

Spring boot

1. Annotation:

- Giúp cấu hình và điều khiển hành vi của ứng dụng
- **@SpringBootApplication**
 - **Ý nghĩa:** Đây là annotation chính để đánh dấu lớp chính của ứng dụng Spring Boot. Nó kết hợp ba annotation quan trọng: **@Configuration**, **@EnableAutoConfiguration**, và **@ComponentScan**.
 - **Sử dụng:** Được sử dụng trên lớp chính của ứng dụng, nơi có phương thức **main** để khởi chạy ứng dụng Spring Boot.
- **@RestController**
 - **Ý nghĩa:** Được sử dụng để đánh dấu lớp là một controller trong Spring MVC và cho phép trả về các dữ liệu (thay vì view) từ các phương thức của controller.
 - **Sử dụng:** Thường được kết hợp với các phương thức xử lý HTTP requests
 - **Annotation Trong Spring Data JPA**
 - **@Entity:** Đánh dấu một lớp là thực thể của cơ sở dữ liệu, tương ứng với một bảng.
 - **@Table:** Xác định tên bảng trong cơ sở dữ liệu mà thực thể sẽ ánh xạ đến.
 - **@Id:** Đánh dấu trường là khóa chính của thực thể.
 - **@GeneratedValue:** Chỉ định chiến lược tự động sinh giá trị cho khóa chính.

- **@Column**: Chỉ định các thuộc tính của cột trong bảng.

-

Annotation Trong Spring Security

- **@EnableWebSecurity**: Kích hoạt bảo mật cho ứng dụng web.
- **@Secured**: Đảm bảo rằng chỉ các người dùng với vai trò được chỉ định mới có thể truy cập phương thức.
- **@PreAuthorize**: Phân quyền truy cập vào các phương thức dựa trên biểu thức SpEL.

- Annotation Trong Spring MVC

- **@RestController**: Đánh dấu lớp là một controller trong ứng dụng RESTful. Tương đương với **@Controller** và **@ResponseBody**.
- **@RequestMapping**: Chỉ định các URL mà phương thức controller sẽ xử lý. Có thể sử dụng các biến thể như **@GetMapping**, **@PostMapping**, **@PutMapping**, và **@DeleteMapping** để chỉ định phương thức HTTP cụ thể.
- **@PathVariable**: Lấy giá trị từ phần của URL được chỉ định.
- **@RequestParam**: Lấy giá trị từ tham số của query string trong URL.
- **@RequestBody**: Chuyển đổi dữ liệu từ request body thành đối tượng Java.
- **@ResponseBody**: Chỉ định rằng giá trị trả về của phương thức sẽ được viết vào response body thay vì được render vào một view.

Các annotation bổ sung

1. RequiredArgsConstructor

- + là một annotation của Lombok được sử dụng để tự động tạo một constructor cho các trường dữ liệu (`fields`) của lớp mà không được khởi tạo trong lớp và được đánh dấu là `final` hoặc `@NonNull`.
- + Khi bạn sử dụng `@RequiredArgsConstructor`, Lombok sẽ tự động tạo một constructor mà tất cả các trường `final` (hoặc được đánh dấu `@NonNull`) đều được khởi tạo thông qua constructor đó.
- + Điều này giúp giảm bớt việc viết mã boilerplate cho các constructor.
- + Sử dụng từ khóa `final` cho các biến phụ thuộc giúp đảm bảo rằng các phụ thuộc không thể bị thay đổi sau khi được khởi tạo, tăng cường tính an toàn và ổn định của ứng dụng.

```
@RequiredArgsConstructor

public class CategoryController {

    private final CategoryService categoryService;
```

2. `@PrePersist` là một annotation JPA được sử dụng để thực hiện các hành động trước khi thực thể được lưu vào cơ sở dữ liệu.

+@PrePersist đảm bảo rằng phương thức onCreate() sẽ được gọi ngay trước khi thực thể được chèn vào cơ sở dữ liệu. Đây là một phần trong vòng đời của thực thể JPA, và nó cho phép bạn thực hiện các hành động cần thiết trước khi dữ liệu được lưu.

3. Annotation @MappedSuperclass trong JPA (Java Persistence API)

- + Được sử dụng để đánh dấu một lớp mà không phải là một thực thể JPA (entity) nhưng chứa các thuộc tính và phương thức chung mà bạn muốn các thực thể khác kế thừa.
- + Đây là một phần trong chiến lược kế thừa của JPA và giúp chia sẻ các thuộc tính giữa các thực thể mà không cần phải tạo ra các bảng riêng cho lớp cha.

Kiến trúc mô hình spring boot

1. Kiến trúc 3 layer, hay còn gọi là mô hình 3 lớp, là một kiến trúc phát triển phần mềm phổ biến trong lập trình ứng dụng. Kiến trúc này bao gồm 3 tầng (layer) logic được phân chia rõ ràng và độc lập với nhau:
 - **Presentation layer (tầng trình diện)**: Là tầng giao diện người dùng, được sử dụng để hiển thị dữ liệu và tương tác với người dùng. Tầng này sử dụng các công nghệ như HTML, CSS, JavaScript, JSP, thư viện front-end như Angular hoặc React, và cung cấp các chức năng như xác thực người dùng, kiểm tra đầu vào và hiển thị dữ liệu.
 - **Business Logic layer (tầng xử lý logic)**: Là tầng xử lý logic của ứng dụng, bao gồm các quy trình kinh doanh và tính năng của ứng dụng. Tầng này thực hiện các tác vụ xử lý dữ liệu, xử lý nghiệp vụ, xử lý lỗi, và tạo ra các kết quả phù hợp. Tầng này sử dụng các công nghệ Java, như Spring Boot, để triển khai logic của ứng dụng.
 - **Data Access layer (tầng truy cập dữ liệu)**: Là tầng truy cập cơ sở dữ liệu, được sử dụng để lưu trữ và truy xuất dữ liệu. Tầng

này sử dụng các công nghệ như JDBC, Hibernate, Spring Data JPA để tương tác với cơ sở dữ liệu.

2. Luồng đi

- **Nhận yêu cầu HTTP** từ client.
- **Controller** xử lý yêu cầu và có thể gọi các services
- **Service Layer** xử lý logic nghiệp vụ và tương tác với repository nếu cần.
- **Repository Layer** thực hiện thao tác với cơ sở dữ liệu.
- **Trả về phản hồi** cho client.

3. DTO

- DTO (Data Transfer Object) là một mẫu thiết kế được sử dụng để truyền dữ liệu giữa các lớp hoặc qua các lớp của ứng dụng, đặc biệt là giữa tầng controller và tầng service hoặc tầng dữ liệu. DTO thường được sử dụng để:
 1. **Tách Biệt Dữ Liệu:** Tách biệt dữ liệu mà bạn gửi qua mạng hoặc từ client, với cách dữ liệu được lưu trữ trong cơ sở dữ liệu. Điều này cho phép bạn tạo ra các lớp dữ liệu tùy chỉnh phù hợp với nhu cầu của giao diện người dùng hoặc API mà không cần phải phụ thuộc vào cấu trúc của entity.
 2. **DTO (Data Transfer Object)** là một mẫu thiết kế dùng để chuyển dữ liệu giữa các lớp hoặc các tầng trong ứng dụng. DTO thường được sử dụng để giao tiếp giữa lớp dịch vụ và lớp điều khiển (controller), hoặc giữa lớp điều khiển và lớp dữ liệu.
 3. giúp làm sạch và phân tách các lớp trong ứng dụng của bạn, giữ cho mã của bạn dễ bảo trì và mở rộng hơn.

Phần cuối: ResponseEntity

- **ResponseEntity** là một lớp trong Spring Framework được sử dụng để xây dựng phản hồi HTTP trong các ứng dụng Spring Boot. Nó cho phép bạn kiểm soát chi tiết hơn về cách phản hồi từ máy chủ được gửi đến client. **ResponseEntity** là một phần quan trọng trong việc xử lý các yêu cầu HTTP và có thể được sử dụng để:
 - **Thiết lập mã trạng thái HTTP:** Bạn có thể chỉ định mã trạng thái HTTP cụ thể cho phản hồi, như 200 OK, 404 Not Found, 500 Internal Server Error, v.v.

