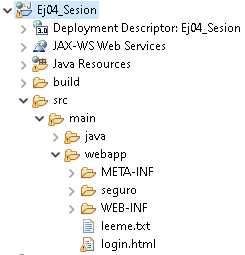
**Proyecto 3 Perdida de Autenticación**

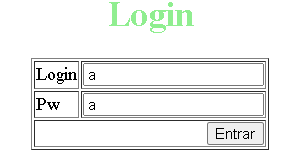
Uno de los problemas que podemos encontrarnos es el del robo de sesión y pérdida de autenticación, esto es que nos roben la cookie de sesión que nosotros tengamos cuando entramos dentro en un servidor y el servidor nos devuelve una petición HTTP.

Para reproducir este error vamos a utilizar el proyecto de sesión y una base de datos H2 que debe estar en la ruta de nuestro equipo. En este caso el sistema operativo es Windows y tenemos ubicada la base de datos H2 dentro de la unidad C.

Abrimos el proyecto de sesión y lo ejecutamos. Como es un proyecto web, necesitaremos de un servidor de aplicaciones o un contenedor servlet, en este caso Tomcat.



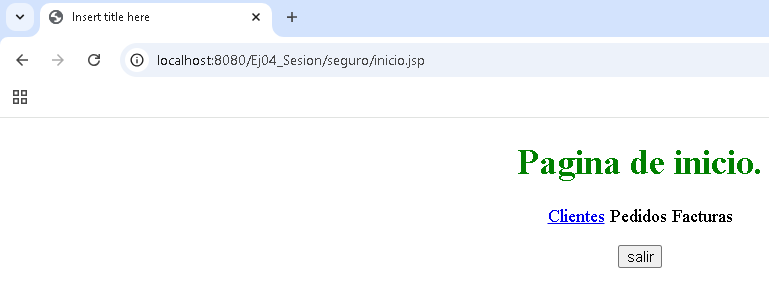
Tenemos una pantalla de login en la base de datos H2 con 2 usuarios, uno con login=a password=a y otro con login=b password=b.



Aquí el problema reside en que ambos usuarios utilicen el mismo equipo y por tanto también utilicen el mismo navegador para entrar dentro de la aplicación.

Esto puede ser, como antiguamente se hacía en algunos equipos que estuvieran dentro del aula para poder, por ejemplo, realizar un examen online en este caso, o para poder, por ejemplo, matricularse en ciertos cursos.

Imaginemos que un usuario acceda con a como login y a como password en la aplicación. Esto va contra la base de datos. Por tanto, si hemos puesto correctamente este login y password, deberíamos entrar.

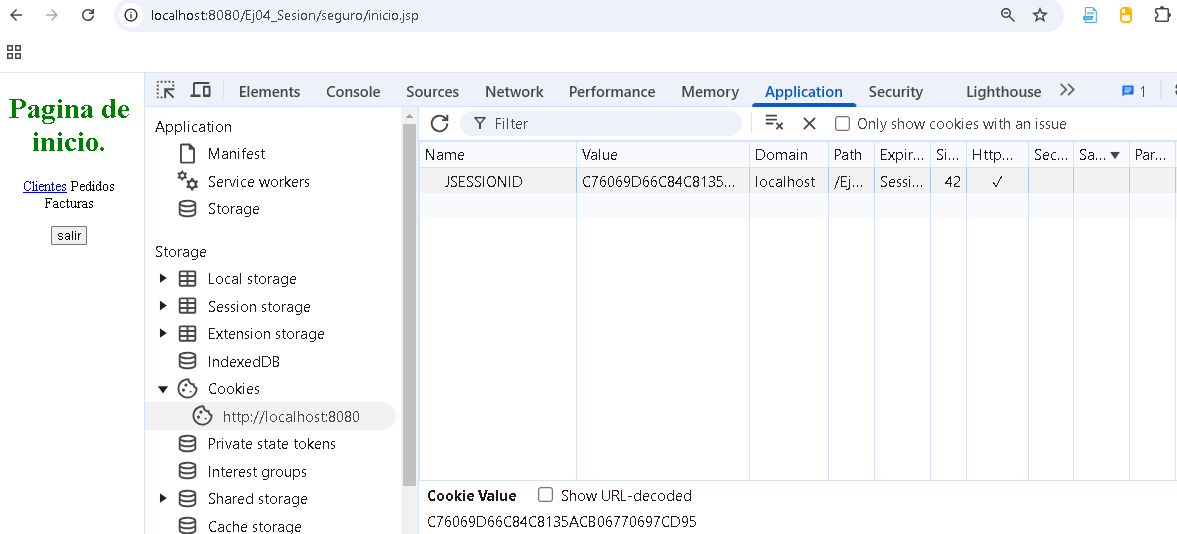


Una vez que hemos ingresado, el usuario que se ha conectado tiene nombre, en este caso Bates Spencer.

