Laboratorio 2

Tema: Hibernate

Profesor: JUAN LOPREIATO

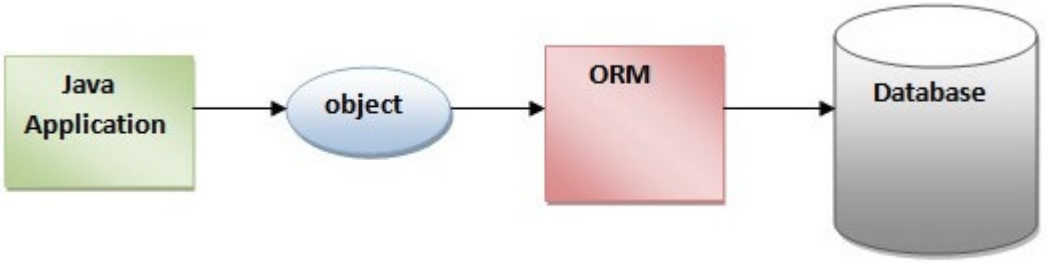
Alumnos:
Victoria Gabay
Sebastian Javier Roca
Tomás Stambulsky
Gian Franco Stigliano
Gastón Cuervo

Cuatrimestre y año: 2do 2022

Introducción

Al momento de crear una aplicación, existen varios paradigmas que se pueden utilizar, uno de ellos, es la programación orientada a objetos. Bajo este paradigma, tenemos la necesidad de crear objetos, a esto, podemos sumarle herencias e interfaces con distintos tipos de patrones si queremos obtener una arquitectura más robusta y escalable en el tiempo.
Una de las necesidades que nace al momento de construir la aplicación, es la de persistir los datos en una base, para esto, una de las herramientas que podemos utilizar en Java es JDBC. Es una API que viene incluida en la JDK de Java y nos permite realizar conexiones de bajo nivel a la base de datos. Para poder utilizarla, es necesario conocer el lenguaje SQL, ya que JDBC es meramente el nexo que permite la comunicación de nuestra aplicación y la base de datos. No nos provee de herramientas de alto nivel para realizar operaciones sobre ella, como puede ser un método para realizar una consulta a una tabla según un parámetro de entrada u otras operaciones similares.
Hibernate ORM (Object Relational Mapping) nace en la búsqueda de facilitar este mapeo y consultas hacia la base de datos. Es un framework que está construido sobre JDBC, es decir, lo implementa y nos abstrae de él, lo que nos permite realizar operaciones de alto nivel con facilidad. Gracias a esto, podemos realizar operaciones de manera sencilla y rápida a la base de datos.

Hibernate es una herramienta de mapeo objeto-relacional (ORM) bajo licencia GNU LGPL para Java, que facilita el mapeo de atributos en una base de datos tradicional, y el modelo de objetos de una aplicación mediante archivos declarativos o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Es una herramienta con gran escalabilidad y muy eficiente que busca optimizar nuestro flujo de trabajo, agilizando la relación entre la aplicación y nuestra base de datos



Como información previa para entender qué es Hibernate, explicamos el concepto de mapeador objeto-relacional (ORM).

La programación orientada a objetos y bases de datos relacionales son dos paradigmas diferentes. El modelo relacional trata con relaciones y conjuntos. Sin embargo, el paradigma orientado a objetos trata con objetos, sus atributos y asociaciones de unos a otros.

Un **ORM** es un sistema que permite almacenar objetos de aplicaciones Java en tablas de sistemas de bases de datos relacionales usando metadatos que describen la relación entre los objetos y la base de datos, y lo hace de una manera transparente y autónoma.

Hibernate parte de una filosofía de mapear objetos Java, también conocidos como “POJOs” (Plain Old Java Objects). Con Hibernate no es necesario escribir código específico en nuestros objetos ni hacer que hereden de clases determinadas. En vez de eso trabajamos con ficheros XML y objetos que proporciona la librería. Una de las principales características de Hibernate es su flexibilidad, envolviéndolo todo bajo un marco de trabajo común.

Existen otros tipos de ORM en el mercado, uno de ellos es **Apache Cayenne** que cuenta con algunas diferencias, una de ellas es que en vez de utilizar POJOs, utiliza OO que significa Object-oriented clases. Otra de las diferencias es que Cayenne realiza una inicialización lazy de todas las relaciones por default. Cuando una de estas relaciones es necesitada y no hay una sesión a la base de datos activa, crea una nueva y realiza la consulta.
Una de las desventajas que podría verse sobre Cayenne, es que no cuenta con las especificaciones JPA (java persistence API). Hibernate hace uso de estas especificaciones , esto quiere decir que podemos cambiar la implementación que utilizamos como acceso a la base por otra que también la implemente.

