Aplicaciones Distribuidas (AD)

Práctica 1: Desarrollo de una aplicación web utilizando páginas html, jsp y servlets que conectan con la base de datos SQLite

En esta práctica se va a desarrollar una aplicación web utilizando *Netbeans* como entorno de desarrollo. El objetivo es realizar una aplicación simple basada en J2EE que conecte con la base de datos SQLite, que utilizaremos en las siguientes prácticas. Esta práctica tiene dos entregas.

En la primera entrega tenéis que indicar quiénes sois los componentes del grupo de prácticas y responder a una serie de preguntas planteadas en el enunciado. El formato del fichero a entregar puede ser texto, un documento Word o un fichero pdf. No es necesario entregar código.

En la segunda entrega deberéis entregar la aplicación web desarrollada en un archivo zip.

Consultad las fechas de cada entrega en el Racó.

Primera Entrega: Creación y modificación de una aplicación web de prueba

Creación

Para desarrollar la aplicación, se deberá crear un nuevo proyecto de tipo Web Application (File → New Project → Java Web → Web application). Se creará un nuevo proyecto que solamente contiene un fichero de tipo .html (index.html) en el directorio raíz. Eliminad ese fichero y cread uno de tipo .jsp (index.jsp) (New → JSP). El servidor de dicha aplicación tiene que ser Glassfish 4.1.1.

Nota: El servidor Glassfish ya está instalado en las máquinas de laboratorio, en el directorio **C:\Archivos de Programa\glassfish-4.1.1**. Tenéis que seleccionarlo cuando creéis el proyecto. Es posible que os cree un dominio local nuevo dentro del directorio **C:\Users\<nombre_usuario>\<directorio_dominio_local>**. Si lo configuráis así, lo podréis ver en todos los ordenadores de laboratorio. El directorio de los proyectos Netbeans tenéis que aseguraros de que está en la unidad **F:**.

En este ejercicio se pide:

a) Comprobar que la aplicación se ha creado correctamente, poniendo en marcha la aplicación web que se ha creado.
   Se tiene que abrir una ventana de navegador que se conecta a la siguiente dirección (*http://localhost:<puerto>/NombreAppWeb/*) y aparece un mensaje en la

página. El puerto dependerá de cómo haya configurado Glassfish el `personal_domain`.

b) Analiza el código fuente que ha recibido el navegador y compáralo con el código de la página `index.jsp`. ¿Qué diferencia o diferencias has encontrado?

Modificación del código

Para añadir código en una página jsp es necesario añadir los siguientes símbolos:
`<% /* Código */ %>`

Dentro del código, se puede añadir código Java con el objetivo de mostrar información en la página web.

a) Añade la siguiente línea de código en la página
`<% out.println("<h2>Primera Práctica - Aplicaciones web – Aplicaciones Distribuidas</h2>"); %>` y ejecuta de nuevo la aplicación. Analiza de nuevo la página html que recibe el navegador, ¿qué se ha añadido a la página? ¿Para qué sirve la etiqueta `<h2>`?
b) Añade otras líneas de código java en la página y observa los cambios que se vayan produciendo en el código generado cuando visualices la página en el navegador.
c) Para finalizar este apartado, busca la opción de menú de *Netbeans* que te permite ver el servlet que se ha generado y comprueba que el código Java del servlet genera la página html que se envía al navegador. ¿Qué instrucciones en lenguaje java muestran el código html no generado con `out.println` en la página jsp?

Segunda Entrega: Conexión de la aplicación web con base de datos SQLite

Para conectar la aplicación web con una base de datos SQLite, necesitaremos un fichero jar que implemente dicha funcionalidad. En el Racó podéis encontrar el fichero **sqlite-jdbc-3.7.2.jar**, que es una librería que permite gestionar una base de datos relacional en un fichero local con Java. El fichero que contiene la base de datos se almacenará en un directorio local de vuestra máquina. También podéis encontrar el fichero ejecutable (para Windows) **sqlitebrowser_200_b1_win.zip** que es un visor de bases de datos SQLite con que el que podréis comprobar el contenido de vuestro fichero.

Para poder utilizar SQLite desde *Netbeans* es necesario añadir la librería al proyecto en el que estamos desarrollando la aplicación web. Para ello, debemos ir al apartado *Libraries* del proyecto e indicar que queremos añadir una biblioteca. Hay que crear una nueva y darle el nombre “sqlite”. Una vez hecho esto, buscamos el fichero .jar que hemos descargado del Racó y aceptamos para que se incorpore al proyecto. En el apartado *Libraries* debe aparecer el nuevo fichero.

