Jorge Iglesias Roldán

Bryan Sanchez Espada

Universidad del País Vasco (UPV)
San Sebastian (Gipuzkoa)

Ingeniería del Software II

Entrega Final - Hibernate y JSF

20 de Diciembre de 2020

ÍNDICE

OBJETIVOS

En este proyecto se busca implementar una aplicación local dada en web. Por ello hemos utilizado el framework Hibernate para facilitar las tareas de creación y mantenimiento de la base de datos de la aplicacion web.

Además de comprender la utilidad y funcionalidad del framework Hibernate, otro de los objetivos de esta parte del proyecto ha sido observar la diferencia de utilizar ObjectDB e Hibernate.

En relación a este aspecto, hemos notado que la diferencia es palpable, ya que Hibernate se encarga de crear las tablas de la base de datos y rellenarlas. Nosotros únicamente tenemos que encargarnos de indicarle a Hibernate qué tablas tiene que crear y con qué información rellenarlas

RESOLUCION DEL LABORATORIO

1. Estructuración del proyecto

La estructura del proyecto se basa en distintos paquetes.

El paquete 'modelo' contiene la clase HibernateUtil, que se encarga de obtener la instancia de SessionFactory y guardarla en un atributo estático para poder usar luego HibernateUtil.getSessionFactory().

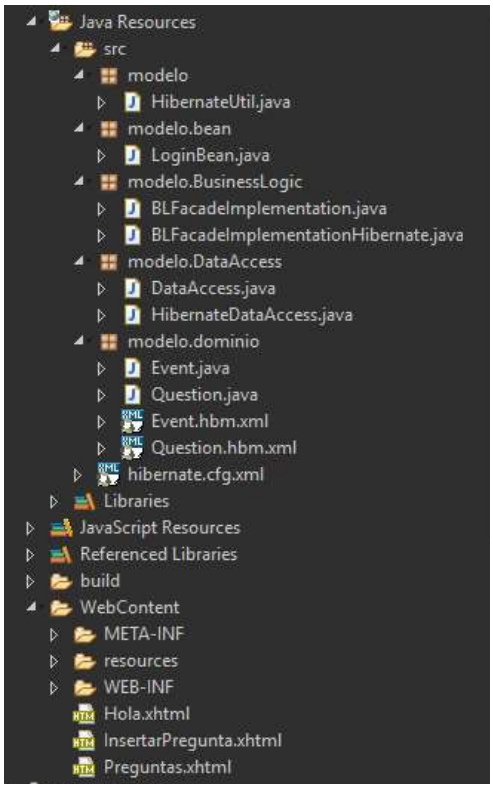
El paquete 'modelo.bean' contiene la clase LoginBean, que es la clase java que contiene las funciones que serán utilizadas en las páginas xhtml del proyecto Hibernate.

El paquete 'modelo.BusinessLogic' contiene la clase BLFacadeImplementationHibernate, que es el controlador que contiene las funciones que conectan la vista y las funciones acceden a la base de datos.

El paquete 'modelo.DataAccess' contiene la clase HibernateDataAccess, que es la clase que contiene las funciones que acceden a la base de datos.

El paquete 'modelo.dominio' contiene las clases Event y Question y los archivos de mapping, que indican a Hibernate las tablas de la base de datos donde se almacenarán los objetos de Event y Question, además de las características de los objetos de estas clases.

Fuera de los paquetes, dentro de 'src', se encuentra el archivo de configuración de hibernate hibernate.cfg.xml.



2. Adecuación de las clases del dominio

Adecuar las clases Event y Question, que se encuentran en el paquete 'modelo.dominio', eliminando las etiquetas JPA y cambiando las estructuras Vector por List.

3. Creación los archivos de mapping

Con clic derecho sobre las clases Event y Question, se crearán los archivos de mapeo, uno por cada clase.

En este caso tendremos dos, 'Event.hbm.xml' y 'Question.hbm.xml'.

En estos archivos xml, se indica a Hibernate las características de la tabla que tiene que crear, añadiendo en ellas los objetos de la clase correspondiente, ya sea objetos Event o Question.

