

UD3: Herramientas de mapeamento objeto-relacional (ORM) Configuración de Hibernate

Para trabajar con Hibernate tendremos que indicar su configuración. Existen varias posibilidades: fichero .properties, ficheros XML, uso de anotaciones, etc.

Si hemos seguido los pasos indicados en la Actividad3.1, habremos generado:

Un fichero **hibernate.cfg.xml**

Una clase dentro del paquete **modelo** por cada tabla que hayamos incluido durante el proceso de "ingeniería inversa": De la BD generamos las clases.

Un fichero ***.hbm.xml** por cada clase creada indicando el **mapeo** entre clase y tabla.

hibernate.cfg.xml

```
--//Hibernate/Hibernate Configuration DTD 3.0//EN (doctype with catalog)
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
3 <hibernate-configuration>
4   <session-factory>
5     <property name="hibernate.connection.driver_class">com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
6     <property name="hibernate.connection.password">abc123</property>
7     <property name="hibernate.connection.url">jdbc:sqlserver://localhost:1433;database=empresa;encrypt=true;trustServerCertificate=true;</property>
8     <property name="hibernate.connection.username">user</property>
9     <property name="hibernate.default_schema">dbo</property>
10    <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>
11    <property name="hibernate.hbm2ddl.auto">none</property>
12    <property name="hibernate.bytecode.use_reflection_optimizer">>false</property>
13    <property name="hibernate.search.autoregister_listeners">>true</property>
14    <property name="hibernate.validator.apply_to_ddl">>false</property>
15  </session-factory>
16  <mapping resource="model/Emp.hbm.xml"/>
17  <mapping resource="model/Account.hbm.xml"/>
18  <mapping resource="model/Dept.hbm.xml"/>
19  <mapping resource="model/AccMovement.hbm.xml"/>
20 </hibernate-configuration>
```

hibernate.cfg.xml es un fichero XML con una DTD pública que permite establecer la configuración de acceso a la BD a través de las etiquetas **<property>**. **Con ellas podremos crear un objeto SessionFactory.**

Finalmente indica dónde se pueden encontrar los ficheros de mapeado entre clases y tablas con las etiquetas **<mapping>**

El fichero generado automáticamente por el plugin Hibernate Tools establece además otras

propiedades que se pueden consultar aquí:

https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate_User_Guide.html#configurations-hbmddl

https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate_User_Guide.html#configurations-bytecode-enhancement

<https://docs.jboss.org/hibernate/search/4.2/reference/en-US/html/search-configuration.html#search-configuration-event>

https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate_User_Guide.html#_bean_validation_options

Clases POJO

Las clases que representan los objetos almacenados en la base de datos son clases POJO (Plain Old Java Objects)

Deben cumplir los siguientes requisitos:

Constructor sin argumentos public o protected. Podría tener constructores adicionales.

Las clase no se recomienda que sea final

Getters y setters para los atributos privados. Normalmente deberá contar con un atributo con funcionalidad de identificador unívoco ID

Implementar la interfaz Serializable: Se recomienda que se añada un [serialVersionUID](#) con la sugerencia del IDE

```
10 public class Dept implements java.io.Serializable {  
11  
12     /**  
13      *  
14      */  
15     private static final long serialVersionUID = 4394934189814545618L;
```

Más información:

https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate_User_Guide.html#

Archivos de mapeado *.hbm.xml

Por ejemplo: **Dept.hbm.xml**:

Este fichero suele encontrarse en el mismo paquete que la clase a persistir.

<hibernate-mapping>: Elemento raíz contiene las clases de los objetos persistentes. En este caso solo Dept

<class>: Pueden aparecer varias clases de objetos persistentes. Los atributos generados son:

- **name**: el nombre Fully Qualified name (FQN) de la clase (incluye el nombre del paquete)
- **table**: nombre de la tabla. Este atributo es opcional si el nombre de la clase Java y el de la tabla coinciden.
- **catalog**: por defecto, el nombre de la BD
- **optimistic-lock**: qué estrategia se usará en caso de que varias transacciones intenten modificar un mismo dato. En la generación automática del fichero se ha usado *none*, por lo que utilizará el nivel de aislamiento por defecto de la BD

<id> Indica la propiedad de la clase que es la **clave primaria**.

