Sesión 10

El objetivo de esta sesión es practicar AJAX, para ello se realizará un *chat* simple: una mini–aplicación web que estará a la escucha en la máquina `www-sew.dyndns.org` y que permitirá intercambiar mensajes entre los clientes que estén conectados. Para que el cliente pueda acceder al servicio, éste deberá conectarse al mismo mediante un navegador (preferiblemente Firefox), solicitando el recurso de URL

http://www-sew.dyndns.org/~*usuario*/chat/chat.php

lo que permitirá al usuario enviar mensajes a otros usuarios mediante un formulario (una petición POST asíncrona por mensaje enviado) y recibir mensajes de otros usuarios en formato XML (mediante un *pool* de peticiones GET asíncronas del recurso `message.php` a intervalos regulares de unos pocos segundos).

Copia de archivos al servidor

En primer lugar, copia todos los archivos y carpetas proporcionados para la práctica al servidor (el contenido del archivo comprimido `practica-10.zip` que puedes descargar del campus virtual). La copia se debe de realizar en el subdirectorio `public_html/chat`, para ello abre el Eclipse, activa la perspectiva `Remote System Explorer` y desde ésta abre una conexión SSH con el servidor. Para abrir la conexión accede al menú contextual (botón derecho del ratón) en la ventana más a la izquierda (pestaña `Remote Systems`) donde también verás, aunque contraída, la máquina `Local`. Cubre los datos de la conexión, básicamente el nombre del servidor (`www-sew.dyndns.org`) y proporciona cuando se requiera el usuario y contraseña de acceso a dicha máquina.

Una vez realizada la conexión, la ventana `Remote Systems` es, a todos los efectos, una aplicación SFTP donde es posible realizar transferencias de archivos entre la máquina `Local` y el servidor, simplemente arrastrando o copiando éstos de una a otra máquina. Ahora sigue los pasos siguientes:

1. Despliega la conexión al servidor y en `My Home` verás la carpeta `public_html`, selecciona éste y desde el menú contextual crea una nueva (opción `New`) carpeta (`Folder`) de nombre `chat`.
2. Despliega la máquina Local hasta alcanzar el archivo comprimido que te has descargado, `practica-10.zip`, y copia todo su contenido a la carpeta `chat` del servidor.
3. Utilizando la barra de *scroll* desplaza la ventana `Remote Systems` hasta que veas, en el árbol de conexión al servidor, el elemento `Ssh shells` (está al final del árbol). Selecciona éste y desde el menú contextual lanza el intérprete de comandos de la máquina remota (opción `Launch Shell`).
4. Ahora proporciona los siguientes comandos en el intérprete de comandos remoto (pestaña `Remote Shell` de la ventana inferior):

   ```
   1) cd public_html/chat
   2) chmod 777 database
   3) cd database
   4) chmod 777 chat.sqlite
   ```

   de forma que cualquiera pueda acceder a la base de datos *sqlite* que utiliza el `chat`.

Otra facilidad que ofrece la conexión remota es el poder editar los archivos remotos directamente en el Eclipse, algo que tendrás que hacer con el archivo `Chat.js` de la práctica.

Explicación de la aplicación `chat`

Se proporciona un prototipo de la práctica con toda la parte del servidor ya realizada, la cual está escrita en PHP. De la parte cliente se proporciona una buena parte del código JavaScript en el archivo `Chat.js`, en el cual se define el objeto `chat` con todos los campos requeridos y métodos necesarios, incluida su interfaz. Únicamente faltan por completar los métodos del `chat` que realizan peticiones asíncronas, tanto para enviar un mensaje al servidor como para recibir mensajes de otros usuarios.

En este caso no se ha optado por realizar una clase porque el `chat` va a ser un objeto único (*singleton*) y, por tanto, es mejor crearlo como un literal. El objeto `chat` y otras funciones auxiliares se declaran en un espacio de nombres independiente (función que se auto-invoca) y únicamente se exportan los miembros que se requieren externamente:

- El método `init` para inicializar el chat, éste requiere como argumento el *nickname* del usuario en el chat. Externamente el método se nombra como `start` y se lanza en el evento `onload` del cuerpo del documento XHTML5 proporcionado por el servidor cuando el usuario se conectar al servicio.

  Este método crea la interfaz del chat y muestra ésta en la sección del documento XHTML5 disponible al efecto. Además, realiza la petición inicial de los últimos mensajes disponibles en el chat lanzando el método `requestMessages`.

- El método `sendMessage` permite enviar un mensaje al servidor mediante una petición POST asíncrona del recurso de URL `insert.php`. Este *script* del servidor se encarga fundamentalmente de almacenar información del mensaje (texto, usuario que lo envía, hora, etc.) en la base de datos de la aplicación y sólo requiere recibir el mensaje escrito por el usuario en la variable `message`.

  Este método se invoca al enviar los datos de formulario que contiene el `textarea` de edición del mensaje (inspecciona éste para verlo).

- El método `requestMessages` realiza una petición GET asíncrona del recurso de URL `message.php`. Este *script* del servidor proporciona al cliente un documento XML con los nuevos (o últimos) mensajes recibidos en el servidor de otros usuarios, para ello se requiere conocer el identificador del último mensaje que se ha mostrado en el cliente (`chat.lastId`). El esquema XML del documento está disponible en el archivo `messages.xsd` y su aspecto es el siguiente.

```
<response xmlns="http://156.35.163.103/sew/messagesSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://156.35.163.103/sew/messagesSchema
 messages.xsd">
   <status>1</status>       <!-- ver archivo config.inc.php -->
   <time>1481057588</time>  <!-- hora del servidor -->
   <users/>                 <!-- usuarios conectados -->
   <messages>
      <message id="30" user="pedro" time="21:51:45">Prueba de Mensaje</message>
      <message id="31" user="pedro" time="21:51:55">Un mensaje más</message>
   </messages>
</response>
```

- El método `showMessages` muestra los mensajes recibidos del servidor en el chat invocando el método `chat.view.show` y lanza el método `requestMessages` en el tiempo establecido.

- El método `submitMessage` envía el mensaje escrito por el usuario al pulsar la tecla `<RETURN>`, si además se pulsa la tecla `<SHIFT>` entonces se añade un salto de línea al mensaje. Inspecciona el formulario que contiene el `textarea` y observa el valor del atributo `action`.