

Práctica 4  
Macedo Borbolla Eduardo Sai  
Sesiones  
Web Application Development  
18 de Septiembre del 2018

**Teoría:** El protocolo HTTP es un protocolo 'stateless' sin estado o sin memoria, es por esto que para establecer una comunicación con 'memorización' o con estado es necesario el uso de sesiones HTTP con uno o más clientes de nuestra aplicación web.

Por definición, una sesión HTTP es una secuencia de transacciones de red de peticiones y respuestas

Para trabajar sesiones podemos hacerlo mediante el objeto de sesión HttpSession o el uso de Cookies dependiendo de si queremos que la persistencia de la sesión sea a nivel navegador o a nivel IP incluyendo todos sus navegadores.

**Desarrollo:**

1.-Implementar un filtro que intercepte cualquier petición que se haga a la aplicación web e imprima en log: Filter: IP fecha y hora método recurso

```

package puntos;

import java.io.IOException;
import java.util.Date;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletResponse;

public class FilterIP implements Filter {
    private ServletContext context;
    public FilterIP() {
        super();
    }

    public void destroy() {
    }
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {
        String ip= request.getRemoteAddr();
        Date date= new Date();
        System.out.println("Filter:"+ip+ " " + date);
        chain.doFilter(request, response);
        //      httpResponse.sendRedirect("/practica3");

    }
    public void init(FilterConfig fConfig) throws ServletException {
        System.out.println("-->Filtro publico monitoreando");
    }
}

```

Este filtro se tomó de la práctica anterior, no hubo problema en su desarrollo, el filtro monitorea todas las peticiones dentro de la página, su contexto es el acceso público.

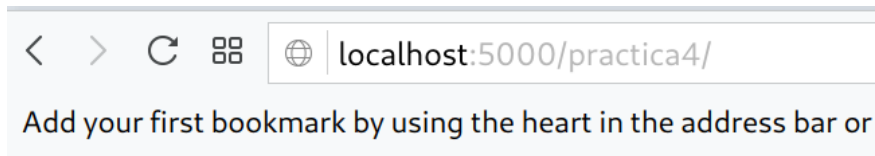
Al correr la aplicación en Jetty obtenemos lo siguiente:

```

2018-10-01 11:01:38.871:INFO::jetty-6.1.26
-->Filtro publico monitoreando
2018-10-01 11:01:39.020:INFO:/practica4:Filtro privado instanciado
Filtro Privado comenzando
2018-10-01 11:01:39.046:INFO::Started SelectChannelConnector@0.0.0.0:5000

```

Ahora accedemos a la página



# Login

- [Ingresar al sitio](#)
- [Salir del sitio](#)
- [Sesion](#)
- [Blue Demon \(Recurso Publico\)](#)
- [Final Spoiler super campeones\(Recurso Privado\)](#)

Y el acceso del cliente genera la siguiente salida en el log:

```
Filtro Privado comenzando  
2018-10-01 11:01:39.046:INFO::Started SelectChannelConnector@0.0.0.0:5000  
Filter:0:0:0:0:0:0:0:1 Mon Oct 01 11:03:00 CDT 2018
```

Al realizar este filtro la única dificultad que tuve fue el plantear lo que sería un recurso público y uno privado.

2.-Implementar un servlet que simule un login. Considerando como parámetros válidos:

- usuario=necaxa
- password=aguinaga

Cuando se ingrese un usuario y contraseña válidos la aplicación deberá subir un objeto *Usuario* a la sesión y mostrar la pantalla de bienvenida.

Para el desarrollo de este servlet definí una cadena con los parámetros pedidos en la especificación y se realizaron las validaciones que solo permiten ingresar al usuario con las credenciales en código duro, de ser válido se le crea un objeto sesión y se “suben” a sesión los atributos “usuario” y “password”

