Práctica 7.6: Sesiones con *Servlets*

En esta práctica utilizaremos *Servlets* para gestionar sesiones en nuestra aplicación.

HTTP es un protocolo sin estado y no orientado a conexión, esto quiere decir que cada petición no conoce que se ha hecho en las peticiones anteriores. Sin embargo la mayoría de las aplicaciones comerciales necesitan:

- Sesión, es decir, el servidor debe ser capaz de diferenciar distintas solicitudes de un mismo cliente.
- Estado, es decir, el servidor debe ser capaz de almacenar/recordar información de anteriores solicitudes de un cliente.

Existen tres formas de gestionar las sesiones:

- *Cookies*. Pequeño fichero de texto almacenado en el cliente que se envia al servidor en cada petición. Se puede colocar la información de la sesión en una *cookie* de sesión. El mayor problema se produce porque los clientes pueden deshabilitar las *cookies*.
- Reescritura de *URLs*. Se añade un texto al final de cada *URL* con la información de la sesión, por ejemplo `http://localhost:8080/Sesiones/inicio.html;jsessionid=131908643`. Es necesario reescribir todas las *URLs* para incluir esta información.
- Campos ocultos en un formulario. Se pasa información de la sesión en campos ocultos del formulario. Hay que incluir esta información en todas las páginas.

Programar la gestión de la sesión mediante estos mecanismos puede resultar tedioso y complejo. Para mejorar esta situación el *API Servlet* proporciona un mecanismo de gestión de sesiones mediante la interfaz *HttpSession* que permite:

- Gestionar la sesión.
- Visualizar información de la sesión como su identificador o fecha de creación.
- Enlazar objetos a la sesión permitiendo que la información del usuario se mantenga a lo largo de múltiples peticiones.

La especificación del *API Servlet* realiza el registro de sesión utilizando el mecanismo de *cookies*, debido a que el servidor configura automáticamente una *cookie* "*jsessionid*" para el control de la sesión. Si el navegador del cliente no admite *cookies* automáticamente se emplea reescritura de *URLs*.

1. Para realizar un ejemplo de sesiones inicia sesión en el equipo **DesarrolloW7XX**.
2. Vamos a probar a crear y recuperar objetos de sesión con los métodos *session.setAttribute* y *session.getAttribute*. También veremos cómo obtener otra informaicón de la sesión como su identificador o su tiempo máximo permitido de inactividad.
3. Crea un nuevo *Dynamic Web Proyect* llamado **Sesiones** y añade un *Servlet* llamado **SessionServlet.java** con el siguiente código:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class SessionServlet extends HttpServlet {
 @Override
 public void doGet(HttpServletRequest request, HttpServletResponse response)
             throws IOException, ServletException {
   // Establece el tipo MIME del mensaje de respuesta
   response.setContentType("text/html");
   // Crea un flujo de salida para escribir la respuesta a la petición del cliente
   PrintWriter out = response.getWriter();

   // Recoge la sesión actual si existe, en otro caso crea una nueva
   HttpSession session = request.getSession();
   Integer contadorAccesos;
   synchronized(session) {
      contadorAccesos = (Integer)session.getAttribute("contadorAccesos");
      if (contadorAccesos == null) {
         contadorAccesos = 0;
      } else {
         contadorAccesos = new Integer(contadorAccesos + 1);
      }
      session.setAttribute("contadorAccesos", contadorAccesos);
   }

   // Escribe el mensaje de respuesta en una página html
   try {
     out.println("<!DOCTYPE html>");
     out.println("<html>");
     out.println("<meta http-equiv='Content-Type' content='text/html; charset=UTF-8'>");
     out.println("<title>Servlet de prueba de sesión</title></head><body>");
     out.println("<h2>Accesos: " +  contadorAccesos + " en esta sesión.</h2>");
     out.println("<p>(Identificador de sesión: " + session.getId() + ")</p>");
     out.println("<p>(Fecha de creación de la sesión: " +
            new Date(session.getCreationTime()) + ")</p>");
     out.println("<p>(Fecha de último acceso a la sesión " +
            new Date(session.getLastAccessedTime()) + ")</p>");
     out.println("<p>(Máximo tiempo inactivo de la sesión: " +
            session.getMaxInactiveInterval() + " seconds)</p>");
     out.println("<p><a href='" + request.getRequestURI() +  "'>Refrescar</a>");
     out.println("<p><a href='" + response.encodeURL(request.getRequestURI()) + "'>
      Refrescar con reescritura de URLs</a>");
     out.println("</body></html>");
   } finally {
      out.close();  // Cerrar el flujo de salida
   }
  }
}
```

4. Añade ahora la siguiente información al descriptor de despliegue:

```
<servlet>
   <servlet-name>ServletPruebaSesion</servlet-name>
   <servlet-class>SessionServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>ServletPruebaSesion</servlet-name>
   <url-pattern>/pruebasesion</url-pattern>
</servlet-mapping>
```

5. Exporta el proyecto **Sesiones** a un fichero **Sesiones.war**, marcando la opción sobreescribir si el fichero ya existe.
6. Inicia ahora el servidor *Tomcat*, ejecutando el *script* de arranque *startup*, situado en **CATALINA_HOME\bin**
7. Para desplegar el proyecto en *Tomcat*, copia el fichero **Sesiones.war** en la carpeta **CATALINA_HOME\webapps**.
8. Accede a la aplicación con la siguiente *URL* `http://localhost:8080/Sesiones/pruebasesion`, Figura 1.

Figura 1: Sesiones

9. Realiza las siguientes pruebas:
   - Actualiza la página pulsado **F5** o el enlace **Refrescar**.
   - Abre otra pestaña en el navegador y comprueba cómo se conserva la sesión.
   - Cierra el navegador y vuelve a abrirlo y comprueba cómo se ha perdido la sesión.
10. A continuación accede utilizando *Firefox* a nuestra aplicación `http://localhost:8080/Sesiones/pruebasesion`.
11. Sobre la página web, pulsando con el botón derecho del ratón en **Ver información de la página**, en la pestaña **Seguridad**, **Ver cookies**, puedes observar la *cookie* que ha enviado nuestra aplicación para gestionar la sesión, Figura 2.

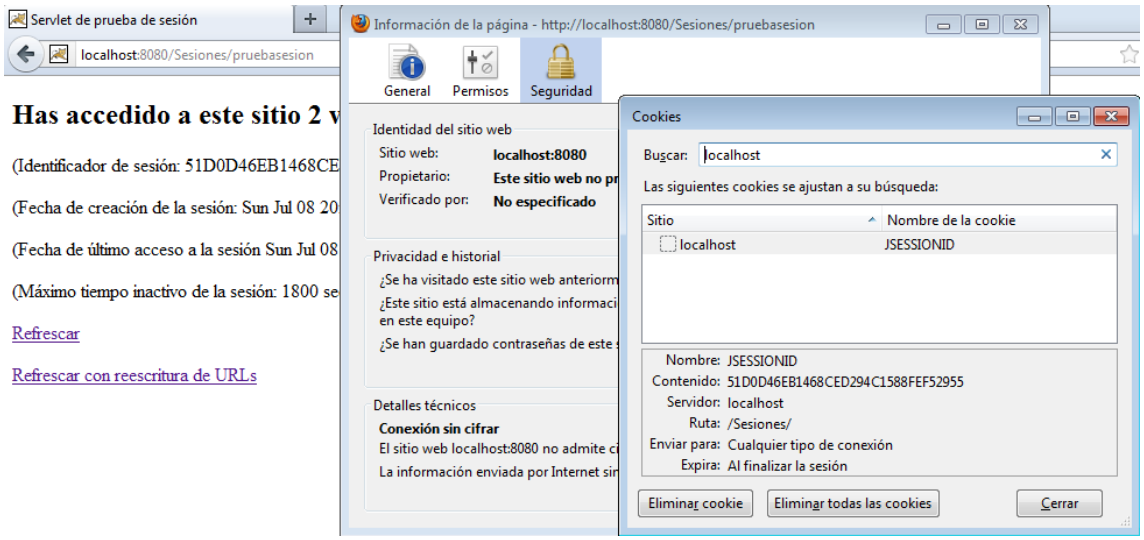

Figura 2: *Cookie* de sesión

12. Si observas la *cookie* puedes ver cómo su nombre es *JSESSIONID* y su contenido es el identificador de la sesión que también puedes ver en la página web.