4. Creación de la clase HibernateUtil

Creamos la clase la clase modelo.HibernateUtil para obtener la instancia de "Session Factory" la cual será guardada por un atributo estáticos para utilizarla posteriormente.

5. Creación del fichero de configuración de Hibernate

Creamos un archivo de configuración de hibernate para manejar todo lo referente a hibernate, es decir, obtener la instancia de SessionFactory y guardarla en un atributo estático.

6. Adaptar la clase del controlador a Hibernate

Utilizando la clase dada en la aplicación local, modificamos el tipo vector ya que Hibernate no trabaja con ellos cambiandolos por un tipo list (java.util.list) y borramos todas las anotaciones JPA. Además modificamos todos la constructora del dataAccess ya que esta no accede al dataAccess de hibernate sino al de object.db, es decir:

```
DataAccess dbh= new DataAccess();
```

Se cambia por:

```
HibernateDataAccess dbh= new HibernateDataAccess();
```

7. Adaptar la clase de acceso a datos a Hibernate

La adaptación de esta clase resulta más compleja ya que no solo hay que cambiar todos los Vector por List sino que encima hay que modificar todos los métodos que se utilizan de object.db por los de hibernate. Además hay que cambiar las 2 primeras líneas de cada método. Eliminar la constructora.

En esta clase, hay que cambiar los db.persist por session.save.

8. Adaptar el bean al nuevo controlador de Hibernate

En la clase LoginBean, hay que actualizar el uso del controlador BLFacade.

Cambiamos el atributo:

```
BLFacadeImplementation facade = new BLFacadeImplementation( );
```

Por:

```
BLFacadeImplementationHibernate facade = new BLFacadeImplementationHibernate();
```

Además, todas las apariciones de DataAccess en el código de la clase, se sustituirá por DataAccessHibernate.

Asímismo, las apariciones de BLFacadeImplementation sustituirlas por BLFacadeImplementationHibernate.

DECISIONES

1. Al traer eventos te traes también las preguntas (lazy)

En el fichero de mapeo ‘Event.hbm.xml’ en la sentencia:

```
<list name="questions" table="QUESTION" lazy="false">
```

Añadimos el atributo lazy=”false” para que al hacer una consulta HQL también podamos traer la tabla de preguntas y extraer las preguntas necesarias sobre ella.

PROBLEMAS

1. Problemas con el JRE y JDK

Al haber hecho el proyecto entre los dos integrantes del grupo, cada uno lo ha hecho en su propio ordenador. Hemos tenido problemas con las versiones de JRE y JDK, ya que teníamos versiones distintas.

Creyendo que con la que teníamos funciona todo correctamente, nos dimos cuenta de que no era así, ya que saltaban errores, varios relacionados con esto.

Finalmente, nos percatamos de la presencia de incompatibilidades entre la versión de JAVA y la versión específica del compilador, ya que eran versiones distintas.

2. Problemas al exportar (librerías)

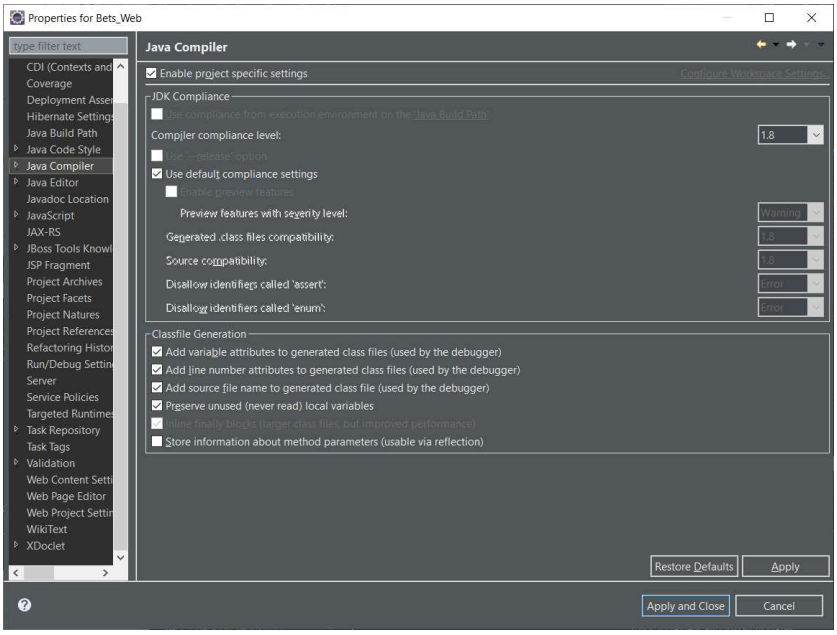
Al exportar las primeras versiones del proyecto, no exportamos correctamente, ya que no exportamos el proyecto con las librerías correspondientes.

