JAVA WEB: SERVLETS

Charly Cimino

Java Web: *Servlets*

Charly Cimino

Versión 2023.09.16

Aplicaciones web con Java

Los ejemplos y explicaciones de esta presentación fueron desarrollados y explicados a partir de las siguientes herramientas y sus respectivas versiones.

JDK 17

Por ser una versión **LTS**
(*Long Time Support*)

NetBeans 17

Instrucciones de instalación

https://youtu.be/2Et13pH2484

Apache Tomcat

Versión 10

Descargar de aquí

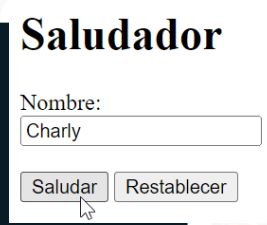
En el siguiente repositorio encontrarás varios de los ejemplos de esta PPT (y más), para poder probarlos en tu máquina.

¿Por qué *servlets*?

Mediante *servlets*, Java permite generar contenido web dinámico desde el servidor, para adaptar las respuestas según el caso.

formSaludo.html

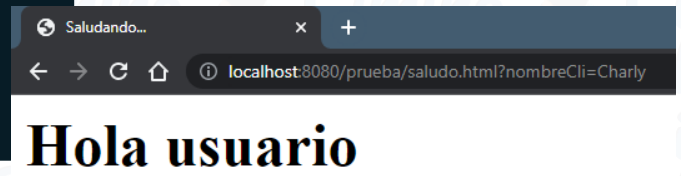
```
<html>
    <head></head>
    <body>
        <h1>Saludador</h1>
        <form action="saludo.html">
            <label>Nombre:</label> <br>
            <input name="nombreCli">
            <br><br>
            <input type="submit" value="Saludar">
            <input type="reset">
        </form>
    </body>
</html>
```



saludo.html

```
<html>
    <head>
        <title>Saludando...</title>
    </head>
    <body>
        <h1>Hola usuario</h1>
    </body>
</html>
```

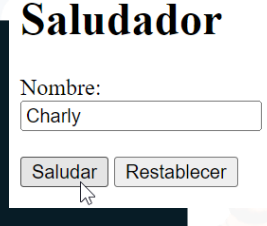
Contenido estático
(Se le presenta al cliente tal y como fue creado previamente)



---

formSaludo.html

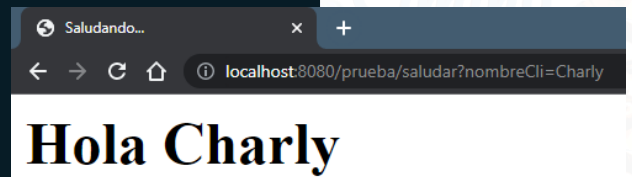
```
<html>
    <head></head>
    <body>
        <h1>Saludador</h1>
        <form action="saludar">
            <label>Nombre:</label> <br>
            <input name="nombreCli">
            <br><br>
            <input type="submit" value="Saludar">
            <input type="reset">
        </form>
    </body>
</html>
```



SaludarServlet.java

```
@WebServlet(name = "saludar", urlPatterns = {"/saludar"})
public class SaludarServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        String nombre = req.getParameter("nombreCli");
        PrintWriter out = resp.getWriter();
        out.println("<html><head>");
        out.println("<title>Saludando...</title>");
        out.println("</head><body>");
        out.println("<h1>Hola " + nombre + "</h1>");
        out.println("</body></html>");     }
}
```

Contenido dinámico
(Se crea al momento de la petición y se le presenta al cliente)



¿Qué es un *servlet*?

La definición de *servlet* tiene varios significados según el contexto

Una tecnología… que permite crear aplicaciones web dinámicas.

Una API… que provee clases, interfaces y documentación.

Una interfaz… que debe ser implementada para crear un *servlet*.

Un componente web… que se despliega en el servidor para generar contenido dinámico.

Ventajas de los *servlets*

Buena *performance*

- Crean un nuevo hilo por cada solicitud, mejorando la concurrencia, frente a la tecnología CGI (*Common Gateway Interface*) que siempre crea nuevos procesos.

Robusto, portable y seguro

- Al utilizar el lenguaje Java, nos podemos abstraer de la pérdida de memoria, recolección de basura (*garbage collector*), etc.

Tecnología madura

- Lleva muchos años en el mercado, por lo que se dispone de una basta comunidad de desarrolladores y gran cantidad de documentación.

Crear un *servlet* para responder peticiones HTTP

Solo se debe crear una clase que extienda de **`HttpServlet`** y sobrescribir los métodos que interesen, de acuerdo a qué tipo de peticiones HTTP escuchar.

En versiones anteriores de *JavaEE* (antes de cambiar de nombre a *JakartaEE*), el paquete se llama `javax`.

PrimerServlet.java

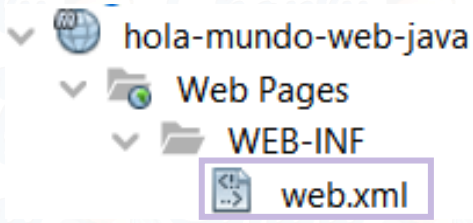
```java
// Se omiten los otros imports y package
import jakarta.servlet.http.HttpServlet;

public class PrimerServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) {
        /*
            Lógica a desarrollar cuando se recibe una petición GET
        */
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) {
        /*
            Lógica a desarrollar cuando se recibe una petición POST
        */
    }

    /*
        Disponibles también los métodos doHead, doOptions, doPut, doTrace y doDelete,
        correspondientes a los otros verbos HTTP.
    */
}
```

En todos los métodos **do...** (**`doGet`**, **`doPost`**, etc.) llegan por parámetro dos objetos de tipos **`HttpServletRequest`** y **`HttpServletResponse`**. Estos representan, respectivamente, la petición (*request*) obtenida desde el cliente y la respuesta (*response*) que enviará el servidor.

Descriptor de implementación (web.xml)

Para que un *servlet* pueda ser accedido por el cliente, este debe estar mapeado a una URL.

Esta declaración (y otras configuraciones del proyecto) se hace dentro del descriptor de implementación (*web deployment descriptor*), que es un archivo llamado **web.xml** y debe encontrarse dentro de la carpeta **WEB-INF**.