El atributo **name** es el nombre de la **propiedad Java** que contiene la clave primaria.

El atributo **type** indica el tipo de la propiedad Java. Este atributo no es necesario puesto que Hibernate puede deducirlo a través de

Reflection. https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate_User_Guide.html#basic-provided

- El tag **<column>** contiene el nombre de la columna de la base de datos asociado a la propiedad. También podría ser un atributo de **<id>** opcional si el nombre de la propiedad Java y el nombre de la columna coinciden. Con **<generator>** se indica cómo se genera el identificador único: En nuestro caso, identity.

<property> se usa para declarar más propiedades Java para ser mapeadas en la base de datos. Si no declaramos las propiedades Java mediante este tag no se leerán o guardarán en la base de datos.

- El atributo **name** es el nombre de la propiedad Java que queremos mapear a la base de datos.
- El tag **<column>** contiene el nombre de la columna de la base de datos asociado a la propiedad. También establece la longitud en base a la longitud de la columna en BD. **column** también podría ser un atributo de **<property>** opcional si el nombre de la propiedad Java y el nombre de la columna coinciden

Relaciones en archivos de mapeado *.hbm.xml

En una relación hay un **lado propietario** ("owning side") y un **lado inverso** ("inverse side").

El lado propietario **es la entidad cuya tabla asociada tiene la clave foránea que mantiene la relación.**

Una relación unidireccional solo tiene un lado propietario: El que permite navegar hacia la otra entidad

En una relación bidireccional:

- Si es 1:N o N:1, **el lado propietario es el lado N**
- Si es 1:1, la entidad cuya tabla asociada tiene la clave foránea es el lado propietario
- Si es N:M, cualquiera puede ser el lado propietario

Departamento y Empleado tienen una relación 1:N. El lado 1 es Departamento y el lado N es Empleado. El siguiente extracto se encuentra en el fichero de mapeo de Dept.hbm.xml:

```
...
<set fetch="select" inverse="true" lazy="true" name="emps" table="EMP">
  <key>
    <column name="DEPTNO"/>
  </key>
  <one-to-many class="model.Emp"/>
</set>
...
```

El generador de las clases ha permitido navegación en ambos lados a través de la Interfaz Set, por lo que, tal y como se ha creado es una relación bidireccional. El lado propietario debe ser el lado N, es decir Empleado. Por lo tanto, Departamento es el lado inverso => inverse=true

¡ATENCIÓN! La nomenclatura en XML de Hibernate puede llevar a confusión: con el atributo **inverse=true** en una colección se indica que la entidad que representa esa colección **es el lado propietario**. En el caso anterior **inverse=true** se encuentra en la colección de `model.Emp` dentro del fichero de mapeo de `Dept.hbm.xml`, indicando que `model.Emp` será el lado propietario en la relación 1:N entre `Dept` y `Emp`. Por extensión, el otro extremo, `Dept` será el lado inverso.

Esto mismo se indica en esta URL : <https://mkyong.com/hibernate/inverse-true-example-and-explanation/> y en la documentación para relaciones bidireccionales M:N: <https://docs.jboss.org/hibernate/stable/core/old/reference/en/html/collections-advancedmappings.html>

Como se ha usado `Set`, se usa `<set>`. Podría haberse usado `List` y se usaría `<list>`. En una lista hay un orden y se permiten duplicados. En un `Set` no se permiten duplicados y no es un conjunto ordenado.

Como atributos de `<set>` se encuentran:

- **name**: La propiedad en `Dept` que permite navegar y obtener los empleados
- **table**: La tabla donde se almacenan los empleados
- **fetch**: Es la estrategia de recuperación de los datos asociados. Mediante un `select` independiente.
- **lazy**: Se indica la estrategia de recuperación de empleados:
 - **Eager**: Se recuperan los empleados cuando se recupere un departamento
 - **Lazy**: Se recuperan los empleados solo cuando se haga uso del método `getEmps()`

El subelemento `<key>``<column>` usan el atributo **name** con el nombre de la columna que mantiene la relación con la entidad del mapeo. En nuestro caso la clave foránea en `EMP` es **DEPTNO**

Del lado empleado en **Emp.hbm.xml**, se han mapeado varias relaciones. Vamos a fijarnos en la relación con `Dept`:

```
<many-to-one class="model.Dept" fetch="select" name="dept">
  <column name="DEPTNO"/>
</many-to-one>
```

Al tratarse del lado muchos, se usa la etiqueta `<many-to-one>` con la clase FQN `model.Dept` a través de la propiedad Java `dept` en `Emp`.

