Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Ingeniería Informática
Tercer curso. Primer cuatrimestre

Programación Web

Práctica 2: Introducción a JSP, JDBC, DAO, DTO

Juan Pedro Muñoz Jimenez
p52mujij@uco.es

Ángel Sevilla Molina
i42semoa@uco.es

Curso académico 2020-2021
Córdoba, 22 de noviembre de 2020

1. Ejercicio 1

1.1. Capa de negocio

La anterior clase `Contacto` ha sido convertida a la clase `User.java`, localizada en el paquete `es.uco.pw.business.user`, manteniendo las mismas operaciones de lectura y escritura, ya que cumplía con el patrón DTO.

La anterior clase `IAnuncio` que definía una interfaz con los métodos de los anuncios ha sido sustituida por la clase abstracta `Bulletin`, localizada en el paquete `es.uco.pw.business.user`, y conformada únicamente por métodos `get` y `set` de toda la información común a los anuncios, es decir, la propia de un anuncio general. Esta clase abstracta es extendida por las clases `GenBulletin`, `ThemBulletin`, `IndBulletin` y `FlashBulletin` para los anuncios generales, temáticos, individualizados y flash, respectivamente. Para su conversión a DTOs, las operaciones de los cambios de fase previas como `archivarAnuncio` o `publicarAnuncio` han sido sustituidas por `setFase`, dejando el control de esta al gestor.

1.2. Capa de datos

La clase `DAO.java` tiene definidos los métodos `getConnection()`, que permite realizar una conexión con la base de datos, y `getStmt(String optStr)`), que recibe una cadena con la operación a realizar y devuelve la declaración en lenguaje SQL con la operación a realizar sobre la base de datos, almacenada en un fichero de propiedades. Así, en cada método de los distintos *DAOs* implementados se invocará a ambos métodos para obtener el conector y saber qué operación debe realizar.

Para el acceso a la información de los usuarios se ha utilizado la clase `DAOUser.java` localizado en el paquete `es.uco.pw.data.dao.user` que realiza operaciones de almacenamiento (`save`), actualización (`update`), borrado (`delete`) y acceso (`getAll` y los distintos `queryBy`), así como una operación de login. `DAOUser` accede a la tabla `User` de la base de datos, sin embargo los intereses del usuario se almacenan en la tabla `Interests_users`, dedicando una fila por interés. Para las operaciones sobre esta tabla se ha definido otro DAO, llamado `DAOInterestsUsers.java`, localizado en el mismo paquete. Los métodos de esta clase son de tipo `protected`, y solo son invocados por `DAOUser`.

Para el acceso a la información de los anuncios se ha utilizado la clase `DAOBulletin.java` localizado en el paquete `es.uco.pw.data.dao.bulletin` que realiza operaciones de almacenamiento (`save`), actualización (`update`), borrado (`delete`) y acceso (`getAll` y los distintos `queryBy`), así como una operación para archivar anuncios (`close`) que actualiza solamente la columna `Bulletins.phase`. Este DAO se encarga de las operaciones de cualquier tipo de anuncios, utilizando como intermedio el DTO abstracto `Bulletin`.

Sin embargo, al igual que ocurre con `DAOUser`, esta clase tiene que hacer frente a otras tablas para poder gestionar las distintas especializaciones de anuncio. Para ello se ha seguido la misma estrategia de DAOs protegidos invocados desde la propia clase.

- Para los anuncios generales no se hace uso de ningún otro DAO, ya que toda la información de estos se encuentra almacenada en la propia tabla `Bulletins`.
- Para los anuncios temáticos se hace uso de `DAOInterestsThemBulletins.java`, cuyo funcionamiento es semejante a `DAOInterestsUsers`.
- Para los anuncios individualizados se hace uso de `DAOReceiversIndBulletins.java`, que accede a la tabla `Receivers_ind_bulletins`, que contiene una fila por cada usuario receptor de un anuncio.
- Para los anuncios flash se hace uso de `DAOFlashBulletins.java`, que accede a la tabla `Flash_bulletins`, que contiene una fila por anuncio con la información de la fecha de finalización del anuncio.

1.3. Clases de gestión

Las clases de gestión, ahora `UserMgr` y `BulletinMgr`, han dejado de utilizar la lista y los ficheros de texto, sustituyéndose por invocaciones a los DAOs correspondientes. Los métodos de Se han simplificado además las operaciones de búsqueda, utilizando ahora operaciones de filtrado. Por otra parte, para simplificar las operaciones que traten sobre las temas de interés, se ha creado la clase `InterestsMgr`, que lee los intereses del fichero de propiedades.

1.4. Programa ejecutable

Los códigos fuente utilizados en el programa ejecutable se localizan en el paquete `es.uco.pw.programs`. Como cambio con respecto a la anterior práctica, los programas principales han sido modularizados, dividiendo en métodos las distintas operaciones a realizar durante la gestión de los usuarios y los anuncios.

El programa de gestión de usuarios, ahora llamado `UserMgrMenu.java`, es accesible sin necesidad de iniciar sesión en el sistema. El programa de gestión de anuncios, ahora llamado `BulletinMgrMenu.java`, si requiere iniciar sesión para hacer uso de su funcionalidad, para los que se proponen los correos electrónicos *i42semoa@uco.es* y *p52mujij@uco.es*, de contraseña *usuario*, que contienen información de ejemplo.

Ambos códigos ahora carecen de método `main`, que han sido sustituidos por el método `menu`, el cual se llamará desde el único programa principal. El ejecutable JAR del primer ejercicio de la práctica se corresponde con el programa `MgrMain.java` localizado en el paquete . Este programa unifica los dos programas de la práctica anterior, permitiendo operaciones de gestión sobre usuarios y anuncios.

2. Ejercicio 2

Para la adecuada utilización de la aplicación web, es necesario ubicar la carpeta raíz de la aplicación dentro del directorio /webapps de Apache Tomcat. Además dependiendo de la versión deberá configurarse adecuadamente el servidor. No se ha podido incluir .war en esta entrega por una serie de problemas técnicos de última hora. El servidor de ThinStation ha estado caído durante estos dos últimos días y no se ha conseguido realizar la compilación desde un servidor instalado en un Windows personal.

2.1. Patron MVC (Modelo-Vista-Controlador)

En este ejercicio se ha tenido que seguir el patrón MVC para la realización de una pagina de login y registro de usuarios, utilizando la base de datos que se ha creado para el ejercicio 1.

2.2. Controladores

Los controladores son los primeros en recibir las peticiones "http-request". Son los que se encargaran de evaluar la lógica de negocio de la pagina, marcando hacia donde dirigir el flujo. Además rescataran los datos necesarios de la base de datos haciendo uso de los DAO y los DTO. También realizaran peticiones de datos al usuario a través de *requests* a las vistas. Se han creado dos controladores necesarios para el tratado de los datos.`loginController.jsp` encargado de revisar los datos de la base de datos para confirmar el login. Y `registroController.jsp`, encargado de recibir los datos del formulario de registro y guardar el usuario en la base de datos.

2.3. Vistas

Las vistas contienen la parte visual y con la que interactúa el usuario. Estas se encargan de recopilar información del usuario en forma de formularios, los cuales después pasan la información a los controladores. Para este ejercicio son como mínimo necesarias dos: `loginView.jsp` y `registroView.jsp`. Cada una dedicada a recoger los formularios de login y registro respectivamente. Coincide con el numero de controladores.

2.4. Modelos

Los javabeans, en general, son los modelos que guardan las credenciales o datos necesarios y apropiados para la sesión de la aplicación. En este caso el javabean se encargará de recoger los datos de la sesión del usuario, de manera que ser podrá consultar en que estado se encuentra (logado o no logado) y cuales son sus datos. Para la gestión de login y registro solo será necesario `CustomerBean.java`, localizado en `es.uco.pw.display.javabean`, para manejar los datos de sesión.