- **Thêm các tiêu đề HTTP:** Bạn có thể thêm hoặc sửa đổi các tiêu đề HTTP trong phản hồi, như Content-Type, Authorization, Cache-Control, v.v.
- **Chứa dữ liệu phản hồi:** Bạn có thể đóng gói dữ liệu mà bạn muốn gửi đến client trong phản hồi HTTP.
 - **Cấu trúc của ResponseEntity**
- ResponseEntity có thể chứa ba thành phần chính:
- **Mã trạng thái HTTP:** Ví dụ: 200 OK, 404 Not Found, 500 Internal Server Error.
- **Tiêu đề HTTP:** Ví dụ: Content-Type, Location, Cache-Control.
- **Thân phản hồi (Body):** Dữ liệu thực tế mà bạn muốn gửi đến client.
- **Các phương thức chính của ResponseEntity**
 - **ResponseEntity.ok():** Trả về mã trạng thái 200 OK với thân phản hồi.
 - **ResponseEntity.status(HttpStatus status):** Chỉ định mã trạng thái HTTP cụ thể cho phản hồi.
 - **ResponseEntity.header(String headerName, String headerValue):** Thêm hoặc thay đổi tiêu đề HTTP.
 - **ResponseEntity.body(T body):** Chỉ định dữ liệu thân phản hồi.



Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy

☒ Maven

Spring Boot

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M2) ☐ 3.3.4 (SNAPSHOT) ☒ 3.3.3

☐ 3.2.10 (SNAPSHOT) ☐ 3.2.9

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 22 ☐ 21 ☒ 17

Language

Dependencies

[ADD DEPENDENCIES...](#) CTRL + B

MySQL Driver SQL

MySQL JDBC driver.

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Lombok DEVELOPER TOOLS

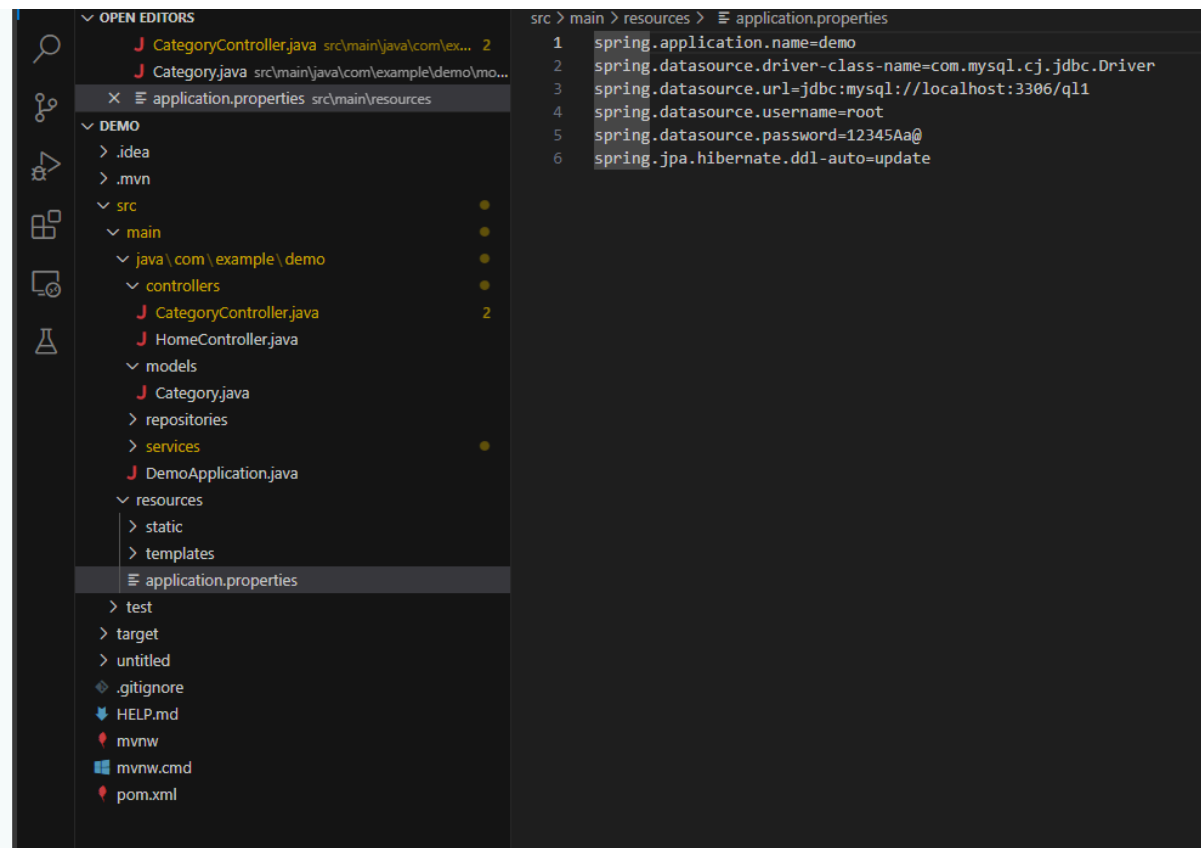
Java annotation library which helps to reduce boilerplate code.

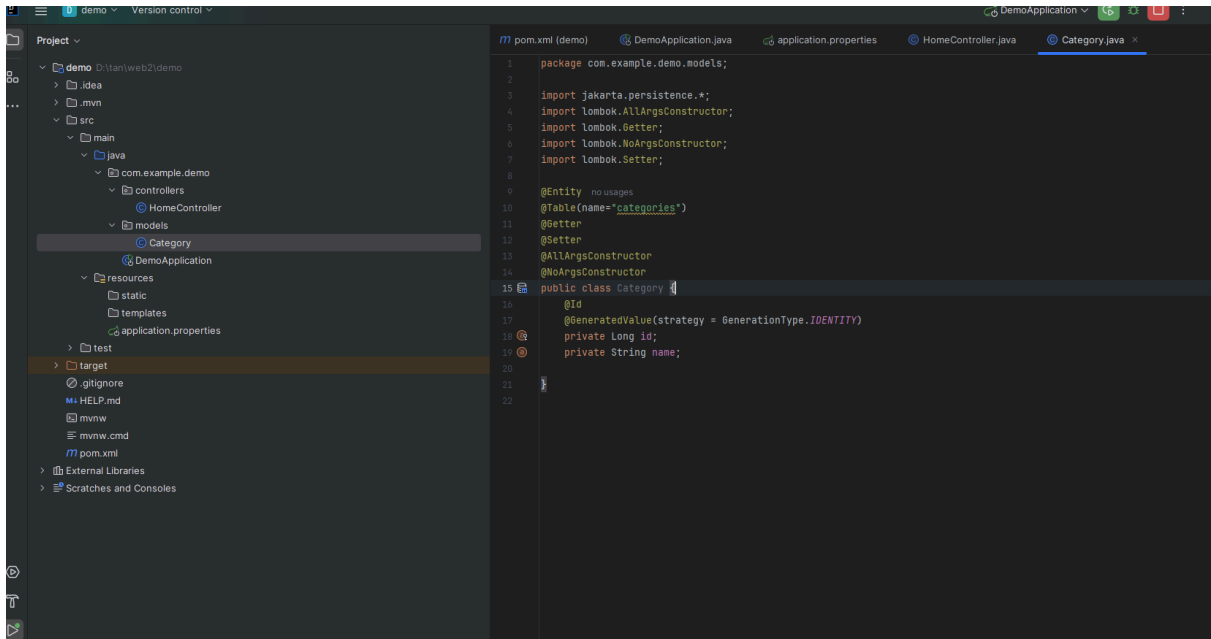
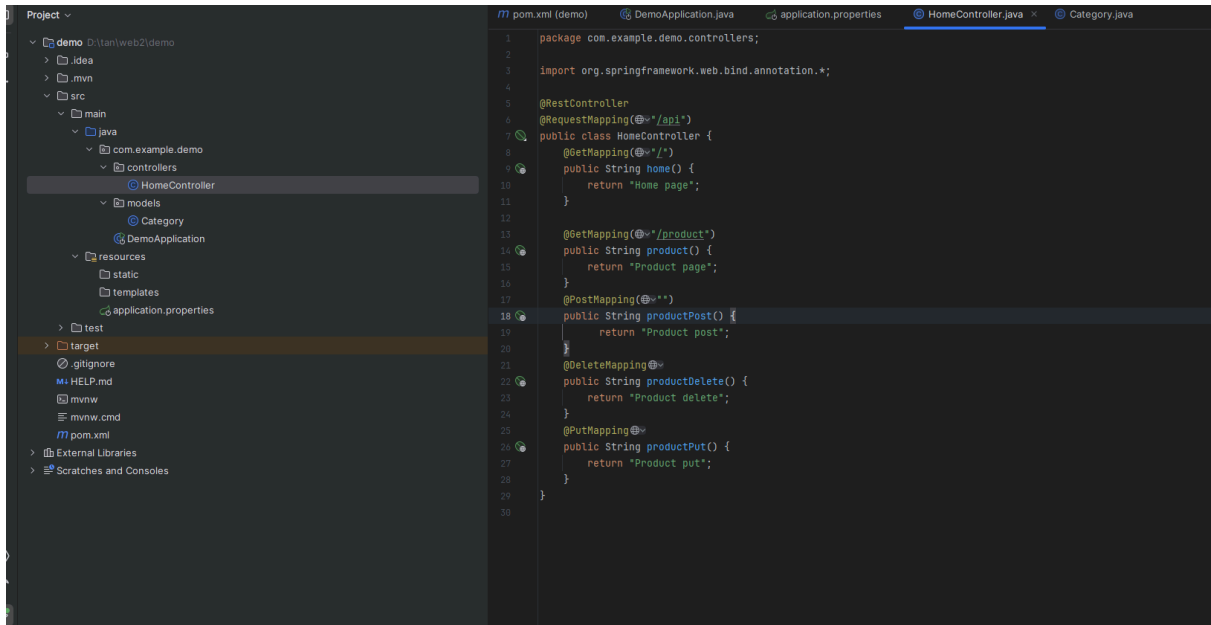