¿Qué ventaja y desventaja tiene?

Entre las principales ventajas técnicas que aporta Hibernate están las siguientes:

1. Permite trabajar con la base de datos por medios de entidades en vez de querys.
2. Nos ofrece un paradigma orientado 100% a objetos.
3. Elimina errores en el tiempo de ejecución.

4. Mejora el mantenimiento del software.
5. Genera gran parte del SQL en tiempo de inicialización del sistema en vez de en tiempo de ejecución.

Luego la desventaja principal sería:

1. Hibernate solo impone una condición para poder utilizarse. Las claves deben tener una clave primaria.
2. No es una solución óptima para proyectos de migración de datos.
3. No ofrece toda la funcionalidad que ofrecería tirar consultas nativas.
4. Puede representar una curva de aprendizaje más grande.
5. Gran cantidad de ficheros de configuración.

Algunos ejemplos de uso reales

Hibernate puede ser utilizado en cualquier aplicación Java que requiere mapear un conjunto de tablas SQL a un conjunto de objetos (instancias de una clase).
Hibernate debería ser el ORM elegido para desarrollar principalmente cuando se requiere de llevar un código rápidamente a producción, a parte de considerar algunos puntos extras como

- La mayor parte de las entidades del sistema requieren de un CRUD (Create, Read, Update, Delete)
- Situaciones en las que no sabemos si el día de mañana la estructura de datos va a ser modificada.
- La aplicación tiene formularios de búsqueda complejos.
- Se deben manejar grandes volúmenes de datos en caché.

En un principio, Hibernate debía ser configurado mediante un archivo xml, que especifique la tabla y la clase a mapear, ahora es posible configurarlo mediante anotaciones dentro de la misma clase.
Las anotaciones. definidas por un @ y la palabra clave pueden ir tanto sobre la clase en sí como sobre las propiedades y son acumulables. Puedo definir una clase como @Entity y asignarle la tabla a la que corresponde con la anotación @Table(name = “MiTabla”).

Dentro de Hibernate están implementadas las funcionalidades de JPA (Java Persistence API), lo que permite definir las entidades, relaciones y comportamientos de las mismas dentro de nuestra aplicación. Esa implementación es la que facilita el uso de las anotaciones.

Sintaxis y forma de uso

- **@Entity:** Le indica a JPA que esa clase puede almacenarse en una tabla.
- **@Table**: le indica a JPA a que tabla hace referencia esa entidad.
- **@Id :** Marca a ese atributo de la clase como Primary Key
- **@GeneratedValue(strategy)**: Especifica la estrategia de generación del atributo marcado como @Id.
    - AUTO: decide que generador usar según como lo haga el motor de base de datos utilizado.
    - IDENTITY: la base de datos es la encargada de determinar y asignar las PKs
    - SECUENCE: Un objeto que puede ser utilizado como una fuente de PKs
    - TABLE: indica que se usa una tabla separada para mantener las PKs.
- **@Column:** Indica a qué columna de la tabla pertenece ese atributo.
- **@ManyToMany:** indica que ese campo va a estar dentro de una relación de muchos a muchos. Se le debe indicar la clase target de esa relación .
- **@JoinTable:** Es la anotación que indica cual es la tabla asociativa que contiene esa relación muchos a muchos.

