TRƯỜNG ĐẠI HỌC HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

TÀI LIỆU THỰC HÀNH

SPRING BOOT

CHƯƠNG 04: TẠO CƠ SỞ DỮ LIỆU BÁN HÀNG

Giảng viên biên soạn: Ths. Nguyễn Minh Tân

2024

BUỔI 4: TẠO CƠ SỞ DỮ LIỆU BÁN HÀNG

I. MỤC TIÊU

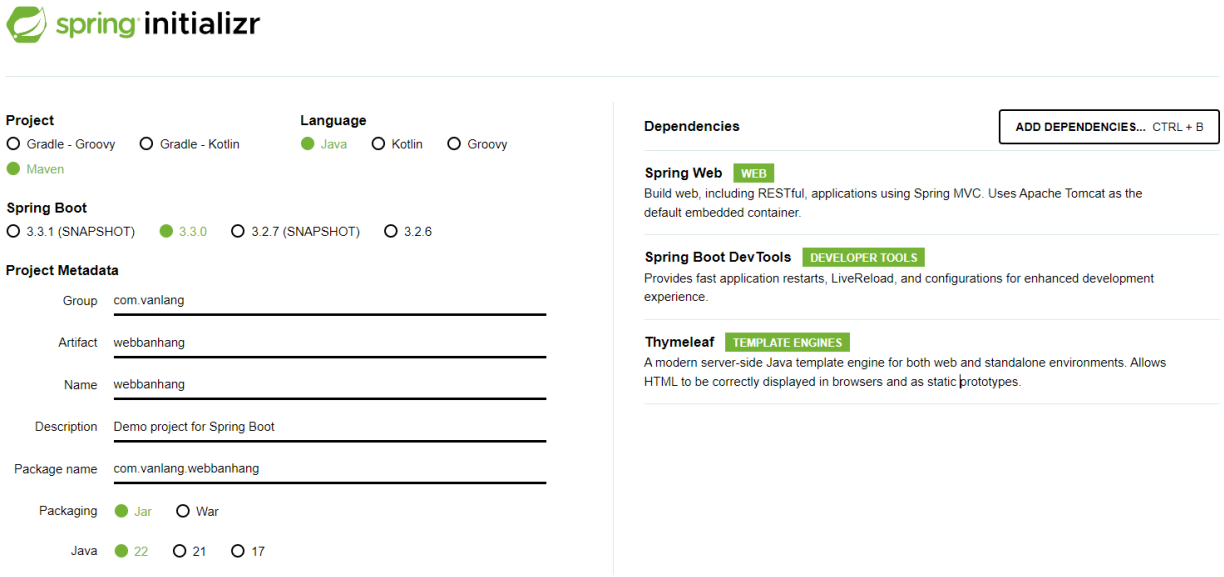
❖ Sau khi học xong bài này, sinh viên có thể:

- Nắm vững các khái niệm cơ bản về cơ sở dữ liệu và mô hình hóa dữ liệu.
- Hiểu được cách thiết kế mô hình ER (Entity-Relationship) và biến đổi mô hình ER thành cấu trúc bảng trong cơ sở dữ liệu.
- Biết cách sử dụng hệ quản trị cơ sở dữ liệu (DBMS) để tạo và quản lý cơ sở dữ liệu.
- Có khả năng tạo bảng, thiết lập khóa chính và khóa ngoại, và định nghĩa các ràng buộc toàn vẹn dữ liệu.
- Phát triển kỹ năng phân tích yêu cầu của hệ thống và áp dụng vào thiết kế cơ sở dữ liệu

