Chạy spring : ./mvnw spring-boot:run

* Khi mà kéo dự án fullstack về , muốn chạy thử thì phải cấu hình lại cho nó , cấu hình database đồ bla blae , **docker compose -f mysql.yml up**  cho cái dự án này
* # config spring sceurity
* spring.sceurity.user.name=hieuvo
* spring.security.user.password=hienvo123

Tạo form với JSTL

* <%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
* Tham khảo: https://www.baeldung.com/spring-mvc-form-tutorial
* Lưu ý về **@ModelAttribute** Annotation : là cách chúng ta convert dữ liệu từ **View** trả cho
* **controller** xử lý
* A computer screen shot of a diagram

  Description automatically generated

**https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywordsreference.html**

[**https://docs.spring.io/spring-data/jpa/reference/repositories/query-methodsdetails.html**](https://docs.spring.io/spring-data/jpa/reference/repositories/query-methodsdetails.html)

**Ví dụ:**

**//repository**

**@Repository**

**public interface UserRepository extends CrudRepository<User, Long> {**

**Optional<User> findOneByActivationKey(String activationKey);**

**List<User> findAllByActivatedIsFalseAndCreatedDateBefore(Instant dateTime);**

**Optional<User> findOneByResetKey(String resetKey);**

**Page<User> findAllByIdNotNullAndActivatedIsTrue(Pageable pageable);**

**}**

**A whiteboard with red writing

Description automatically generated**

1. **Tạo relationship**

1.The **inverse side** cần sử dụng **mappedBy**

-> the **owner side** không được sử dụng mappedBy (tức là table có FK - khóa ngoại)

**Mối quan hệ Users và Roles**: N -1

1 user có 1 role và **table user chứa role\_id** (owner side)

User -> ManyToOne - lưu là Role

@ManyToOne private Role role;

1 role có nhiều user -> Role - OneToMany - lưu là List<User>

@OneToMany List<User> users;

**Bước 2:** chỉ rõ column mapping

Sử dụng **mappedBy** cho inverse side:

**@One**ToMany( mappedBy = “role”) List<User> users;

Sử dụng **joinColumns** cho owner side:

@**Many**ToOne

@JoinColumn(name = "role\_id") private Role role;

1. Many to many <https://www.bezkoder.com/jpa-many-to-many/>

@ManyToMany(fetch = FetchType.LAZY, cascade = { CascadeType.PERSIST, CascadeType.MERGE }, mappedBy = "tags")

@Entity

public class Post {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToMany(fetch = FetchType.LAZY, cascade = { CascadeType.PERSIST, CascadeType.MERGE })

@JoinTable(

name = "post\_tag",

joinColumns = @JoinColumn(name = "post\_id"),

inverseJoinColumns = @JoinColumn(name = "tag\_id"))

private Set<Tag> tags = new HashSet<>();

}

@Entity

public class Tag {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToMany(fetch = FetchType.LAZY, cascade = { CascadeType.PERSIST, CascadeType.MERGE }, mappedBy = "tags")

private Set<Post> posts = new HashSet<>();

}

**3. cascade = { CascadeType.PERSIST, CascadeType.MERGE }**

* cascade cho phép chỉ định các hành động mà Hibernate/JPA sẽ tự động thực hiện trên các thực thể liên quan khi có thay đổi đối với thực thể chính.
  + **CascadeType.PERSIST**: Khi bạn lưu (persist) một thực thể chính (chẳng hạn như Post), các thực thể liên quan (chẳng hạn như tags) cũng sẽ được tự động lưu nếu chúng chưa tồn tại trong cơ sở dữ liệu.
  + **CascadeType.MERGE**: Khi bạn cập nhật (merge) một thực thể chính, các thực thể liên quan cũng sẽ được cập nhật nếu chúng thay đổi.

Điều này giúp quản lý sự phụ thuộc giữa các thực thể một cách thuận tiện hơn, tự động hóa các thao tác lưu và cập nhật.