Si ingresamos al Developer tools con F12, vayamos a la pestaña de Storage

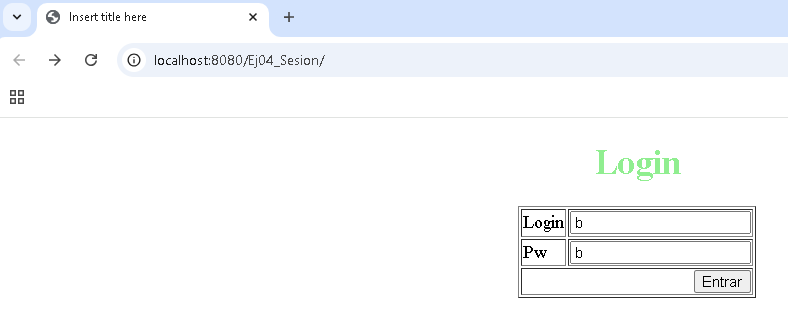
Barra superior -> Application -> menú izquierdo -> Storage

Encontraremos en este caso una cookie que está metida dentro del formato de cookies. Es una cookie de sesión y expira con la sesión de nuestro navegador.



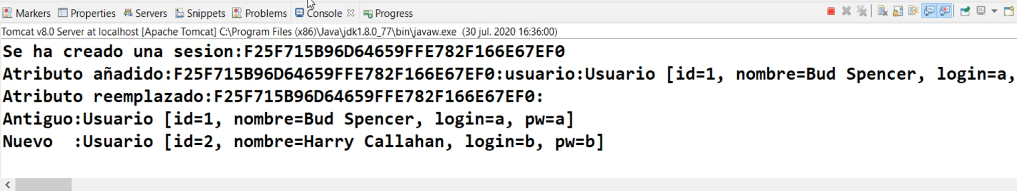
Inicialmente, si yo no le hubiéramos puesto nombre, esta cookie tendría de nombre JSesiónID, porque en este caso sería una sesión de Java, porque un servidor Java es el que nos la habría dado. Cambiarle el nombre no está de más para darle menos idea a los hackers de qué tipo de servidor y en qué lenguaje lo tenemos.

Si ahora volvemos hacia atrás con el navegador, siendo este usuario Spencer y llegamos hasta la pantalla inicial de login, podría ser que un segundo usuario accediera. Imaginemos que el segundo usuario, en este caso nuestro usuario B accede. El usuario que entra es el señor Harry Callahan. Sin embargo, la sesión (F12), tendría exactamente el mismo valor. Podrían entonces hacernos una fijación de sesión, un robo de nuestra sesión y obtener todos los datos que fuéramos colocando.



Por tanto, esto sería el problema más grande que nos podemos encontrar.

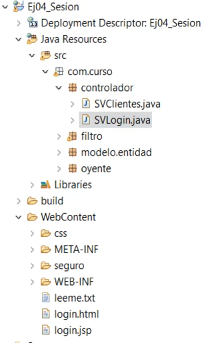
Vemos como se ha creado una sesión, cuando hemos agregado esa sesión a un usuario y cómo se ha reemplazado el antiguo usuario por el nuevo, sin embargo, el valor de la sesión es el mismo.



Vamos a ver posibles soluciones que podemos hacer con nuestra sesión.

Una de las primeras soluciones que tenemos es la invalidación de la sesión en SVLogin.

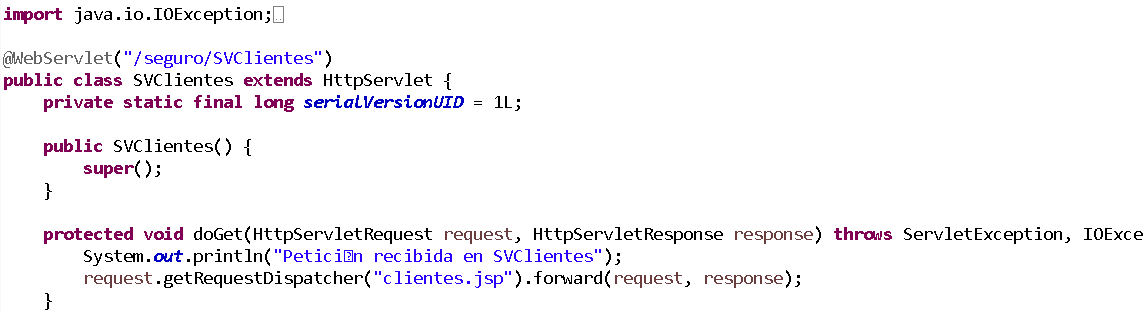
Cuando hagamos Logout, por ejemplo, podemos hacer un botón para desconectarnos y entonces invalidar esa sesión con la cual estamos entrando en la aplicación.



Aquí debemos descomentar el código a continuación.



También en SVClientes, que es el otro controlador, vemos cómo existe la petición hasta clientes.jsp. Aquí inicialmente en la ruta del web servlet pusimos un prefijo de seguro. Esto es así porque podemos validar estos distintos servlet mediante filtros o listener (oyentes). Ambos son componentes que podemos implementar para ofrecer una mayor seguridad en nuestras aplicaciones web.



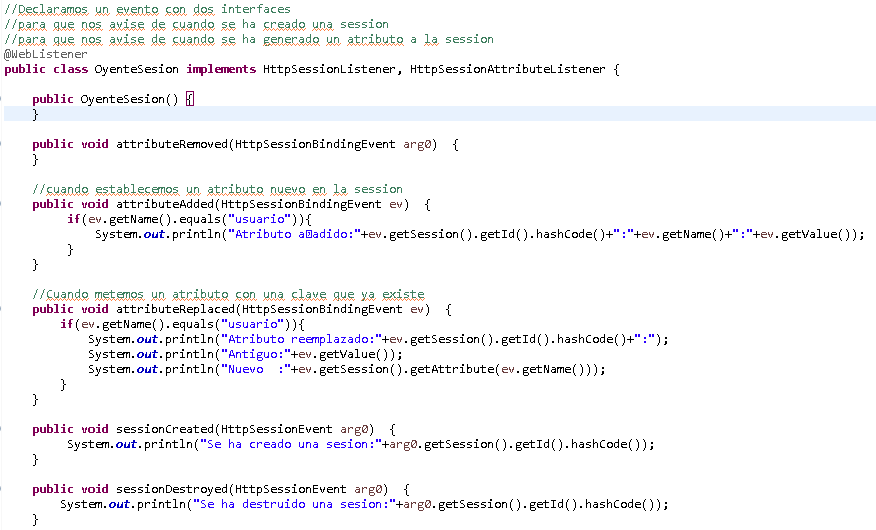
En la capa de filtro tenemos los distintos filtros, uno de ellos es el filtro de seguridad (FiltroSeguridad.java). Si descomentamos la línea 15, todo servlet que adopte la ruta de /seguro/\* implementara el patrón Intercept Filter.



Sólo tendría que descomentar la anotación @WebFilter("/seguro/\*") y de ese modo podemos comprobar, por ejemplo, si hemos perdido nuestro usuario o la sesión, nos redirigimos al Login.

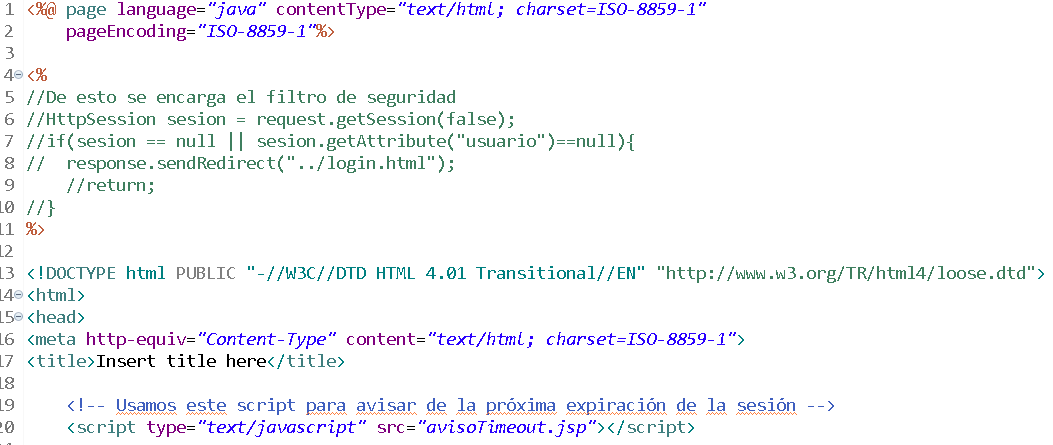
Como SVClientes está ligado al modelo de negocio (interactúa con la vista clientes.jsp), es conveniente que este enlazada al filtro de seguridad, en cambio con SVLogin, no es necesario.

Por otro lado, en la capa oyente (listener), podemos hacer un oyente de sesión. Se decora con la anotacion @WebListener. Aquí implementamos las interfaces HttpSessionListener y HttpSessionAttributeListener para que nos avise cuando se ha creado una session y un atributo en la sesion (por ejemplo, usuario).



Todo esto nos permite monitorizar cuando se creado la sesión, cuando se ha agregado un atributo en la sesión, cuando se reemplaza un atributo de la sesión (por ejemplo, el usuario) y cuando se destruye la sesión.

Finalmente, en la vista clientes.jsp un scriptlet que permite invalidar una sesión, en el caso en el cual no esté definida la sesión o el usuario. De este modo, se redirecciona al login.



En nuestro descriptor de despliegue web.xml podemos encontrar la configuración de nuestra sesión. Se puede observar el nombre otorgado a la cookie (para evitar el que viene por defecto el cual es JSESSIONID), el tiempo de expiración de la sesión (expresada en minutos) y el tracking- mode para que se comporte en este caso como una cookie.

