Entity Mapping:

Describe the concept of entity mapping in Hibernate. Explain how to map a Java class (User) to a database table (users) using annotations (@Entity, @Table). Include examples of mapping fields (@Id, @Column) and associations (@ManyToOne, @OneToMany).

Entity mapping in Hibernate is the process of defining how Java objects (entities) are mapped to database tables. This involves specifying the relationship between Java class fields and database table columns, as well as defining relationships between different entities (such as one-to-many or many-to-one relationships).

Basic Annotations for Entity Mapping

- \*@Entity\*: This annotation specifies that the class is an entity and is mapped to a database table.

- \*@Table\*: This annotation specifies the name of the database table to which the entity is mapped.

- \*@Id\*: This annotation specifies the primary key of an entity.

- \*@Column\*: This annotation specifies the details of the column to which a field is mapped.

Example: Mapping a User Entity

1. Basic Mapping

Here is a simple example of how to map a User class to a users table:

import javax.persistence.Entity;

import javax.persistence.Table;

import javax.persistence.Id;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Column;

@Entity

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "username", nullable = false)

private String username;

@Column(name = "password", nullable = false)

private String password;

@Column(name = "email", nullable = false)

private String email;

// Getters and Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

Mapping Relationships

2. One-to-Many and Many-to-One Relationships

Suppose we have two entities, User and Post, where one user can have multiple posts. Here is how we can map this relationship using annotations.

\*User Entity:\*

import javax.persistence.Entity;

import javax.persistence.Table;

import javax.persistence.Id;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Column;

import javax.persistence.OneToMany;

import java.util.List;

@Entity

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "username", nullable = false)

private String username;

@Column(name = "password", nullable = false)

private String password;

@Column(name = "email", nullable = false)

private String email;

@OneToMany(mappedBy = "user")

private List<Post> posts;

}

\*Post Entity:\*

import javax.persistence.Entity;

import javax.persistence.Table;

import javax.persistence.Id;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Column;

import javax.persistence.ManyToOne;

import javax.persistence.JoinColumn;

@Entity

@Table(name = "posts")

public class Post {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "title", nullable = false)

private String title;

@Column(name = "content", nullable = false)

private String content;

@ManyToOne

@JoinColumn(name = "user\_id", nullable = false)

private User user;

// Getters and Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getContent() {

return content;

}

public void setContent(String content) {

this.content = content;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

}