VD: cách làm này mình khỏi cần tạo class khóa tổ hợp, giờ mình làm chung luôn nó có một khóa chính

**Order\_Detail:**

id: long

**order\_id**: long

product\_id: long

quantity: long

price: double

Order\_detail

**@ManyToOne**

**@JoinColumn(name = "orderId", insertable = false, updatable = false) // Chỉ định không insert hoặc update**

**private Order order;**

**@ManyToOne**

**@JoinColumn(name = "productId", insertable = false, updatable = false) // Chỉ định không insert hoặc update**

**private Product product;**

**1 order hasMany order\_detail (1 - N)**

@ManyToOne

@JoinColumn(name = “order\_id”)

private Order order;

@OneToMany( mappedBy = “order”)

List<OrderDetail> orderDetails;

**Quan hệ giữa Product và Order\_detail**

**1 product có thể thuộc nhiều order\_detail (1 - N)**

@ManyToOne

@JoinColumn(name = “product\_id”)

private Product product;

**CÓ THỂ TẠO MỘT BẢNG TRUNG GIAN NHƯ THẾ NÀY ( HỎI CHATGPT)**

**1. Class OrderProductPK (Khóa chính tổng hợp)**

@Embeddable

public class OrderProductPK implements Serializable {

@Column(name = "orderId")

private Long orderId;

@Column(name = "productId")

private Long productId;

// Constructors, getters, setters, equals, and hashCode

// Getters and Setters, equals, hashCode

}

**2. Class OrderProduct (Bảng trung gian)**

@Entity

@Table(name = "Order\_Product")

public class OrderProduct {

@EmbeddedId

private OrderProductPK id;

@ManyToOne

@MapsId("orderId")

@JoinColumn(name = "orderId")

private Order order;

@ManyToOne

@MapsId("productId")

@JoinColumn(name = "productId")

private Product product;

private Integer quantity;

// Constructors, getters and setters

// Getters and Setters

}

**3. Class Order**

@Entity

@Table(name = "Orders")

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long orderId;

private Double totalPrice;

@Column(name = "userId")

private Long userId;

@OneToMany(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval = true)

private Set<OrderProduct> orderProducts = new HashSet<>();

// Constructors, getters and setters

public Order() {}

public Order(Double totalPrice, Long userId) {

this.totalPrice = totalPrice;

this.userId = userId;

}

public void addProduct(Product product, Integer quantity) {

OrderProduct orderProduct = new OrderProduct(this, product, quantity);

orderProducts.add(orderProduct);

}

public void removeProduct(Product product) {

orderProducts.removeIf(op -> op.getProduct().equals(product));

}

// Getters and Setters

}

**4. Class Product**

@Entity

@Table(name = "Products")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long productId;

private String nameProduct;

private Double price;

private String image;

private String detailDesc;

private String shortDesc;

private Integer quantity\_spconlai;

private Integer sold;

private String factory;

private String tager;

@OneToMany(mappedBy = "product", cascade = CascadeType.ALL, orphanRemoval = true)

private Set<OrderProduct> orderProducts = new HashSet<>();

// Constructors, getters, setters

// Getters and Setters

}

**Tóm tắt:**

* **OrderProductPK**: Định nghĩa khóa chính tổng hợp cho bảng trung gian.
* **OrderProduct**: Bảng trung gian lưu trữ thông tin mối quan hệ giữa Orders và Products với thuộc tính bổ sung quantity.
* **Order** và **Product**: Hai thực thể chính có quan hệ **Many-to-Many** thông qua bảng trung gian.

1. **Tái sử dụng layout**

<https://stackoverflow.com/questions/60956605/jsp-include-not-working-file-not-found-status-500/60957169#60957169>

* Để lấy dữ liệu khi mình chọn 1-N
* <div class="mb-3 col-12 col-md-6">
* <label class="form-label">Role:</label>
* <form:select class="form-select" path="role.roleName">
* <form:option value="ADMIN">ADMIN</form:option>
* <form:option value="USER">USER</form:option>
* </form:select>

Vì nó là dữ liệu Object nên mình phải . để lấy

1. Để upload, gửi file thì mình truyền nó dưới dạng Form-Data

A screenshot of a computer program

Description automatically generated

**Về error tags:**

https://docs.spring.io/spring-framework/reference/web/webmvc-view/mvcjsp.html#mvc-view-jsp-formtaglib-errorstag

https://docs.spring.io/spring-framework/docs/6.1.5/javadocapi/org/springframework/web/servlet/tags/form/ErrorsTag.html

**Validate với Bootstrap 5:**

https://getbootstrap.com/docs/5.0/forms/validation/#server-side