Descarga del Racó el fichero **servletTestAD.java**, que contiene un servlet de ejemplo de conexión y uso de la base de datos. Este fichero no se puede incorporar directamente a vuestra aplicación web, sólo se proporciona como ejemplo de las instrucciones Java necesarias para conectarse con la base de datos, crear tablas,

insertar y consultar datos, etc. Tenéis que crear un nuevo servlet dentro de vuestra aplicación web y copiar las instrucciones que necesitéis del servlet ejemplo.

Una vez instalada la librería y creado el servlet, realiza las siguientes operaciones:

a) En la operación `processRequest` del servlet que habéis creado, implementad la creación de una base de datos, creación de tablas e inserción de datos (podéis utilizar la misma base de datos, tablas y datos que en **servletTestAD.java**).
b) Crea un fichero html. Añade un formulario html que llame al servlet que acabáis de crear, para que cree la base de datos y las tablas. En el anexo 1 hay una breve explicación de cómo se crea un formulario y los parámetros que necesita. En el atributo `action` del formulario tenéis que poner la URL del servlet que será algo como *http://localhost:<puerto>/NombreAppWeb/<nombreServlet>*, donde `nombreServlet` es el valor que se haya definido en la Anotación Java `@WebServlet(name = "nombreServlet", urlPatterns = {"/nombreServlet"})`, dentro del código del servlet.
c) Pon en marcha la aplicación y comprueba que se ha creado la base de datos. Realiza distintas pruebas, como añadir nuevos datos en la base de datos o consultar los datos existentes. En el ejemplo la base de datos se crea cada vez que se llama al servlet, modifica el código para que sólo se cree una vez y no se añadan datos repetidos. Puedes crear nuevos servlets, ficheros html o Java Server Pages (jsp) como necesites.

Anexo 1: Cómo enlazar servlets, jsp's y ficheros html

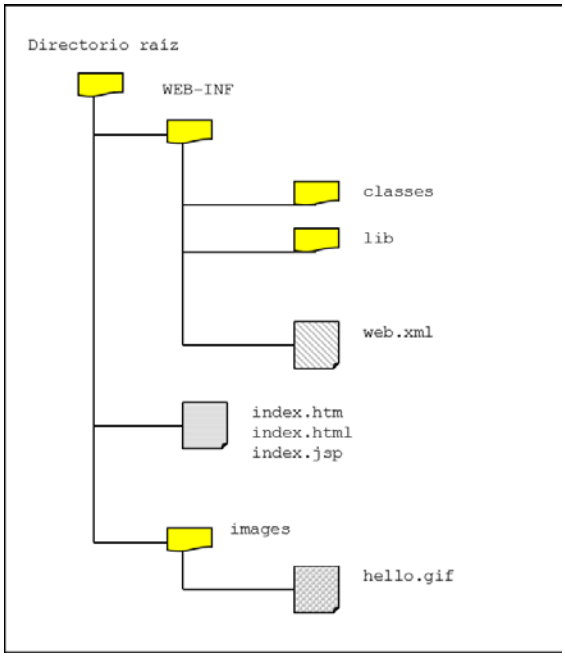
Para relacionar los ficheros html o jsp con los servlets, deberemos añadir el siguiente código html a nuestro formulario:

```
<form action = "login" method = "POST">
<!-- Campos del formulario -->
</form>
```

En este formulario estamos indicando que los datos que introduzca el usuario tenemos que enviarlos a un servlet identificado con la URL `login` y que utilizará el método POST de HTTP para el envío de los datos. En Windows la URL no es case sensitive, en Linux, sí.

Anexo 2: Estructura general de una aplicación web J2EE

La siguiente figura muestra la estructura básica de una aplicación web basada en J2EE. En esta figura se muestra el directorio raíz, de donde pueden colgar páginas html o jsp y también directorios donde tendremos ficheros "estáticos" (imágenes, documentos, etc.). Finalmente, tenemos el directorio WEB-INF, donde estará el fichero `web.xml`, que nos permite configurar distintos aspectos de nuestra aplicación web. (NOTA: En las últimas versiones de aplicaciones web puede que este fichero no aparezca en la estructura de directorios.) Dentro de este directorio, hay dos subdirectorios, lib y classes. En lib colocaremos las librerías necesarias para que nuestra aplicación web funcione bien (por ejemplo, la librería de SQLite). En classes estarán ubicados los ficheros .class de los servlets que hayamos desarrollado en nuestra aplicación. Netbeans genera automáticamente la estructura de ficheros necesaria.



Los distintos entornos de programación que podamos utilizar (Netbeans, Eclipse, etc.) gestionan de forma bastante automatizada este tipo de aplicaciones y puede que dentro de una aplicación web no estén todos los ficheros indicados en la imagen.

Ejemplo de fichero `web.xml` que cambia la página inicial de la aplicación para que sea `editor-canvas.html`.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>editor-canvas.html</welcome-file>
    </welcome-file-list>
</web-app>
```