II. NỘI DUNG THỰC HÀNH:

Tạo một dự án Spring Boot mới

Download Template Spring Boot từ spring.io



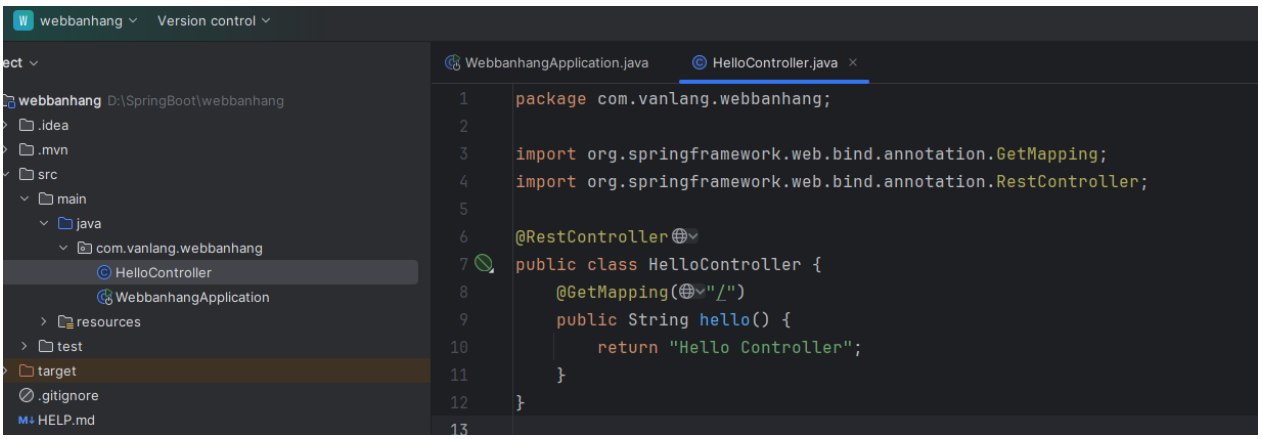
Cấu hình dự án

Tiến hành giải nén thư mục dự án Spring boot template có tên webbanhang.zip đã được tải về trước đó. Sau khi giải nén thành công, ta có thể thấy thư mục **webbanhang** xuất hiện. Tiếp theo, mở IntelliJ IDEA và chọn Open. Duyệt đến thư mục dự án **webbanhang**. IntelliJ IDEA sẽ tải và mở dự án **webbanhang**.

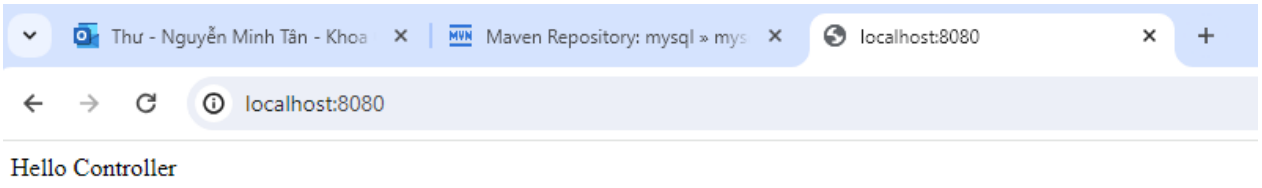
Bổ sung thêm một vài thư viện sử dụng trong dự án ban đầu là

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>

<dependency>
    <groupId>jakarta.validation</groupId>
    <artifactId>jakarta.validation-api</artifactId>
</dependency>

<dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
</dependency>
```

Tạo mới một class HelloController trong package mặc định, sau đó thêm đoạn code sau để test thử server



Kết quả khi thực hiện chương trình



4.1 Cài đặt Workbench để quản lý cơ sở dữ liệu MySQL

MySQL Workbench là công cụ giúp người dùng thao tác dễ dàng, trực quan hơn với database. Thay vì phải làm việc với database thông qua giao diện dòng lệnh nhàm chán như trước đây thì bây giờ bạn có thể nhìn database một cách trực quan hơn qua giao diện của MySQL Workbench.

4.2 Thêm các Dependency

Để kết nối và làm việc với cơ sở dữ liệu MySQL trong một ứng dụng Spring Boot sử dụng Maven, bạn cần thêm các dependency liên quan đến Spring Data JPA và JDBC driver cho MySQL vào file pom.xml.

4.2.1 Dependency cho Spring Data JPA

Spring Data JPA giúp quản lý dữ liệu trong các ứng dụng Java qua JPA. Spring Boot cung cấp một starter kit giúp dễ dàng tích hợp và sử dụng JPA:

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Link maven: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa/3.3.0

4.2.2 Dependency cho MySQL JDBC Driver

JDBC driver cho MySQL để Spring Boot có thể kết nối và thao tác với cơ sở dữ liệu MySQL. Sử dụng dependency sau:

```
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>8.0.33</version>
</dependency>
```

Link maven: https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.33

4.2.3 Dependency cho Lombok

Lombok là một thư viện Java giúp tự sinh ra các hàm setter/getter, hàm khởi tạo, toString… và tinh gọn chúng. Sử dụng dependency sau:

```
<dependency>
   <groupId>org.projectlombok</groupId>
   <artifactId>lombok</artifactId>
   <scope>provided</scope>
</dependency>
```

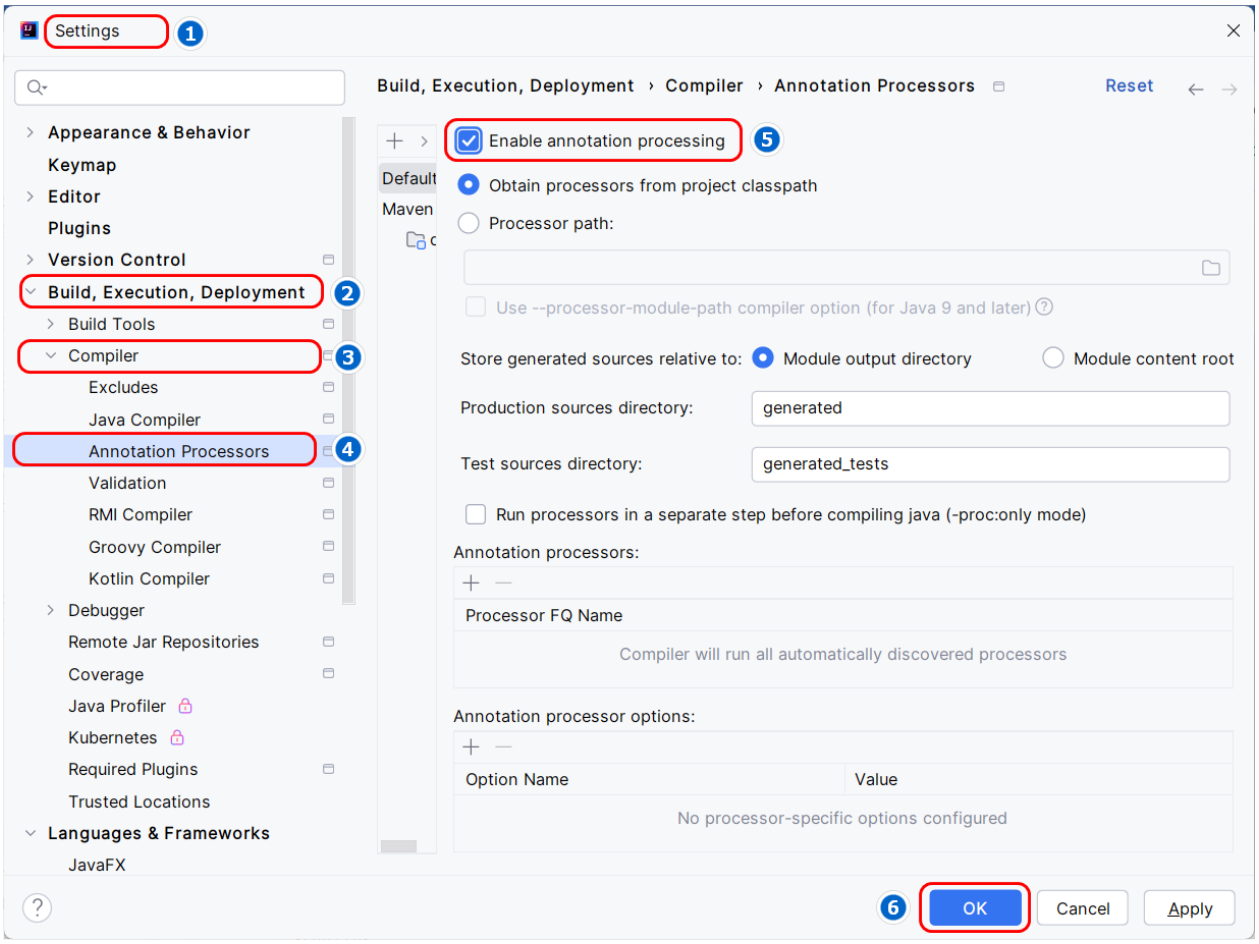
Link maven: https://mvnrepository.com/artifact/org.projectlombok/lombok/1.18.32

Cấu hình IDE:

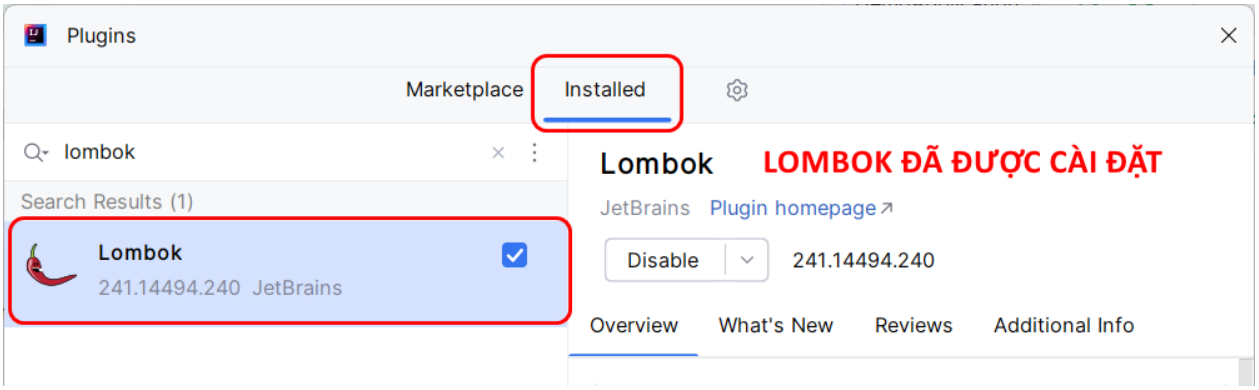
Khi sử dụng Lombok phải đảm bảo rằng đã kích hoạt xử lý annotation trong IDE.

Trong **IntelliJ IDEA: File → Settings → Build, Execution, Deployment → Compiler → Annotation Processors** → Tích vào **"Enable annotation processing".**

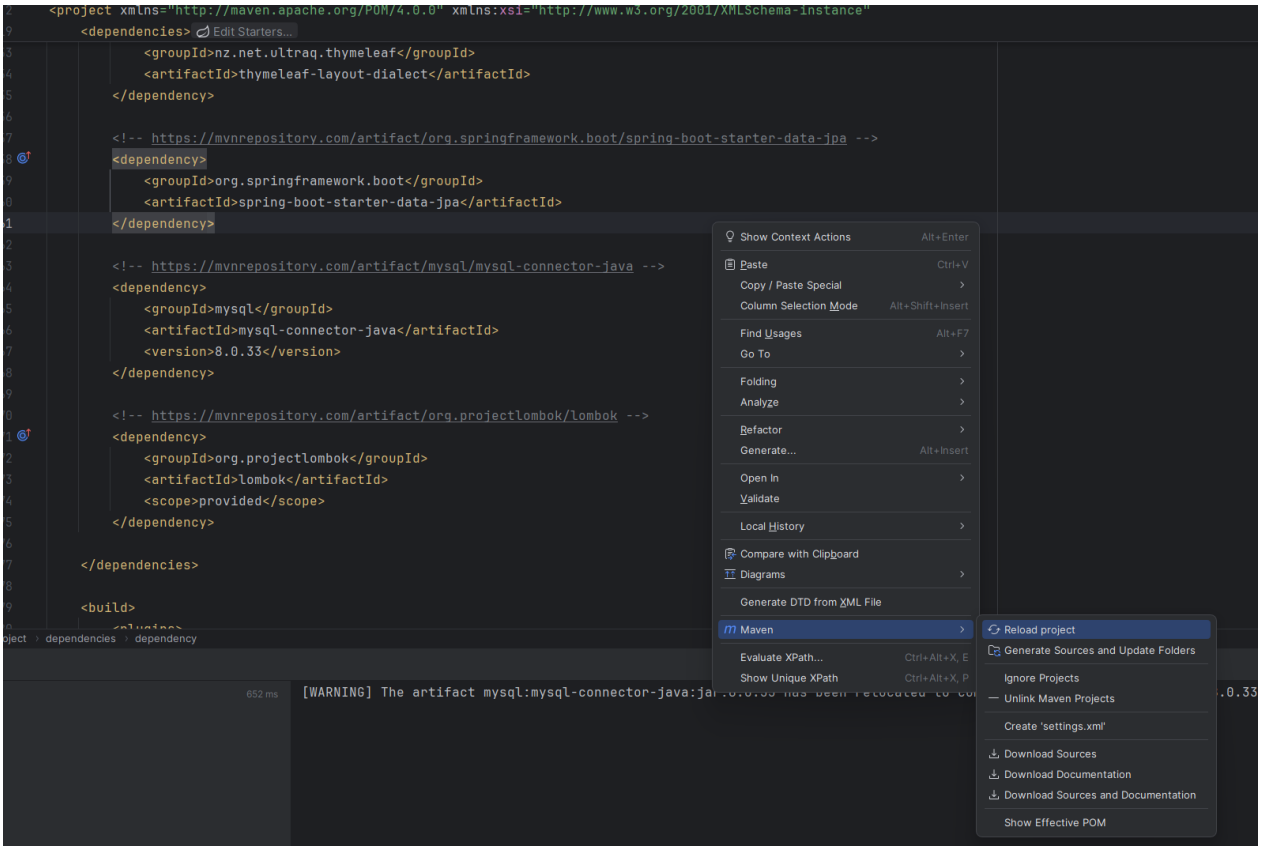
Kết quả như hình bên dưới:



Cuối cùng vào **Setting →Plugin** → Kiểm tra xem **Plugin Lombok** đã được cài đặt hay chưa, nếu chưa chọn và tiến hành cài đặt:



Kết quả sau khi đã thêm các Dependency cần thiết:



4.3 Tạo các entity class cho các bảng:

- Tạo một package mới trong dự án của bạn và đặt tên là `model`.

- Trong package `model`, tạo các class `Product`, `Category`, `Order`, và `OrderDetail`.

- Định nghĩa các thuộc tính và phương thức cho mỗi class theo yêu cầu của bảng tương ứng trong cơ sở dữ liệu.

Ví dụ, class `**Product**` có thể có các thuộc tính như `id`, `name`, `price`, `description`, và `category`.