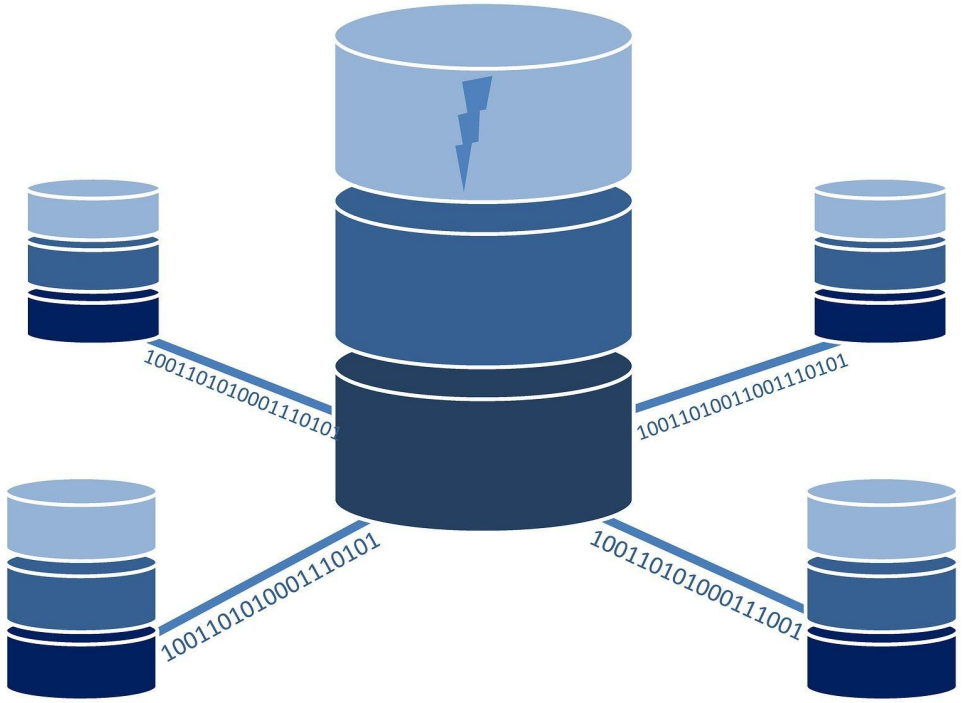
```java
@Entity
@Table(name = "characters")

public class Characters {

  @Id //SET ID AS PRIMARY KEY
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;

  @Column(name = "nombre")
  private String name;

  @Column(name = "imagen")//SPECIFY COLUMN NAME
  private String imageUrl;

  @Column(name = "edad")
  private int age;
```

```
  @Column(name = "peso")
  private float weight;

  @Column(name = "historia")
  private String history;
```

Si tuviera que realizar una aplicación para una compañía médica que debe permitir cosas como dar de alta usuarios (médicos, pacientes, administradores), solicitar, cancelar y modificar turnos, entre otras cosas utiliza Hibernate como ORM, además de agregar Spring JPA que implementa el patrón Repository y permite una gran agilización (a nivel de desarrollo de código) para consultas más complejas.

```
// Ejemplo de un repositorio que implementa spring JPA y que
// permite realizar una búsqueda de personajes por nombre
// simplemente poniendo el nombre del método como findByName y
// enviando nombre como parámetro.
public interface CharactersRepository extends
JpaRepository<Characters,Integer> {

  public Optional<Characters> findByName(String name);
  public List<Characters> findByAge(int age);
}
```

¿Qué tan escalable es hibernate?



Existe una gran variedad de bases de datos NoSQL, muchas de las cuales han surgido de las necesidades específicas de distintas compañías. Una de las características más importantes para dichas compañías suele ser lograr una buena escalabilidad. Aquí es donde entra Hibernate, dado que tiene una gran escalabilidad y resulta ser muy eficiente. Esto se debe a su arquitectura de doble capa y al hecho de que podría ser usado en un cluster.

Además, Hibernate le da mucha importancia a la concurrencia. Porque asegura la identidad sólo a nivel de unidades de trabajo en un hilo simple no requiere un bloqueo costoso ni otros medios de sincronización. Esto se termina traduciendo en una mayor facilidad para la escalabilidad.

Usualmente usar el caché ayuda mucho a aumentar el rendimiento de la base de datos, dado que al guardar las estructuras de objetos en memoria, permite un acceso más rápido a ellas. Pero todos tienen problemas que se remarcan cuanto mayor sea la concurrencia de la base de datos. Estos problemas pueden llevar a una caída del rendimiento sustancial. Hibernate le da mucha importancia a optimizar y arreglar estos temas mediante su arquitectura.

En el primer nivel del cache se guardan los objetos a recuperar de la base de datos. Además de otras ventajas más técnicas. La segunda capa del caché permite una gran configuración y seleccionar estrategias a medida para la situación.

También Hibernate da la opción de habilitar individualmente la caché que será para cada una de las entidades. Al hacer esto se debe establecer una estrategia de concurrencia, que será utilizada al sincronizar la caché de primer nivel con la segunda y esta con la base de datos. Dándole más versatilidad al poder elegir la estrategia.

Fuentes

1. https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/97
2. https://prezi.com/vwadwvpqgmak/hibernate-es-una-herramienta-de-mapeo-objeto-relacional-orm/
3. https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html_single/
4. https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/97
5. https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/180
6. http://jcesarperez.blogspot.com/2008/09/cuando-usar-no-usar-hibernatede.html
7.