De no ser válido el login se da un mensaje de retroalimentación que se incluye al mismo servlet mediante un include del RequestDispatcher.

```

package puntos;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class ServletLogin extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final String us="necaxa";
    private final String p="aguinagua";

    public ServletLogin() {
        super();
    }

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

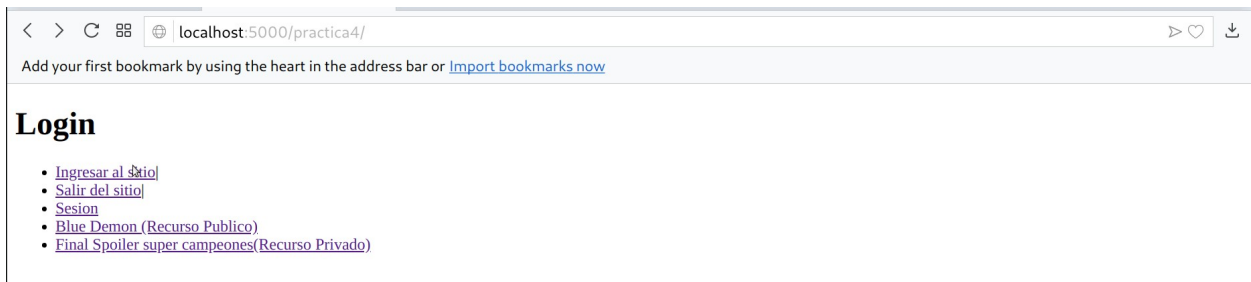
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        PrintWriter out= response.getWriter();
        String usuario=request.getParameter("usuario");
        String password=request.getParameter("password");
        System.out.println(usuario+" "+password);

        if(us.equals(usuario)&&p.equals(password)){
            HttpSession sesion=request.getSession();
            sesion.setAttribute("usuario",usuario);
            sesion.setAttribute("password", password);
            response.sendRedirect("/practica4/Sesion");        }
        else {
            RequestDispatcher rd=getServletContext().getRequestDispatcher("/login.html");
            out.println("<font color=red>Nombre o contraseña incorrectos</font>");
            rd.include(request, response);
        }
    }
}

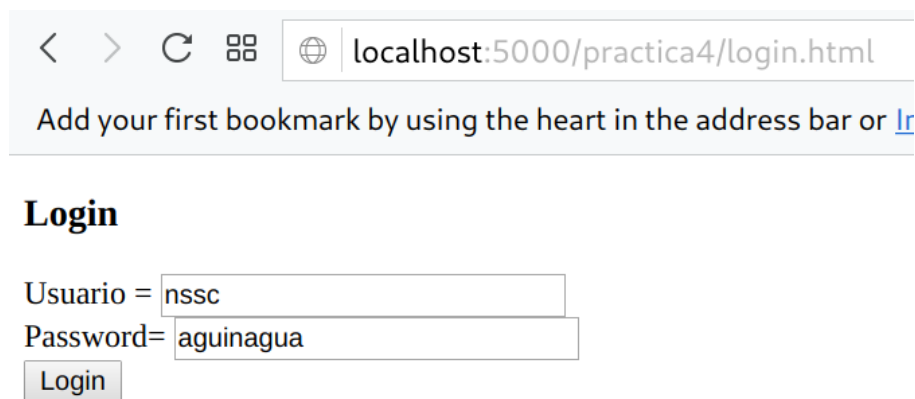
```

Ahora procedemos a ejecutar el servlet incorrecta y correctamente.

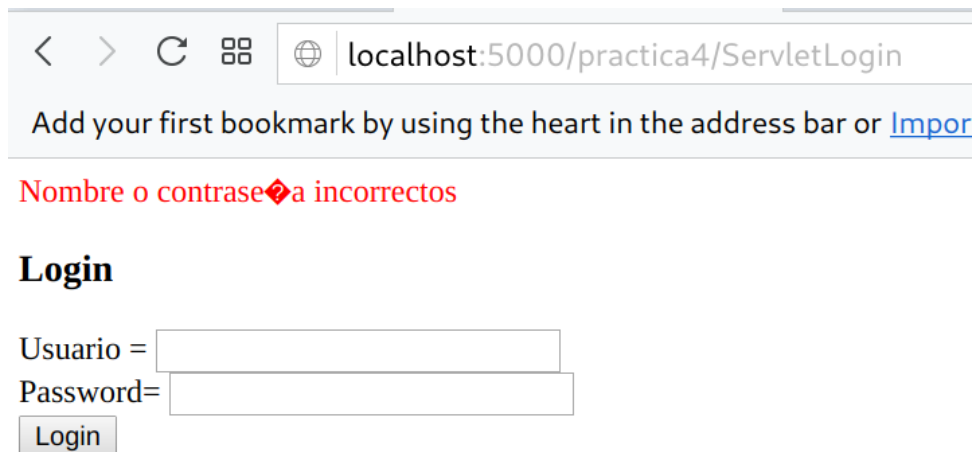
Primero vamos al login.html mediante el hipervíncul “Ingresar al sitio”



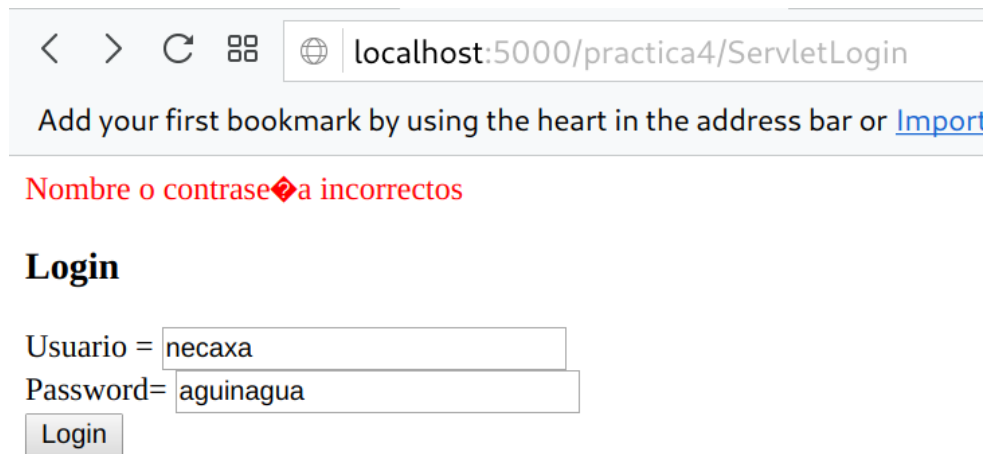
Luego proporcionamos credenciales incorrectas para verificar nuestra validación.



Obtenemos un mensaje de error volviendo a la misma URL.



Ahora ingresamos las credenciales correctamente.



< > ↺ ☰ | localhost:5000/practica4/ServletLogin

Add your first bookmark by using the heart in the address bar or [Import](#)

Nombre o contraseña incorrectos

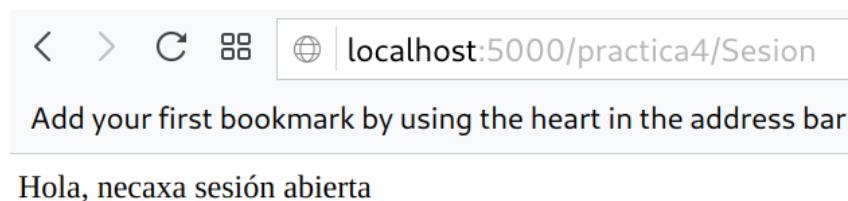
## Login

Usuario =

Password=

Login

Y damos click a login.



< > ↺ ☰ | localhost:5000/practica4/Sesion

Add your first bookmark by using the heart in the address bar

Hola, necaxa sesión abierta

Podemos ver el nombre de nuestro usuario obtenido por sesión exitosamente.

3.-Tome como base los ejercicios anteriores e implemente un filtro para el control de acceso que bloquee el acceso de la siguiente manera:

- Recursos públicos:
  - LoginServlet
- Recursos privados:
  - InfoServlet

En caso de que se intente ingresar a un recurso privado deberá mostrar en pantalla el mensaje "acceso no permitido".

Dentro de este filtro monitoreamos que la sesión esté iniciada y sea válida para poder ver este recurso.

Primero de la petición ponemos la sesión en “false” para que si no hay una sesión cree una.