Las declaraciones se harán con etiquetas XML anidadas dentro de las etiquetas `<web-app></web-app>`

web.xml

```
<web-app>
    ...
</web-app>
```

Los atributos `xmlns`, `xmlns:xsi`, `xsi:schemaLocation` y `version` definen el espacio de nombres XML.

No es relevante comprenderlo para seguir adelante.

web.xml

```
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
   version="6.0">
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

Se está estableciendo el tiempo que dura la sesión HTTP (30 minutos).

Archivo web.xml que genera NetBeans al inicio del proyecto, con *JakartaEE* 10.0

Mapeo de *servlet*

Para que un *servlet* pueda ser accedido por el cliente, este debe estar mapeado a una URL.

web.xml

```xml
<web-app>
    <servlet>
        <servlet-name>bienvenida</servlet-name>
        <servlet-class>
            ar.charlycimino.ejemplos.javaservlets.ppt.BienvenidaServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>bienvenida</servlet-name>
        <url-pattern>/welcome</url-pattern>
    </servlet-mapping>
</web-app>
```

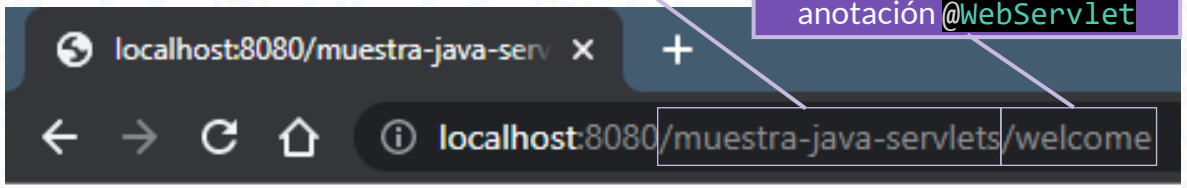
Etiqueta	Descripción	Subetiqueta	Descripción
`<servlet>`	Describe la información de uno o más *servlets*. (Ver detalle completo)	`<servlet-name>`	Especifica el nombre del *servlet*, que se usará para referirse a él en otras partes del documento.
		`<servlet-class>`	Especifica el nombre completo de la clase Java (incluyendo el paquete) que representa al *servlet*.
`<servlet-mapping>`	Describe la forma de acceder a uno o más *servlets*. (Ver detalle completo)	`<servlet-name>`	Especifica el nombre del *servlet* que se va a mapear.
		`<url-pattern>`	Especifica la URL que se debe colocar (tras el *context path*) para acceder al *servlet*.

BienvenidaServlet.java