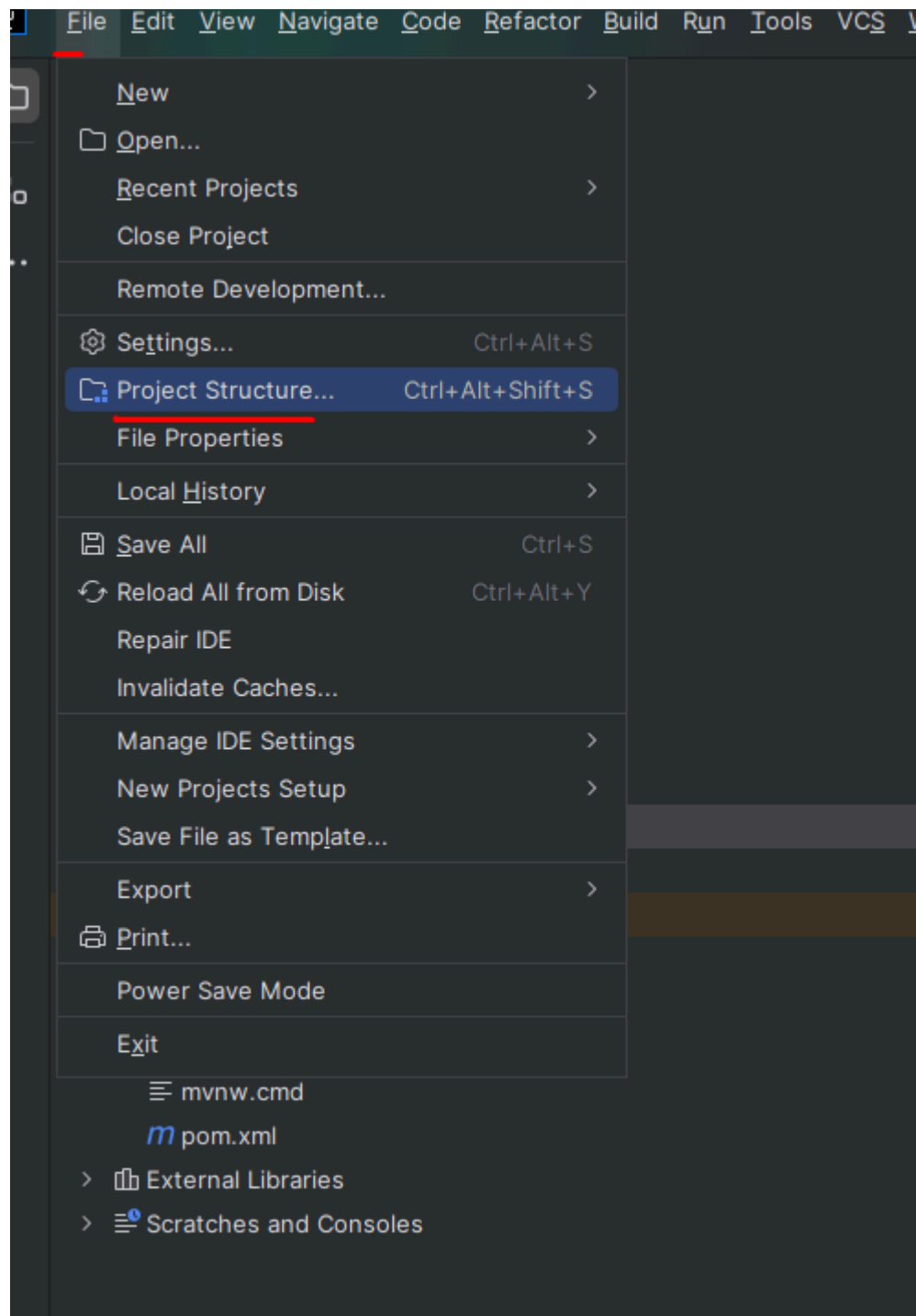
[GENERATE](#) CTRL + G

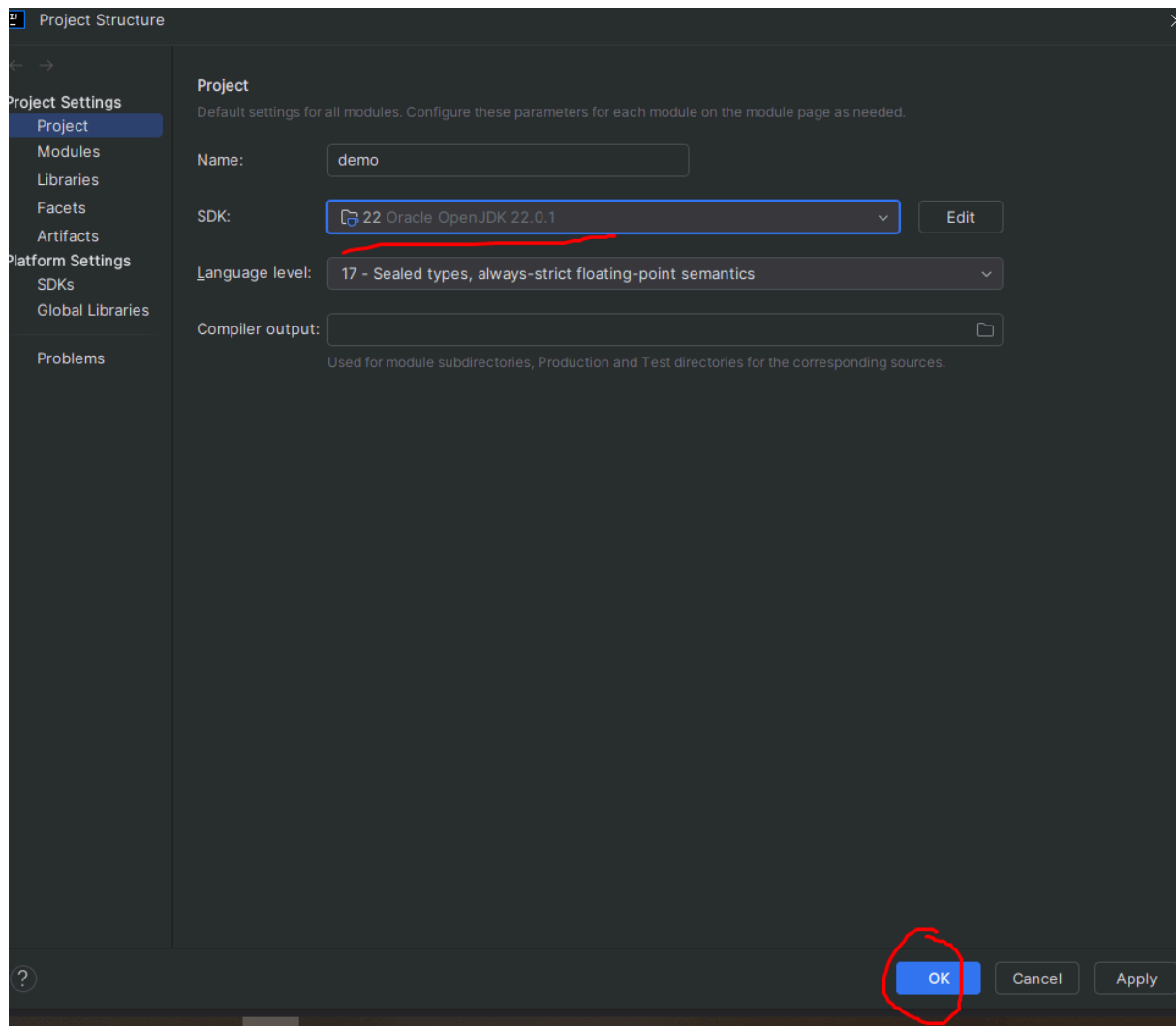
