".

```
public class Punto1 extends HttpServlet {
	private static final long serialVersionUID = 1L;

   public Punto1() {
      super();
   }

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

		response.setContentType("text/html");
	PrintWriter out= response.getWriter(); //Impresión mediante flujo out del
                                                objeto response
	out.print("<html><body>");//Inicio HTML
	out.print("<h3>Bonjour monde</h3>"); //Hola mundo en fránces
	out.print("</body></html>");//Fin HTML
	}
protected void doPost(HttpServletRequest request, HttpServletResponse response)
	throws ServletException, IOException {
		doGet(request, response);
	}
}
```

Al desarrollar este servlet no tuve problema alguno dado que se vio en clase.

2.-Implementar un servlet que muestre en el navegador los headers del protocolo HTTP que se están enviando desde el cliente al servidor.

```
import java.util.Enumeration;
public class Punto2 extends HttpServlet {
	private static final long serialVersionUID = 1L;

   public Punto2() {
      super();
   }

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

	response.setContentType("text/html"); //Mime HTML
	PrintWriter out=response.getWriter();//objeto respuesta para manejar idem
	out.print("<html><body>");// //impresion HTML por flujo Print Writer

	Enumeration headerNames= request.getHeaderNames(); //numeracion de los
encabezados de la peticion
	out.print("PETICION TIPO<br>");
	out.print(request.getMethod());//Accesamos al tipo de peticion en la que se programo
                                       el servlet
	out.print("<br>");

	/ /Mientras haya un encabezado en nuestra numeracion por el cual no hayamos pasado
	while(headerNames.hasMoreElements()) { //hasMoreElements is boolean
	String clave =(String) headerNames.nextElement(); //Cast a cadena para la clave del
                                                          encabezado
	String valor=request.getHeader(clave);//Obtenemos el valor del encabezado con su
                                                          clave
	out.println(clave+":"+valor+"<br>"); //USer-agent:Mozilla Firefox
	out.print("</body></html>");//Cerramos campo HTML
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		doGet(request, response);
	}

}
```

Para realizar este servlet tuve que investigar el metodo getHeaders() visto en clase y su aplicación utilizando un objeto Enum para poder guardar en orden los encabezados e imprimir uno por uno en pares (clave, valor) mediante el cast a cadenas.

3.-Implemente un servlet que atienda una peticion GET y en el método service de atención implemente una variable contador inicializada en 0, y posteriormente muestre su valor en pantalla y en cada petición incrementar su valor.

```
public class Punto3 extends HttpServlet {
	private static final long serialVersionUID = 1L;


   public Punto3() {
      super();
   }
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		Integer x=0; //Declaramos una variable entera en el ambito del metodo
		String str="<br>Aumentando variable X<br>"; //Cadena a concatenar cada que
aumente el valor
		PrintWriter out=response.getWriter();//Instancia de objeto de flujo de respuesta
		out.print("<html><body>");//COmienza cuerpo HTML
		out.print("<h2>Variable dentro de doGet</h2>");
		out.print(x+"\n");
		x++;//Aumento de la variable en cada llamada a doGet
		out.print(str);
		str=str+str;//Concatenacion de la cadena en cada llamada a doGet
		out.print("</body></html>");//fin cuerpo HTML


	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
	}

}
```

Para el desarrollo de este servlet solo estuve un buen rato intentando descifrar si era correcto que no aumentara el valor de la variable, esto pasa dado que esta dentro del método do del servlet el cual es reenviado cada vez que la instancia de este responde por tal método digase doGET para peticiones GET.

4.-Con base en el punto anterior mueva la variable implementada en el método service como propiedad del servlet. ¿Qué observa? ¿Qué diferencia existe entre la salida de éste ejercicio y el anterior?, justifique sus respuestas.

```
public class Punto4 extends HttpServlet {
	private static final long serialVersionUID = 1L;

	Integer x=0; //Declaramos una variable entera en el ambito del servlet
	String str="<br>Aumentando variable X"; //Cadena a concatenar cada que
aumente el valor

   public Punto4() {
      super();

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

		PrintWriter out=response.getWriter();//Instancia de objeto de flujo de
                                                     respuesta
		out.print("<html><body>");//COmienza cuerpo HTML
		out.print("<h2>Variable dentro de doGet</h2>");
		out.print(x+"\n");
		x++;//Aumento de la variable en cada llamada a doGet
		out.print(str);
		str=str+str;//Concatenacion de la cadena en cada llamada a doGet
		out.print("</body></html>");//fin cuerpo HTML
	}


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

		doGet(request, response);
	}
}
```

Con base en el punto anterior puedo decir que como ahora la variable pertenece a la instancia del servlet que llamamos cada que recargamos la página o pedimos el recurso, esta se sigue guardando ya que no hay reinicialización de la variable cada vez que se llama el servlet ni el método.

Conclusiones

Al utilizar servlets podemos manejar el módulo controlador del patrón de diseño modelo-vista-controlador para poder gestionar las respuestas y comportamientos a seguir por parte de la página en el servidor al cual se están realizando las peticiones, cabe mecionar que debe ser de cuidado el ámbito de cada variable así como el control de flujo de datos dentro de la misma para poder manejar de manera correcta y no tan elaborada dentro de cada componente de la arquitectura.

Bibliografía:

Curso Web Application Development 2019-1. M. en C. Hermes Montes Casiano