```java
// Se omiten imports y package
public class BienvenidaServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        PrintWriter out = resp.getWriter();
        out.println("<html><head></head><body>");
        out.println("<h1>Hola mundo web</h1>");
        out.println("</body></html>");
    }
}
```

Recordar que Tomcat monta la aplicación en una ruta (llamada *context path*) que lleva el mismo nombre que el proyecto.

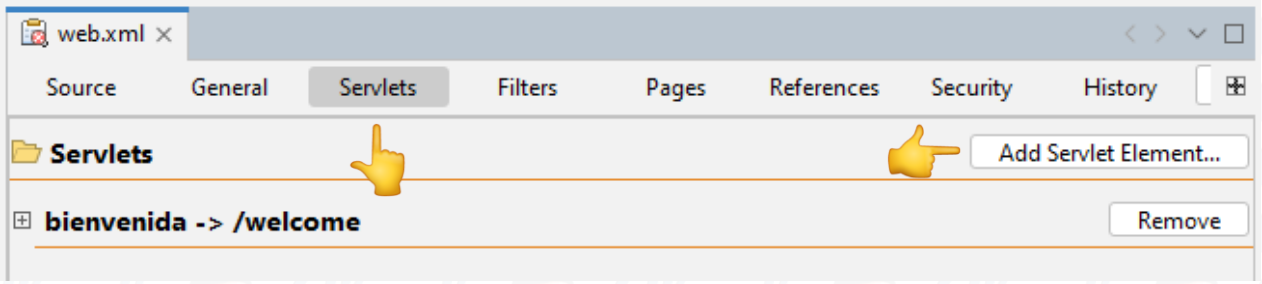
Como se especificó en el atributo `urlPatterns` de la anotación `@WebServlet`

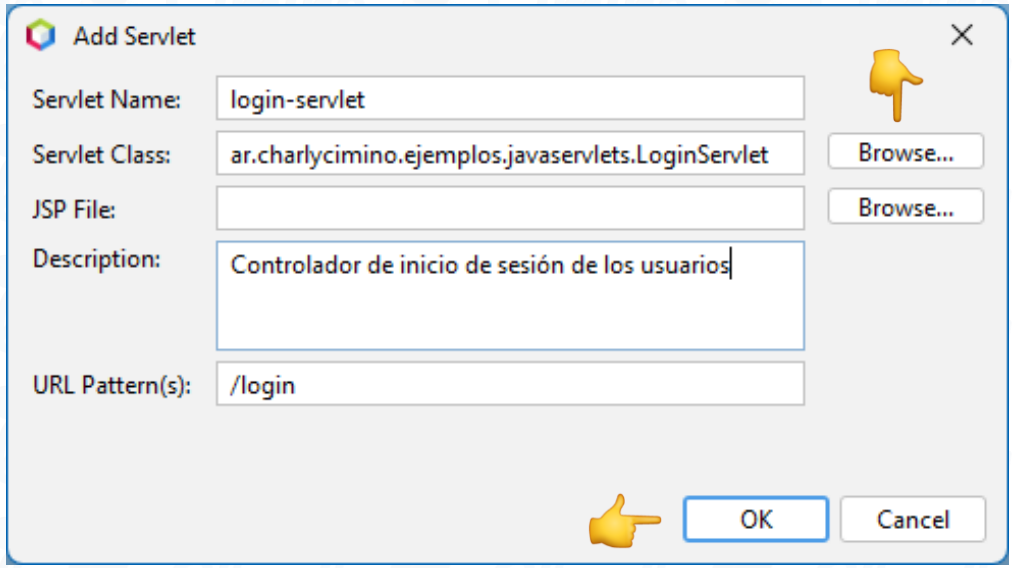


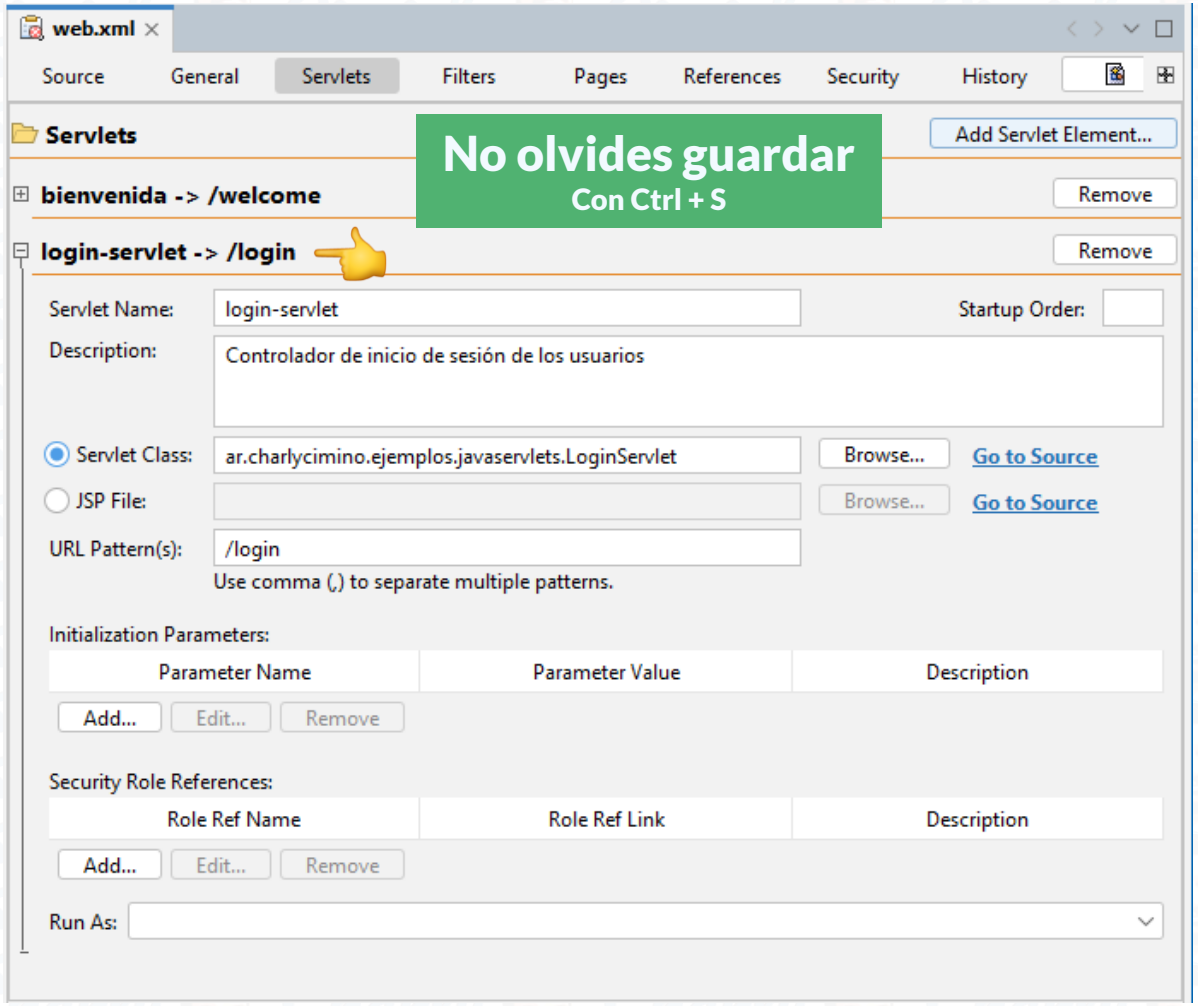
Hola mundo web

Mapeo de *servlet* mediante interfaz gráfica

NetBeans provee de un asistente para agregar y configurar *servlets* sin necesidad de manipular directamente el código XML







Mapeo de *servlet* con *annotations*

A partir de la especificación 3.0 de la API de Servlet (2009), pueden describirse algunos datos (no todos) de los *servlets* directamente en sus clases Java, a través de anotaciones (*annotations*).

BienvenidaServlet.java

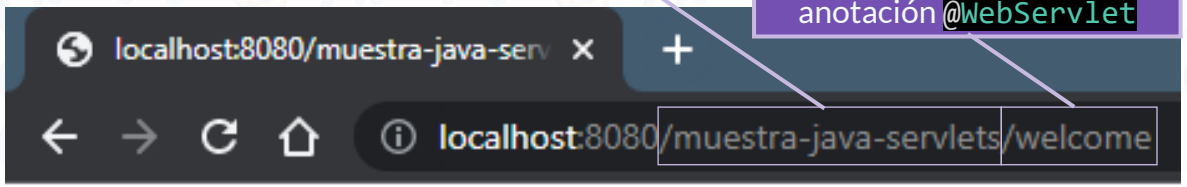
```java
// Se omiten imports y package
@WebServlet(name = "bienvenida", urlPatterns = "/welcome")
public class BienvenidaServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        PrintWriter out = resp.getWriter();
        out.println("<html><head></head><body>");
        out.println("<h1>Hola mundo web</h1>");
        out.println("</body></html>");
    }
}
```

Hay que importarla.

Recordar que Tomcat monta la aplicación en una ruta (llamada *context path*) que lleva el mismo nombre que el proyecto.

Como se especificó en el atributo `urlPatterns` de la anotación `@WebServlet`

Anotación	Descripción	Atributo	Descripción
`@WebServlet`	Describe la información de un *servlet*. (Ver detalle completo)	`name`	Especifica el nombre del *servlet*.
		`urlPatterns`	Especifica la URL que se debe colocar (tras el *context path*) para acceder al *servlet*. Puede ser más de una, si se listan las URLs dentro de un par de llaves y separadas por comas.