El atributo **name** de **<column>** identifica la clave foránea que mantiene la relación.

Obtener una Session

Para obtener una Session, tendremos que obtener una SessionFactory.

Esto lo podemos conseguir a través del fichero de configuración hibernate.cfg.xml.

Vamos a hacerlo con **el siguiente fichero**.

```
1 package util;
2
3 import org.hibernate.SessionFactory;
4 import org.hibernate.cfg.Configuration;
5
6 public class SessionFactoryUtil {
7     private static final SessionFactory sessionFactory;
8     /**
9      * El bloque static es un bloque de instrucción dentro de una clase Java que se
10     * ejecuta cuando una clase se carga por primera vez en la JVM. Inicializa la variable SessionFac
11     * de tipo static para que solo haya una instancia para todas las consultas en toda la aplicaciór
12     * Seguimos el patrón Singleton: https://es.wikipedia.org/wiki/Singleton
13     */
14     static {
15         try {
16             sessionFactory = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory()
17         } catch (Throwable ex) {
18             System.err.println("Initial SessionFactory creation failed." + ex);
19             throw new ExceptionInInitializerError(ex);
20         }
21     }
22
23     public static SessionFactory getSessionFactory() {
24         return sessionFactory;
25     }
26
27 }
28
```

Operaciones sobre la Session

Session proporciona métodos para leer, guardar o borrar entidades de la BD.

- **load()**: recupera una instancia persistente conociendo la clase y el identificador. Carga el objeto si existe a un estado persistente. Si no existe, lanza una excepción. Ej:

```
Dept dept = (Dept) session.load(Dept.class, (int) id);
```

- **get()** – hace lo mismo que load(), pero devuelve null en caso de no encontrar el objeto.
Dept dept = (Dept) session.get(Dept.class, (int) id);
- **Transaction tx = beginTransaction()** – para comenzar transacciones y **rollback()** para recuperar el estado previo a las modificaciones de una transacción

- **save()** – para hacer persistente objeto en BD
`session.save(departamento);`
- **saveOrUpdate()** para actualizar un objeto que ya era persistente.
- **delete()** – para eliminar un objeto
- **update()** – para modificar objeto
- Sobre un objeto de clase **Transaction**, se llamará a **commit()**: Intenta finalizar una transacción de forma exitosa
- **close()**: Cierra la sesión
- **createQuery()** – para consulta HQL (Hibernate Query Language) y ejecutar sobre BD

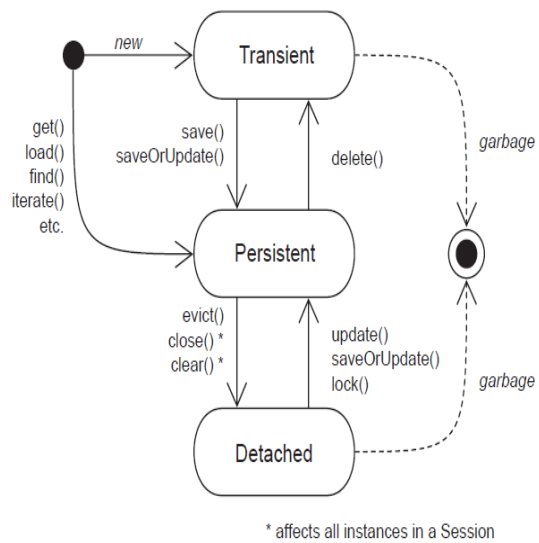
Estados en el contexto de persistencia

Existen diferentes estados

Transient: El objeto está creado en memoria con `new ...()`, pero no está asociado a una Session de Hibernate. No tiene representación en BD

Persistent o **managed**: El objeto tiene representación en BD, cuenta con un ID y se encuentra en el ámbito de una Session

Detached: Una instancia de un objeto persistente, pero que no se encuentra actualmente asociado a una Session.



Fuente: <https://www.mysoftkey.com/wp-content/uploads/2017/07/hibernate-object-lifecycle.png>

Para saber más:

[https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate User Guide.html#pc](https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate%20User%20Guide.html#pc)