Síntesis conceptual

Asignatura: Acceso a datos
Unidad: 4. Correspondencia objeto-relacional

Resumen

Existen diferentes procesos para hacer persistente un objeto en una base de datos. Una de ellas es a la **ORM**, que emplea técnicas y herramientas para persistir un objeto según una correspondencia o mapa que se asocia cada atributo de un objeto a su correspondiente tabla y columnas en la base de datos relacional.

Uno de los framework más importantes para aplicar ORM es Hibernate. Hibernate en su arquitectura simula una capa intermedia entre la aplicación y la base de datos, permitiendo la conexión con la base de datos, gestión de las sesiones, transacciones, ficheros de correspondencia, entre otras características.

Hibernate maneja los siguientes ficheros:

- **hibernate.cfg.xml** – Fichero de configuración
- **hibernate.reveng.xml** – Fichero base para crear las clases y ficheros de mapeo a partir de las tablas existentes en la base de datos.
- **POJOs** – Clases con sus atributos que representan los objetos correspondiente a la correspondencia de las tablas y sus columnas de una base de datos.
- Ficheros **.hbm.xml** – Ficheros de correspondencia o mapeo que indican la relación entre los POJOs y las tablas de la base de datos relacionales.
- **HibernateUtil.java** – Clase en java que centraliza el arranque de Hibernate y la disponibilidad de sesiones.

Un objeto según el estado de uso en que se encuentre podemos diferenciar:

- **Transitorio**. Es un objeto nuevo, está en la aplicación, pero no está en el contexto de hibernate ni en la base de datos.
- **Persistente o gestionado**. Es un objeto que está en el contexto de hibernate y está en la base de datos. Cualquier cambio se verá reflejado en la base de datos.
- **Separado**. Es un objeto que no está en el contexto de hibernate pero si existe en la base de datos. Cualquier cambio no se verá reflejado en la base de datos.
- **Eliminado.** Es un objeto que está en el contexto de hibernate y existe en la base de datos, pero está pendiente de ser eliminado en la base de datos.

Existen diferentes métodos para guarda, actualizar, obtener, cerrar, y otros, que una vez ejecutados cambiarán de estado el objeto.

A parte, hibernate dispone de su propio lenguaje HQL para realizar cualquier tipo de sentencia, similar a SQL. Y un sublenguaje JPQL para la persistencia.

Con la interfaz Query y usando los métodos .createQuery() y executeUpdate() podemos ejecutar sentencias de HQL. Aun así, la sesión que crea hibernate dispone del método .createNativeQuery para ejecutar sentencia en SQL.

Una de las dificultades que contienen la bases de datos relacionales con respecto a la representación en objetos, es que no existe una forma exacta de representar la herencia de objetos. Existe dos formas de simular la herencia:

- Eliminación de subtipos. Una sola tabla representa la herencia, tiene todos los campos que componen la clase principal y la subclase. A demás se añade un campo para indicar el tipo o clasificación, es decir, el nombre de la clase de la herencia.
- Eliminación de la jerarquía. Hay una tabla por cada clase de la herencia. Y las tablas que representan las subclases de la herencia, tendrán como clave primaria la misma que la tabla de la clase padre.

Conceptos fundamentales

- **ORM (Object-Relational Mapping):** Técnicas y herramientas para hacer un objeto persistente, mediante un proceso de mapeo o correspondencia a partir de los datos del tipo de lenguaje de programación orientado a objetos a una base de datos relacional.

- **Mapping o correspondencia:** Es la relación entre una clase y sus atributos y la tabla y sus columnas que la representa en la base de datos relacional.

- **Ficheros hbm**: Ficheros de hibernate en los que se especifica los mapeos de los objetos.

- **HQL (Hibernate Query Language)**: Lenguaje propio de hibernate, similar a SQL. En vez de tratar datos, tratan objetos.

- **JPA (Java Persistence API):** API estándar de JAVA para ORM

- **JPQL:** Es un lenguaje propio de hibernate. Subconjunto de HQL con partes de especificación JPA