Hola mundo web

Peticiones del cliente: Datos y URL

Métodos en `HttpServletRequest`

Categoría	Método	Descripción
Datos de la petición	`getProtocol()`	Devuelve el nombre y la versión del protocolo que usa la solicitud en el formato *protocol/majorVersion.minorVersion*, por ejemplo, **HTTP/1.1**.
	`getMethod()`	Devuelve el nombre del método HTTP con el que se realizó esta solicitud, por ejemplo, **GET**, **POST**, **PUT**, etc.
	`getScheme()`	Devuelve el nombre del esquema utilizado para realizar esta solicitud, por ejemplo, **http**, **https** o **ftp**.
Datos del servidor	`getServerName()`	Devuelve el nombre de host (o dirección IP) del servidor al que se envió la solicitud.
	`getServerPort()`	Devuelve el número de puerto al que se envió la solicitud.
Datos del cliente	`getRemoteAddr()`	Devuelve la dirección IP del cliente o último proxy que envió la solicitud.
	`getRemoteHost()`	Devuelve el nombre de host (o dirección IP) del cliente o último proxy que envió la solicitud.
	`getRemotePort()`	Devuelve el número de puerto de origen del cliente o último proxy que envió la solicitud.
Partes de la URL	`getRequestURL()`	Reconstruye la URL que el cliente usó para realizar la solicitud.
	`getRequestURI()`	Devuelve desde el *context path* (inclusive) hasta la *query string* (sin incluir) de la URL de la solicitud
	`getContextPath()`	Devuelve el *context path* de la URL de la solicitud.
	`getServletPath()`	Devuelve el *servlet* al que llama la URL de esta solicitud
	`getPathInfo()`	Devuelve la información adicional de la URL de esta solicitud (solo si el *servlet* fue mapeado con un `/*` al final, sino devuelve `null`)
	`getQueryString()`	Devuelve *query string* de la URL de la solicitud (si no existe, devuelve `null`).

Peticiones del cliente: Datos y URL (ejemplo)

MuestraRequestServlet1.java

El * es un comodín que permite que cualquier URL a partir de su aparición haga coincidencia con este *servlet*.

