Práctica 1. Servlets y JSP

Aplicación web para la gestión de concursos convocados en la UBU.

Objetivo

Desarrollar una aplicación web con nombre `concursos` (WebApp) que permita la creación de nuevas convocatorias de concursos por parte de los usuarios.

Funcionalidad

Casos de uso 1. Validación: el usuario se valida en el sistema. Si la validación es correcta se pasa a la ventana de gestión de concursos.

Casos de uso 2. El usuario visualizará el listado de concursos ya registrados en la herramienta. Podrá seleccionar editar cualquiera de los existentes o crear nuevos concursos.

Caso de uso 3: Crear/Editar un concurso: en esta pantalla el usuario podrá modificar la información requerida para el concurso o crear uno nuevo. Deberá proporcionar al menos la información siguiente:

- Título del concurso
- Propósito/Objetivo
- Bases del concurso
- Premios
- Plazo
- Jurado
- Ficheros con información adicional

Caso de uso 4: El usuario introduce el título del fichero y lo adjunta para que sea almacenado en el sistema asociado al concurso correspondiente.

Requisitos no funcionales

La aplicación se puede resolver utilizando los contenidos vistos en las sesiones de prácticas previas:

servlets, filter, JSP, JavaBeans, lenguaje de expresiones, bibliotecas de etiquetas, clases java, etc.

Existe libertad en el diseño de la aplicación, pero se valorará positivamente el seguir una arquitectura Modelo-Vista-Controlador.

La persistencia de los datos se gestionará mediante conexiones JDBC contra una base de datos MySql.

Como resultado final se obtendrá un fichero war que deberá obtenerse desde Maven con la opción:

```
mvn package
```

Prototipos

La aplicación web se compone básicamente de cuatro pantallas principales (correspondientes a los casos de uso). Estas ventanas se corresponderán con formularios HTML.

Modelo de datos

Se utilizará un modelo de dos entidades/clases. Se presenta a continuación el modelo relacional.

- `Usuarios(`**`id`**`, login, password, perfilAcceso)`
- `Concursos(`**`id`**`, titulo, objetivo, bases, premios, plazo, jurado,`*`idUsuario`*`)`
- `Ficheros(`**`id`**`, titulo, ruta,` *`idConcurso`*`)`

En la tabla `Concursos` se tiene como clave foránea *idUsuario* que referencia al campo **id** de la tabla de Usuarios. En la tabla `Ficheros` se tiene como clave foránea a *idConcurso* que referencia al campo **id** de la tabla Concursos.

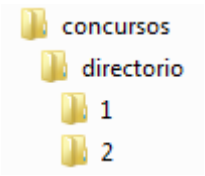
En UBUVirtual se dispone de ejemplos de código Java para el manejo de consultas e inserciones con JDBC.

Simplificaciones

No se solicitan validaciones de los datos del formulario, en el lado del cliente.

Los ficheros subidos al servidor se alojan a partir de un subdirectorio que cuelga del directorio de la aplicación web (`concursos`) de nombre *directorio.* Dentro de este subdirectorio, existirá un directorio por usuario, coincidiendo en nombre con el identificador del concurso.

Ej: suponiendo que los concursos ya creados son *1* y *2*



Alternativa a la creación dinámica de subdirectorios: Estas carpetas estarán **ya creadas previamente** y la aplicación **no se encarga de gestionarlas**, sólo de alojar ficheros en el correspondiente directorio y de insertar la correspondiente entrada en la base de datos.

Condiciones de entrega

Se entregará **un fichero `concursos.zip`** a través de la tarea habilitada en UBUVirtual. Dicho proyecto contendrá un proyecto Eclipse de naturaleza Maven y los scripts de creación de las tablas de base de datos en MySql.

El pool de conexiones JDBC se deberá llamar: *ConcursosPool* y el recurso JDBC tendrá como nombre JNDI: *jdbc/Concursos*

El proyecto tendrá en su directorio raíz un fichero `pom.xml` que permite el desarrollo completo del fichero `war` a desplegar en un servidor Glassfish. Antes de la entrega se recomienda `mvn clean`.

Adicionalmente se entregará un **video** en el que alumno explica cómo ha desarrollado la práctica, mostrando cómo está organizado el proyecto, arrancándolo y probando que funciona.

Revisar la documentación de la tarea para detalles adicionales y cambios en la fecha y hora de envío límite (1**0/12/2021 a las 21:30**). La práctica tiene una valoración máxima de 1 punto.

Anexo 1. Creación del poyecto

Crear el proyecto con *archetype*: `maven-archetype-webapp`

Cambiar las propiedades del proyecto *Project Facets* a mínimo:
- Dynamic Web Module 3.0
- Java 1.8

En las propiedades del proyecto seleccionar *Maven / Update Project*.
Revisar las dependencias de Maven. Comprobar que está añadido:

```
<dependency>
	<groupId>javax</groupId>
	<artifactId>javaee-web-api</artifactId>
	<version>6.0</version>
</dependency>
```

Si no está añadida la dependencia añadirla manualmente, sobre el proyecto *Maven / Add Dependency*.

Por defecto no se crea la carpeta `src\main\java`, así que debemos crearla:

Refrescamos la vista del proyecto y la carpeta debería aparecer en Java Resources. En caso contrario seleccionar con el botón derecho, *Build Path / Us as Source Folder*.

Añadir la carpeta `src\main\webapp` de la misma forma si no está en Java Resources.

Con eso el proyecto debería estar preparado para trabajar con:

- La carpeta `src\main\java` para las clases java (servlets, javabeans, clases de utilidad, etc).
- La carpeta `src\main\webapp` para las desarrollar los .jsp, .html, ficheros de imágenes, CSS, etc.

En UBUVirtual se deja un fichero pom.xml de ejemplo, con la configuración que se debería utilizar en el desarrollo de la práctica, con los ajustes que sean necesarios. El usuario administrador de Glassfish debe tener login "`admin`" y password "`admin`".

Anexo 2. Creación de Pool de Conexiones y Recurso JDBC

Consultar la correspondiente documentación existente en UBUVirtual y crearlos con los nombres indicados en las condiciones de entrega.+

Anexo 3. Subida de ficheros con Servlet 3.0

Previamente a la especificación 3.0 de servlets, la subida de ficheros se apoyaba en bibliotecas de terceros (Apache Commons FileUpload). En esta nueva especificación la subida de ficheros está integrada en la API permitiendo gestionar los envíos de formularios con distintos tipos de datos.

Para permitir el procesamiento de formularios que envían datos en formato texto y datos binarios (`multipart`) se deben realizar los siguientes ajustes:

- modificar en el fichero web.xml la entrada del servlet de la forma:
  - Ej:

```
<servlet>
   <servlet-name>nombredelservlet</servlet-name>
   <servlet-class>paquete1.paquete2.NombreClase</servlet-class>
   <multipart-config>
         <max-file-size>[nºbytes]</max-file-size>
         <max-request-size>[nºbytes]</max-request-size>
         <file-size-threshold>[nºbytes]</file-size-threshold>
   </multipart-config>
</servlet>
```

- o bien modificar el código fuente del *servlet* añadiendo como anotación de clase (y añadir opcionalmente atributos de tamaño, etc.):
  - `@MultipartConfig`

Como resultado, se pueden obtener las distintas partes que forman el envío (los elementos del formulario). Se dejan códigos parciales de ejemplo en UBUVirtual del procesado de formularios *multipart* así como el código necesario para procesar el título y el fichero en el ejercicio planteado.