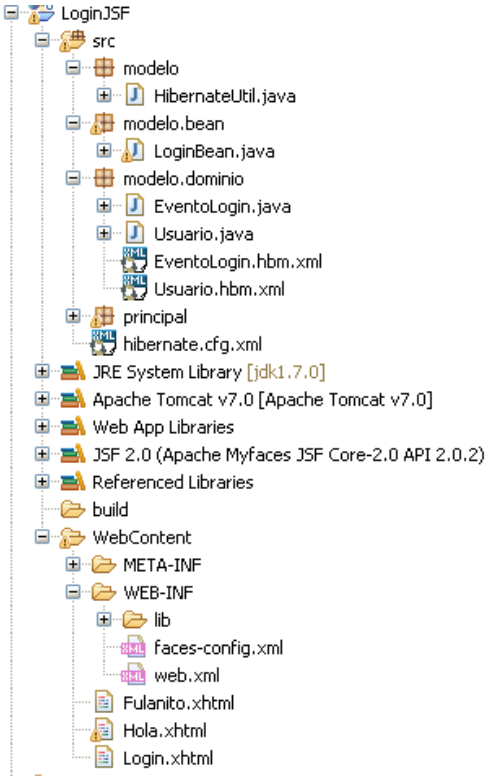
Ejercicios propuestos Framework Hibernate:

1.-) Como resultado del laboratorio Hibernate se tiene un proyecto con una estructura similar a la siguiente:



En dicho laboratorio se han añadido al proyecto LoginJSF las clases del dominio EventoLogin y Usuario y los ficheros de configuración necesarios para darles persistencia en una BD MySQL (los ficheros de mapping EventoLogin.hbm.xml y Usuario.hbm.xml), así como el fichero de configuración de Hibernate (hibernate.cfg.xml).

Como resultado del proyecto LoginJSF se tiene un JSF de inicio llamado Login.xhtml, que junto con su correspondiente bean (LoginBean.java) y ficheros de configuración (faces-config.xml y web.xml) permite ejecutar una aplicación web donde se permita o no la entrada del usuario.

Sin embargo, la implementación de la lógica del negocio reside en el método `comprobar()` del bean LoginBean y es ciertamente muy pobre: deja entrar a todos los usuarios excepto al usuario de nombre `"fulanito"`.

```
public String comprobar() {
      if (nombre.equals("fulanito")) return "error";
      else return "ok";}
```

En este ejercicio, se pide modificar este método `comprobar()` para que compruebe si existe un usuario en la BD que tiene el mismo nombre, password y tipo de usuario que los valores almacenados en los atributos `nombre`, `password` y `tipo` `del bean` LoginJSF.

AYUDA:

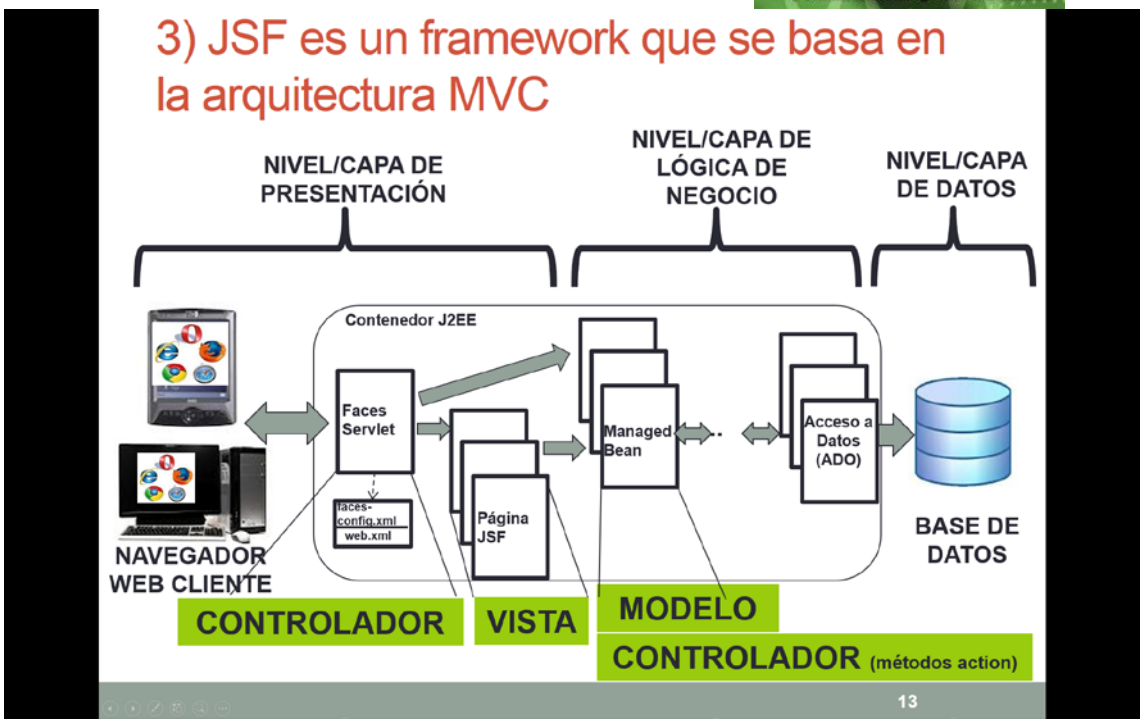
1) En la clase CrearEventos (en el paquete `principal`) existe el método `createAndStoreUsuario` que permite añadir usuarios a la BD, y de hecho ya debería existir el usuario `"Pepe"`, creado al ejecutarse la siguiente instrucción: `e.createAndStoreUsuario("Pepe", "125", "estudiante");`

2) En la implementación del método `createAndStoreEventoLogin(String usuario, boolean login, Date fecha)` de la clase CrearEventos se realiza una comprobación de si existe un usuario de nombre `usuario`. Para esto caso habría que comprobar si existe un usuario con un determinado nombre, password y tipo.

Además, se pide que en el caso en que el usuario con esos datos no exista, se le muestre una vista donde se le diga que el usuario no puede entrar en el sistema, dando explícitamente el nombre del usuario. Esto es, si en la vista JSF de Login se da un nombre de usuario con password o tipo de usuario incorrecto:

Debería mostrarse una vista que diga lo siguiente:

2.-) Una vez realizado el ejercicio anterior, se pide rehacerlo para que desde el bean no se invoque directamente a la BD, sino que se siga el diseño propuesto en la siguiente diapositiva correspondiente al tema de Framework JSF de este curso.



Como se puede ver, desde los modelos (bean gestionados o managed bean) no se interactúa directamente con la base de datos, sino que se invoca a algunos métodos de la lógica de negocio, que a su vez pueden invocar a otros métodos de acceso a datos (a clases ADO o DAO en inglés).

En este caso, se pide modificar el método `comprobar()` del bean LoginBean para que no ejecute instrucciones de Hibernate, sino que delegue en una clase DAO.

AYUDA: Esta podría ser la clase CrearEventos realizada en el laboratorio de Hibernate. En realidad, no tiene un nombre muy apropiado para ser una clase DAO, pero sí tiene unos métodos típicos de clases DAO como son `createAndStoreUsuario(String nombre, String password, String tipo)` o `createAndStoreEventoLogin(String usuario, boolean login, Date fecha)`. A esa clase se le podría añadir un método `hacerLogin(String nombre, String password, String tipo)` que devuelva si existe o no ese usuario con ese password y tipo en la BD.