**Giới thiệu**

Hibernate là framework được sử dụng nhiều nhất hiện nay để giúp lập trình viên Java có thể map các class (Pojo) với một cơ sở dữ liệu bất kỳ.

Trước khi Hibernate ra đời, chúng ta thường thao tác với cơ sở dữ liệu thông qua JDBC. Theo thời gian, JDBC bộc lộ nhiều điểm yếu như:

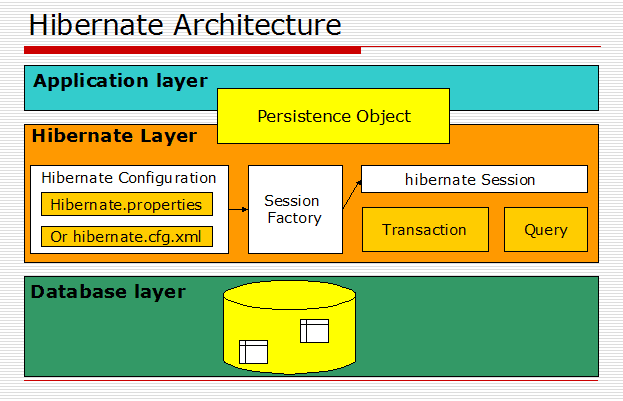
* Có nhiều code thừa mà chỉ phục vụ mục đích là lấy dữ liệu.
* Mất nhiều thời gian map dữ liệu vào object Java.
* Sẽ tốn nhiều công sức khi hệ thống thay đổi CSDL (yêu cầu jdbc mới, code mới)
* Giao tiếp giữa các bảng thường khó, thiếu tính OOP trong đó.

Từ đây, để giảm tải gánh nặng cho dev khi thao tác với database. Hibernate ra đời!

**Định nghĩa**

Hibernate là một thư viện ORM (Object Relational Mapping) mã nguồn mở giúp lập trình viên viết ứng dụng Java có thể map các objects (pojo) với hệ quản trị cơ sở dữ liệu quan hệ, và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cớ dữ liệu quan hệ.

Hiểu ngắn gọn thì Hibernate sẽ là một layer đứng trung gian giữa ứng dụng và database, và chúng ta sẽ giao tiếp với Hibernate thay vì giao tiếp với database



Để giao tiếp với Hibernate, chúng ta sẽ tạo ra một Class đại diện cho một Table. Và mọi dữ liệu từ Table trong database sẽ được Hibernate bind vào Class đó cho chúng ta.

**POJO**

Pojo (plain old Java object) là class đại diện cho một Table, thuật ngữ này để định nghĩa chính xác thì mình không dám chắc, nhưng về ý nghĩa thì nó là một class java thuần túy, rất thuần túy:

1. All properties must public setter and getter methods (mọi biến đều phải có get/set)
2. All instance variables should be private (mọi biến là thuộc tính thì nên là private)

public class MyFirstPojo

{

private String name;

public static void main(String [] args)

{

for (String arg : args)

{

MyFirstPojo pojo = new MyFirstPojo(arg); // Here's how you create a POJO

System.out.println(pojo);

}

}

public MyFirstPojo(String name)

{

this.name = name;

}

public String getName() { return this.name; }

public String toString() { return this.name; }

}

**Mapping dữ liệu**

Khi đã có Class đại diện cho Table rồi, chúng ta sẽ định nghĩa các trường trong class đó tương ứng với column nào trong database bằng tập hợp các Annotaion mà Hibernate cung cấp.

@Entity // Đánh dấu đây là một Entity, chịu sự quản lý của Hibernate

@Table(name = "USER") //Entity này đại diện cho table USER trong db

public class UserModel {

@Id // Đánh dấu biến ở dưới là primary key của table này

@GeneratedValue(strategy = GenerationType.IDENTITY) // Tự động tăng giá trị khi insert

private Long id;

@Column(name = "email", unique = true) // trường email ở dưới đại diện cho cột email trong database

private String email;

@Column(name = "name")

private String name;

public Long getId() {

return this.id;

}

public void setId(Long id) {

this.id = id;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getName() {

return this.name;

}

public void setName(String name) {

this.name = name;

}

}

Bây giờ việc bạn lấy dữ liệu từ database sẽ đại loại như này:

public List<User> findAll() {

return session.createQuery("SELECT a FROM User a", User.class).getResultList();

}

Chúng ta tiết kiệm được rất nhiều thời gian cho việc mapping dữ liệu từ database sang class java, và đặc biệt là khi thay đổi Database thì cũng sẽ không ảnh hưởng gì tới đoạn code ở trên cả, chúng ta gần như trong suốt với tầng database, mà chỉ cần nói chuyện với Hibernate là đủ!

**Session**

Đối tượng chính của việc truy xuất hay insert dữ liệu bằng Hibernate chính là Session và được tạo ra từ Session Factory.

Session Factory Là một interface giúp tạo ra session kết nối đến database bằng cách đọc các cấu hình trong một file xml và mỗi loại Database khác nhau sẽ có một cấu hình khác nhau.

File cấu hình hibernate.cfg.xml có dạng như sau:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration><session-factory><propertyname="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property><propertyname="hibernate.connection.url">jdbc:mysql://192.168.10.13:3306/loda

</property><propertyname="hibernate.connection.username">root</property><propertyname="hibernate.connection.password">root</property><propertyname="hibernate.connection.pool\_size">10</property><propertyname="dialect">org.hibernate.dialect.MySQLDialect</property><propertyname="hibernate.current\_session\_context\_class">thread</property></session-factory></hibernate-configuration>

Khi đã có file config này rồi, chúng ta sử dụng nó để tạo ra Session Factory như sau:

public class SessionFactoryProvider {

private static SessionFactory buildSessionFactory() {

ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()//

.configure("hibernate.cfg.xml").build();

Metadata metadata = new MetadataSources(serviceRegistry).getMetadataBuilder().build();

return metadata.getSessionFactoryBuilder().build();

}

}

Từ đó, mỗi lần cần query hay insert dữ liệu, chúng ta sẽ tạo ra Session và sử dụng.

SessionFactory factory = HibernateSessionUtils.getSessionFactory();

Session session = factory.getCurrentSession();

try {

session.getTransaction().begin();

List<User> users = session.createQuery("SELECT a FROM User a", User.class).getResultList();

session.getTransaction().commit();

}catch (Exception e) {}

**Hibernate Query Language (HQL)**

Hibernate sử dụng ngôn ngữ Hibernate Query Language (HQL) để query dữ liệu. Nó chỉ khác SQL bình thường ở chỗ, đối tượng tác động lúc này là Entity chứ không còn là Table nữa:

ví dụ:

-- SQL

-- from table name

Select u.id, u.email from USER u;

-- HQL

-- from class name

Select u.id, u.email from User u;

-- query toàn bộ object

Select u from User u;