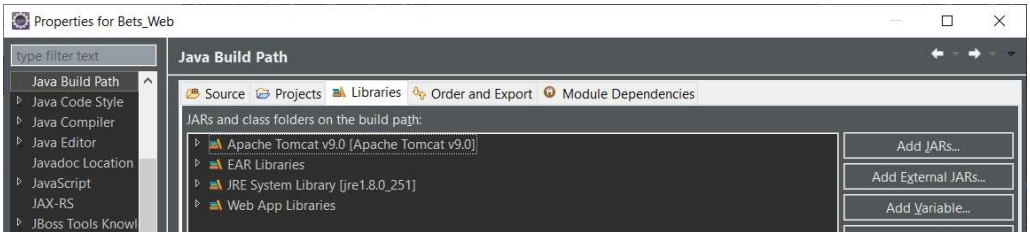
De esta forma, cuando importamos el .war del proyecto, no se importaban las librerías y el proyecto fallaba, daba errores.

SOLUCIÓN A LOS PROBLEMAS

1. Problemas con el JRE y JDK

La solución ha sido indicar la misma versión en ambos apartados, tanto de compilador como de JAVA.





2. Problemas al exportar (librerías)

La solución a este problema, fue simplemente tener en cuenta las librerias a la hora de exportar el proyecto, en formato .war, seleccionando la opción de incluir las librerías del proyecto en el proceso de exportación.

CONCLUSIÓN

Por un lado, el estudio de los JSF y la utilización de hibernate nos ha posibilitado indagar en el proceso tanto del funcionamiento de los mismos como en la logística de una aplicación java web, como a obtener práctica y conocimientos en el proceso de elaborar una metodología para afrontar los problemas a los que nos enfrentaremos en un futuro. Más concretamente, nos ha permitido trabajar con un enfoque en el que el usuario es el centro de atención y cuyas necesidades hemos de satisfacer de una forma meticulosa y rigurosa.

Por otro lado, el trabajo en sí ha posibilitado nuestra capacidad de organizarnos como equipo, y en nuestra situación en particular, a habituarnos a medidas de comunicación entre nosotros de las que antes no éramos conocedores. Sin embargo, esta misma situación ha afectado a nuestra obtención de resultados, ya que la gran carga de trabajo en esta época del curso, los problemas mencionados y la coordinación no han sido óptimas.

HORAS EMPLEADAS

Tareas		Tiempo empleado
Realización del laboratorio	Estructuración del proyecto	2h
	Adecuación de las clases del dominio	3h
	Crear los archivos de mapping	1h
	Crear la clase HibernateUtil	10 mins
	Crear fichero de configuración de Hibernate	2h
	Adaptar la clase del controlador a Hibernate	3h
	Adaptar la clase de acceso a datos a Hibernate	2h
	Adaptar el bean al nuevo controlador de Hibernate	1h
Redacción y documentación		3h
	TOTAL	**17h 30 mins**

Nota: En el tiempo de realización además del tiempo de implementación, se incluye el tiempo de búsqueda de información y el de resolución de problemas.

Además ha sido imposible la separación del tiempo entre los diferentes integrantes del grupo ya que el trabajo ha sido en paralelo.