[EXPLORE](#) CTRL + SPACE

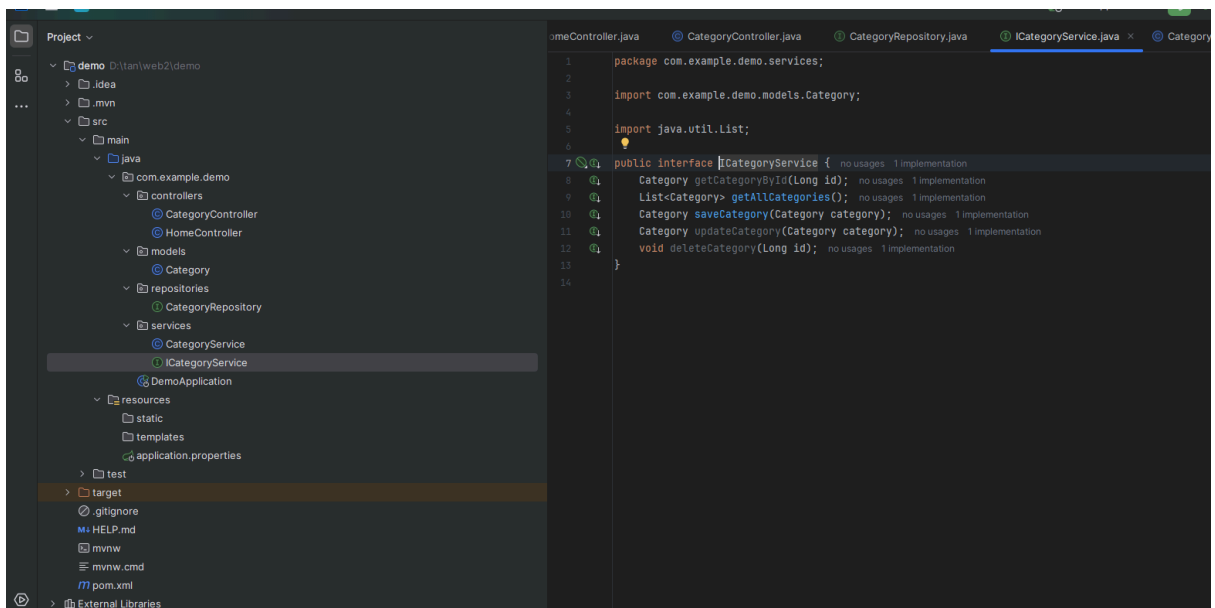
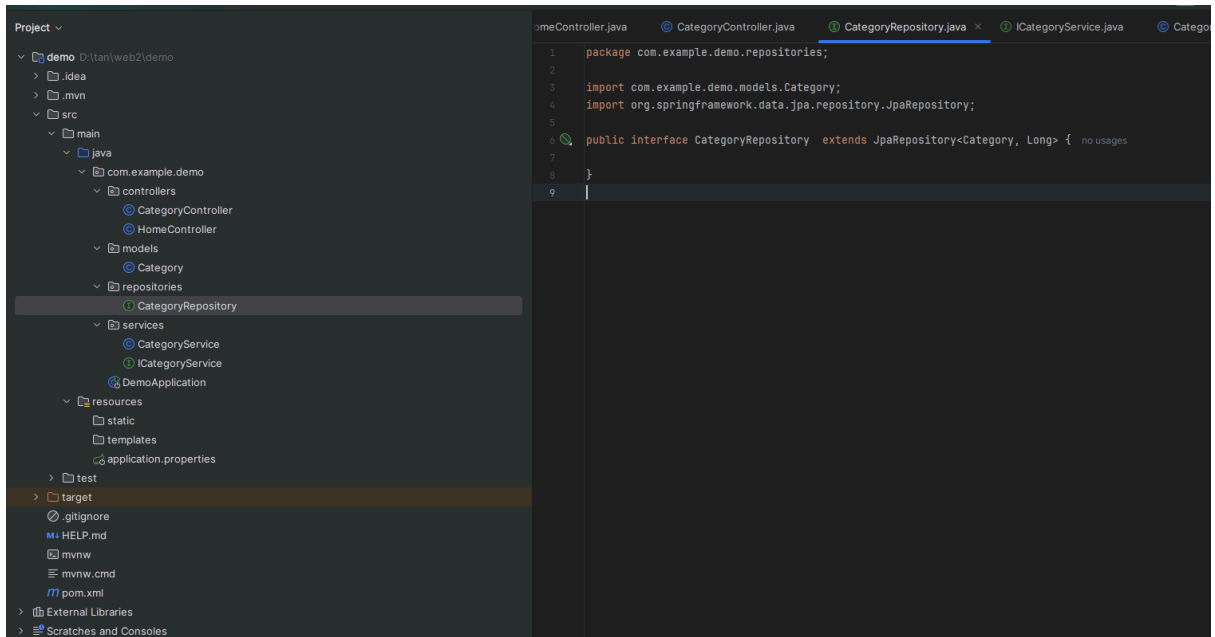
[SHARE...](#)

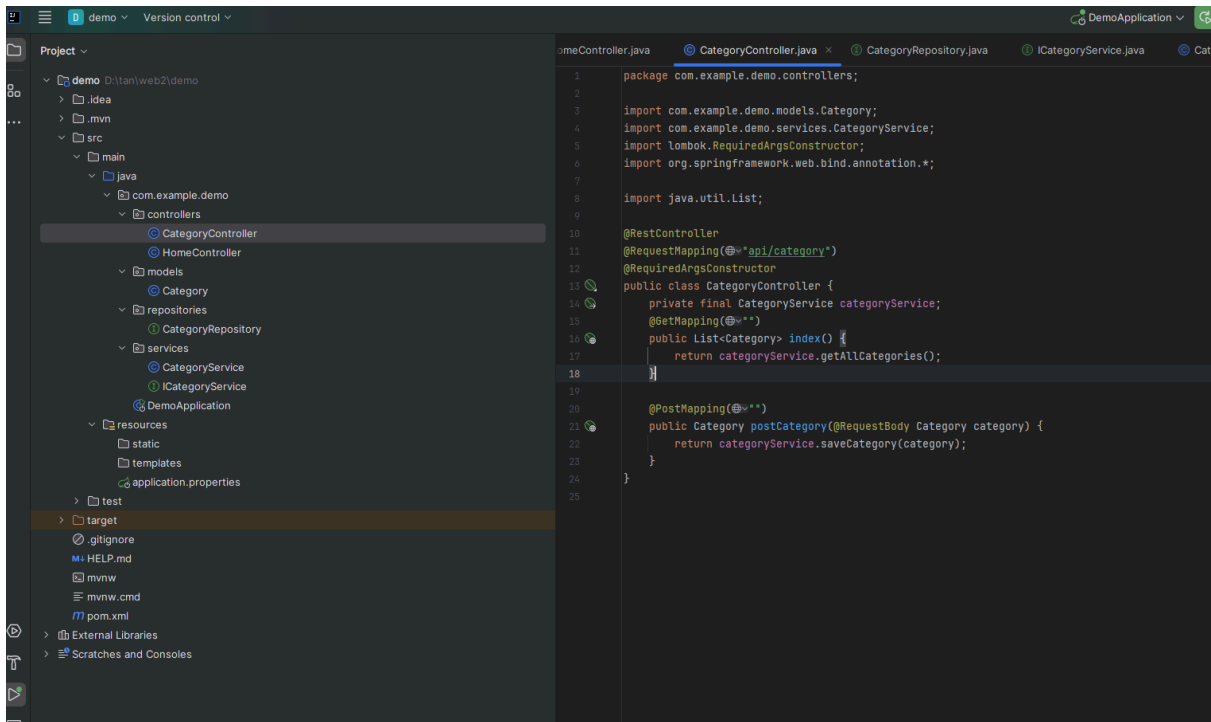
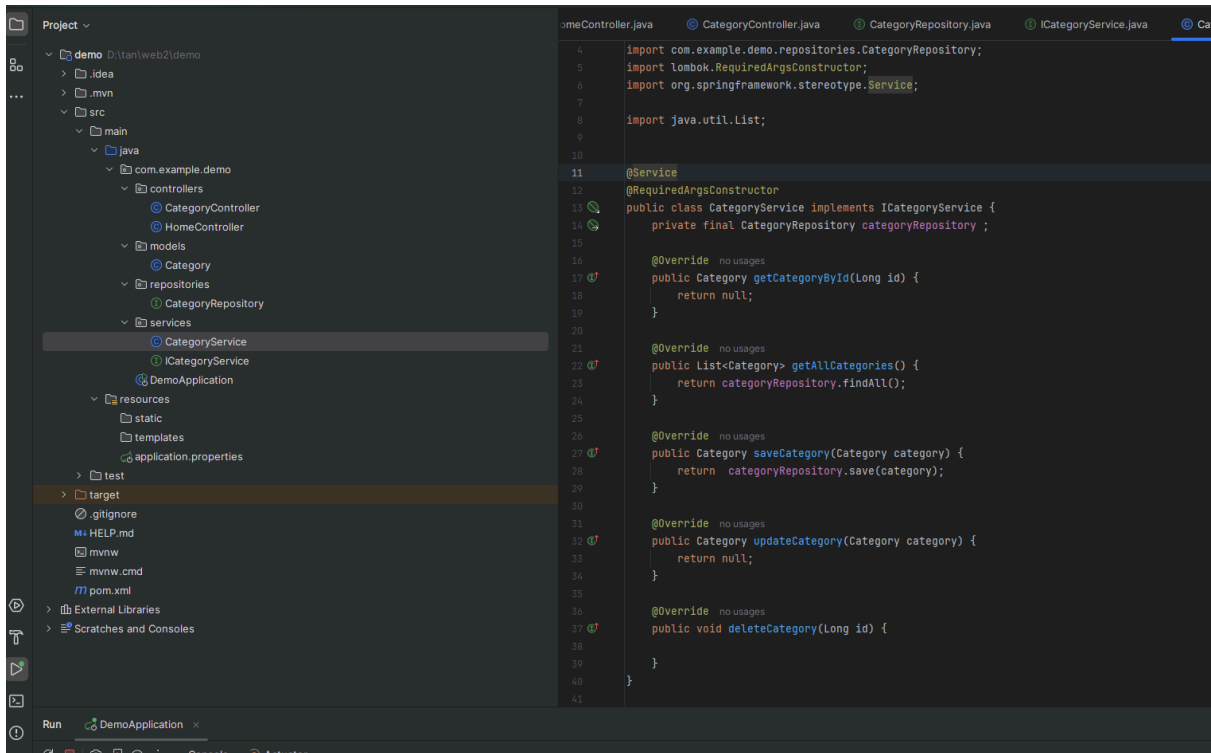


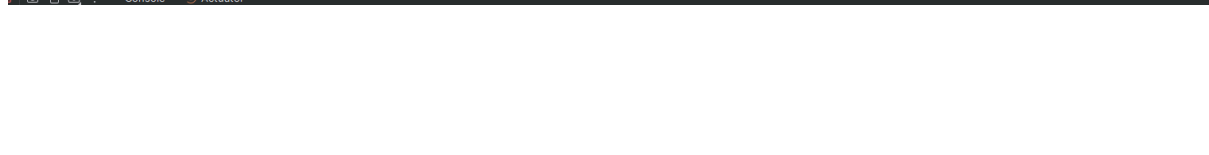


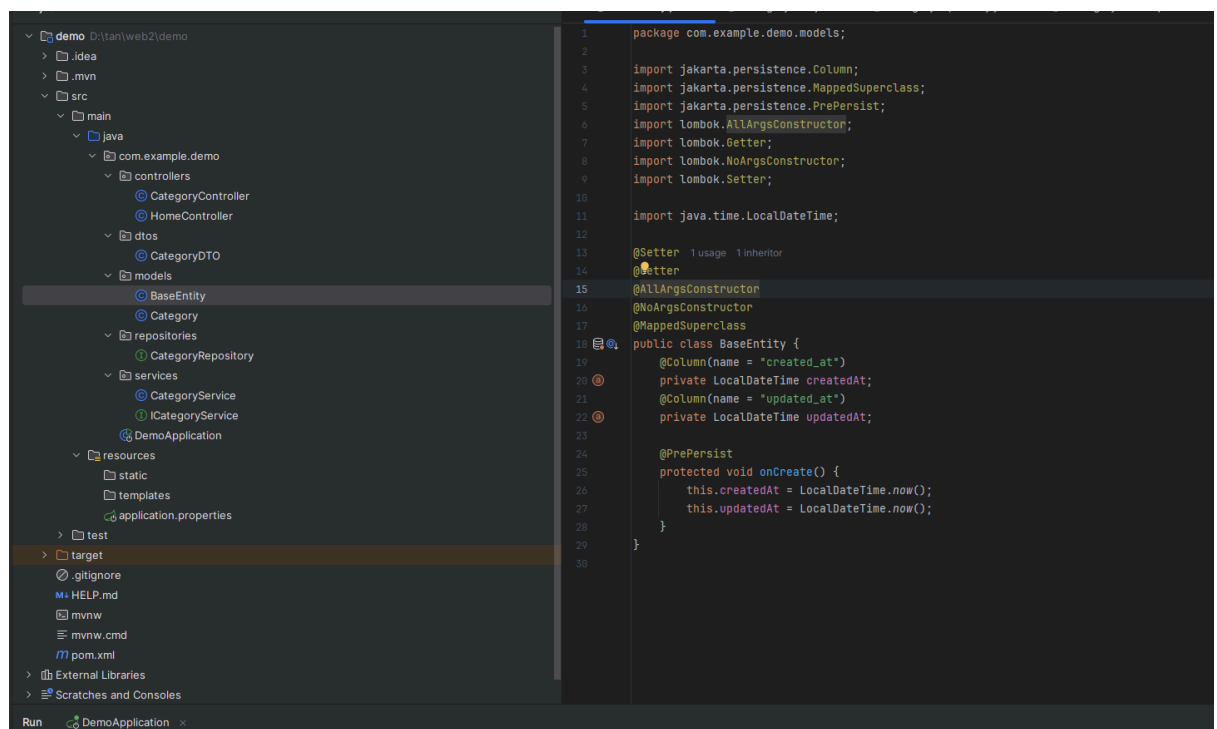
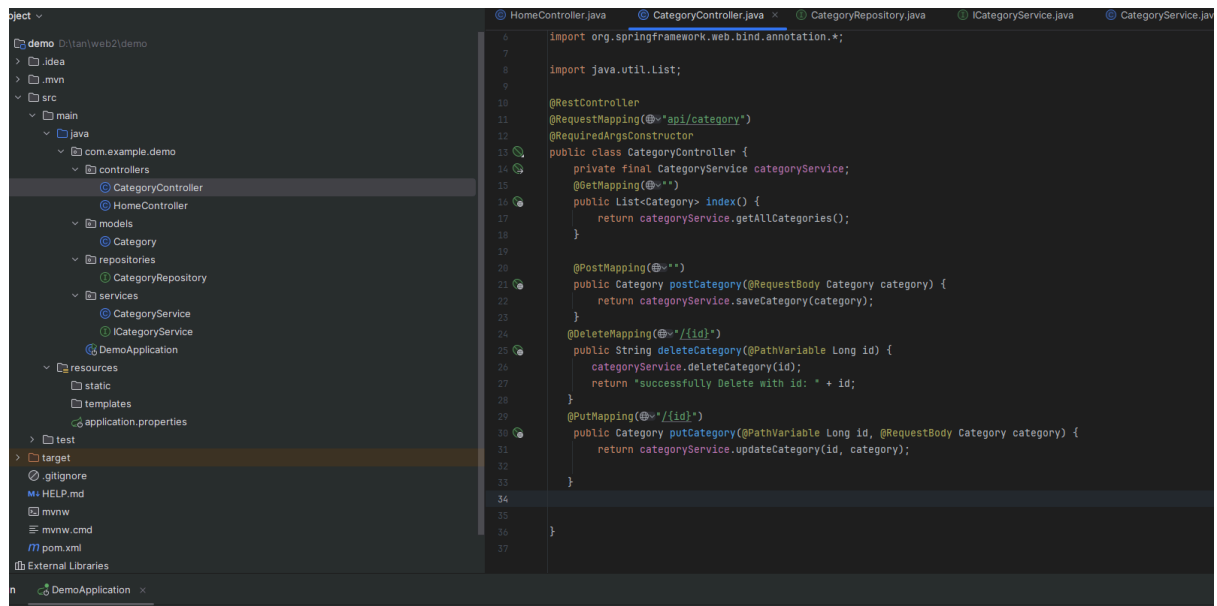