13. Por defecto el método que utiliza *API Servlet* para gestionar la sesión son las *cookies*, pero si estas estuvieran deshabilitadas en el navegador del cliente, automaticamente utilizaría reescritura de *URLs*.

14. Cuando se utiliza este método de gestión de sesiones, todas las *URLs* reescritas emitidas por el servidor deben pasar a través del método *response.encodeURL( )* para añadir la información de la sesión.

15. Para comprobar cómo funciona la gestión de sesión utilizando reescritura de *URLs* deshabilita las *Cookies* en *Firefox*, Figura 3.

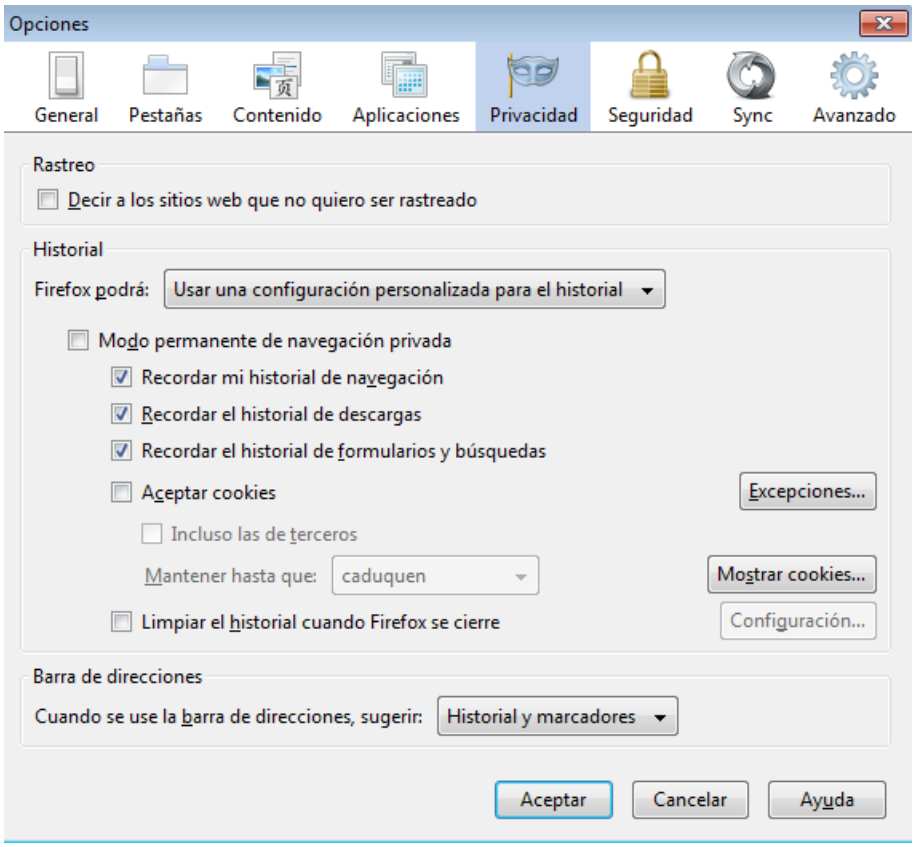

Figura 3: Deshabilita *Cookies*

16. Ahora accede a la aplicación `http://localhost:8080/Sesiones/pruebasesion` y comprueba cómo no funciona el seguimiento de la sesión actualizando la página o pulsando en el enlace **Refrescar**. Solo utilizando el enlace **Refrescar con reescritura de URLs** funcionará la gestión de la sesión sin utilizar *Cookies*, Figura 4.

Figura 4: Sesión con reescritura de URLs

17. Observa cómo el identificador de la sesión escrito en la URL es el mismo que el mostrado en la página.

18. Recuerda que si utilizamos el método *response.encodeURL( )* y las *cookies* no están deshabilitadas entonces este método no añadirá el identificador de la sesión a la *URL*.

19. Como prueba final, vamos a acceder al descriptor de despliegue, **web.xml**, para modificar el tiempo máximo, en minutos, que dura inactiva una sesión.

```
...
 <session-config>
  <session-timeout>
        1
  </session-timeout>
 </session-config>
</web-app>
```

20. Una vez hecha esta modificación vuelve a desplegar la aplicación y comprueba cómo la sesión caduca tras un minuto de inactividad.

◇