```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-req1/*"})
public class MuestraRequestServlet1 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        PrintWriter out = resp.getWriter();
        out.print("<html><head>");
        out.print("<title>Viendo una petición HTTP</title>");
        out.print("</head><body>");
        out.print("<p><strong>getProtocol</strong>: " + req.getProtocol() + "</p>");
        out.print("<p><strong>getMethod</strong>: " + req.getMethod() + "</p>");
        out.print("<p><strong>getScheme</strong>: " + req.getScheme() + "</p>");
        out.print("<hr>");
        out.print("<p><strong>getServerName</strong>: " + req.getServerName() + "</p>");
        out.print("<p><strong>getServerPort</strong>: " + req.getServerPort() + "</p>");
        out.print("<hr>");
        out.print("<p><strong>getRemoteAddr</strong>: " + req.getRemoteAddr() + "</p>");
        out.print("<p><strong>getRemoteHost</strong>: " + req.getRemoteHost() + "</p>");
        out.print("<p><strong>getRemotePort</strong>: " + req.getRemotePort() + "</p>");
        out.print("<hr>");
        out.print("<p><strong>getRequestURL</strong>: " + req.getRequestURL() + "</p>");
        out.print("<p><strong>getRequestURI</strong>: " + req.getRequestURI() + "</p>");
        out.print("<p><strong>getContextPath</strong>: " + req.getContextPath() + "</p>");
        out.print("<p><strong>getServletPath</strong>: " + req.getServletPath() + "</p>");
        out.print("<p><strong>getPathInfo</strong>: " + req.getPathInfo() + "</p>");
        out.print("<p><strong>getQueryString:</strong> " + req.getQueryString() + "</p>");
        out.print("</body></html>");
    }
}
```

Respuesta recibida tras enviar una petición desde un *smartphone* conectado a la misma red local que el servidor:

http://192.168.0.82:8080/muestra-java-servlets/muestra-request-servlet1?nom=Alan&ape=Turing

Respuesta recibida tras enviar una petición desde una PC conectada a la misma red local que el servidor:

http://localhost:8080/muestra-java-servlets/servlet-req1/algunaAccion

Peticiones del cliente: *Headers* y Parámetros

Más métodos en `HttpServletRequest`

Categoría	Método	Descripción
Cabeceras (*headers*)	`getHeader(String nombre)`	Devuelve el valor del *header* correspondiente (como `String`) en la petición
	`getIntHeader(String nombre)`	Devuelve el valor del *header* correspondiente (como `int`) en la petición
	`getDateHeader(String nombre)`	Devuelve el valor del *header* correspondiente (como número de milisegundos en formato `long`) en la petición
	`getHeaderNames()`	Devuelve una `Enumeration<String>` de todos los nombres de los *headers* de la petición.
Parámetros*	`getParameter(String nombre)`	Devuelve el valor del parámetro correspondiente como un `String`, (o `null` si no existe ningún parámetro para el nombre dado) en la petición.
	`getParameterNames()`	Devuelve una `Enumeration<String>` que contiene todos los nombres de los parámetros de la petición.
	`getParameterMap()`	Devuelve un `Map<String,String[]>` que contiene todos los pares **<nombre,valores>** de los parámetros de la petición.

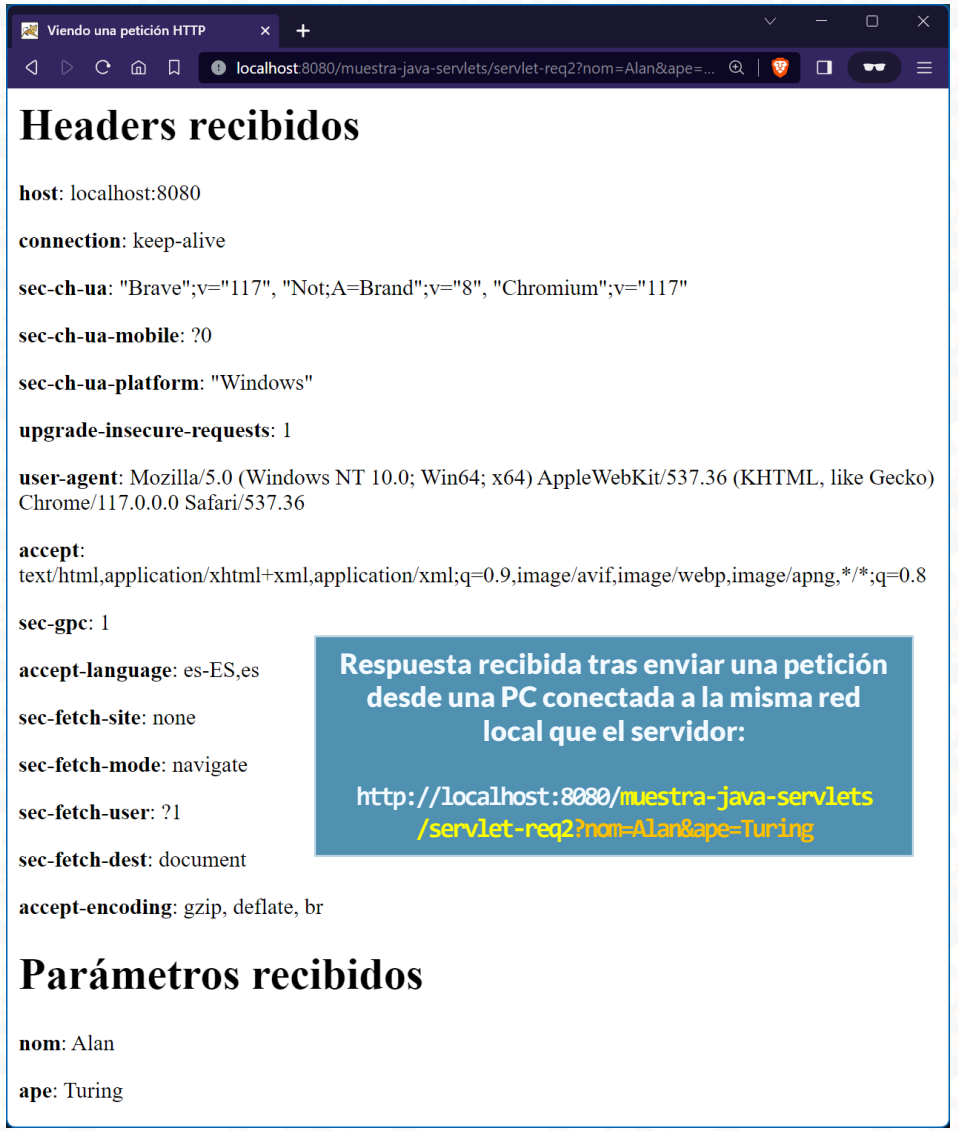
* Los parámetros son datos que envía el cliente (generalmente a través de formularios) como parte de la petición.

Pueden llegar en la *query string* de la URL (comúnmente en peticiones **GET**) o en el cuerpo de la petición HTTP (comúnmente en peticiones **POST**).

Peticiones del cliente: *Headers* y Parámetros (ejemplo)

MuestraRequestServlet2.java

```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-req2"})
public class MuestraRequestServlet2 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        PrintWriter out = resp.getWriter();
        out.print("<html><head>");
        out.print("<title>Viendo una petición HTTP</title>");
        out.print("</head><body>");

        Enumeration<String> nombresHeaders = req.getHeaderNames();
        out.print("<h1>Headers recibidos</h1>");
        while (nombresHeaders.hasMoreElements()) {
            String nH = nombresHeaders.nextElement();
            out.println("<p><strong>" + nH + "</strong>: " + req.getHeader(nH) + "</p>");
        }

        Enumeration<String> nombresParams = req.getParameterNames();
        out.print("<h1>Parámetros recibidos</h1>");
        while (nombresParams.hasMoreElements()) {
            String nP = nombresParams.nextElement();
            out.println("<p><strong>" + nP + "</strong>: " + req.getParameter(nP) + "</p>");
        }

        out.print("</body></html>");
    }
}
```



Peticiones del cliente: Atributos y Redirección

Aún más métodos en `HttpServletRequest`

Categoría	Método	Descripción
Atributos	`getAttribute(String nombre)`	Devuelve el valor del atributo correspondiente como un `Object`, (o `null` si no existe ningún atributo para el nombre dado) de la petición.
	`getAttributeNames()`	Devuelve una `Enumeration<String>` que contiene todos los nombres de los atributos de la petición.
	`removeAttribute(String nombre)`	Elimina el atributo correspondiente de la petición.
	`setAttribute(String nombre, Object valor)`	Almacena un atributo con su respectivo valor en la petición.
Redirección	`getRequestDispatcher(String path)`	Devuelve un objeto `RequestDispatcher` que permitirá incluir o redireccionar a otro recurso en la ruta dada.

Métodos en `RequestDispatcher`

Método	Descripción
`forward(ServletRequest request, ServletResponse response)`	Redirecciona desde el *servlet* actual a otro reenviando la petición y respuesta.
`include(ServletRequest request, ServletResponse response)`	Incluye el contenido de otro recurso (*servlet*, JSP, HTML) en la respuesta.

Peticiones del cliente: Atributos y Redirección (ejemplo)

MuestraRequestServlet3.java

```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-req3"})
public class MuestraRequestServlet3 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setAttribute("pais", "Argentina");
        req.setAttribute("codigo", 54);
        req.setAttribute("el10", new Persona("Leo","Messi"));
        req.setAttribute("hoy", LocalDate.now());

        req.getRequestDispatcher("servlet-req4").forward(req, resp);
    }
}
```

MuestraRequestServlet4.java

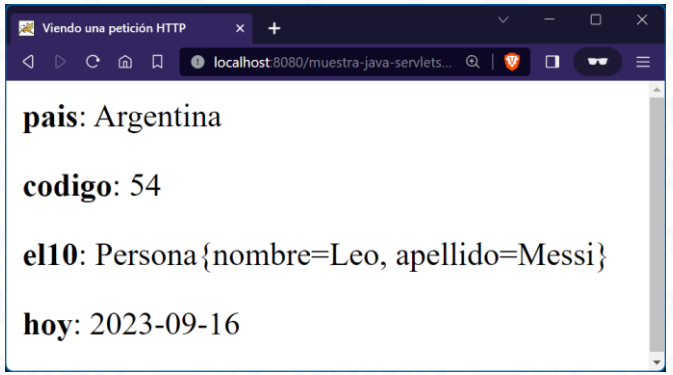
```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-req4"})
public class MuestraRequestServlet4 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        PrintWriter out = resp.getWriter();
        out.print("<html><head>");
        out.print("<title>Viendo una petición HTTP</title>");
        out.print("</head><body>");
        out.println("<p><strong>pais</strong>: " + req.getAttribute("pais") + "</p>");
        out.println("<p><strong>codigo</strong>: " + req.getAttribute("codigo") + "</p>");
        out.println("<p><strong>el10</strong>: " + req.getAttribute("el10") + "</p>");
        out.println("<p><strong>hoy</strong>: " + req.getAttribute("hoy") + "</p>");
        out.print("</body></html>");
    }
}
```

Persona.java

```java
public class Persona {

    private String nombre;
    private String apellido;

    public Persona(String nombre, String apellido) {
        this.nombre = nombre;
        this.apellido = apellido;
    }

    @Override
    public String toString() {
        return "Persona{" + "nombre=" + nombre
                + ", apellido=" + apellido + '}';
    }
}
```

Respuesta recibida tras enviar una petición desde una PC conectada a la misma red local que el servidor:

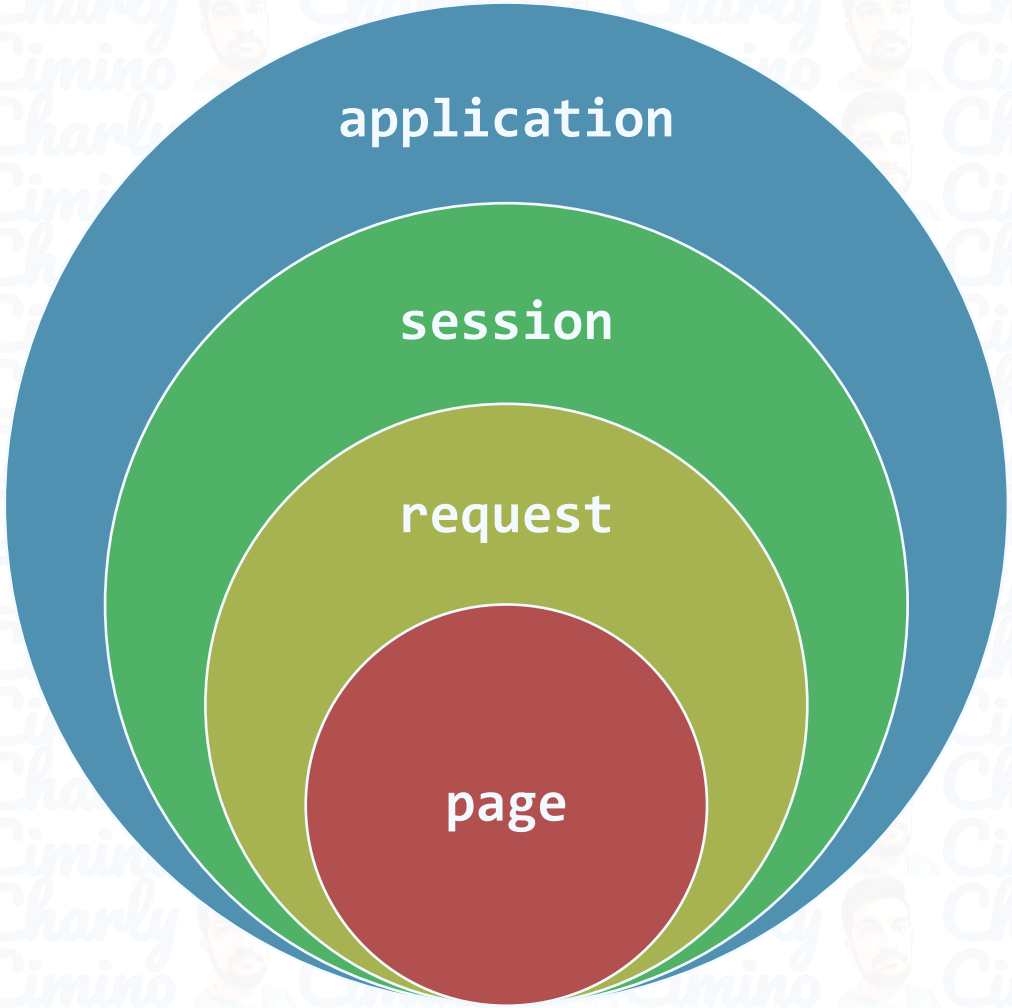
http://localhost:8080/muestra-java-servlets/servlet-req3



Parámetros vs. Atributos

Parámetros	Atributos
Son datos recibidos desde el cliente, que pueden llegar a través de la *query string* de la URL (comúnmente en peticiones GET) o en el cuerpo de la petición HTTP (comúnmente en peticiones POST).	Son datos generados en el servidor y que solo pueden ser accesibles dentro de él.
Solo pueden ser cadenas de caracteres	Pueden ser cualquier tipo de dato (número, booleano, caracter, cadena, objeto, lista, etc)
Son de solo lectura (no se pueden eliminar)	Se pueden generar, reemplazar y borrar
Solo están ligados a la petición (*request*)	Pueden estar ligados a la página (*page*), petición (*request*), sesión (*session*) o aplicación (*application*) **Detalle en la próxima diapositiva**

Ámbito (*scope*) de los atributos



Ámbito	Duración	Ejemplo
Aplicación (*application*)	El dato persiste durante todo el ciclo de vida de la **aplicación** (o hasta que sea borrado)	`getServletContext().setAttribute("atrr1", "valor1")`
Sesión (*session*)	El dato persiste durante todo el ciclo de vida de la **sesión** (o hasta que sea borrado)	`req.getSession().setAttribute("atrr2", "valor2")`
Petición (*request*)	El dato persiste durante todo el ciclo de vida de la **petición** (o hasta que sea borrado)	`req.setAttribute("atrr3", "valor3")`
Página (*page*)	**Solo para JSP.** El dato persiste durante todo el ciclo de vida de la **página** (o hasta que sea borrado)	**(Ver en la PPT "Java Web JSP").**

Respuestas al cliente: *status* y *headers*

Métodos en `HttpServletResponse`

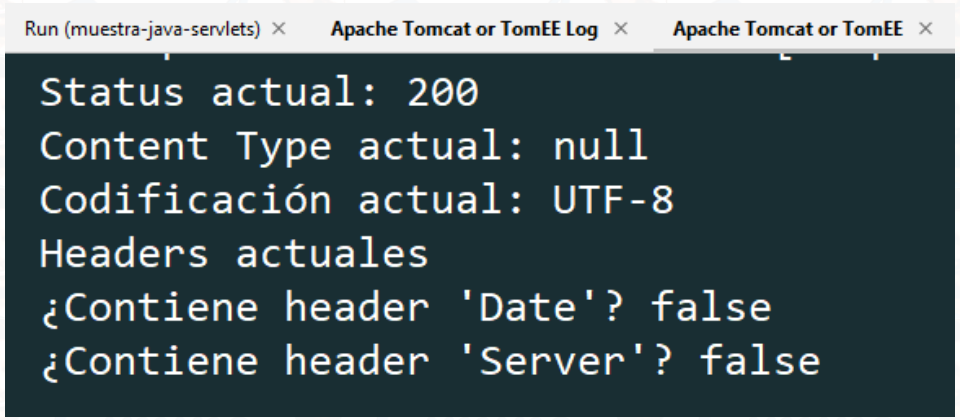
Categoría	Método	Descripción
Estado (*status*)	`getStatus()`	Obtiene el código de estado actual de la respuesta.
	`setStatus()`	Establece el código de estado para la respuesta.
Cabeceras (*headers*)	`setContentType(String tipo)`	Establece el tipo de contenido MIME de la respuesta en el *header* `Content-Type`
	`getContentType()`	Devuelve el tipo de contenido MIME de la respuesta.
	`setContentLengthLong(long len)` `setContentLengthLong(int len)`	Establece la longitud del cuerpo de la respuesta (en bytes) en el *header* `Content-Length`
	`setCharacterEncoding(String charset)`	Establece la codificación de caracteres de la respuesta, por ejemplo, `"UTF-8"`.
	`getCharacterEncoding()`	Devuelve la codificación de caracteres utilizada en la respuesta.
	`addHeader(String nombre, String valor)`	Agrega un nuevo *header* de respuesta con el nombre y el valor (cadena) dados.
	`addIntHeader(String nombre, int valor)`	Agrega un nuevo *header* de respuesta con el nombre y el valor (entero) dados.
	`addDateHeader(String nombre, long ms)`	Agrega un nuevo *header* de respuesta con el nombre y el valor (número de milisegundos) dados.
	`setHeader(String nombre, String valor)`	Establece un *header* de respuesta con el nombre y el valor (cadena) dados.
	`setIntHeader(String nombre, int valor)`	Establece un *header* de respuesta con el nombre y el valor (entero) dados.
	`setDateHeader(String nombre, long ms)`	Establece un *header* de respuesta con el nombre y el valor (número de milisegundos) dados.
	`containsHeader(String nombre)`	Devuelve un booleano que indica si el *header* de respuesta correspondiente ya se ha establecido.
	`getHeader(String nombre)`	Devuelve el valor del *header* correspondiente (como cadena) en la respuesta.
	`getHeaderNames()`	Devuelve una colección de cadenas con los nombres de los *headers* actualmente en la respuesta.

Respuestas al cliente: *status* y *headers* (ejemplo)

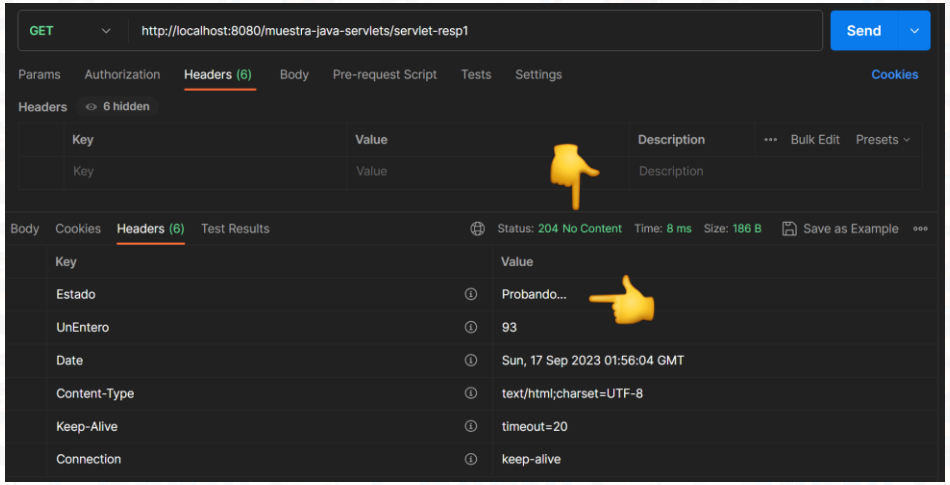
MuestraResponseServlet1.java

```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-resp1"})
public class MuestraResponseServlet1 extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        System.out.println("Status actual: " + resp.getStatus());
        System.out.println("Content Type actual: " + resp.getContentType());
        System.out.println("Codificación actual: " + resp.getCharacterEncoding());

        Collection<String> nombresHeaders = resp.getHeaderNames();
        System.out.println("Headers actuales");
        for (String nombreHeader : nombresHeaders) {
            System.out.println("\t" + nombreHeader + ": " + resp.getHeader(nombreHeader));
        }

        System.out.println("¿Contiene header 'Date'? " + resp.containsHeader("Date"));
        System.out.println("¿Contiene header 'Server'? " + resp.containsHeader("Server"));

        resp.addHeader("Estado", "Probando...");
        resp.addIntHeader("UnEntero", 93);
        resp.addDateHeader("Date", System.currentTimeMillis());

        resp.setStatus(204); // Status 'No Content'
    }
}
```

Las salidas por consola iniciales se observan en la consola de Tomcat.



Los *headers* y *status* de respuesta pueden verse fácilmente con Postman.



Respuestas al cliente: *Buffer* y redirección

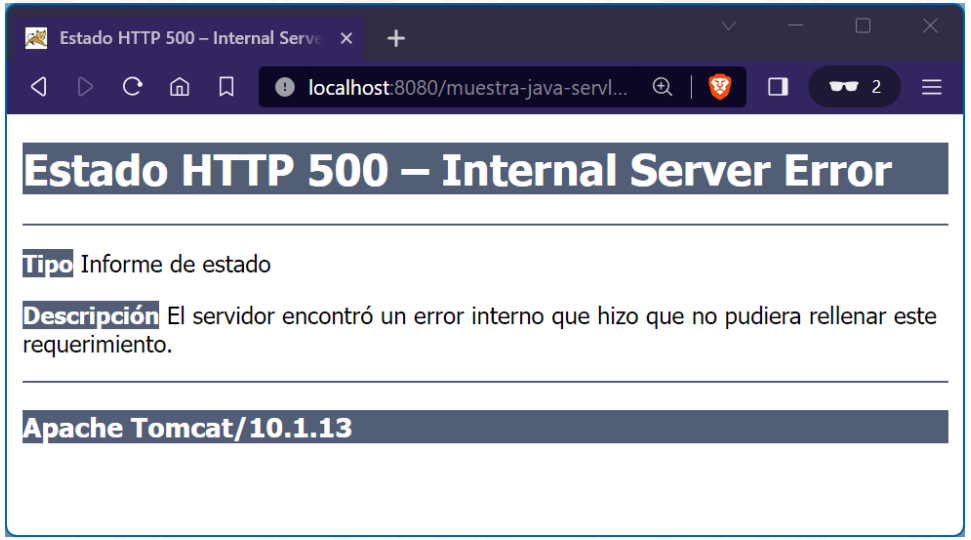
Más métodos en `HttpServletResponse`

Categoría	Método	Descripción
Buffer	`getWriter()`	Devuelve un objeto `PrintWriter` que permite colocar texto en el cuerpo de la respuesta (mediante los métodos `print`).
	`reset()`	Borra cualquier dato que exista en el búfer, así como el código de estado y encabezados.
	`resetBuffer()`	Borra el contenido del búfer subyacente en la respuesta sin borrar los encabezados ni el código de estado.
	`setBufferSize(int size)`	Establece el tamaño de búfer preferido para el cuerpo de la respuesta.
	`getBufferSize()`	Devuelve el tamaño de búfer real utilizado para la respuesta.
	`flushBuffer()`	Obliga a que cualquier contenido del búfer se escriba en el cliente.
Redirección	`sendError(int codigo)`	Envía una respuesta de error al cliente utilizando el código de estado especificado, luego borra el búfer.
	`sendError(int codigo, String mensaje)`	Envía una respuesta de error al cliente utilizando el código de estado especificado y un mensaje, luego borra el búfer.
	`sendRedirect(String path)`	Envía una respuesta de redireccionamiento temporal al cliente utilizando la URL de redireccionamiento especificada, luego borra el búfer.

Respuestas al cliente: *Buffer* y redirección (ejemplo 1)

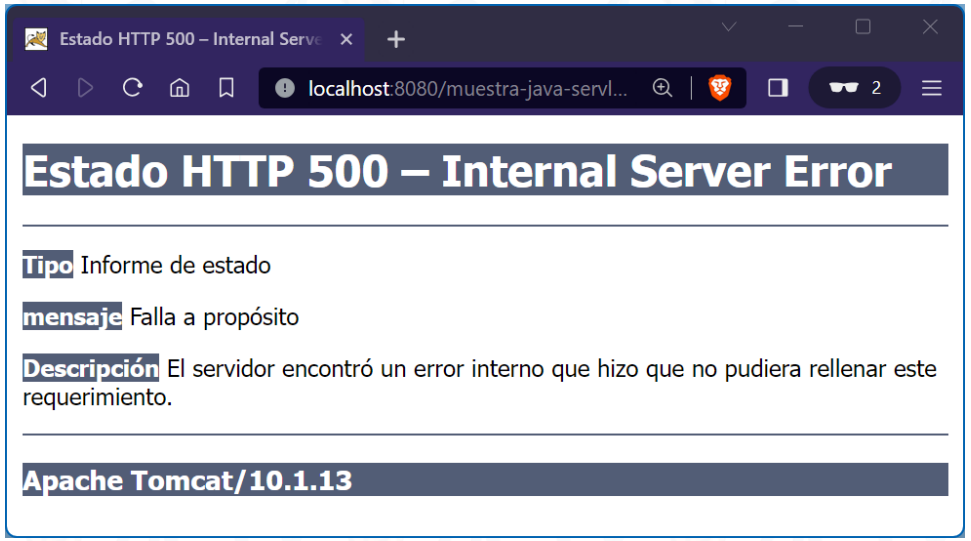
MuestraResponseServlet2.java

```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-resp2"})
public class MuestraResponseServlet2 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.sendError(500); // Status 'Internal Server Error'
    }
}
```



MuestraResponseServlet3.java

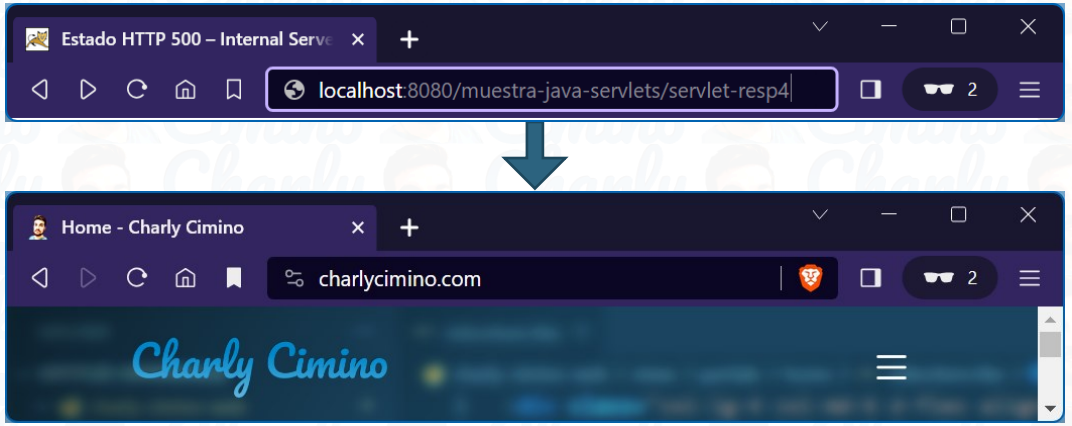
```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-resp3"})
public class MuestraResponseServlet3 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.sendError(500, "Falla a propósito");
    }
}
```



Respuestas al cliente: *Buffer* y redirección (ejemplo 2)

MuestraResponseServlet4.java

```java
// Se omiten imports y package
@WebServlet(urlPatterns = {"/servlet-resp4"})
public class MuestraResponseServlet4 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.sendRedirect("https://charlycimino.com");
    }
}
```



`req.getRequestDispatcher("…").forward(req, resp);`	`resp.sendRedirect("…");`
Redirección en el lado del **servidor**	Redirección en el lado del **cliente**
Envía las mismas referencias a los objetos *request* y *response*, permitiendo continuar la cadena de petición/respuesta.	Inicia una nueva *request* en el *browser* del cliente (es como si hubiera colocado la URL en la barra de direcciones).

FIN DE LA PRESENTACIÓN

Encontrá más como estas en mi sitio web.