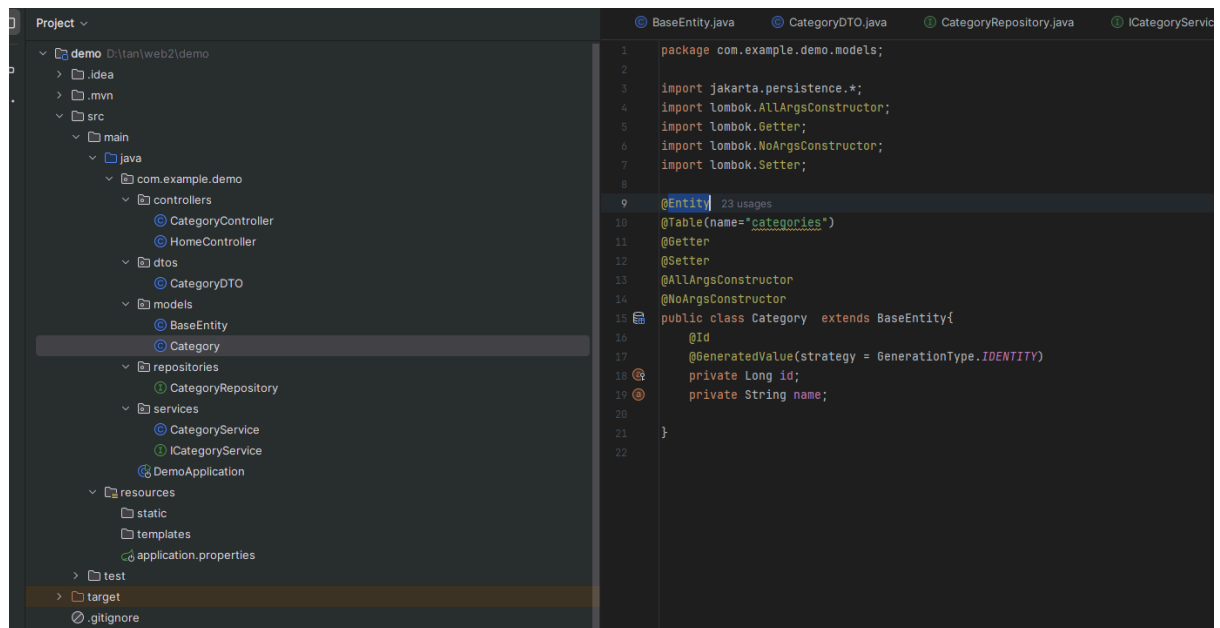




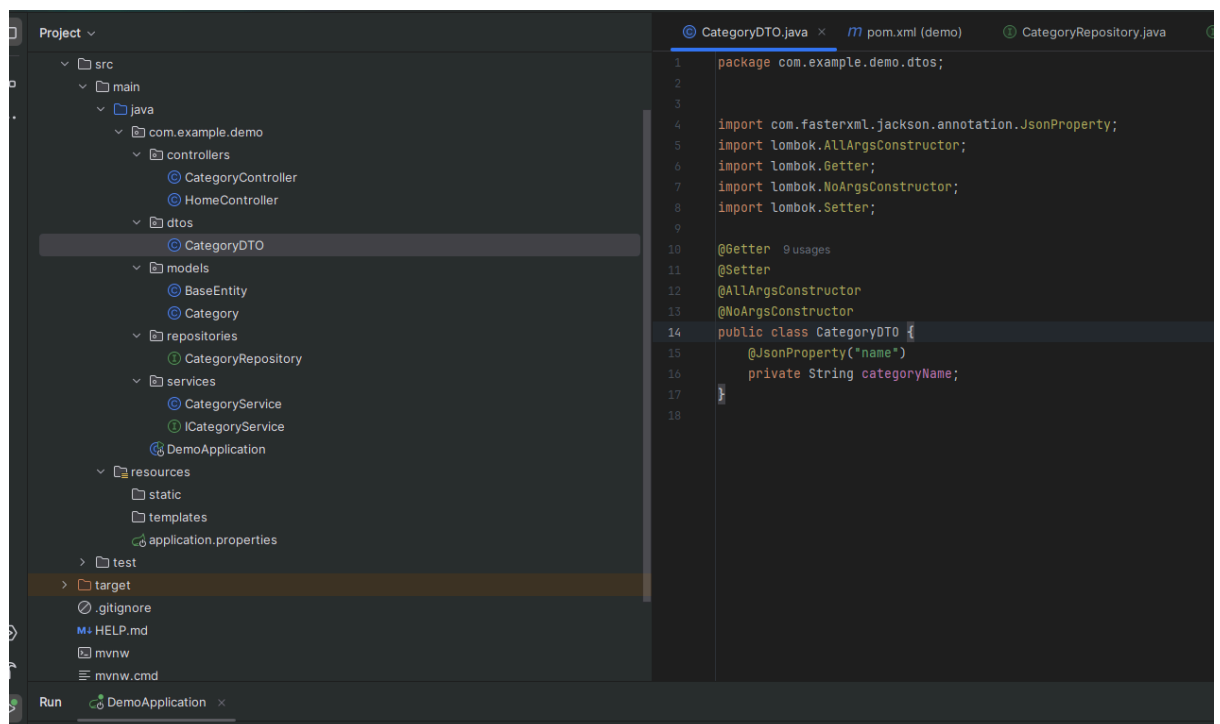


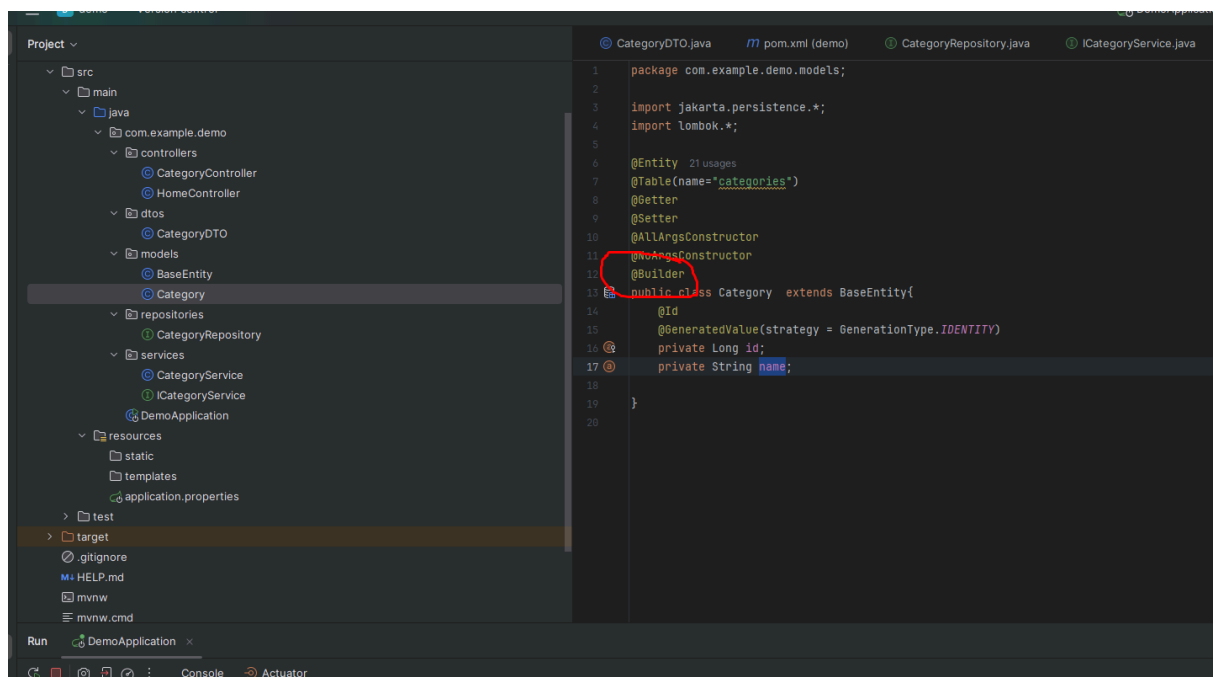
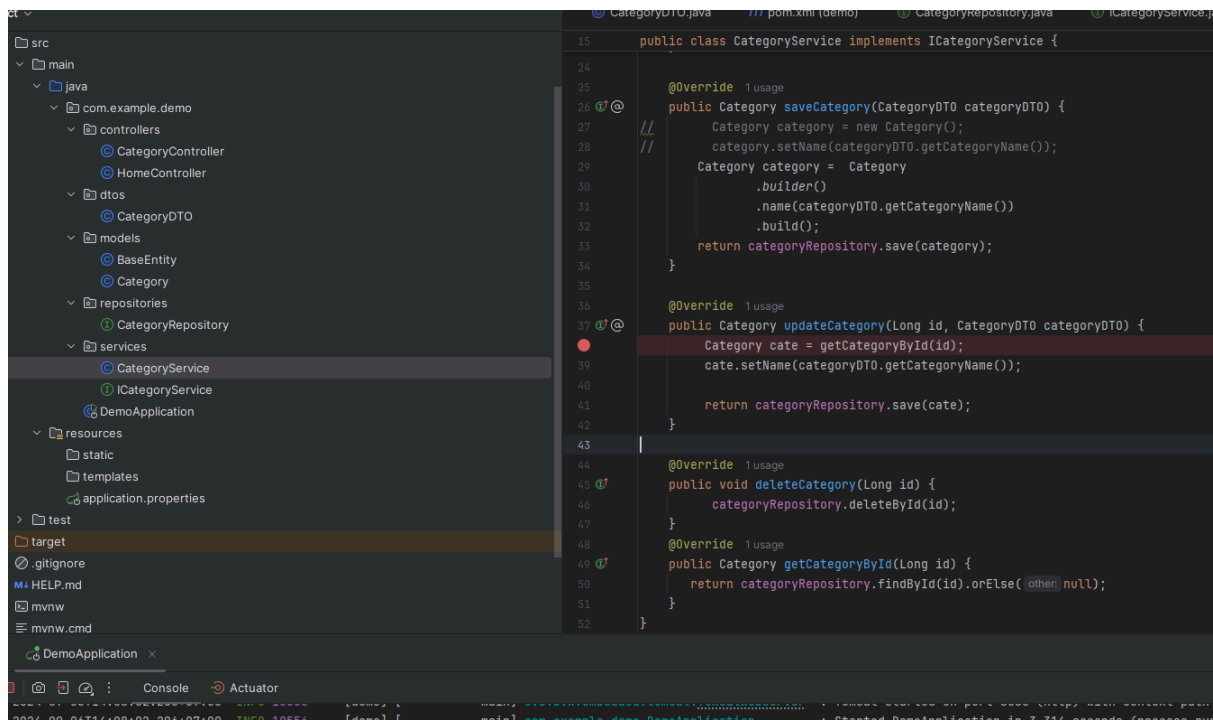






Thực hành DTO






```

40         return successfullyDeleteWithId(id);
41     }
42
43     @PostMapping("/{id}")
44     public Category putCategory(@PathVariable Long id, @RequestBody CategoryDTO categoryDTO) {
45         return categoryService.updateCategory(id, categoryDTO);
46     }
47
48
49 }
50

```

Validation

https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation/3.3.3

MVN REPOSITORY Search for groups, artifacts, categories

Indexed Artifacts (41.1M)

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JVM Languages
- JSON Libraries
- Language Runtime
- Core Utilities
- Mocking
- Web Assets
- Annotation Libraries
- HTTP Clients
- Logging Bridges
- Dependency Injection
- XML Processing
- Web Frameworks
- I/O Utilities
- Defect Detection Metadata
- Code Generators
- Configuration Libraries
- OSGi Utilities
- Concurrency Libraries
- Reflection Libraries

Home » org.springframework.boot » spring-boot-starter-validation » 3.3.3

Spring Boot Starter Validation » 3.3.3
Starter for using Java Bean Validation with Hibernate Validator

License: Apache 2.0

Categories: Validation Libraries

Tags: spring framework starter validation

Organization: VMware, Inc.

HomePage: https://spring.io/projects/spring-boot

Date: Aug 22, 2024

Files: pom (2 KB) jar (4 KB) View All

Repositories: Central

Ranking: #208 in MvnRepository (See Top Artifacts)
#3 in Validation Libraries

Used By: 2,515 artifacts

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-validation</artifactId>
  <version>3.3.3</version>
</dependency>

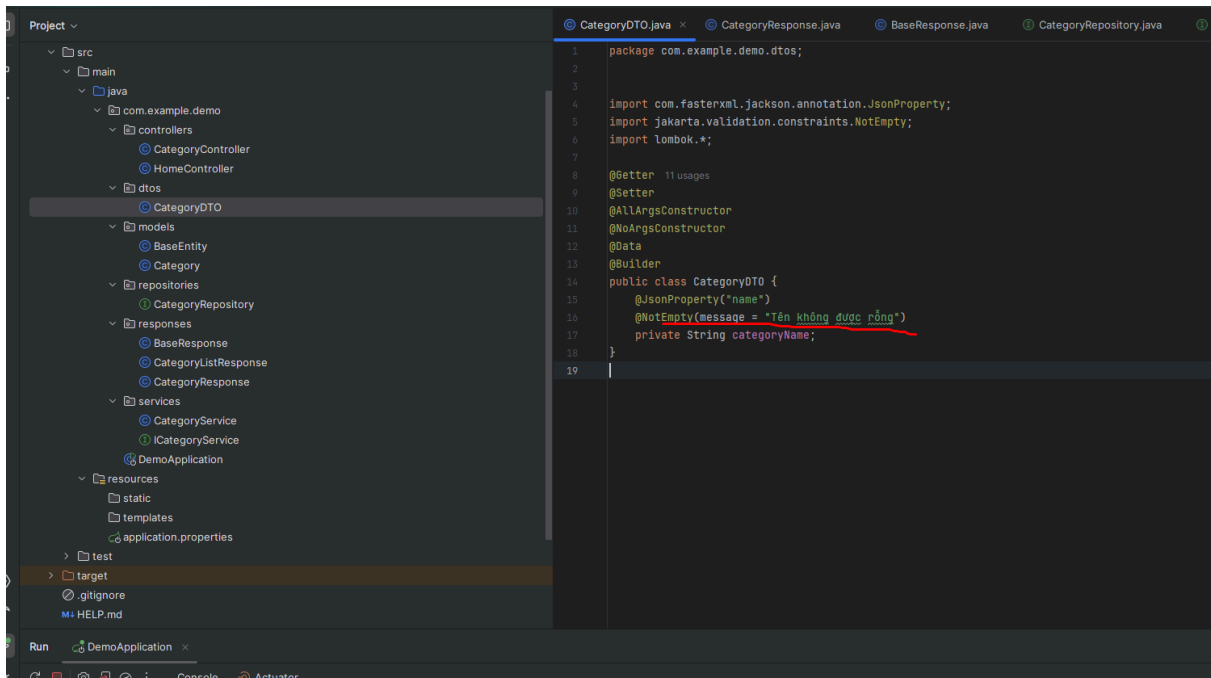
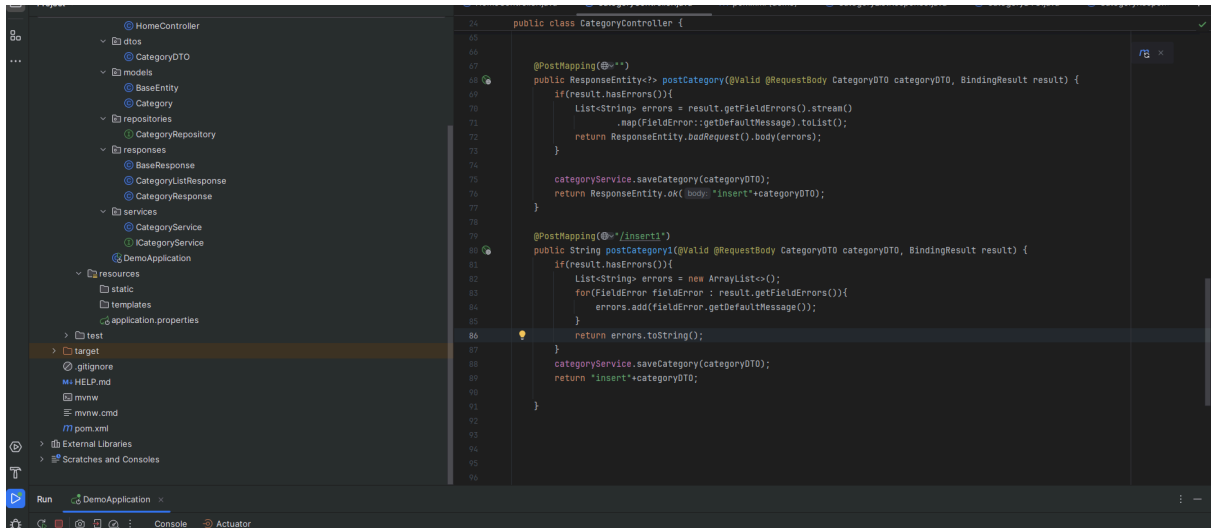
```

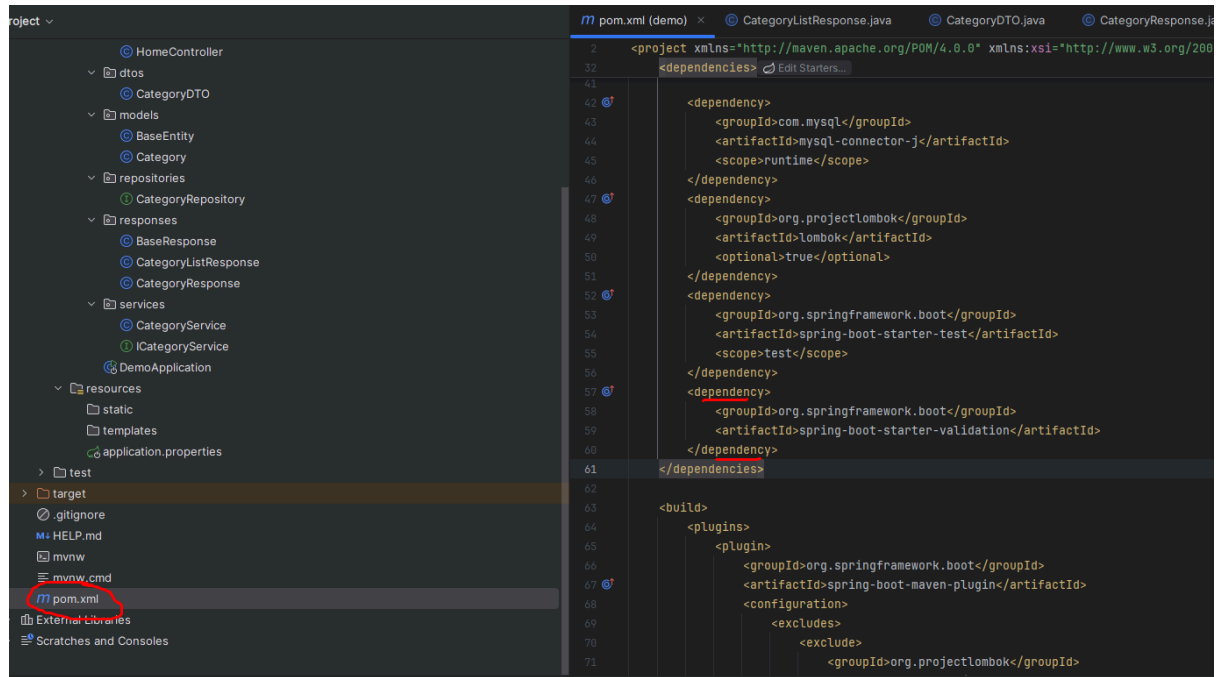
☒ Include comment with link to declaration

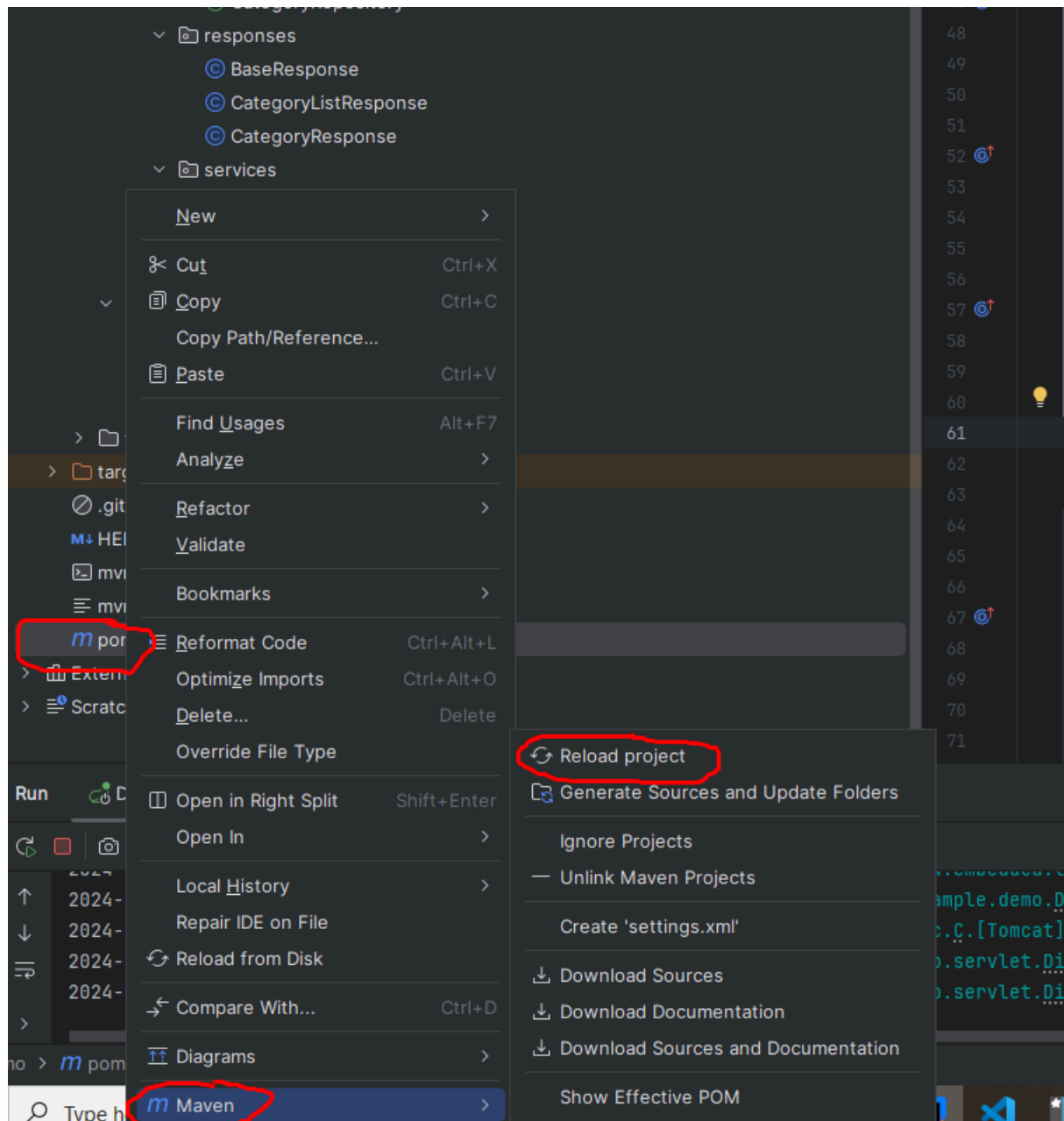
Compile Dependencies (3)

Category/License	Group / Artifact	Version
Apache 2.0	org.apache.tomcat.embed » tomcat-embed-el	10.1.28

nhà xinh BIG SALE 70% TẤT CẢ SẢN PHẨM 30.08 - 22.09 XEM NGAY







pages