Después verificamos si la sesión sigue siendo nula, de ser así no permitimos el acceso al recurso al usuario de la página no autenticado.

```

package puntos;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class FiltroPrivado implements Filter {
    private ServletContext context;

    public FiltroPrivado() {

    }

    public void destroy() {

    }

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
    throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;

        HttpSession session = req.getSession(false);

        if (session == null) {
            this.context.log("Intento de acceso no autorizado");
            RequestDispatcher rd= req.getRequestDispatcher("/practica4/index.html");
            ((HttpServletResponse) response).sendError(404,"Acceso no permido");
            // rd.forward(req, res);
        } else {

            chain.doFilter(request, response);
        }
    }

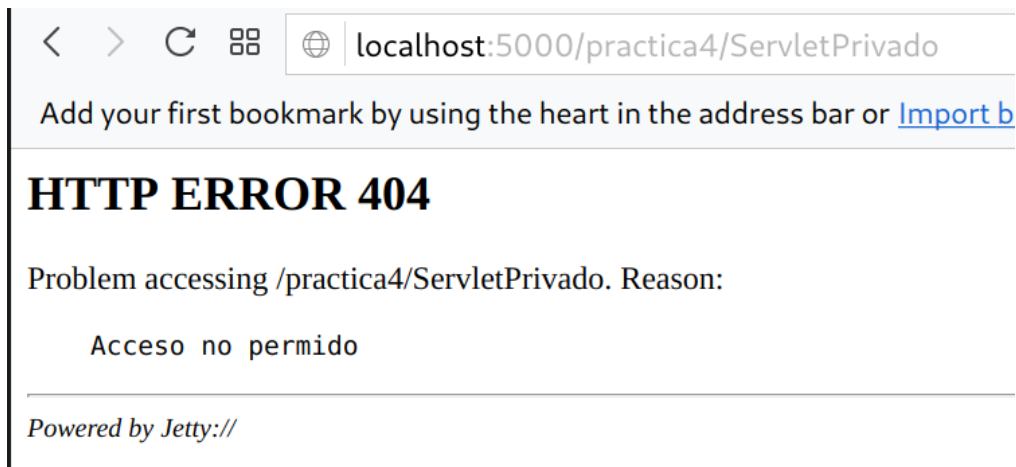
    public void init(FilterConfig fConfig) throws ServletException {
        this.context = fConfig.getServletContext();
        this.context.log("Filtro privado instanciado");
        System.out.println("Filtro Privado comenzando");

    }
}

```

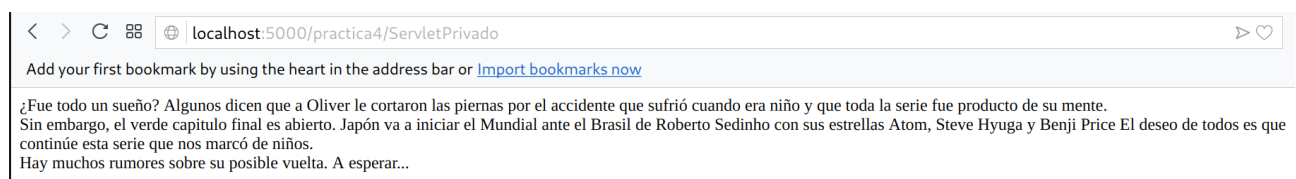
Procedemos a probar el filtro con nuestro recurso privado.

Al entrar sin sesión tenemos la siguiente salida del lado del cliente:



Y la siguiente salida del lado del servidor:

```
Filtro Privado comenzando
2018-10-01 11:37:49.162:INFO::Started SelectChannelConnector@0.0.0.0:5000
Filter:0:0:0:0:0:0:0:1 Mon Oct 01 11:37:56 CDT 2018
2018-10-01 11:37:56.250:INFO:/practica4:Intento de acceso no autorizado
```



Al ingresar con la sesión de necaxa:aguinagua tenemos esta salida para el recurso privado.

## Conclusiones:

La combinación de filtros con el manejo de sesiones Http nos permite tener una herramienta que nos facilite el trabajo de gestión de permisos usando los medios comunes de acceso a una página web (navegador) a nuestro favor, de igual manera podemos verificar los accesos y peticiones que realizan nuestros usuarios y público en general sin dar noticia de ello.



**Bibliografía:**

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