```
package com.vanlang.webbanhang.model;

public class Product {
    private Long id;
    private String name;
    private double price;
    private String description;
    private Category category;
}
```

Class `**Category**` có thể có thuộc tính như `id` và `name`.

```
package com.vanlang.webbanhang.model;

public class Category {
    private Long id;
    private String name;
}
```

Class `**Order**` có thể có thuộc tính như `id` , `customerName` và danh sách `orderDetails`.

```
package com.vanlang.webbanhang.model;

import java.util.List;

public class Order {
    private Long id;
    private String customerName;
    private List<OrderDetail> orderDetails;
}
```

Class `**OrderDetail**` có thể có thuộc tính như `id` , `quantity` , `product` và `order`.

```
package com.vanlang.webbanhang.model;

public class OrderDetail {
    private Long id;
    private int quantity;
    private Product product;
    private Order order;
}
```

4.4 Ánh xạ các entity với cơ sở dữ liệu

Trong Spring Boot để liên kết các class Java với các bảng trong cơ sở dữ liệu, ta sử dung JPA (Java Persistence API). Mỗi class entity được ánh xạ tới một bảng thông qua các annotation sau:

'@Entity': Khai báo class là một entity, làm đại diện cho một bảng trong cơ sở dữ liệu.

'@Table': Xác định tên của bảng cơ sở dữ liệu mà entity này ánh xạ đến.

Ví dụ, trong class `**Product**`, có thể sử dụng annotation như sau:

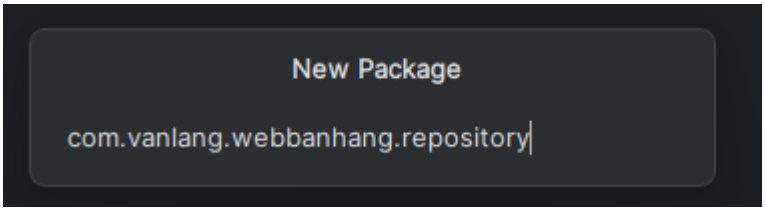
```
@Entity
@Table(name = "products")
public class Product {
    //cac thuộc tính và phương thức của class
}
```

4.5 Tạo các repository interfaces cho các entity

Repository là một pattern thiết kế cho phép bạn trừu tượng hóa logic truy cập dữ liệu. Trong Spring Boot, bạn sẽ tạo các '**repository'** interface cho mỗi entity:

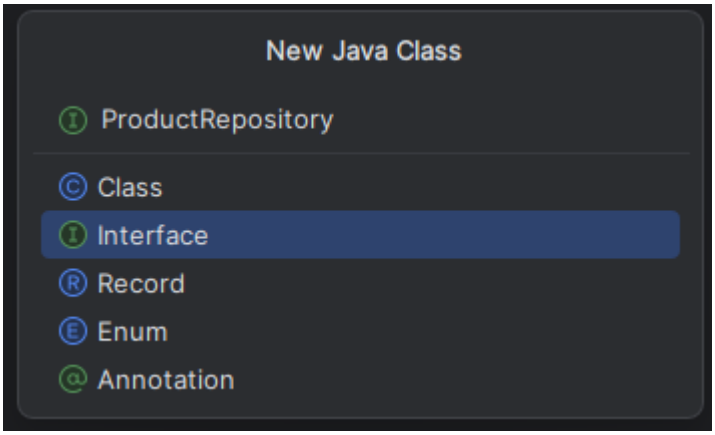
• Tạo một package mới tên là '**repository'**.

Tạo package **repository** tại **src/main/java/com.vanlang.webbanhang**. Nhấn chuột phải vào **com.vanlang.webbanhang**, chọn **New → Package →** đặt tên là **repository**.

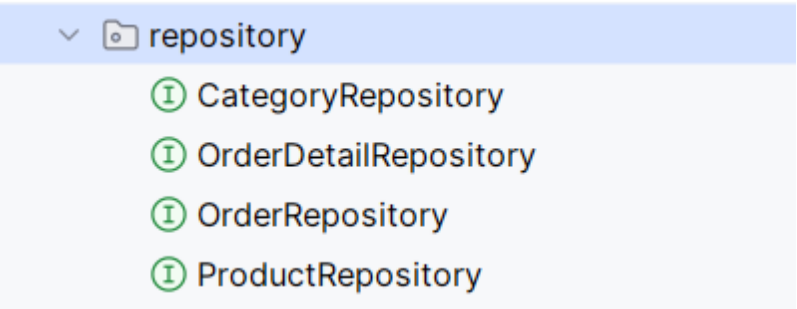


• Trong package này, tạo các interface như '**ProductRepository'**, '**CategoryRepository'**, '**OrderRepository'**, và '**OrderDetailRepository'**.

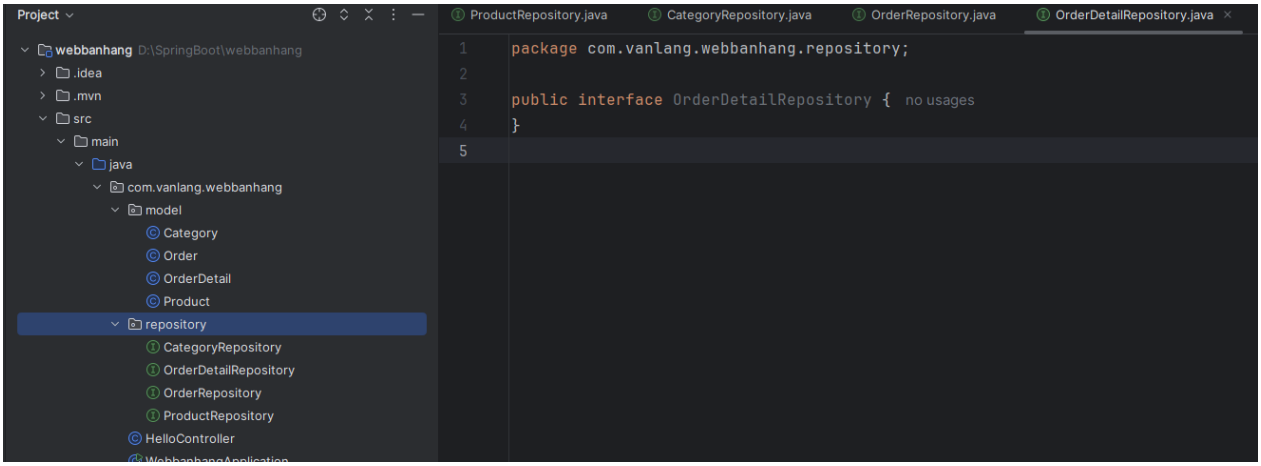
Lưu ý: đây là interface chứ không phải class. Quan sát cho kĩ theo lựa chọn ở hình.



Package sau khi hoàn thành:



Kết quả các bước tạo interface trong **package repository**



• Mỗi interface sẽ kế thừa từ '**JpaRepository**', cung cấp các phương thức **CRUD** sẵn có.

Ví dụ, interface `**ProductRepository**` có thể được định nghĩa như sau:

```
package com.vanlang.webbanhang.repository;

import com.vanlang.webbanhang.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

4.6 Triển khai phương thức CRUD bằng JpaRepository

Kế thừa từ JpaRepository không chỉ giúp cung cấp các phương thức CRUD cơ bản mà còn giúp đảm bảo tính nhất quán và dễ dàng quản lý trong ứng dụng. Annotation @Repository chỉ ra rằng interface đó là một phần của lớp truy cập dữ liệu, cho phép Spring tự động phát hiện và cấu hình bean cho nó.

- Sử dụng annotation `@Repository` trước mỗi interface repository để đánh dấu rằng interface đó là một repository.

- Interface repository sẽ kế thừa từ `JpaRepository` và sử dụng generic type tương ứng với entity tương ứng.

Ví dụ, trong interface `ProductRepository`, bạn có thể sử dụng annotation như sau:

```
@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

4.7 Bổ sung các yêu cầu còn thiếu

Khi đã định nghĩa xong các entity và repositories cơ bản, bước tiếp theo là xác định và ánh xạ các quan hệ giữa các entity. Các quan hệ này giúp phản ánh cấu trúc và mối quan hệ của dữ liệu trong cơ sở dữ liệu.

Xác Định Quan Hệ Giữa Các Entities

• **Many-to-One và One-to-Many**: Thường gặp trong các mô hình dữ liệu nơi một bản ghi có thể liên kết với nhiều bản ghi khác. Ví dụ, một **Order** có thể chứa nhiều **Product** và mỗi **Product** có thể thuộc về một **Category**.

• Sử dụng @**ManyToOne** để chỉ định quan hệ nhiều đến một. Áp dụng annotation này trên thuộc tính trong class con.

• Sử dụng @**OneToMany** để chỉ định quan hệ một đến nhiều. Áp dụng annotation này trên thuộc tính trong class chính.

Tùy Chỉnh Repository Interfaces

• **Phương thức tùy chỉnh**: Đôi khi, các phương thức CRUD cơ bản không đủ để xử lý các yêu cầu phức tạp hoặc logic kinh doanh đặc thù của ứng dụng.

• Có thể thêm các phương thức tùy chỉnh vào repository interfaces để thực hiện truy vấn đặc biệt hoặc xử lý các nghiệp vụ cụ thể.

• Sử dụng @**Query** để định nghĩa các truy vấn SQL hoặc JPQL trực tiếp trong interface.

Với các bước trên đã thiết lập cơ bản và tùy biến cho các entity class và repository interfaces của mình, đảm bảo chúng được ánh xạ chính xác với cơ sở dữ liệu trong dự án Spring Boot.

4.8 Hướng dẫn Code mẫu

Code mẫu trong package model với các class: **Product, Category, Order, OrderDetail**

4.8.1 Entity class `Product`:

```
package com.vanlang.webbanhang.model;

import jakarta.persistence.*;
import lombok.*;

@Setter
```

```
@Getter
@RequiredArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "products")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private double price;
    private String description;

    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;
}
```

4.8.2 Entity class `Category`:

```
package com.vanlang.webbanhang.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import lombok.*;

@Setter
@Getter
@RequiredArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "categories")
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank(message = "Tên là bắt buộc")
    private String name;
}
```

4.8.3 Entity class `Order`:

```
package com.vanlang.webbanhang.model;

import java.util.List;
import jakarta.persistence.*;
import lombok.*;
```

```
@Setter
@Getter
@RequiredArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "orders")
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String customerName;

    @OneToMany(mappedBy = "order")
    private List<OrderDetail> orderDetails;
}
```

4.8.4 Entity class `OrderDetail`:

```
package com.vanlang.webbanhang.model;

import jakarta.persistence.*;
import lombok.*;

@Setter
@Getter
@RequiredArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "order_details")
public class OrderDetail {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private int quantity;

    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product product;

    @ManyToOne
    @JoinColumn(name = "order_id")
    private Order order;
}
```

Sau đây là code mẫu:

4.8.5 Repository interface `ProductRepository`:

```
package com.vanlang.webbanhang.repository;
```

```
import com.vanlang.webbanhang.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

4.8.6 Repository interface `CategoryRepository`:

```
package com.vanlang.webbanhang.repository;

import com.vanlang.webbanhang.model.Category;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CategoryRepository extends JpaRepository<Category, Long> {
}
```

4.8.7 Repository interface `OrderRepository`:

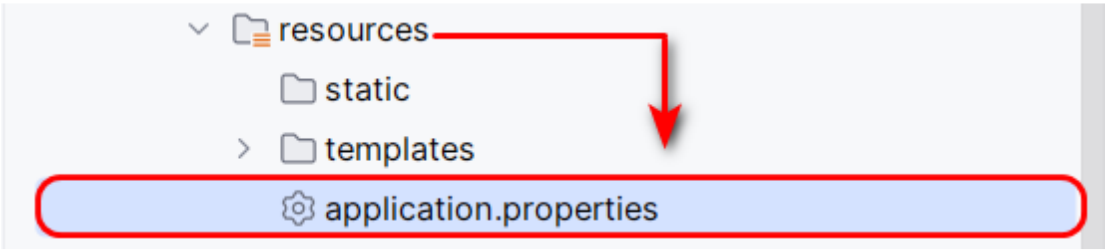
```
package com.vanlang.webbanhang.repository;

import com.vanlang.webbanhang.model.Order;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface OrderRepository extends JpaRepository<Order, Long> {
}
```

4.8.8 Repository interface `OrderDetailRepository`:

```
package com.vanlang.webbanhang.repository;

import com.vanlang.webbanhang.model.OrderDetail;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface OrderDetailRepository extends JpaRepository<OrderDetail, Long> {
}
```

4.8.9 Cấu hình file application.properties

File **application.properties** nằm trong thư mục **resources:**



Cấu hình trong **application.properties** để kết nối database **MySQL** trong **Workbench**:

Link thư viện: https://spring.io/guides/gs/accessing-data-mysql

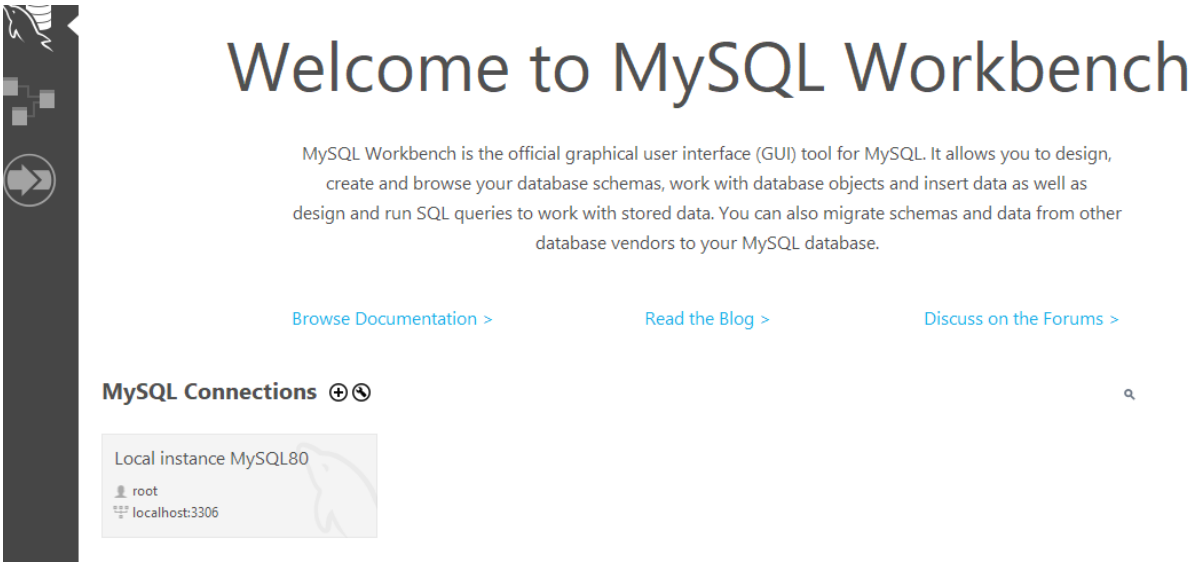
Code mẫu

```
spring.application.name=webbanhang

#Database connection properties
spring.datasource.url=jdbc:mysql://localhost:3306/webbanhang
spring.datasource.username=root
spring.datasource.password=123456
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql=true

#Hibernate properties
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

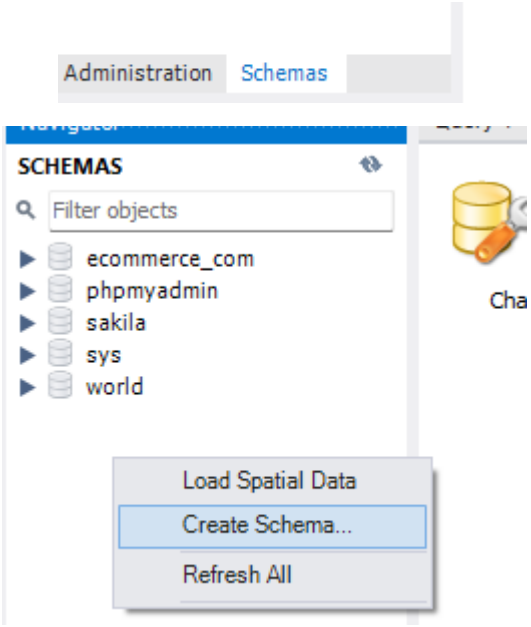
Tiến hành khởi tạo database webbanhang trong MySQL:

Bước 1: Mở MySQL trên Workbench

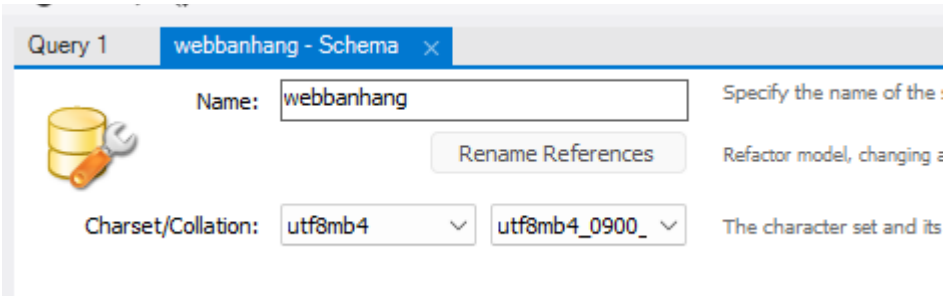
Bước 2: Cấu hình và khởi động MySQL. Cấu hình như sau:



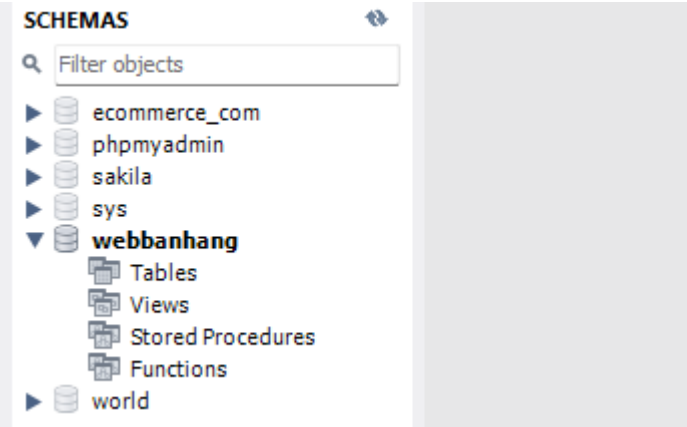
Bước 3: Tiến hành tạo database. Chọn vào Schemas → Chọn Create Schema → Database:



Ví dụ trường hợp này là **Database** tên **webbanhang**:

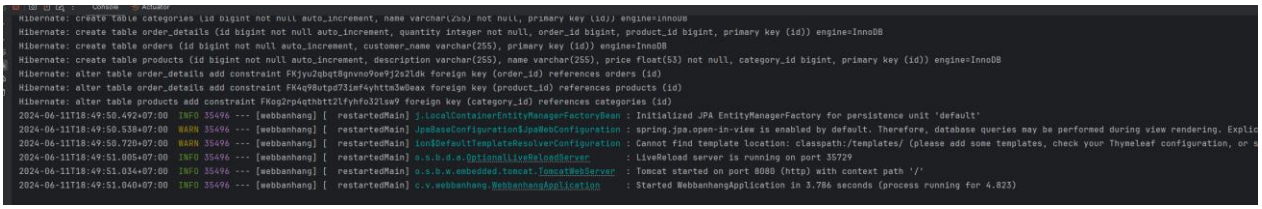


Kết quả khởi tạo thành công database '**WebBanHang**'



Bước 4: Tiến hành khởi tạo dự án trong Intellij IDEA và kiểm tra kết nối database:

Dự án Build thành công:



Kiểm tra kết nối Database MySQL trong Workbench: