

# BÀI 2 LÀM QUEN VỚI SPRING FRAMEWORK

**Bài này giúp người học nắm được các nội dung sau:**

- Spring Framework là gì?
- Cài đặt môi trường và công cụ để phát triển ứng dụng Web bằng Spring Framework
- Lập trình cơ bản ứng dụng Website với Spring Framework

## 2.1 Cơ bản về Spring Framework và các cài đặt cần thiết

### 2.1.1 Yêu cầu

Máy tính phải cài đặt:

- Java JDK (Java Development Kit)
- Cài đặt Laragon (Quản lý CSDL MySQL)
- Cài đặt IntelliJ IDEA.

*(Tham khảo cài đặt tại phần [PHỤ LỤC CÀI ĐẶT MÔI TRƯỜNG VÀ CÔNG CỤ CẦN THIẾT](#) Ở CUỐI TRANG)*

Máy tính chạy trên nền tảng hệ điều hành:

- Windows 7
- Windows 8
- Windows 10
- Windows 11.

## 2.2 HƯỚNG DẪN THIẾT LẬP PHPMYADMIN

**PhpMyAdmin** là một công cụ quản trị cơ sở dữ liệu miễn phí và mã nguồn mở được viết bằng ngôn ngữ PHP, được sử dụng để quản lý cơ sở dữ liệu MySQL.

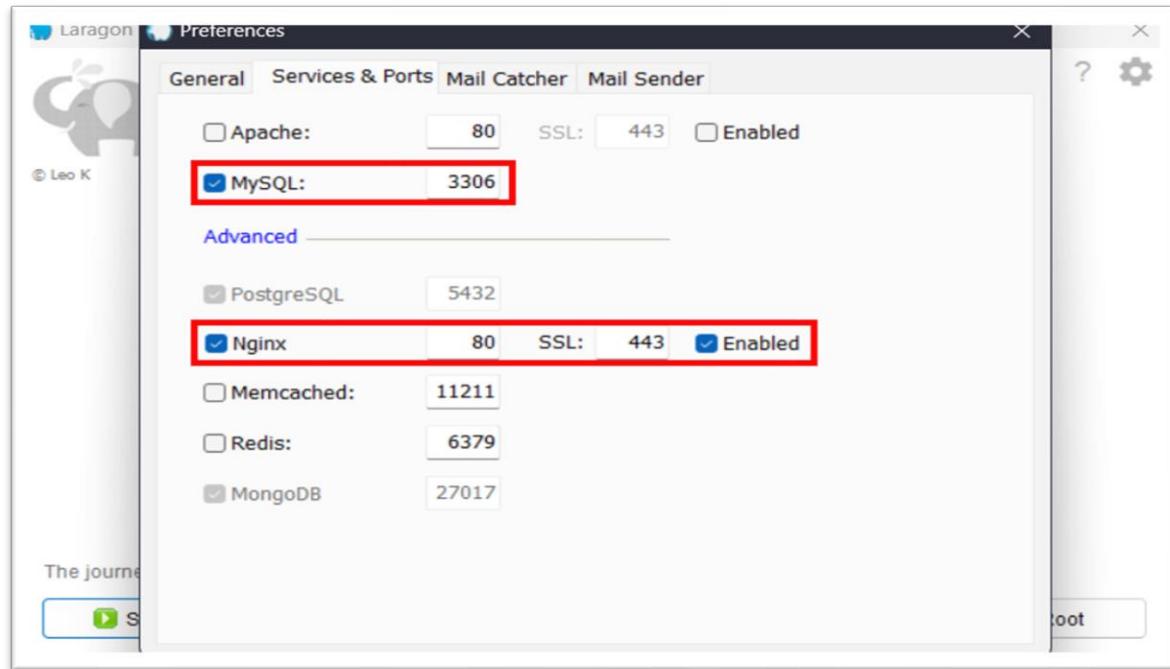
Nó cho phép người dùng thực hiện các tác vụ quản lý cơ sở dữ liệu như **tạo, xóa hoặc chỉnh sửa** cơ sở dữ liệu và bảng, tạo người dùng và **phân quyền**, thực hiện truy vấn SQL, sao lưu và khôi phục cơ sở dữ liệu và nhiều tác vụ khác thông qua giao diện web đơn giản và thân thiện.

Các bạn khởi động ứng dụng **Laragon**, nhấp vào biểu tượng cài đặt ở góc bên phải như ảnh bên dưới.



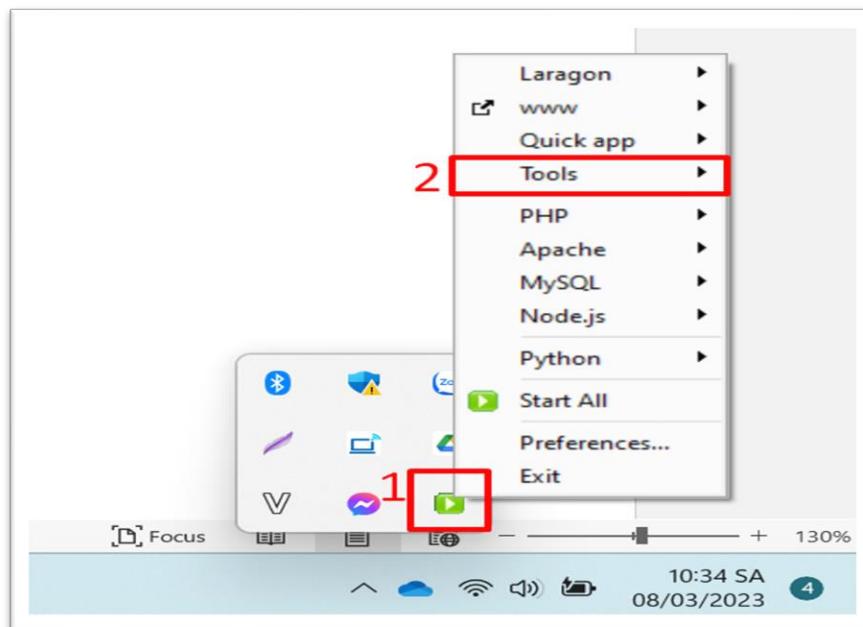
Hình 14. Khởi động ứng dụng Laragon

Lựa chọn cấu hình như bên dưới: tích chọn **MySQL, Nginx**.

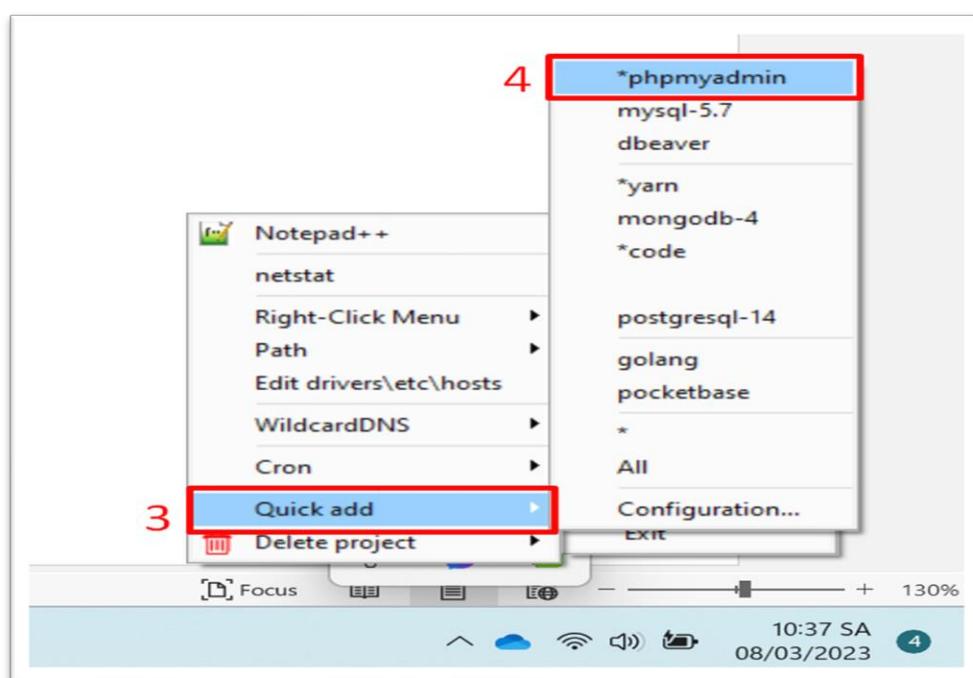


Hình 15. Cấu hình MySQL, Nginx cho Laragon

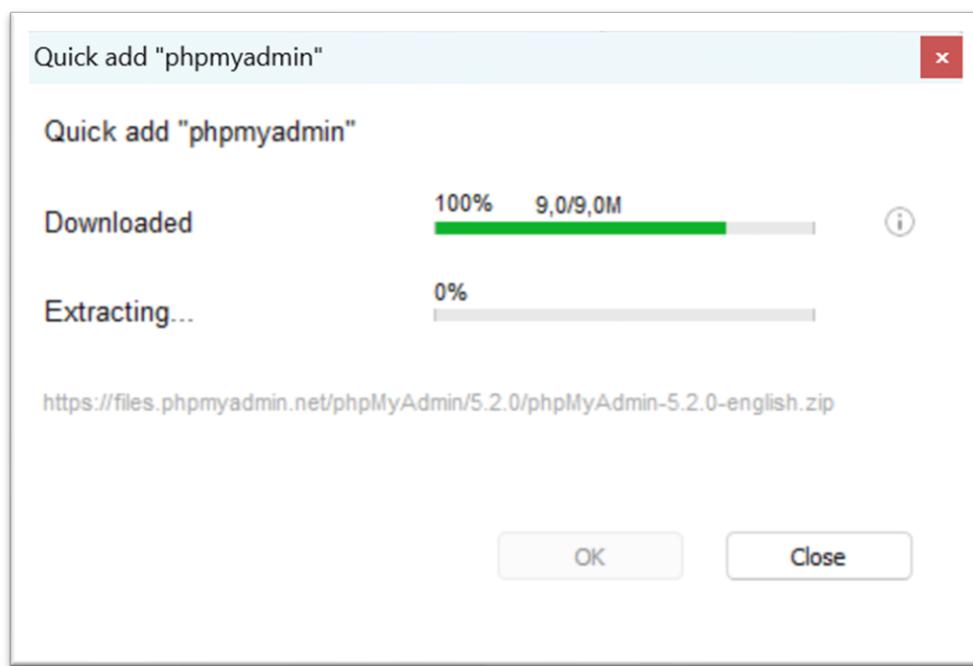
Sau khi làm xong các bước trên, ta tiến hành cài đặt chọn **PhpMyAdmin** theo các bước như bên dưới.



Hình 16. Cài đặt chọn PhpMyAdmin 1

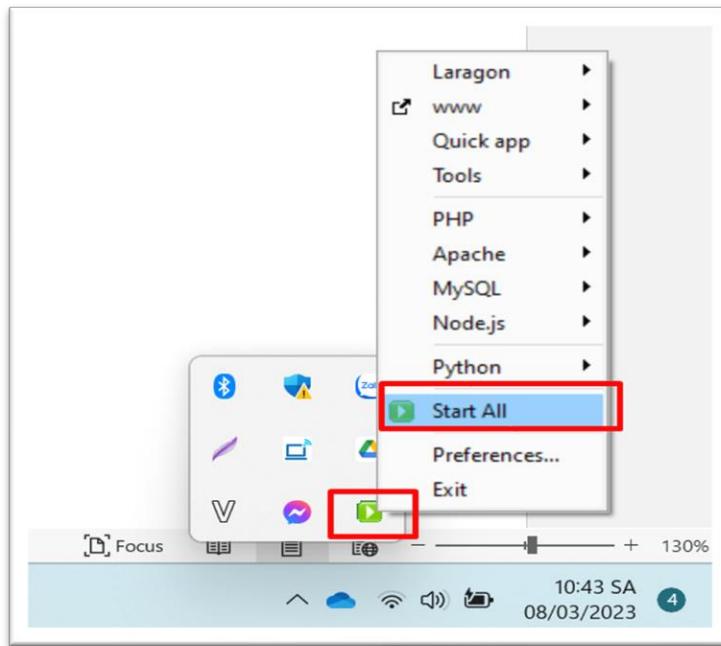


Hình 17. Cài đặt chọn PhpMyAdmin 2



Hình 18. Quá trình Quick add "Phpmyadmin"

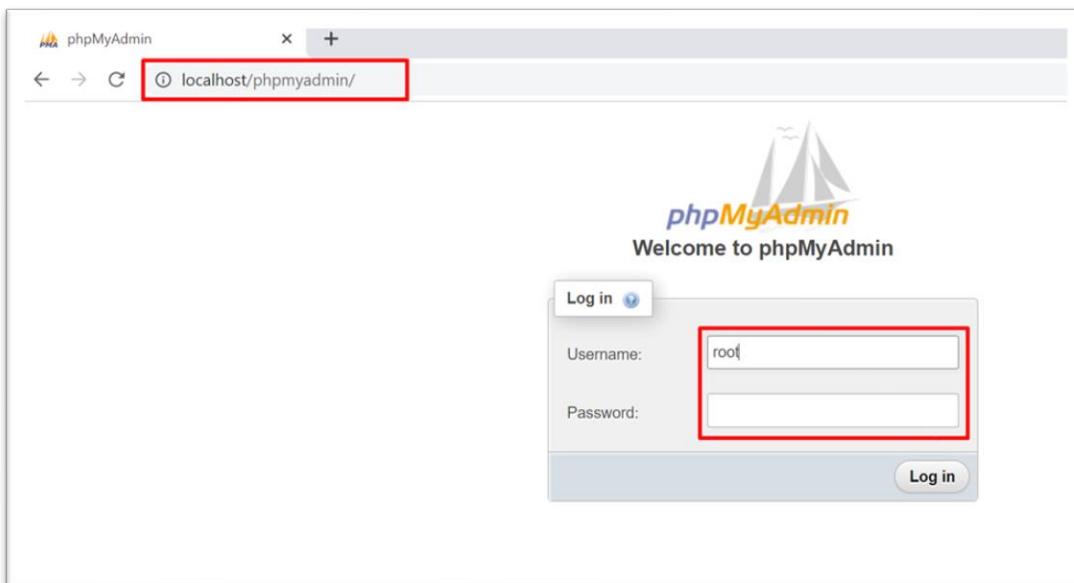
Sau khi thiết lập xong ấn **Start All** để khởi động các dịch vụ.



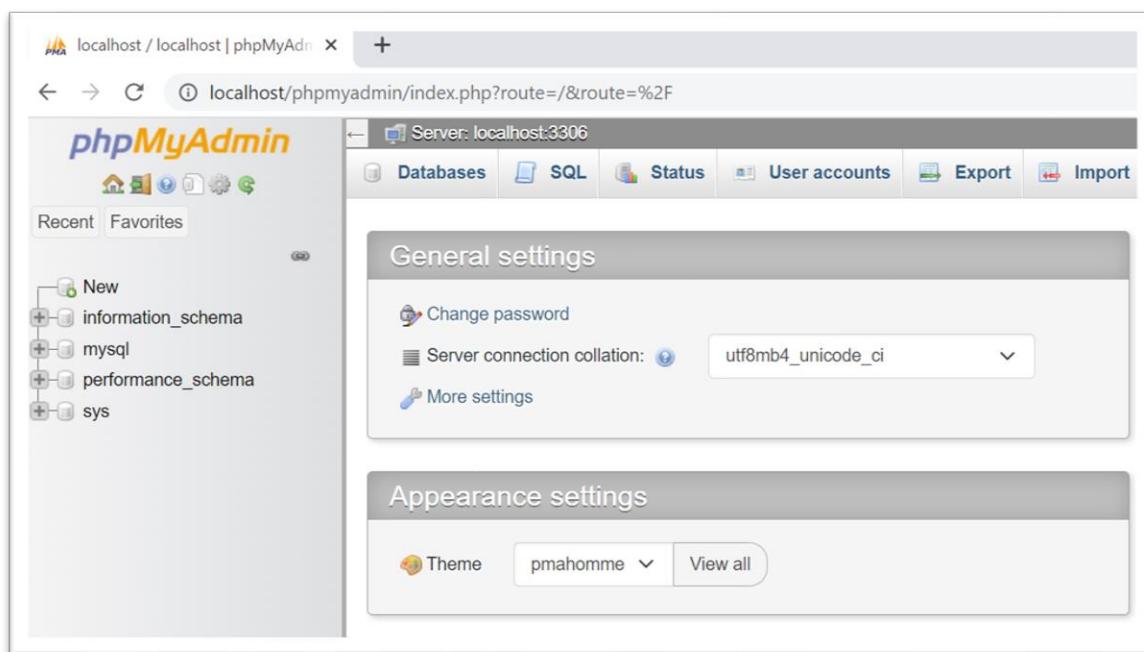
Hình 19. Khởi động dịch vụ PhpMyAdmin

Như vậy bạn đã cài đặt và thiết lập thành công dịch vụ PhpMyAdmin trên máy tính. Bây giờ truy cập vào đường dẫn: <http://localhost/phpmyadmin/>

Đăng nhập vào **phpMyAdmin**. Theo mặc định tên người dùng là **root** và mật khẩu **để trống**



Hình 20. Giao diện đăng nhập PhpMyAdmin



Hình 21. Giao diện trang Index PhpMyAdmin

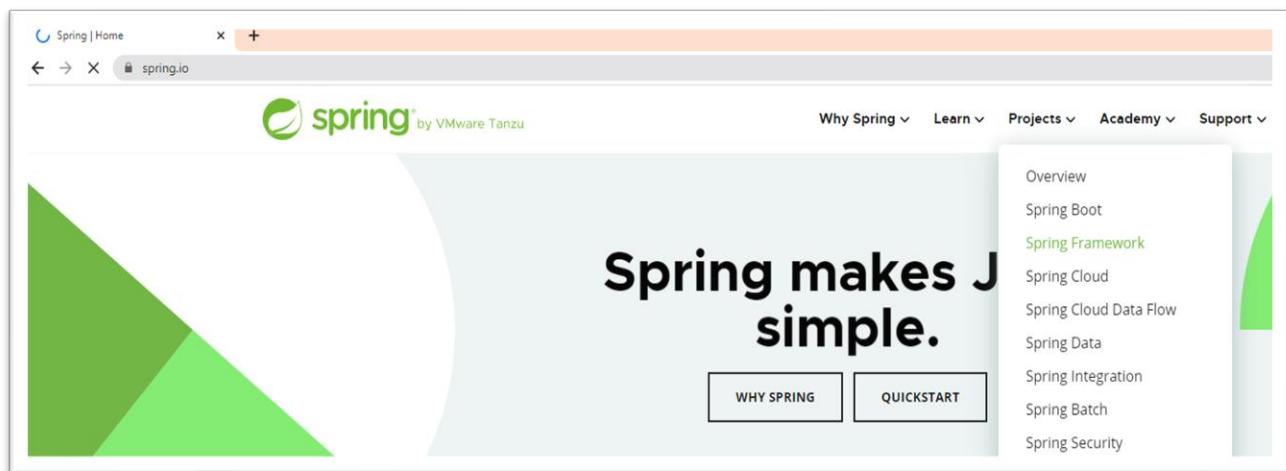
Quá trình cài đặt và khởi động PhpMyAdmin thành công, chúng ta sẽ quay PhpMyAdmin sau.

## 2.3 Download Template Spring từ spring.io

Template Spring từ spring.io là một dự án mẫu (template project) sẵn có cho việc phát triển ứng dụng sử dụng framework Spring. Dự án mẫu này cung cấp một số bộ khung cơ bản cho việc phát triển ứng dụng, giúp giảm thời gian và công sức của lập trình viên khi bắt đầu phát triển một dự án mới.

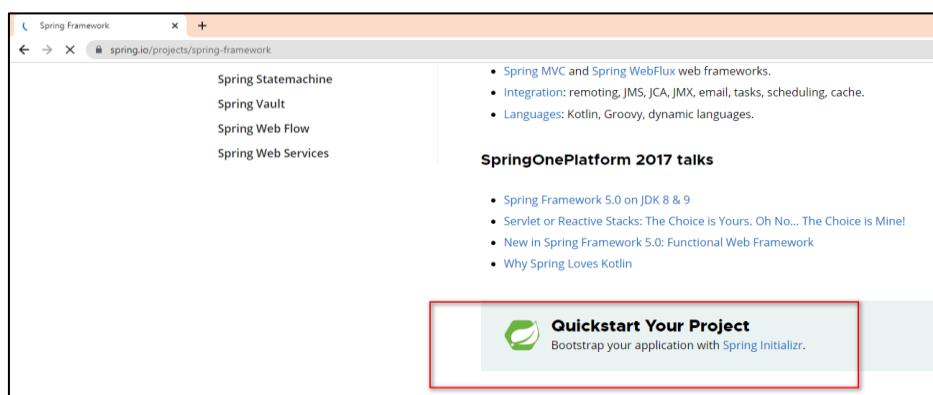
Template Spring bao gồm một số module cơ bản như Spring Boot, Spring Security, Spring Data, Spring MVC, và nhiều hơn nữa.

Truy cập trang web: <https://spring.io/>, tại mục **Project** chọn **Spring Framework**



Hình 22. Open Spring Framework tại [spring.io](https://spring.io/)

Tiếp theo, lướt xuống dưới chọn **Spring Initializr**.



Hình 23. Khởi động Quikstart you Project

Lựa chọn cấu hình, tích chọn các lựa chọn như ảnh bên dưới,

The screenshot shows the Spring Initializr web application at <https://start.spring.io>. The interface allows users to configure a new Spring Boot project. Key settings visible include:

- Project:** Maven (selected)
- Language:** Java (selected)
- Spring Boot:** 3.0.5 (selected)
- Project Metadata:**
  - Group: com.example
  - Artifact: demo
  - Name: demo
  - Description: Demo project for Spring Boot
  - Package name: com.example.demo
- Packaging:** Jar (selected)
- Java:** 20 (selected)

Hình 24. Lựa chọn cấu hình cho spring initializr

Tiếp theo nhấn Tiếp, nhấn **Add Dependencies...**

The screenshot shows the 'Dependencies' section of the Spring Initializr interface. It displays a message: "No dependency selected". To the right, there is a prominent yellow button labeled "ADD DEPENDENCIES... CTRL + B".

Cửa sổ mở ra, tìm và thêm các **Dependencies** như ảnh bên dưới.

**Dependencies**

**ADD DEPENDENCIES... CTRL + B**

- Spring Boot DevTools** **DEVELOPER TOOLS**  
Provides fast application restarts, LiveReload, and configurations for enhanced development experience. -
- Lombok** **DEVELOPER TOOLS**  
Java annotation library which helps to reduce boilerplate code. -
- Spring Web** **WEB**  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container. -
- Thymeleaf** **TEMPLATE ENGINES**  
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes. -
- Spring Security** **SECURITY**  
Highly customizable authentication and access-control framework for Spring applications. -
- Spring Data JPA** **SQL**  
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate. -
- MySQL Driver** **SQL**  
MySQL JDBC driver. -

Hình 25. thêm các Dependencies trong spring.io

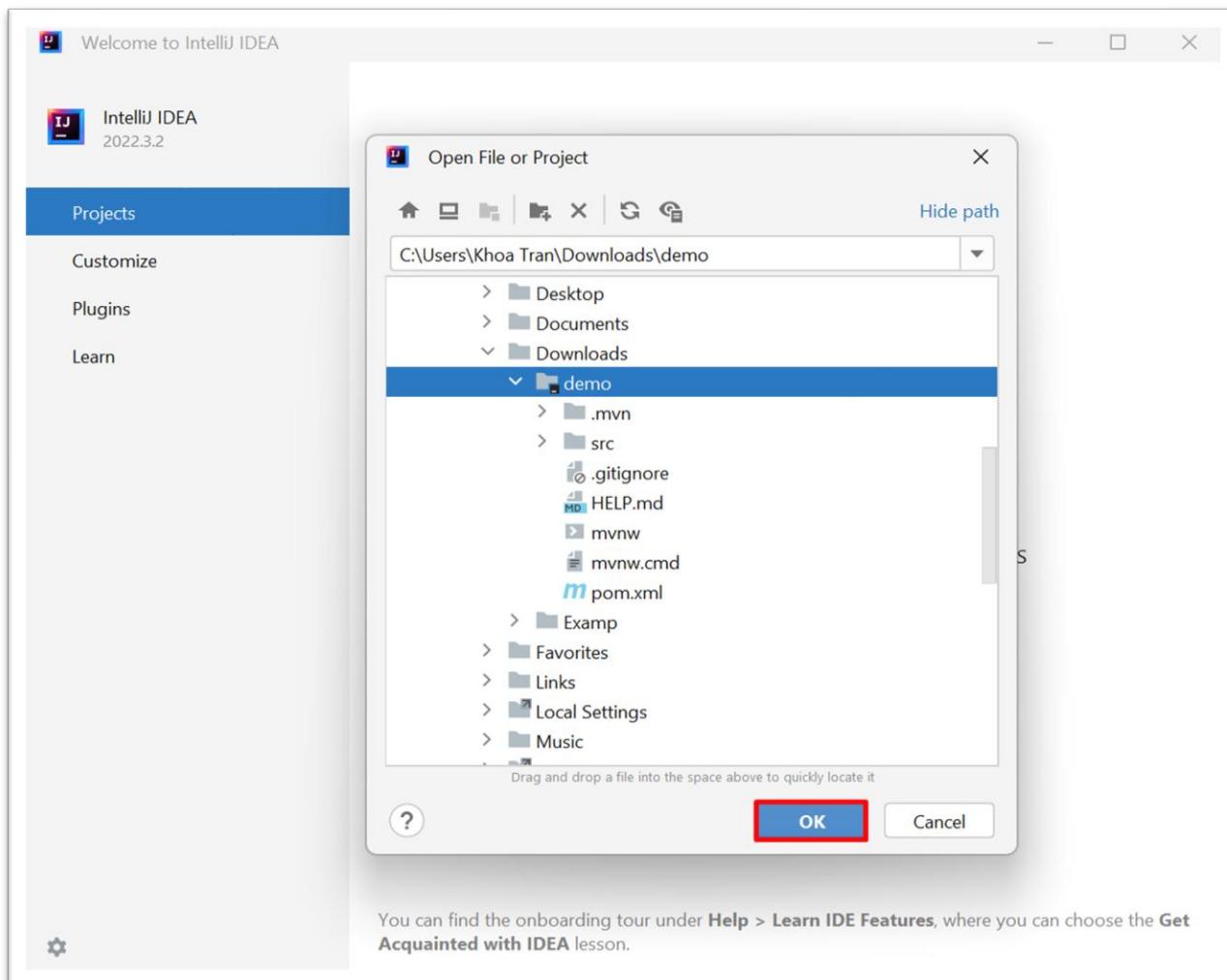
Sau đó, nhấn **GENERATE** để tải về



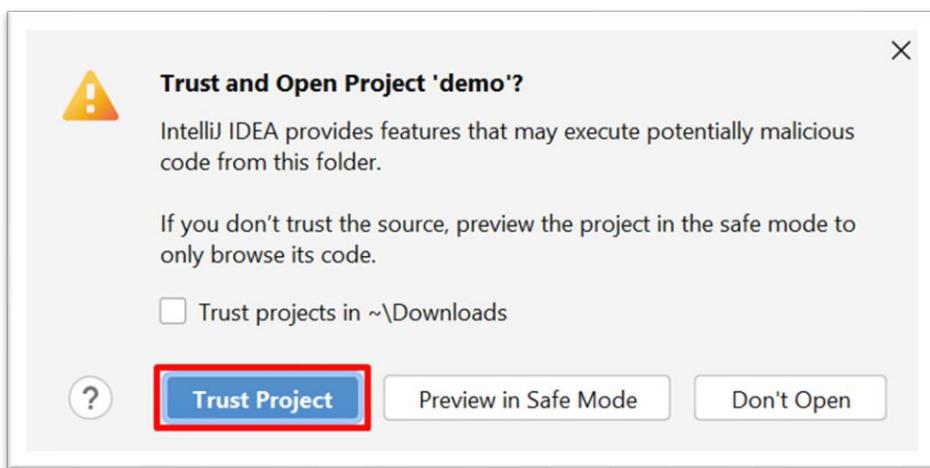
Một file **demo.zip** được tải về máy tính,



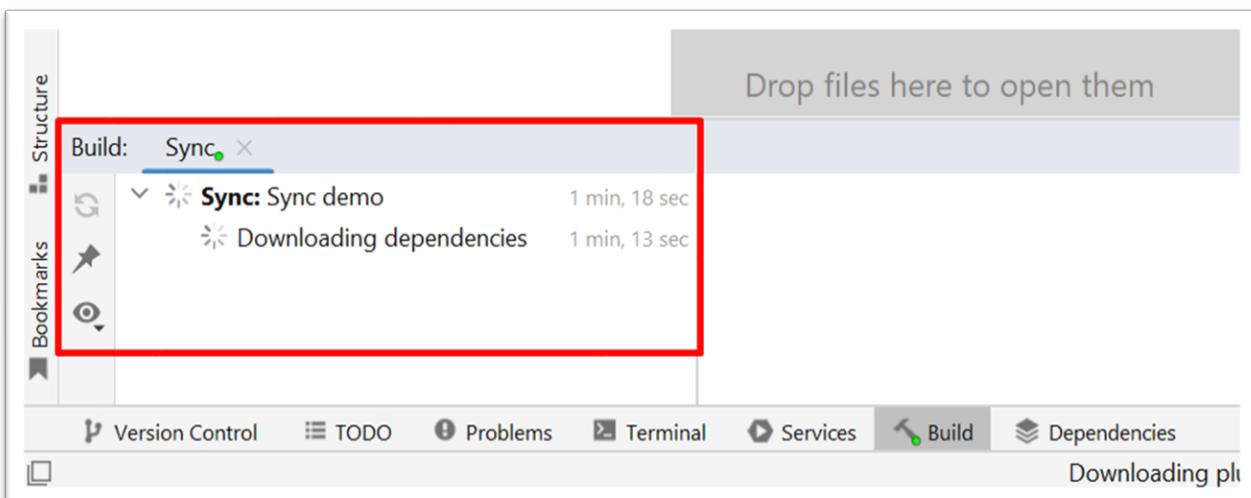
Giải nén thư mục **Demo.zip Template** đã tải về và mở Spring Teamplate bằng **IntelliJ IDEA**.



Hình 26. Giải nén và mở Spring Teamplate bằng IntelliJ IDEA.

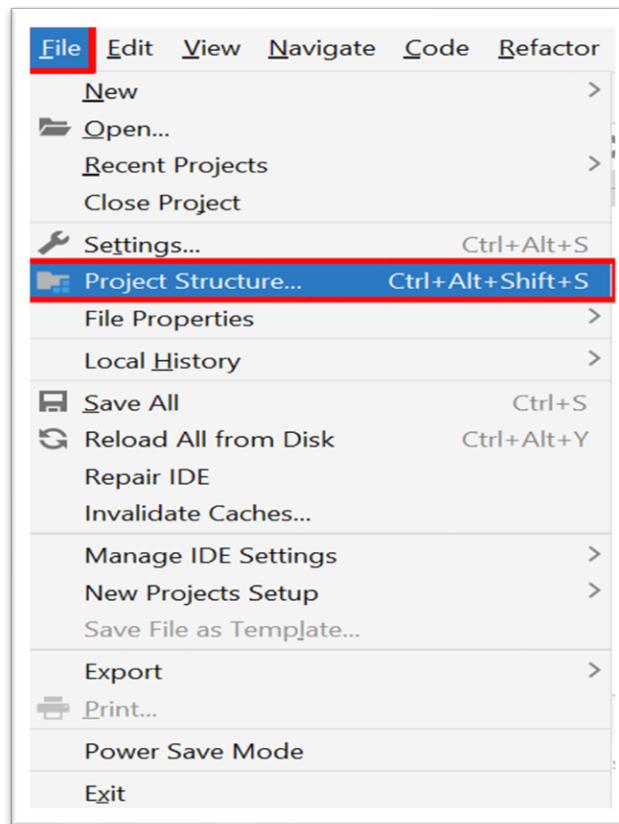


Tiến trình tải các **Dependencies** cần thiết. Bạn đợi cho các cài đặt hoàn tất, bao gồm sẽ cài đặt đường dẫn tới thư mục JDK đã cài ở bước trước.



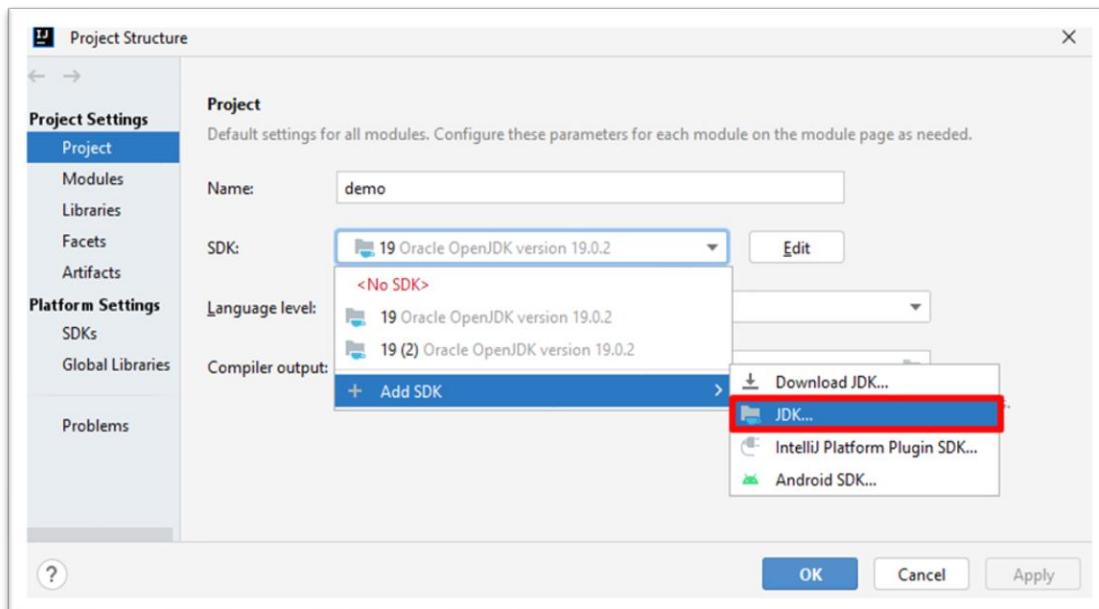
Hình 27. Bước cài đặt các Dependencies cần thiết

Tại thẻ **File** → chọn **Project Structure...** Để tạo cấu trúc dự án.



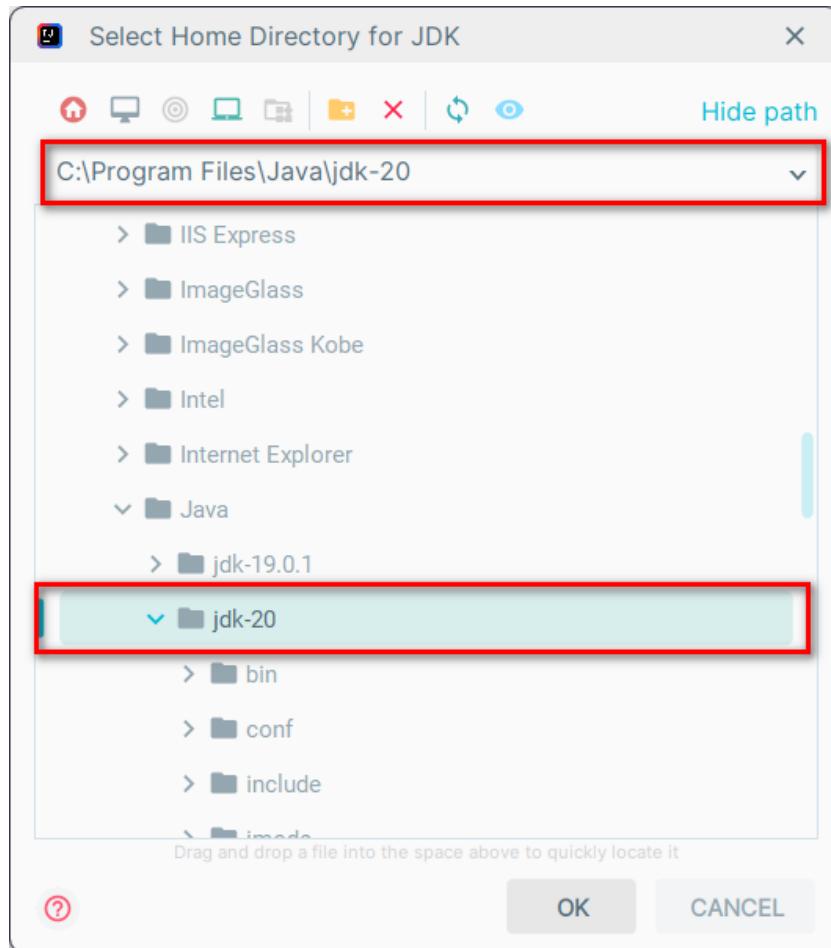
Hình 28. Lựa chọn Project Structure

Tiếp theo, làm theo hướng dẫn bên dưới để Add đến đường dẫn chứa file JDK.



Hình 29. Lựa chọn phiên bản JDK

Cửa sổ xuất hiện, chọn thư mục **jdk-20** theo đường dẫn **C:\Program Files\Java\jdk-20**.



Hình 30. Lựa chọn phiên bản JDK - 20

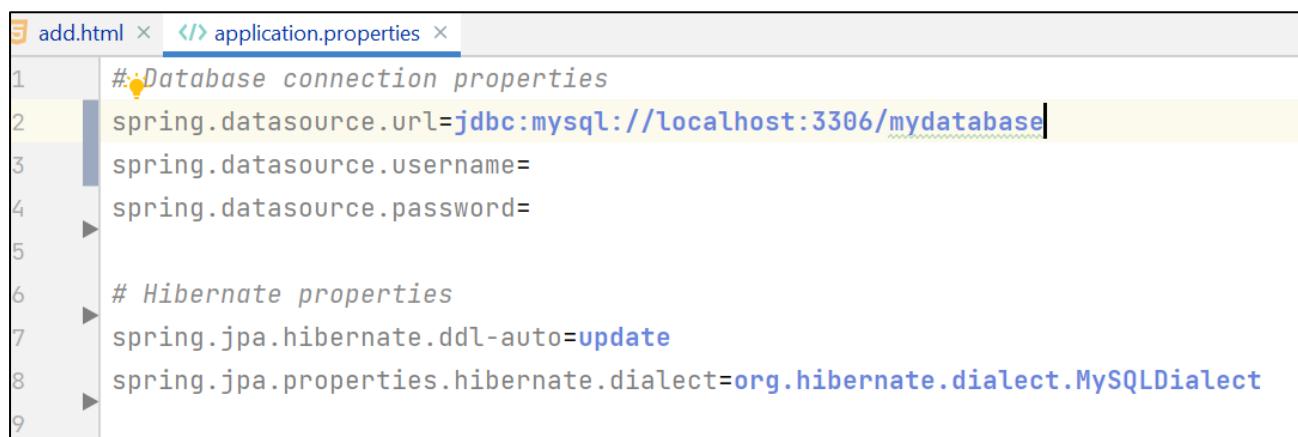
Thêm nội dung vào file **application.properties**

**Application.properties** có thể được đặt trong thư mục **src/main/resources** của ứng dụng web Java. Nó được sử dụng để cấu hình các thông số như cơ sở dữ liệu, cổng kết nối, tên đăng nhập và mật khẩu.

```
# Database connection properties
spring.datasource.url=jdbc:mysql://localhost:3306/mydatabase
spring.datasource.username=
spring.datasource.password=

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

Tìm đến file **application.properties** tại đường dẫn **src/main/resources**

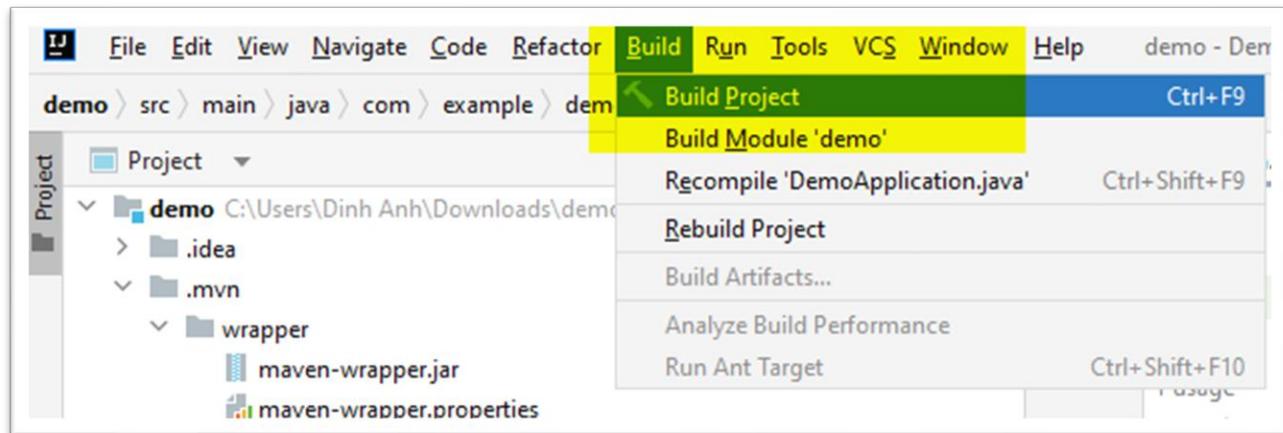


```
# Database connection properties
spring.datasource.url=jdbc:mysql://localhost:3306/mydatabase
spring.datasource.username=
spring.datasource.password=

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

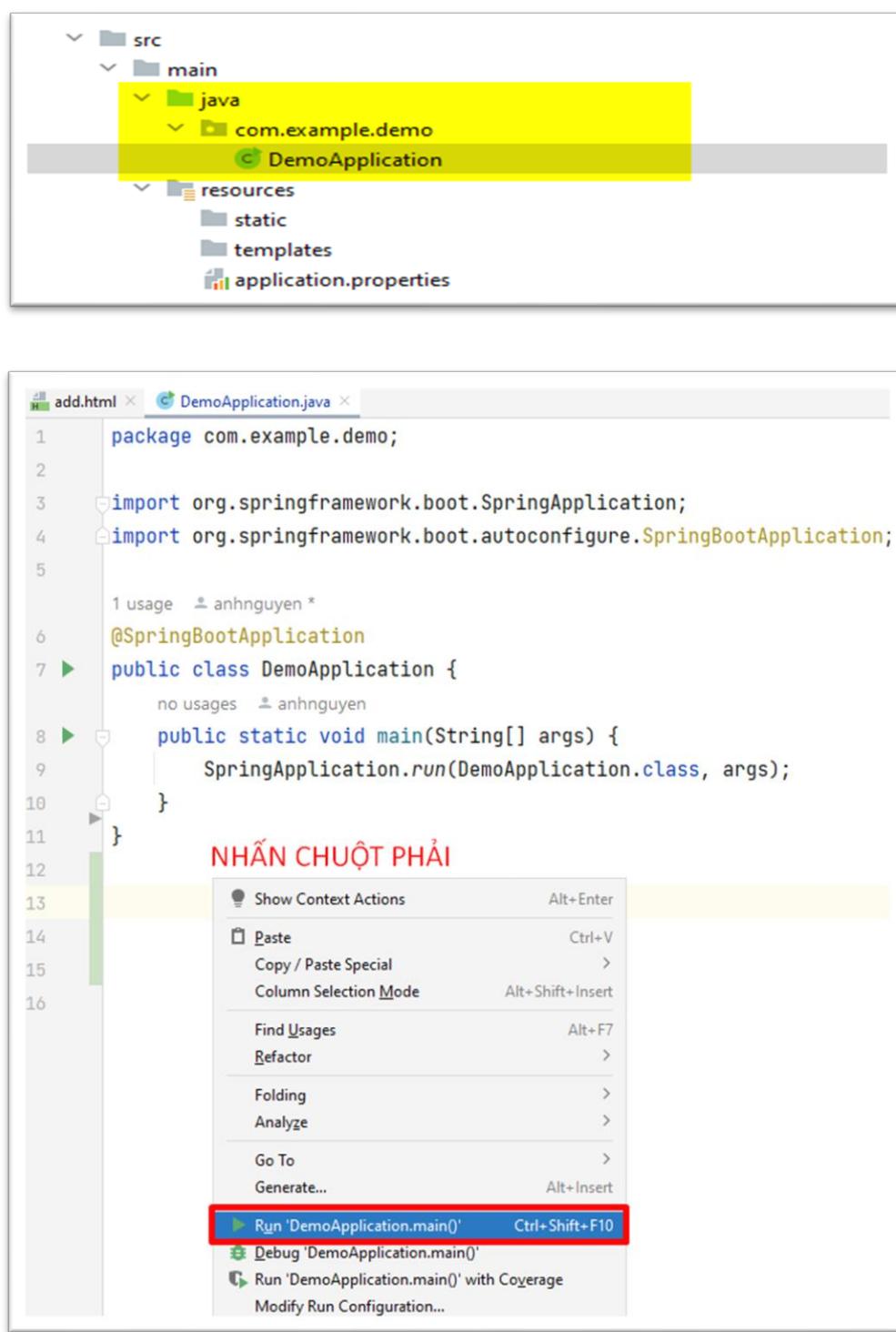
Hình 31. Bổ sung đoạn code cấu hình cho application.properties

Tiếp theo, **Build Project** như ảnh bên dưới.



Hình 32. Build project chương trình

Tiến hành Run chương trình tại file **DemoApplication.java** theo ảnh hướng dẫn bên dưới.



Hình 33. Run chương trình tại file DemoApplication

# BÀI 4 THAO TÁC VỚI CƠ SỞ DỮ LIỆU

**Bài này giúp người học nắm được các nội dung sau:**

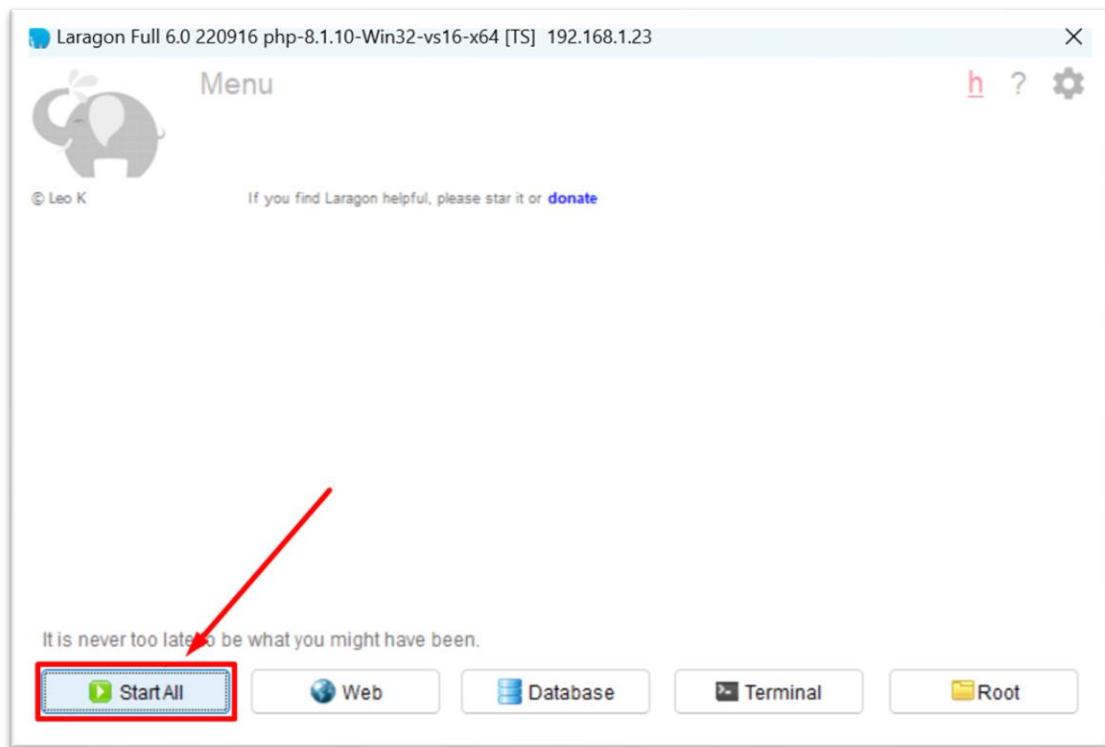
- Làm quen thao tác thêm, xoá sửa dữ liệu trên cơ sở dữ liệu
- Làm quen với trình Quản trị cơ sở dữ liệu MySQL phpMyAdmin
- Kết nối cơ sở dữ liệu Project với phpMyAdmin
- Làm cơ sở xây dựng ứng dụng website quản lý sách với các chức năng giao tiếp cơ bản với Cơ sở dữ liệu: chức năng thêm, chức năng xóa, chức năng sửa.

## 4.1 Khởi động phpMyAdmin – Kết nối CSDL

Mở chương trình quản lý cơ sở dữ liệu phù hợp, trong hướng dẫn này dùng chương trình quản lý cơ sở dữ liệu **phpMyAdmin** (*Tham khảo lại Lab thực hành 01 cách cài đặt phpMyAdmin*).

Khởi động phần mềm **Laragon**, nhấn **Start All** để khởi động để bật các dịch vụ Server.

**Lưu ý:** Nếu như webserver khởi động không lên do port. Sinh viên vui lòng thay đổi port của webserver trong setting.



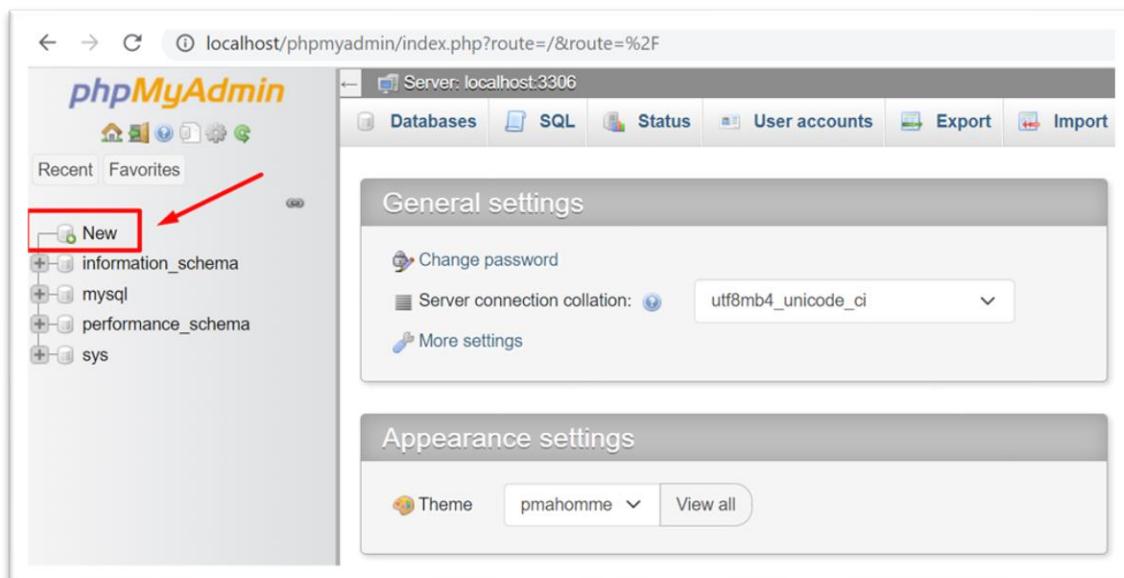
Hình 66. Khởi động phpMyAdmin

Truy cập địa chỉ website: <http://localhost/phpmyadmin/> , nhập tài khoản

**Username: root**

**Pass:** “để trống”

Tiếp theo, tạo CSDL có tên **bookstore**, Chọn **New** ở góc bên trái.



Hình 67. Tạo mới CSDL trong phpMyAdmin

Đặt tên và tạo, nhớ chọn đúng **utfmb4\_0900\_ai\_ci**



Hình 68. Đặt tên cho CSDL trong phpMyAdmin mới và tạo

Tiếp theo, mở dự án lên và tìm đến file **application.properties** và cấu hình lại như bên dưới:

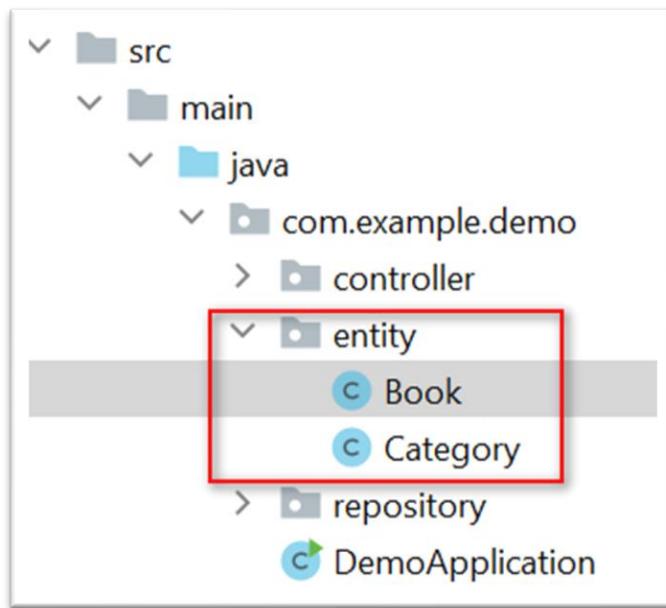
```
# Database connection properties
spring.datasource.url=jdbc:mysql://localhost:3306/bookstore
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

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

#Spring Security
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration
```

## 4.2 Khởi tạo các đối tượng

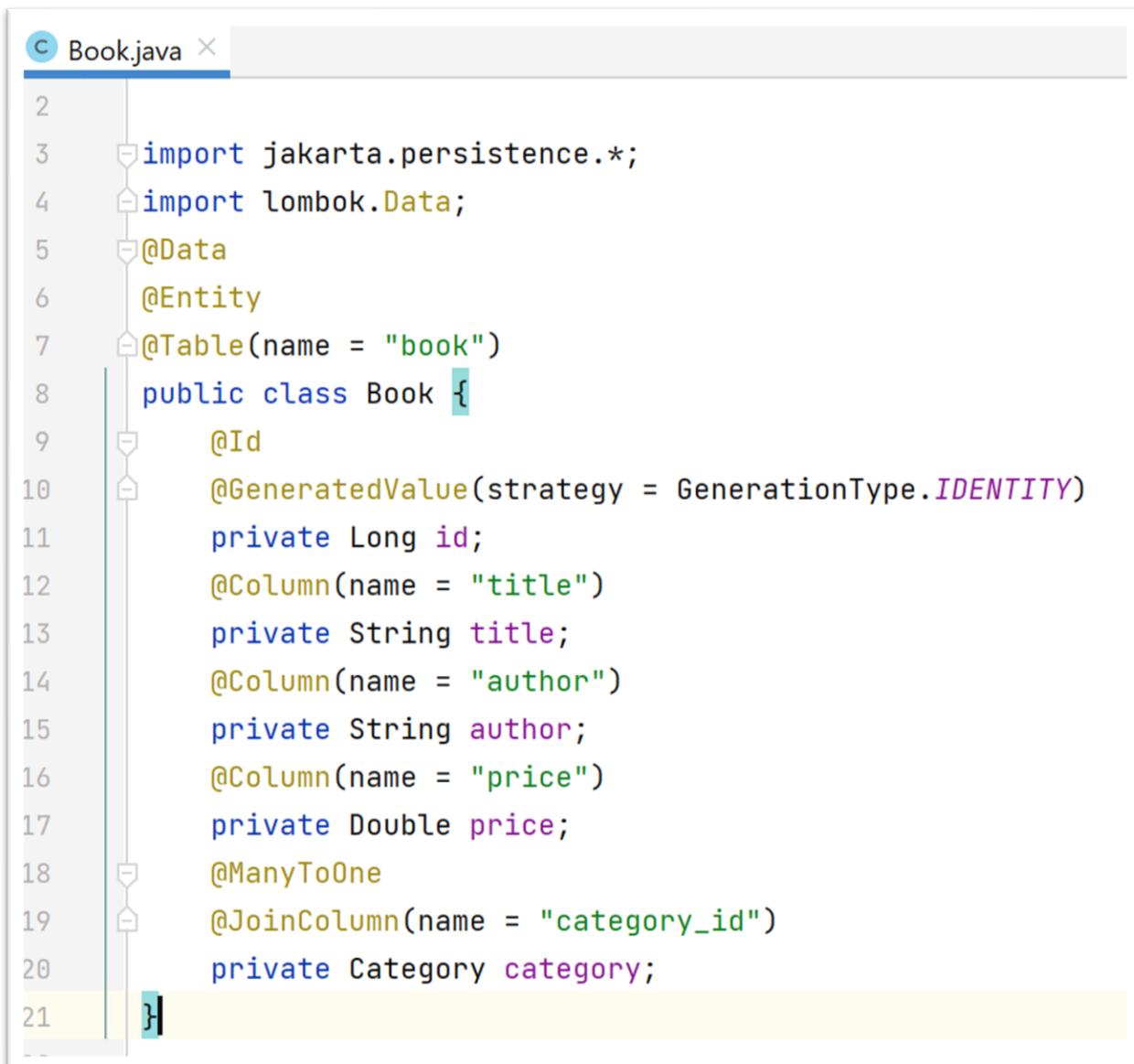
Tạo 2 file model **Book.java** và **Category.java** đặt tại thư mục **src/main/java/com.example.demo/entity**.



Hình 69. Tạo 2 file Book.java và Category.java trong thư mục entity

Tiến hành thêm đoạn code khai báo các thuộc tính như bên dưới cho 2 file

**Book.java** và **Category.java**



```
2
3 import jakarta.persistence.*;
4 import lombok.Data;
5 @Data
6 @Entity
7 @Table(name = "book")
8 public class Book {
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private Long id;
12    @Column(name = "title")
13    private String title;
14    @Column(name = "author")
15    private String author;
16    @Column(name = "price")
17    private Double price;
18    @ManyToOne
19    @JoinColumn(name = "category_id")
20    private Category category;
21 }
```

Hình 70. Thêm các thuộc tính cho Book.java

```
c Category.java ×
1 package com.example.demo.entity;
2
3 import jakarta.persistence.*;
4 import lombok.Data;
5
6 import java.util.List;
7
8 @Data
9 @Entity
10 @Table(name = "category")
11 public class Category {
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15
16     @Column(name = "name")
17     private String name;
18
19     @OneToMany(mappedBy = "category", cascade = CascadeType.ALL)
20     private List<Book> books;
}
```

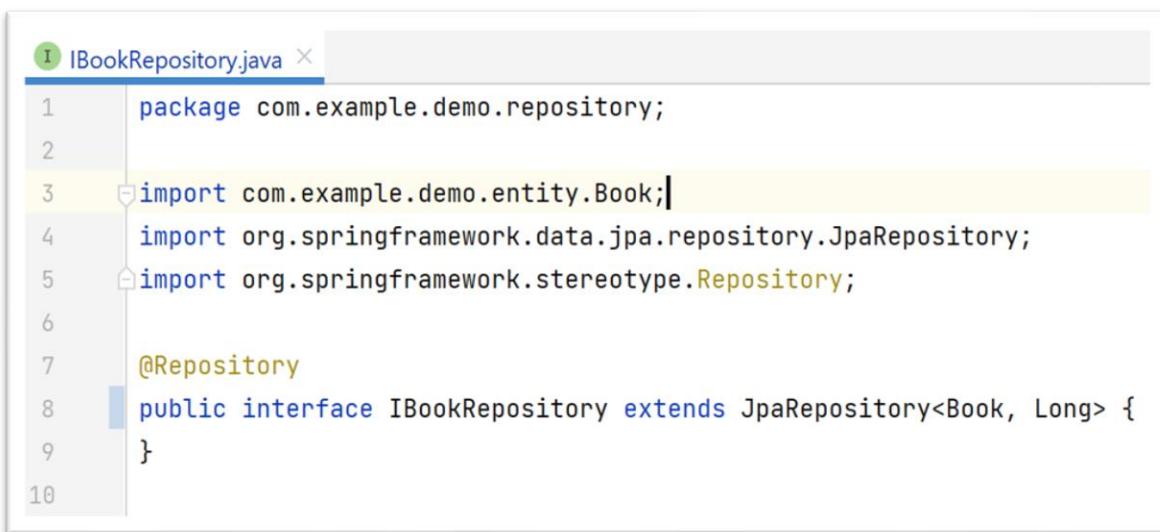
Hình 71. Thêm các thuộc tính cho Category.java

Tiếp theo tạo thêm 2 file **IBookRepository.java** và **ICategoryRepository.java** đặt tại thư mục **src/main/java/com.example.demo/repository**

**IBookRepository.java** và **ICategoryRepository.java** là các file interface trong một ứng dụng phần mềm quản lý sách. Chúng được sử dụng để tương tác với các đối tượng sách (books) và các đối tượng danh mục (categories) trong cơ sở dữ liệu.

**IBookRepository.java** cung cấp các phương thức để thêm, sửa, xóa, lấy danh sách các cuốn sách trong cơ sở dữ liệu.

Code nội dung vào file **IBookRepository.java**

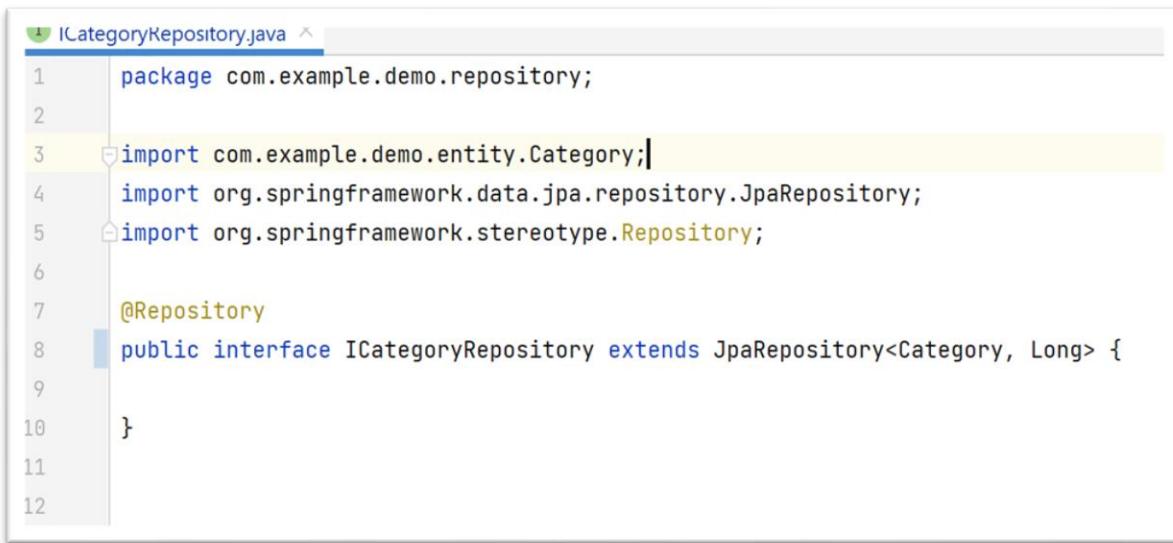


```
1 package com.example.demo.repository;
2
3 import com.example.demo.entity.Book;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface IBookRepository extends JpaRepository<Book, Long> {
9 }
10
```

Hình 72. Tạo thêm thư mục *IBookRepository.java*

**ICategoryRepository.java** cung cấp các phương thức để thêm, sửa, xóa, lấy danh sách các danh mục trong cơ sở dữ liệu.

Code nội dung vào file **ICategoryRepository.java**

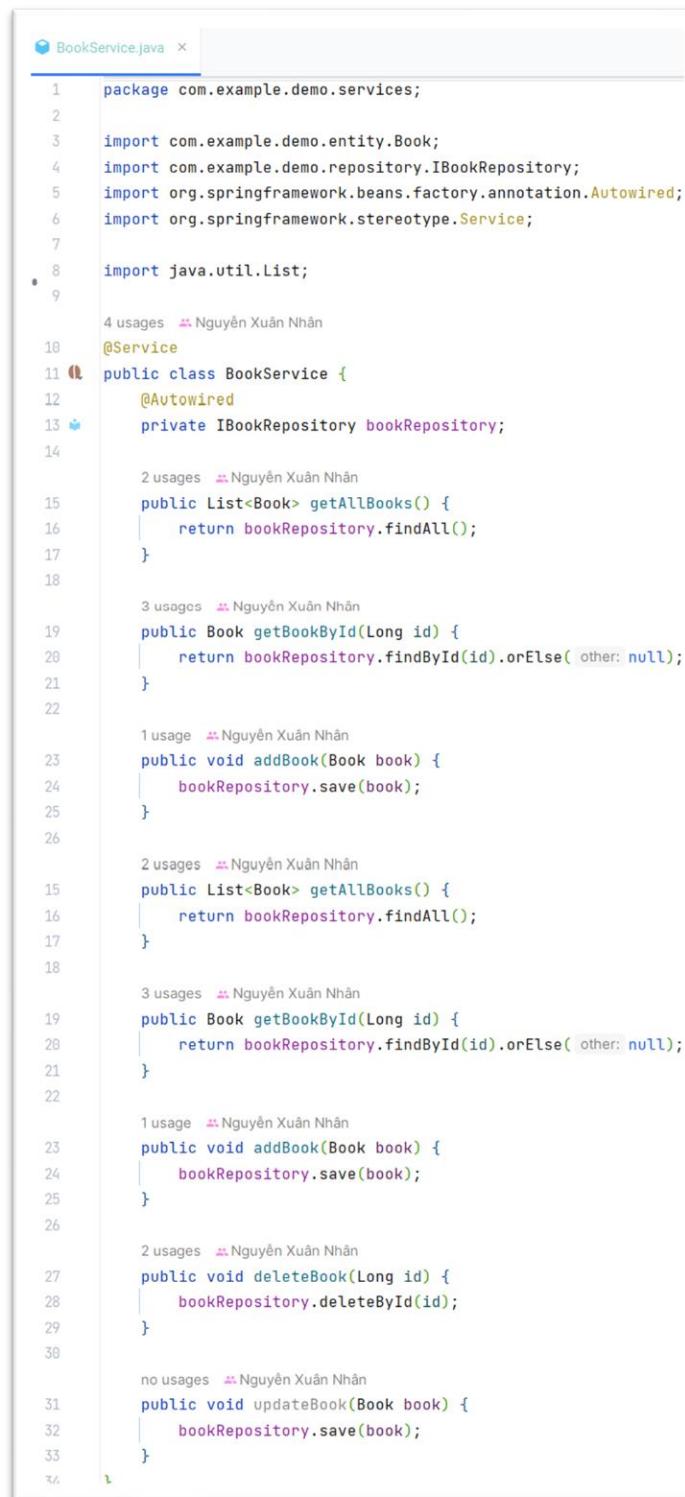


```
ICategoryRepository.java
1 package com.example.demo.repository;
2
3 import com.example.demo.entity.Category;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface ICategoryRepository extends JpaRepository<Category, Long> {
9
10 }
```

Hình 73. Tạo thêm thư mục ICategoryRepository.java

Tạo file **BookService.java** đặt tại thư mục theo đường dẫn:

**src/main/java/com.example.demo/services**



```
BookService.java x
1 package com.example.demo.services;
2
3 import com.example.demo.entity.Book;
4 import com.example.demo.repository.IBookRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import java.util.List;
9
10 4 usages ↗ Nguyễn Xuân Nhân
11 @Service
12 public class BookService {
13     @Autowired
14     private IBookRepository bookRepository;
15
16     2 usages ↗ Nguyễn Xuân Nhân
17     public List<Book> getAllBooks() {
18         return bookRepository.findAll();
19     }
20
21     3 usages ↗ Nguyễn Xuân Nhân
22     public Book getBookById(Long id) {
23         return bookRepository.findById(id).orElse( other: null );
24     }
25
26     1 usage ↗ Nguyễn Xuân Nhân
27     public void addBook(Book book) {
28         bookRepository.save(book);
29     }
30
31     2 usages ↗ Nguyễn Xuân Nhân
32     public void deleteBook(Long id) {
33         bookRepository.deleteById(id);
34     }
35
36     no usages ↗ Nguyễn Xuân Nhân
37     public void updateBook(Book book) {
38         bookRepository.save(book);
39     }
40
```

Hình 74. Tạo thêm file BookService.java

Trong lớp **BookService.java**, thường sẽ có các phương thức để **thêm, sửa, xóa, lấy danh sách các cuốn sách**, cùng với việc tìm kiếm các cuốn sách theo các tiêu chí khác nhau (ví dụ: tên sách, tác giả, danh mục). Lớp này thường sử dụng **IBookRepository.java** để tương tác với cơ sở dữ liệu và thực hiện các tác vụ liên quan đến sách.

Các phương thức trong lớp **BookController.java** thường được thiết kế để xử lý các yêu cầu từ người dùng *như thêm, sửa, xóa, hiển thị thông tin chi tiết và tìm kiếm*

Tạo file **BookController.java** đặt tại **src/main/java/com.example.demo**



```

1 package com.example.demo.controller;
2
3 import com.example.demo.entity.Book;
4 import com.example.demo.services.BookService;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.RequestMapping;
10
11 import java.util.List;
12
13 @Controller
14 @RequestMapping("/books")
15 public class BookController {
16     @Autowired
17     private BookService bookService;
18
19     @GetMapping
20     public String showAllBooks(Model model) {
21         List<Book> books = bookService.getAllBooks();
22         model.addAttribute(attributeName: "books", books);
23         return "book/list";
24     }
25 }

```

Hình 75. Tạo thêm file BookController.java

## 4.3 Khởi tạo các thành phần giao diện

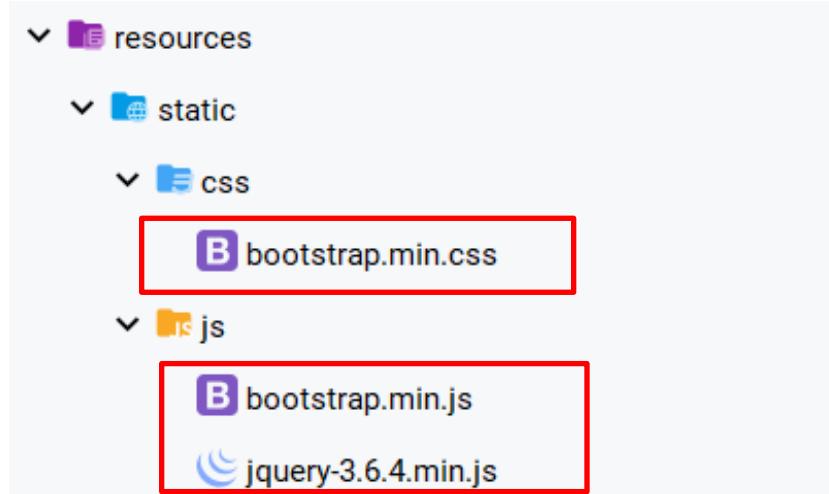
Tạo file **layout.html** đặt tại **src/main/resources/templates**



```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>My App</title>
    <link th:fragment="link-css" rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
</head>
<body>
    <header th:fragment="header">
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="container-fluid">
                <a class="navbar-brand" href="#">HUTECH</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
                    data-bs-target="#navbarSupportedContent"
                    aria-controls="navbarSupportedContent"
                    aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarSupportedContent">
                    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                        <li class="nav-item">
                            <a class="nav-link active" aria-current="page" href="/">Home</a>
                        </li>
                        <li class="nav-item"><a class="nav-link" href="/books">List Book</a></li>
                        <li class="nav-item"><a class="nav-link" href="/books/add">Add Book</a></li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div th:insert="~{::${content}}">
        <!-- NỘI DUNG TRANG CON-->
    </div>
    <footer th:fragment="footer">
        <script th:src="@{/js/bootstrap.min.js}"></script>
    </footer>
</body>
</html>
```

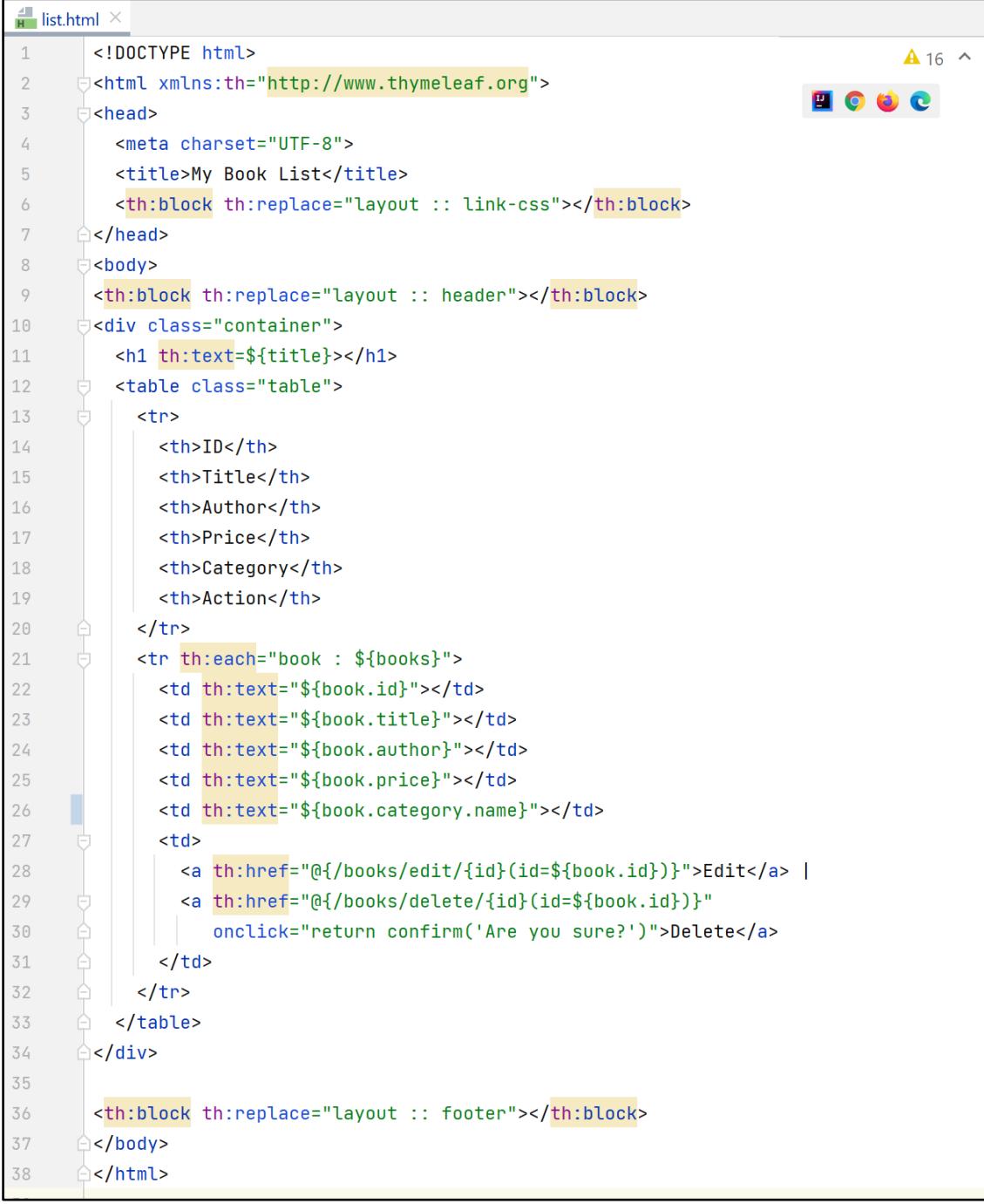
Hình 76. Tạo file layout.html

Kiểm tra đường dẫn đặt hai file tĩnh **css** và **js** của **bootstrap** và **jquery**  
(như đã học ở bài lab 1, 2 đã làm)



Hình 77. Thư mục css và js

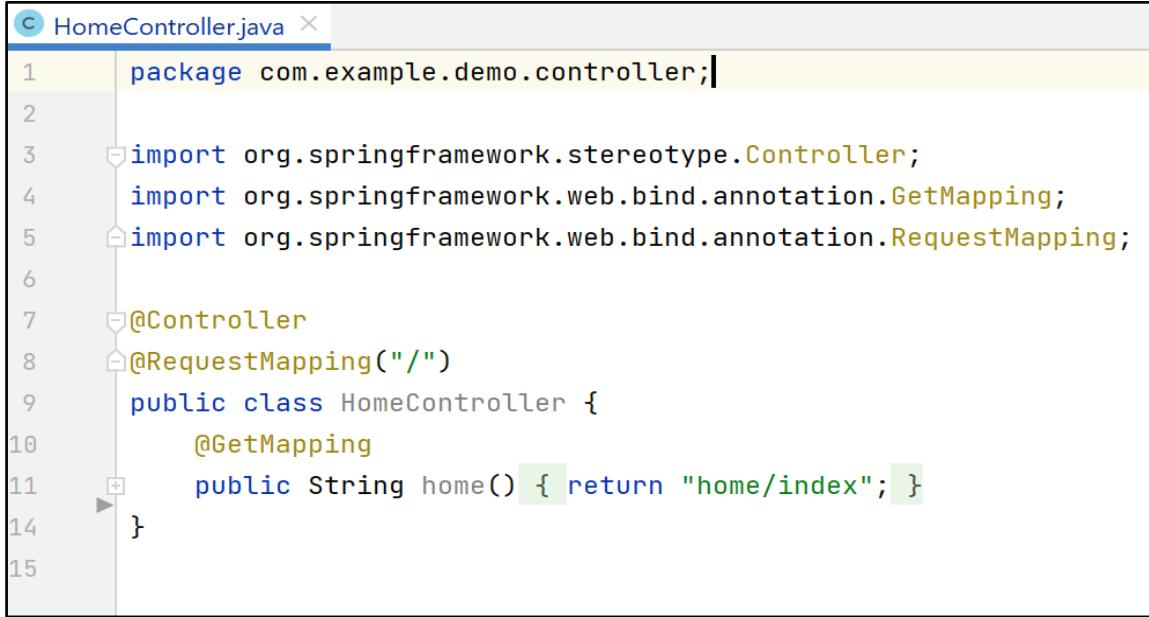
Tạo file **list.html** đặt tại **src/main/resources/templates/book**



```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>My Book List</title>
    <th:block th:replace="layout :: link-css"></th:block>
</head>
<body>
    <th:block th:replace="layout :: header"></th:block>
    <div class="container">
        <h1 th:text=${title}></h1>
        <table class="table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Title</th>
                    <th>Author</th>
                    <th>Price</th>
                    <th>Category</th>
                    <th>Action</th>
                </tr>
            <tbody>
                <tr th:each="book : ${books}">
                    <td th:text="${book.id}"></td>
                    <td th:text="${book.title}"></td>
                    <td th:text="${book.author}"></td>
                    <td th:text="${book.price}"></td>
                    <td th:text="${book.category.name}"></td>
                    <td>
                        <a th:href="@{/books/edit/{id}(id=${book.id})}">Edit</a> | 
                        <a th:href="@{/books/delete/{id}(id=${book.id})}"
                           onclick="return confirm('Are you sure?')">Delete</a>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
    <th:block th:replace="layout :: footer"></th:block>
</body>
</html>
```

Hình 78. Tạo file *list.html* hiển thị danh sách Books

Tạo file **HomeController.java** đặt tại thư mục controller  
**src/main/java/com.example.demo/controller**



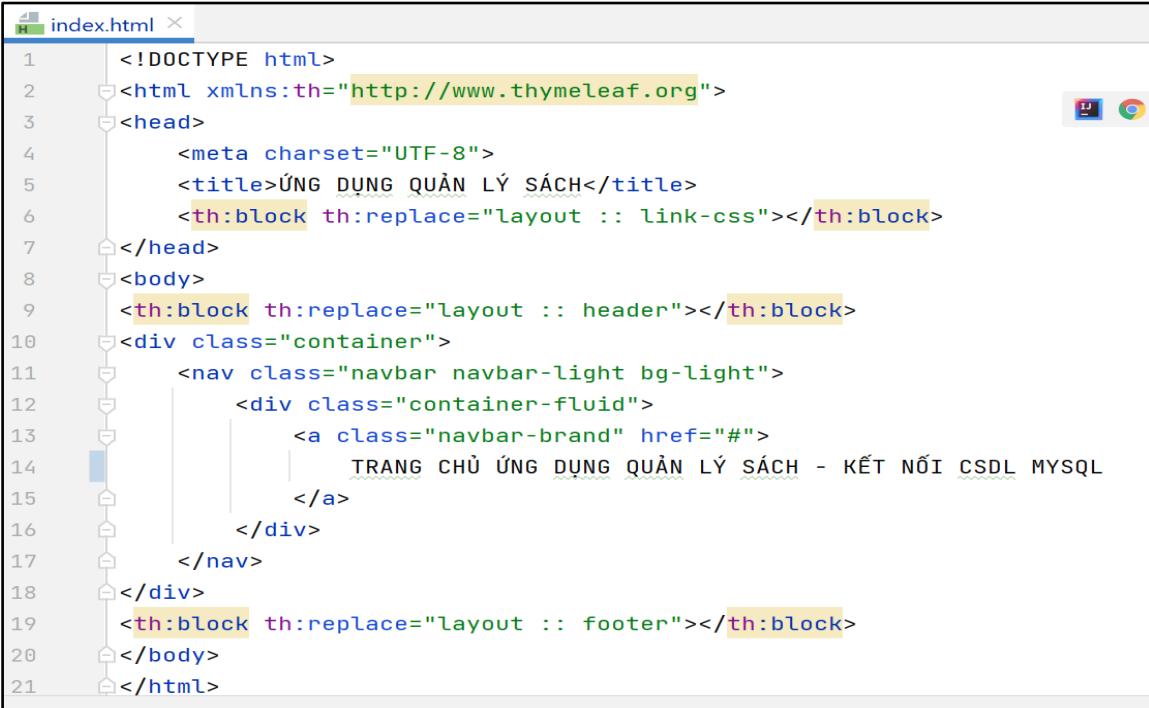
```

1 package com.example.demo.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RequestMapping;
6
7 @Controller
8 @RequestMapping("/")
9 public class HomeController {
10     @GetMapping
11     public String home() { return "home/index"; }
12 }
13
14
15

```

Hình 79. Tạo thêm file HomeController.java

Tạo file **index.html** đặt tại **src/main/resources/templates/home**



```

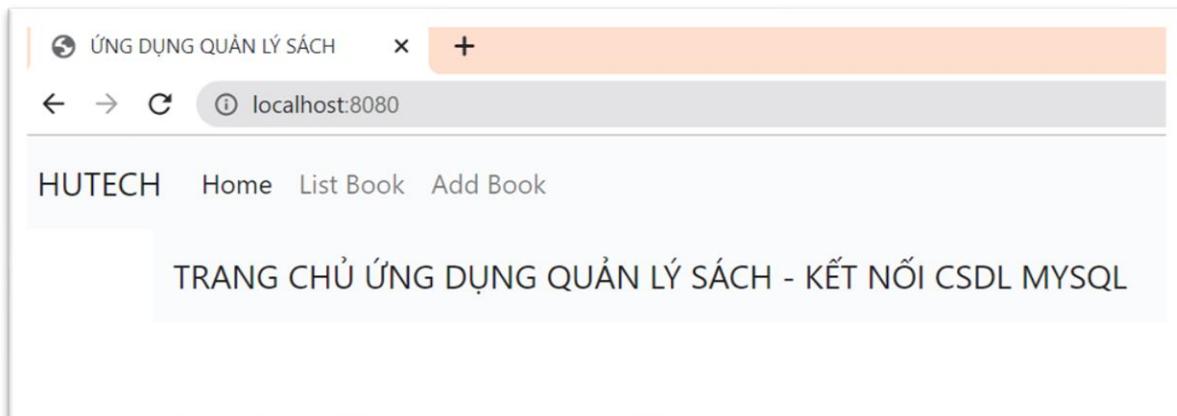
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title>ỨNG DỤNG QUẢN LÝ SÁCH</title>
6     <th:block th:replace="layout :: link-css"></th:block>
7 </head>
8 <body>
9     <th:block th:replace="layout :: header"></th:block>
10    <div class="container">
11        <nav class="navbar navbar-light bg-light">
12            <div class="container-fluid">
13                <a class="navbar-brand" href="#">
14                    TRANG CHỦ ỨNG DỤNG QUẢN LÝ SÁCH - KẾT NỐI CSDL MYSQL
15                </a>
16            </div>
17        </nav>
18    </div>
19    <th:block th:replace="layout :: footer"></th:block>
20 </body>
21 </html>

```

Hình 80. Tạo file index.html

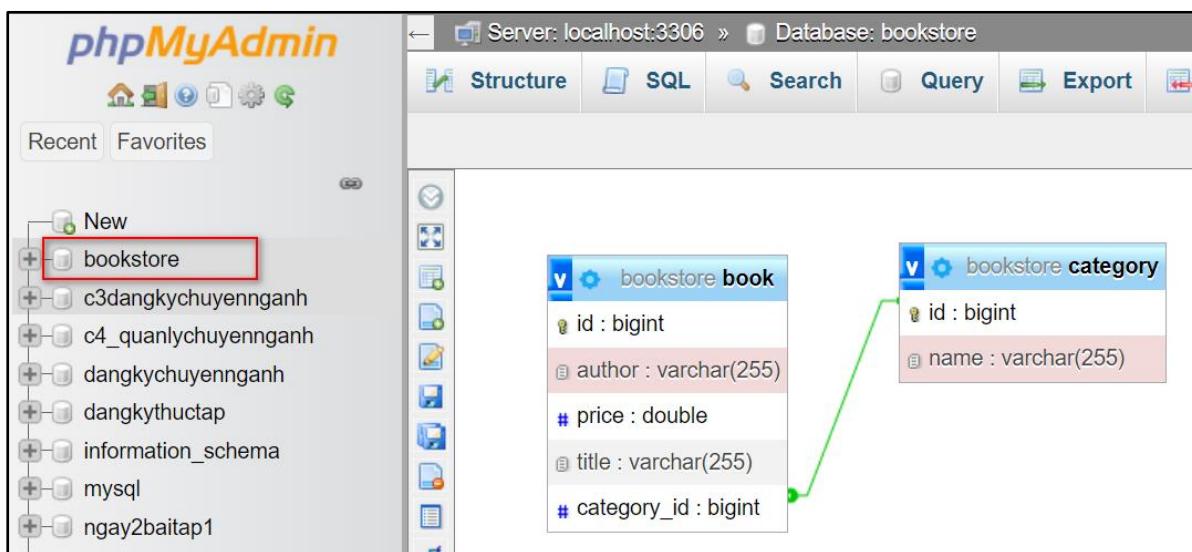
## 4.4 Build và chạy ứng dụng

Build và chạy ứng dụng tại địa chỉ **localhost:8080**



Hình 81. Build và chạy ứng dụng

Kiểm tra CSDL thông qua **phpmyadmin** <http://localhost/phpmyadmin>, hệ thống tự động tạo các bảng trong CSDL **BookStore** từ ánh xạ của model.



Hình 82. Add database bookstore

Tiếp theo, chúng ta thêm thông tin mẫu vào bảng Category, bảng Book trong cơ sở dữ liệu vừa tạo.

### Thêm dữ liệu cho bảng Category

id	name
1	CNTT
2	Kinh Tế

Hình 83. Thêm dữ liệu cho bảng Category

### Thêm dữ liệu cho bảng Book

id	author	price	title	category_id
1	Khoa CNTT HUTECH	100,000	Thực hành LT Java	1
2	Khoa Kinh Tế	200,000	Kinh Tế Vĩ Mô	2

Hình 84. Thêm dữ liệu cho bảng Book

Truy cập trang danh sách tại địa chỉ **localhost:8080/books**

My Book List						
HUTECH Home List Book Add Book						
ID	Title	Author	Price	Category	Action	
1	Thực hành LT Java	Khoa CNTT HUTECH	100000.0	CNTT	<a href="#">Edit</a>   <a href="#">Delete</a>	
2	Kinh Tế Vĩ Mô	Khoa Kinh Tế	200000.0	Kinh Tế	<a href="#">Edit</a>   <a href="#">Delete</a>	

Hình 85. Truy cập trang danh sách

## TÓM TẮT

Java Spring Framework là một framework phát triển ứng dụng web được viết bằng Java. Nó cung cấp các công cụ và thư viện để phát triển ứng dụng web hiệu quả và nhanh chóng. Spring Framework có nhiều tính năng mạnh mẽ bao gồm Dependency Injection, AOP, MVC, JDBC, Security và nhiều hơn nữa.

PHPMyAdmin là một công cụ quản lý cơ sở dữ liệu MySQL được viết bằng PHP. Nó cung cấp cho người dùng một giao diện web để quản lý các cơ sở dữ liệu MySQL. PHPMyAdmin cho phép người dùng thực hiện các tác vụ như tạo, xóa, sửa đổi các bảng dữ liệu, quản lý người dùng và phân quyền truy cập.

Khi sử dụng Java Spring Framework và PHPMyAdmin cùng nhau, người lập trình có thể phát triển các ứng dụng web mạnh mẽ với cơ sở dữ liệu MySQL và sử dụng các tính năng mạnh mẽ của Spring Framework để tăng hiệu suất và tăng tính bảo mật của ứng dụng.

## CÂU HỎI ÔN TẬP

**Câu 1:** PHPMyAdmin là gì? Nó được sử dụng để làm gì trong quản lý cơ sở dữ liệu MySQL?

**Câu 2:** Các tính năng chính của PHPMyAdmin là gì? Tại sao nó lại được ưa chuộng trong quản lý cơ sở dữ liệu MySQL?

**Câu 3:** Làm thế nào để sử dụng Java Spring Framework và PHPMyAdmin cùng nhau để phát triển một ứng dụng web?

# BÀI 5 XÂY DỰNG ỨNG DỤNG QUẢN LÝ SÁCH – CHỨC NĂNG THÊM, XÓA, SỬA

**Bài này giúp người học nắm được các nội dung sau:**

- ✓ Sử dụng trình Quản trị cơ sở dữ liệu MySQL phpMyAdmin
- ✓ Kết nối cơ sở dữ liệu Project với phpMyAdmin
- ✓ Xây dựng ứng dụng website quản lý sách với các chức năng giao tiếp cơ bản với Cơ sở dữ liệu: chức năng thêm, chức năng xóa, chức năng sửa.

**Mô tả chức năng:**

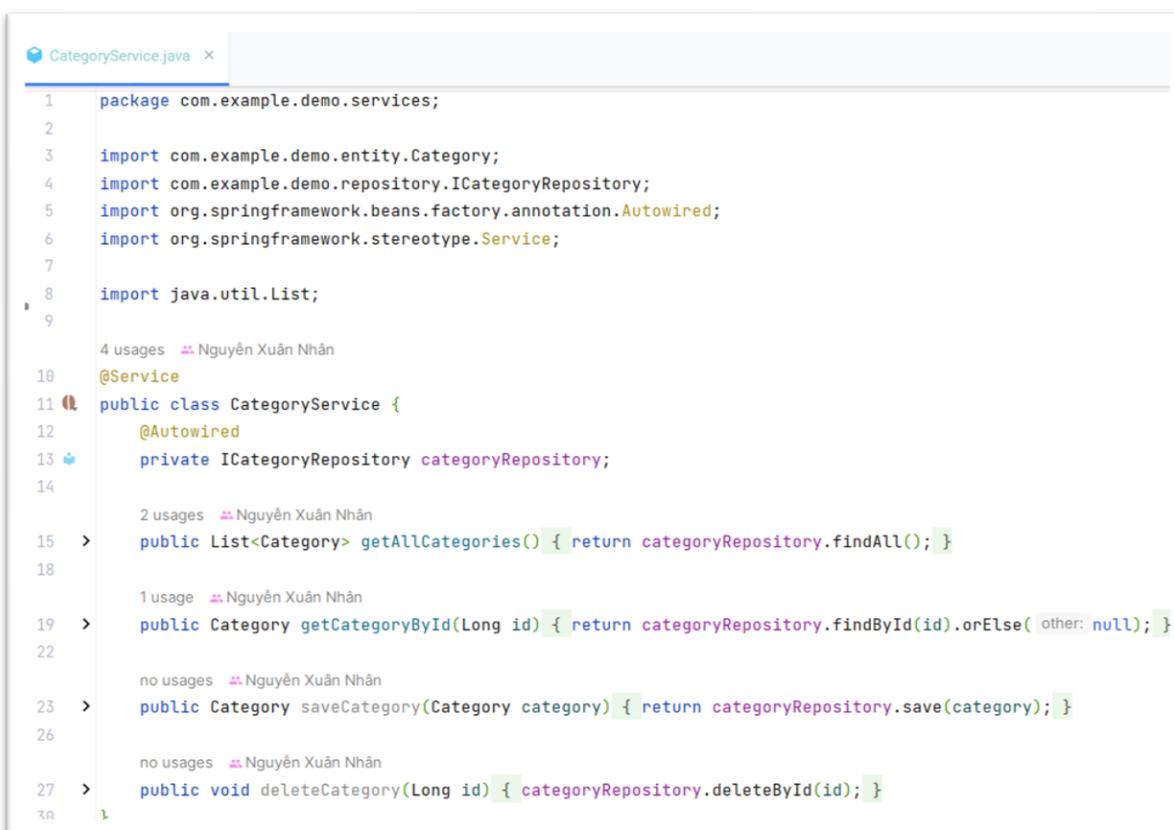
- ✓ **Thêm sách:** Cho phép người dùng nhập thông tin của một quyển sách gồm tiêu đề, tác giả, giá, danh mục của sách. Thông tin này sẽ được lưu vào một biến toàn cục trong ứng dụng
- ✓ **Xóa sách:** Cho phép người dùng chọn một quyển sách trong danh sách hiển thị và xóa khỏi danh sách.
- ✓ **Sửa sách:** Cho phép người dùng chọn một quyển sách trong danh sách hiển thị và cập nhật thông tin của nó.
- ✓ **Hiển thị sách:** Hiển thị danh sách tất cả các quyển sách có trong danh sách, bao gồm tiêu đề, tác giả, giá và danh mục của sách.

## 5.1 Viết trang thêm sách – Add Book

Đầu tiên chúng ta tạo thêm 1 file **CategoryService.java** đặt tại thư mục **src/main/java/com.example.demo/services**

Trong lớp **CategoryService.java**, các phương thức thường được định nghĩa để thao tác với cơ sở dữ liệu, như *thêm, sửa, xóa và truy vấn dữ liệu*. Lớp này cung cấp các chức năng để quản lý danh mục sản phẩm và đảm bảo tính nhất quán của dữ liệu trong toàn bộ ứng dụng web.

Thêm code như ảnh bên dưới:



```
CategoryService.java
1 package com.example.demo.services;
2
3 import com.example.demo.entity.Category;
4 import com.example.demo.repository.ICategoryRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import java.util.List;
9
10 import NguyenXuanNhien;
11 @Service
12 public class CategoryService {
13     @Autowired
14     private ICategoryRepository categoryRepository;
15
16     public List<Category> getAllCategories() { return categoryRepository.findAll(); }
17
18     public Category getCategoryById(Long id) { return categoryRepository.findById(id).orElse( other: null ); }
19
20     public Category saveCategory(Category category) { return categoryRepository.save(category); }
21
22     public void deleteCategory(Long id) { categoryRepository.deleteById(id); }
```

Hình 86. Tạo thêm file Category Services

### 4.1.1. Chỉnh sửa BookController

Tiến hành tinh chỉnh lại nội dung file BookController, thêm 2 phương thức để add book.



```

1 package com.example.demo.controller;
2
3 import com.example.demo.entity.Book;
4 import com.example.demo.services.BookService;
5 import com.example.demo.services.CategoryService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Controller;
8 import org.springframework.ui.Model;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.ModelAttribute;
11 import org.springframework.web.bind.annotation.PostMapping;
12 import org.springframework.web.bind.annotation.RequestMapping;
13
14 import java.util.List;
15
16 @Controller
17 @RequestMapping("/books")
18 public class BookController {
19     @Autowired
20     private BookService bookService;
21
22     @Autowired
23     private CategoryService categoryService;
24
25     @GetMapping
26     public String showAllBooks(Model model) {
27         List<Book> books = bookService.getAllBooks();
28         model.addAttribute(attributeName: "books", books);
29         return "book/list";
30     }
31
32     @GetMapping("/add")
33     public String addBookForm(Model model) {
34         model.addAttribute(attributeName: "book", new Book());
35         model.addAttribute(attributeName: "categories", categoryService.getAllCategories());
36         return "book/add";
37     }
38
39     @PostMapping("/add")
40     public String addBook(@ModelAttribute("book") Book book) {
41         bookService.addBook(book);
42         return "redirect:/books";
43     }
44 }

```

Hình 87. Chỉnh sửa nội dung trong file BookController.java

## Phương thức addBookForm và phương thức addBook

Đoạn code ở ảnh trên chứa **2 phương** thức dùng để Add thông tin Book.

Phương thức **addBookForm()** này trả về một trang HTML cho phép người dùng thêm mới một đối tượng Book vào cơ sở dữ liệu.

Phương thức **addBook()** này được sử dụng để xử lý dữ liệu đăng ký từ trang HTML "/add" được trả về bởi phương thức "addBookForm" trong câu hỏi trước đó.

**@GetMapping** và **@PostMapping** là các **annotation** được sử dụng để ánh xạ các yêu cầu HTTP GET và POST tương ứng đến phương thức xử lý của controller. Ở đây, phương thức addBookForm sẽ xử lý yêu cầu GET đến đường dẫn "/add", và phương thức addBook sẽ xử lý yêu cầu POST đến đường dẫn "/add".

```
5 import com.example.demo.services.CategoryService;
```

Sử dụng để import và sử dụng các phương thức và thuộc tính được định nghĩa trong class **CategoryService** trong các lớp khác của ứng dụng.

```
22 @Autowired  
23 private CategoryService categoryService;  
24
```

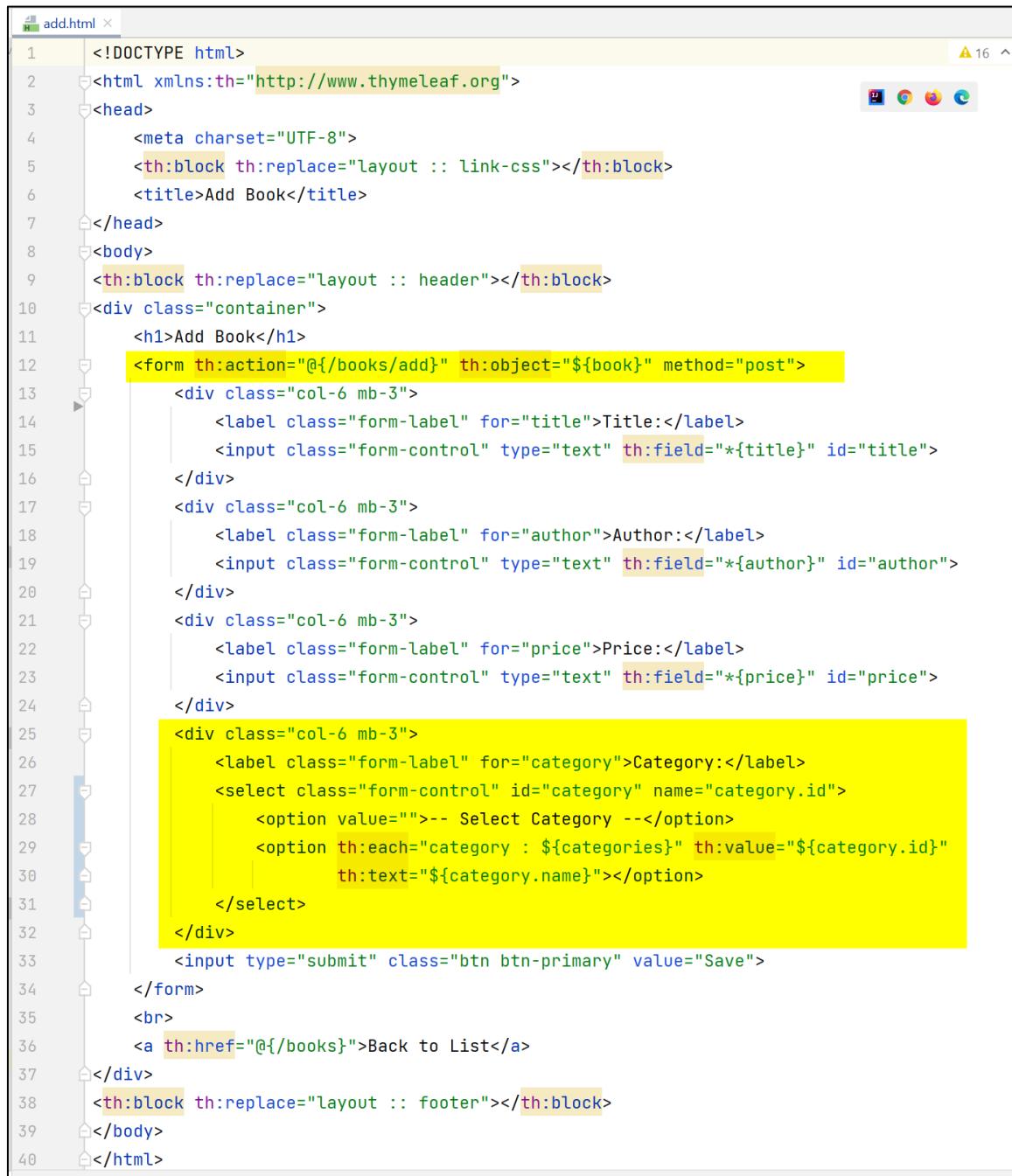
**@Autowired** được sử dụng để chú thích một thuộc tính trong một class

Đoạn code **@Autowired private CategoryService categoryService;** cho phép đối tượng implement (thực thi) CategoryService được tự động chèn vào thuộc tính **categoryService** của class hiện tại, để đối tượng này có thể được sử dụng trong các phương thức của class đó. Điều này giúp cho việc sử dụng CategoryService trở nên dễ dàng và tiện lợi hơn trong quá trình lập trình.

Tạo mới hoặc chỉnh sửa file **add.html** đặt tại đường dẫn  
**src/main/resources/templates/book**

Người dùng sẽ chọn một danh mục từ danh sách.

Bổ sung đoạn code như bên dưới.



```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <th:block th:replace="layout :: link-css"></th:block>
    <title>Add Book</title>
</head>
<body>
    <th:block th:replace="layout :: header"></th:block>
    <div class="container">
        <h1>Add Book</h1>
        <form th:action="@{/books/add}" th:object="${book}" method="post">
            <div class="col-6 mb-3">
                <label class="form-label" for="title">Title:</label>
                <input class="form-control" type="text" th:field="*{title}" id="title">
            </div>
            <div class="col-6 mb-3">
                <label class="form-label" for="author">Author:</label>
                <input class="form-control" type="text" th:field="*{author}" id="author">
            </div>
            <div class="col-6 mb-3">
                <label class="form-label" for="price">Price:</label>
                <input class="form-control" type="text" th:field="*{price}" id="price">
            </div>
            <div class="col-6 mb-3">
                <label class="form-label" for="category">Category:</label>
                <select class="form-control" id="category" name="category.id">
                    <option value="">-- Select Category --</option>
                    <option th:each="category : ${categories}" th:value="${category.id}" th:text="${category.name}"></option>
                </select>
            </div>
            <input type="submit" class="btn btn-primary" value="Save">
        </form>
        <br>
        <a th:href="@{/books}">Back to List</a>
    </div>
    <th:block th:replace="layout :: footer"></th:block>
</body>
</html>
```

Hình 88. Thêm nội dung vào file add.html

Build, chạy ứng dụng tại địa chỉ **localhost:8080/books/add**

The screenshot shows a web application interface for adding a book. At the top, there are navigation icons and a URL bar showing 'localhost:8080/books/add'. Below the header, the page title is 'HUTECH' followed by 'Home', 'List Book', and 'Add Book'. The main section is titled 'Add Book'. It contains four input fields: 'Title' with the value 'TH Lập trình UD Java', 'Author' with the value 'Bộ Môn CNPM - Khoa CNTT HUTECH', 'Price' with the value '80000', and 'Category' with the value 'CNTT'. A blue 'Save' button is located below the inputs. At the bottom left, there is a link 'Back to List'.

Hình 89. Giao diện add Book

Nhấn **Save** và xem kết quả.

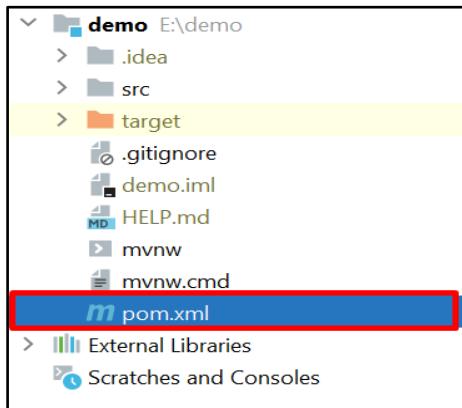
The screenshot shows a table of books. The columns are labeled 'ID', 'Title', 'Author', 'Price', 'Category', and 'Action'. There are seven rows of data. The last row, which was added, has the following values: ID 7, Title 'TH Lập trình UD Java', Author 'Bộ Môn CNPM - Khoa CNTT HUTECH', Price '80000.0', Category 'CNTT', and Action 'Edit | Delete'.

ID	Title	Author	Price	Category	Action
1	Thực hành LT Java	Khoa CNTT HUTECH	100000.0	CNTT	<a href="#">Edit</a>   <a href="#">Delete</a>
2	Kinh Tế Vĩ Mô	Khoa Kinh Tế	200000.0	Kinh Tế	<a href="#">Edit</a>   <a href="#">Delete</a>
3	ádf	sfasd	80000.0	CNTT	<a href="#">Edit</a>   <a href="#">Delete</a>
4	pppp	pppppp	9999.0	Kinh Tế	<a href="#">Edit</a>   <a href="#">Delete</a>
5	9	9	9.0	CNTT	<a href="#">Edit</a>   <a href="#">Delete</a>
7	TH Lập trình UD Java	Bộ Môn CNPM - Khoa CNTT HUTECH	80000.0	CNTT	<a href="#">Edit</a>   <a href="#">Delete</a>

Hình 90. Danh sách Book sau khi Add Book mới

## 5.2 Thêm ràng buộc khi lưu sách

Mở file **pom.xml** thêm **dependency**

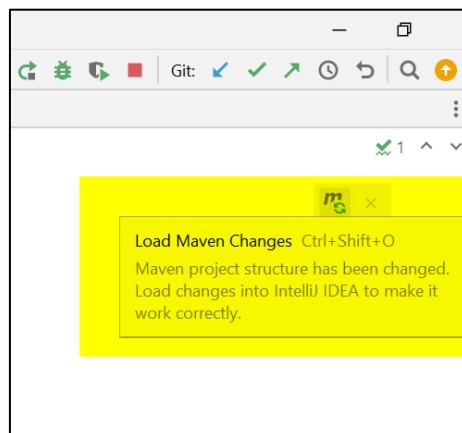


Hình 91. Thêm ràng buộc vào file pom.xml

Với Spring, khi chúng ta sử dụng dependency này, việc xác thực validation sẽ được tự động thực hiện trước khi các thao tác được thực hiện trong controller của ứng dụng. Nếu dữ liệu không hợp lệ, Spring sẽ trả về các thông báo lỗi tương ứng.

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

Tại góc phải màn hình của file **pom.xml**, cập nhật maven hoặc nhấn **CTRL + SHIFT + O**

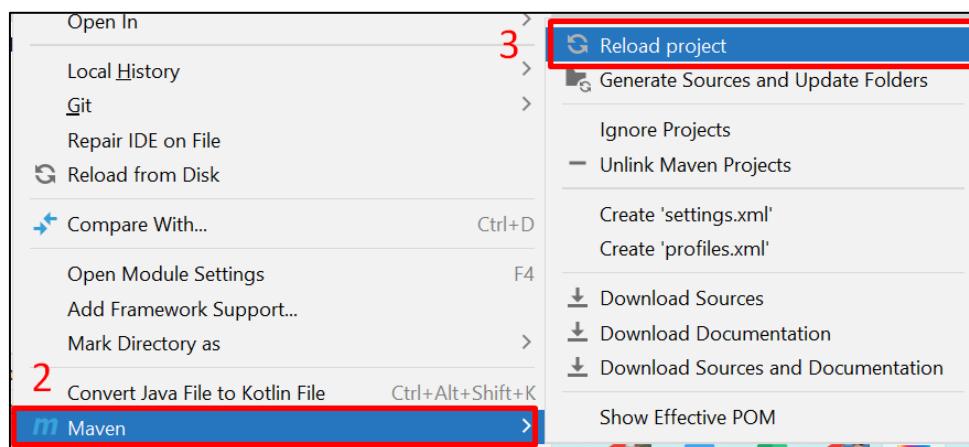


Hình 92. Cập nhật maven

Nếu vẫn còn có lỗi xuất hiện, thì rất có thể là **dependencies** chưa cập nhập.

Vậy để cập nhập làm theo các bước sau.

- Bước 1: Chọn **chuột phải vào dự án (Project)** của mình
- Bước 2: Tìm và chọn vào **Maven**
- Bước 3: Chọn **Reload project**



Hình 93. **Reload project** Maven

Tạo 1 thư mục **Validator** đặt tại **src/main/java/com.example.demo**

Tạo file **ValidCategoryIdValidator.java** đặt tại đường dẫn

**src/main/java/com.example.demo/Validator**

Hàm **isValid()** Kiểm tra các đối tượng "Category" được sử dụng trong hệ thống là hợp lệ và có tồn tại ID không.

```

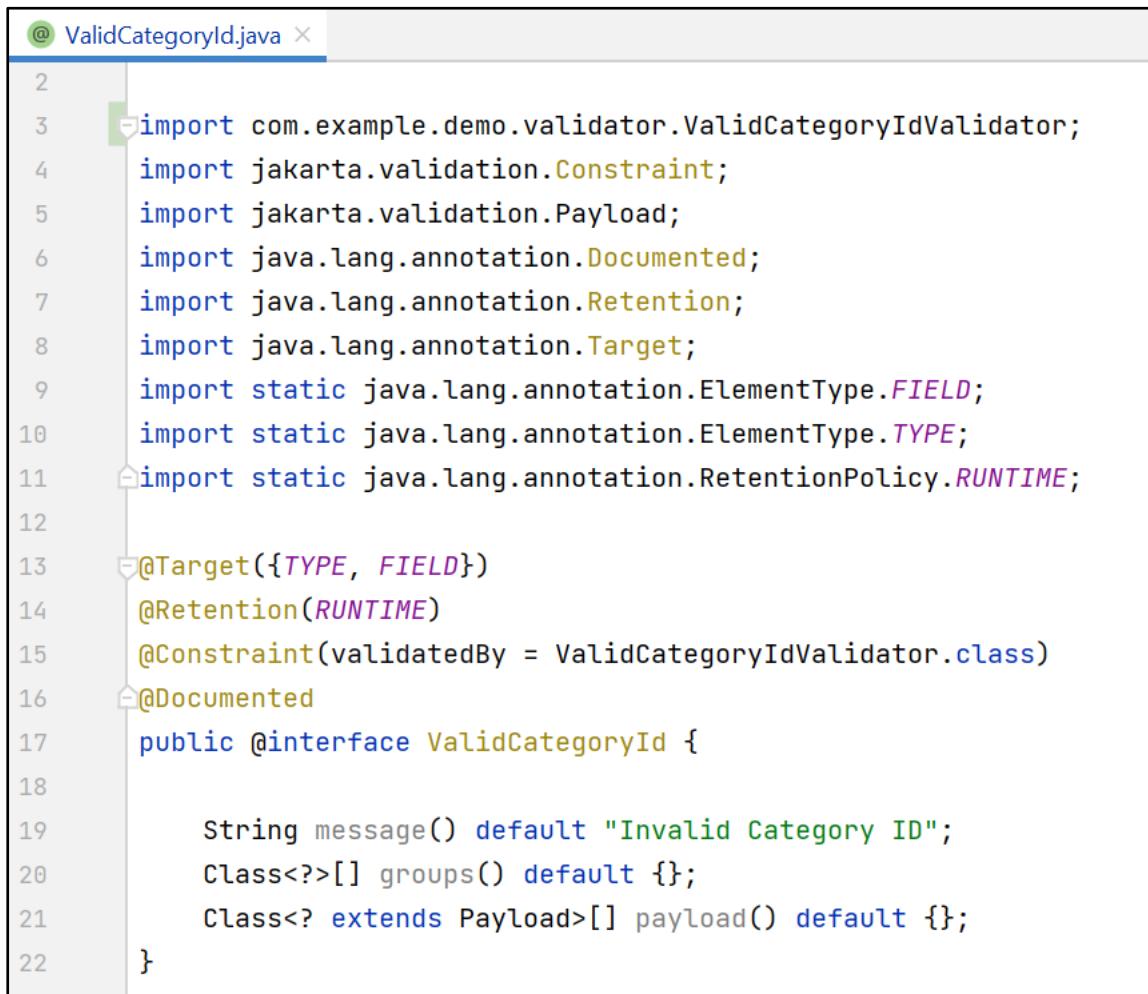
ValidCategoryIdValidator.java
1 package com.example.demo.validator;
2
3 import com.example.demo.entity.Category;
4 import com.example.demo.validator.annotation.ValidCategoryId;
5 import jakarta.validation.ConstraintValidator;
6 import jakarta.validation.ConstraintValidatorContext;
7
8 public class ValidCategoryIdValidator implements ConstraintValidator<ValidCategoryId, Category> {
9     @Override
10    public boolean isValid(Category category, ConstraintValidatorContext context) {
11        return category != null && category.getId() != null;
12    }
13}

```

Hình 94. Kiểm tra tính hợp lệ của một đối tượng "Category"

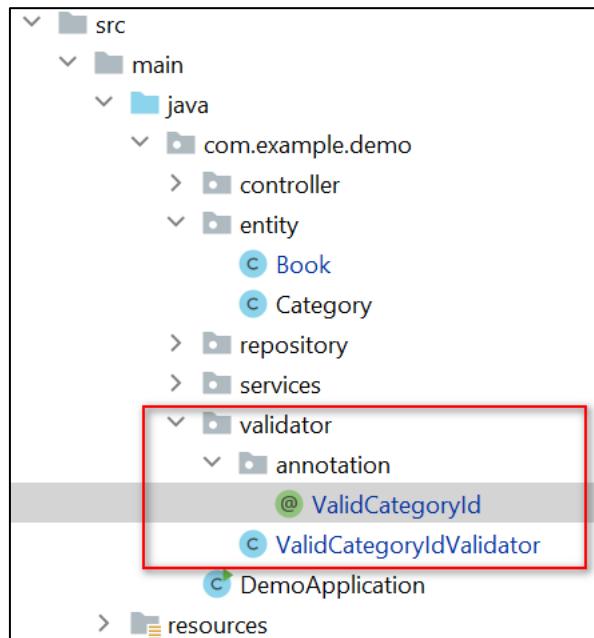
Tiếp theo, Tạo thư mục **annotation** (*thư mục con của validator*) đặt tại đường dẫn **src/main/java/com.example.demo/validator**

Tạo file **ValidCategoryId.java** đặt tại đường dẫn  
**src/main/java/com.example.demo/validator/annotation**



```
2
3 import com.example.demo.validator.ValidCategoryIdValidator;
4 import jakarta.validation.Constraint;
5 import jakarta.validation.Payload;
6 import java.lang.annotation.Documented;
7 import java.lang.annotation.Retention;
8 import java.lang.annotation.Target;
9 import static java.lang.annotation.ElementType.FIELD;
10 import static java.lang.annotation.ElementType.TYPE;
11 import static java.lang.annotation.RetentionPolicy.RUNTIME;
12
13 @Target({TYPE, FIELD})
14 @Retention(RUNTIME)
15 @Constraint(validatedBy = ValidCategoryIdValidator.class)
16 @Documented
17 public @interface ValidCategoryId {
18
19     String message() default "Invalid Category ID";
20     Class<?>[] groups() default {};
21     Class<? extends Payload>[] payload() default {};
22 }
```

Cấu trúc thư mục Validator chưa annotation sẽ như bên dưới.



Chỉnh sửa file **Book.java** trên đường dẫn  
**src/main/java/com/example/demo/entity**

Thêm các **annotation** để ràng buộc việc nhập liệu.

**Annotation** được hiểu là một dạng chú thích hoặc một dạng siêu dữ liệu (metadata) được dùng để cung cấp thông tin dữ liệu cho mã nguồn Java. Các chú thích không có ảnh hưởng trực tiếp đến hoạt động của mã mà chúng chú thích.

```
c Book.java ×
  5 import jakarta.validation.constraints.*;
  6 import lombok.Data;
  7 @Data
  8 @Entity
  9 @Table(name = "book")
10 public class Book {
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14
15     @Column(name = "title")
16     @NotEmpty(message = "Title must not be empty")
17     @Size(max = 50, min = 1, message = "Title must be less than 50 characters")
18     private String title;
19
20     @Column(name = "author")
21     private String author;
22
23     @Column(name = "price")
24     @NotNull(message = "Price is required")
25     private Double price;
26
27     @ManyToOne
28     @JoinColumn(name = "category_id")
29     @ValidCategoryId
30     private Category category;
31 }
```

Hình 95. Thêm các annotation để ràng buộc việc nhập liệu

### Chỉnh sửa file Add.html view đặt tại **templates/book/add.html**



```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <th:block th:replace="layout :: link-css"></th:block>
    <title>Add Book</title>
</head>
<body>
    <th:block th:replace="layout :: header"></th:block>
    <div class="container">
        <h1>Add Book</h1>
        <form th:action="@{/books/add}" th:object="${book}" method="post">
            <div class="col-6 mb-3">
                <label class="form-label" for="title">Title:</label>
                <input class="form-control" type="text" th:field="*{title}" id="title">
                <span class="text-danger" th:if="#{fields.hasErrors('title')}" th:errors="*{title}"></span>
            </div>
            <div class="col-6 mb-3">
                <label class="form-label" for="author">Author:</label>
                <input class="form-control" type="text" th:field="*{author}" id="author">
            </div>
            <div class="col-6 mb-3">
                <label class="form-label" for="price">Price:</label>
                <input class="form-control" type="text" th:field="*{price}" id="price">
                <span class="text-danger" th:if="#{fields.hasErrors('price')}" th:errors="*{price}"></span>
            </div>
            <div class="col-6 mb-3">
                <label class="form-label" for="category">Category:</label>
                <select class="form-control" id="category" name="category.id">
                    <option value="">-- Select Category --</option>
                    <option th:each="category : ${categories}" th:value="${category.id}"
                           th:text="${category.name}"></option>
                </select>
                <span class="text-danger" th:if="#{fields.hasErrors('category')}" th:errors="*{category}"></span>
            </div>
            <input type="submit" class="btn btn-primary" value="Save">
        </form>
        <br>
        <a th:href="@{/books}">Back to List</a>
    </div>
    <th:block th:replace="layout :: footer"></th:block>
</body>
</html>
```

Hình 96. Chỉnh sửa file Add.html

Build, chạy ứng dụng và truy cập tới trang <http://localhost:8080/books/add>

Không nhập nhưng vẫn nhấn nút **SAVE** để kiểm tra ràng buộc

The screenshot shows a web browser window with the URL [localhost:8080/books/add](http://localhost:8080/books/add) in the address bar. The page title is "HUTECH". Below it, there are links for "Home", "List Book", and "Add Book". The main content is titled "Add Book". There are four input fields: "Title", "Author", "Price", and "Category". Each field has a red validation message below it. A blue "Save" button is at the bottom.

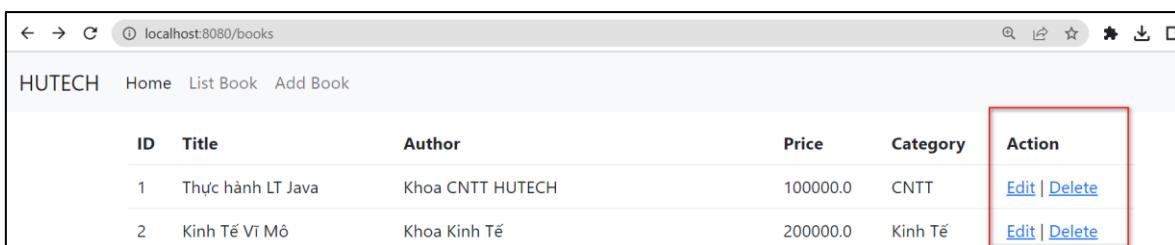
Field	Validation Message
Title	Title must not be empty Title must be less than 50 characters
Author	(empty)
Price	Price is required
Category	-- Select Category -- Invalid Category ID

**Save**

Hình 97. Kiểm tra ràng buộc

### Câu hỏi bài tập cần thực hiện:

Sinh viên tự thực hiện trang **Edit** và trang **Delete**, tương tự như trên



The screenshot shows a web browser window with the URL 'localhost:8080/books'. The page title is 'HUTECH Home List Book Add Book'. Below the title is a table with the following data:

ID	Title	Author	Price	Category	Action
1	Thực hành LT Java	Khoa CNTT HUTECH	100000.0	CNTT	<a href="#">Edit</a>   <a href="#">Delete</a>
2	Kinh Tế Vĩ Mô	Khoa Kinh Tế	200000.0	Kinh Tế	<a href="#">Edit</a>   <a href="#">Delete</a>

Hình 98. Thực hiện chức năng Edit và Delete

## 5.3 Gợi ý chức năng Edit và delete Book

### 4.3.1. Delete Book

Tìm Book dựa trên Id truyền vào.

```
Book book = bookService.getBookById(id);
```

Xoá Book dựa vào Book có Id vừa tìm được

```
bookService.deleteBook(id);
```

### 4.3.1. Edit Book

Tạo 2 phương thức:

Trong **phương thức GET**, truyền vào đường dẫn có định dạng `/edit/{id}` để lấy thông tin của đầu sách cần chỉnh sửa. Sau đó, chúng ta thêm đối tượng book và danh sách các category vào Model và trả về trang view để hiển thị thông tin đầu sách cần chỉnh sửa.

Trong **phương thức POST**, chúng ta cũng truyền vào đường dẫn có định dạng `/edit/{id}` để xác định đầu sách cần cập nhật thông tin, tiếp chúng ta cập nhật thông tin của đầu sách và chuyển hướng người dùng đến trang danh sách đầu sách.

## TÓM TẮT

PHPMyAdmin là một công cụ quản lý cơ sở dữ liệu MySQL được viết bằng PHP. Nó cung cấp cho người dùng một giao diện web để quản lý các cơ sở dữ liệu MySQL. PHPMyAdmin cho phép người dùng thực hiện các tác vụ như tạo, xóa, sửa đổi các bảng dữ liệu, quản lý người dùng và phân quyền truy cập.

Trong Java Spring framework, chức năng thêm, xoá, sửa dữ liệu được thực hiện thông qua các tương tác với cơ sở dữ liệu.

Để thêm mới dữ liệu, người lập trình có thể sử dụng các phương thức được cung cấp bởi framework, chẳng hạn như save() hoặc saveAll() để lưu đổi tượng mới vào cơ sở dữ liệu.

Để xoá dữ liệu, người lập trình cần truy vấn đến đổi tượng cần xoá, sau đó sử dụng phương thức delete() để xoá đổi tượng đó khỏi cơ sở dữ liệu.

Để sửa đổi dữ liệu, người lập trình có thể truy vấn đến đổi tượng cần sửa đổi và thực hiện các thay đổi trên đổi tượng đó. Sau đó, sử dụng phương thức save() để cập nhật thông tin đã thay đổi vào cơ sở dữ liệu.

## CÂU HỎI ÔN TẬP

1. Làm thế nào để thêm mới dữ liệu vào cơ sở dữ liệu trong Java Spring framework?
2. Có bao nhiêu phương thức dùng để xoá dữ liệu trong Java Spring framework? Và chúng khác nhau như thế nào?
3. Làm thế nào để sửa đổi dữ liệu trong cơ sở dữ liệu bằng Java Spring framework?
4. Làm thế nào để kiểm tra tính hợp lệ của dữ liệu trước khi lưu vào cơ sở dữ liệu?

# BÀI 6 XÂY DỰNG CHỨC NĂNG TẠO TÀI KHOẢN VÀ ĐĂNG NHẬP CHO NGƯỜI DÙNG

**Bài này giúp người học nắm được các nội dung sau:**

Thực hiện được chứng năng đăng nhập và tạo tài khoản người dùng. Mục tiêu phát triển một chức năng đăng ký, an toàn và linh hoạt cho người dùng. Chức năng này sẽ cho phép người dùng tạo tài khoản mới, đặt mật khẩu và xác nhận mật khẩu, nhập thông tin cá nhân, và xác thực thông tin của họ. Điều này sẽ giúp chúng ta đảm bảo tính toàn vẹn và bảo mật thông tin cá nhân của người dùng.

## Mô tả chức năng:

Thêm chức năng đăng ký vào tạo tài khoản người dùng trên Spring Boot là một mục tiêu quan trọng để đáp ứng nhu cầu của người dùng và cải thiện trải nghiệm của họ khi sử dụng ứng dụng. Với chức năng đăng ký, người dùng có thể tạo tài khoản cá nhân và lưu trữ thông tin của mình trên hệ thống của chúng ta.

Thêm vào đó, việc thêm chức năng đăng ký sẽ giúp chúng ta thu thập thông tin từ người dùng. Chúng ta có thể sử dụng thông tin đăng ký của người dùng để tùy chỉnh trải nghiệm của họ, cung cấp các tính năng mới, và nâng cao chất lượng dịch vụ của chúng ta.

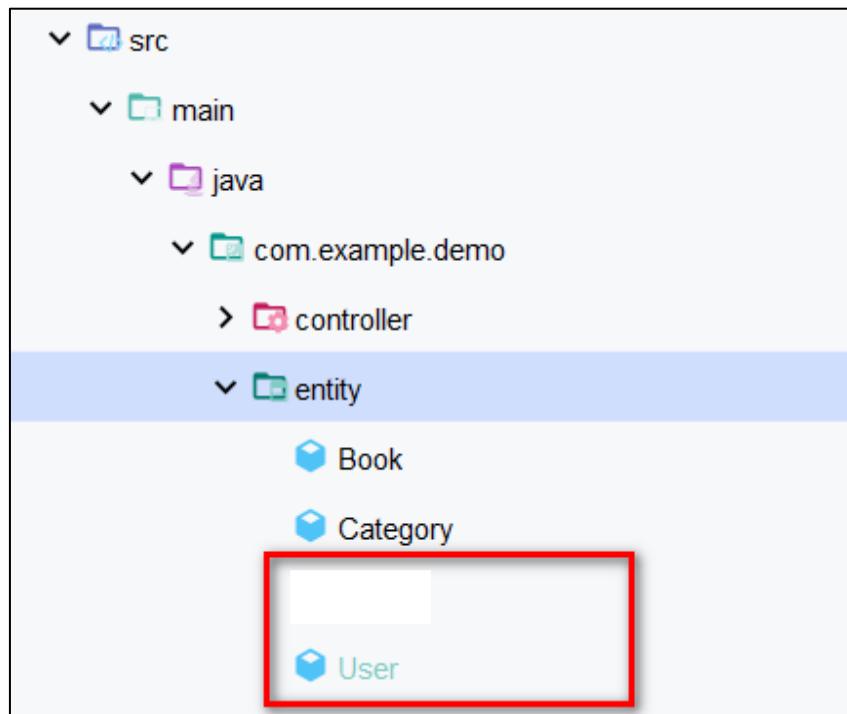
Vì vậy, mục tiêu của chúng ta khi thêm chức năng đăng ký vào tạo tài khoản người dùng trên Spring Boot là tạo ra một trải nghiệm dễ sử dụng và bảo mật cho người dùng, thu thập thông tin và cải thiện tính năng của ứng dụng của chúng ta, và đáp ứng nhu cầu của người dùng.

## 6.1 Xây dựng thêm table mới là User

Tạo class **User.java** trong thư mục **Entity** trên đường dẫn  
**src/main/java/com.example.demo/entity**

**User.java** lớp Java để xác định thông tin người dùng và quyền truy cập.

Lớp **User.java** được sử dụng để đại diện cho thông tin người dùng, bao gồm các thuộc tính như tên đăng nhập, mật khẩu, địa chỉ email và các thông tin cá nhân khác. Nó cũng có thể chứa các phương thức để xử lý các hoạt động liên quan đến người dùng như đăng ký, đăng nhập, cập nhật thông tin người dùng và quên mật khẩu.



Hình 99. Tạo 2 class User trong thư mục entity

Thêm nội dung code vào file **User.java** như bên dưới

```
package com.example.demo.entity;

import com.example.demo.validator.annotation.ValidUsername;
import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;
import lombok.Data;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Data
@Entity
@Table(name = "user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "username", length = 50, nullable = false, unique = true)
    @NotBlank(message = "Username is required")
    @Size(max = 50, message = "Username must be less than 50 characters")
    @ValidUsername
    private String username;

    @Column(name = "password", length = 250, nullable = false)
    @NotBlank(message = "Password is required")
    private String password;

    @Column(name = "email", length = 50)
    @Size(max = 50, message = "Email must be less than 50 characters")
    private String email;

    @Column(name = "name", length = 50, nullable = false)
    @Size(max = 50, message = "Your name must be less than 50 characters")
    @NotBlank(message = "Your name is required")
    private String name;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private List<Book> books = new ArrayList<>();
}
```

Hình 100. Thêm nội dung vào file User.java

Nếu có lỗi đỏ xuất hiện như bên dưới, là do chưa khai báo hoặc định nghĩa **ValidUsername**.

The screenshot shows a Java code editor with a file named 'User.java'. The code is as follows:

```
1 package com.example.demo.entity;
2
3 import com.example.demo.validator.annotation.ValidUsername; // Error
4 import jakarta.persistence.*;
5 import jakarta.validation.constraints.NotBlank;
6 import jakarta.validation.constraints.Size;
7 import lombok.Data;
8
9 import java.util.ArrayList;
10 import java.util.HashSet;
11 import java.util.List;
12 import java.util.Set;
13
```

The line 'import com.example.demo.validator.annotation.ValidUsername;' is highlighted with a red box, indicating a compilation error. A yellow circular icon is also present near the start of the line.

Hình 101. Thông báo lỗi chưa định nghĩa lớp ValidUsername

**ValidUsername** dùng để kiểm tra tính hợp lệ của tên người dùng trong ứng dụng. Có thể được định nghĩa là một chuỗi ký tự chỉ chứa các ký tự chữ cái, số và dấu gạch dưới, không được bắt đầu bằng số hoặc dấu gạch dưới, và không quá dài hoặc quá ngắn.

Tiến hành tạo thêm file tên là **ValidUsername.java** trong thư mục annotation theo đường dẫn

**src/main/java/com/example/demo/validator/annotation**

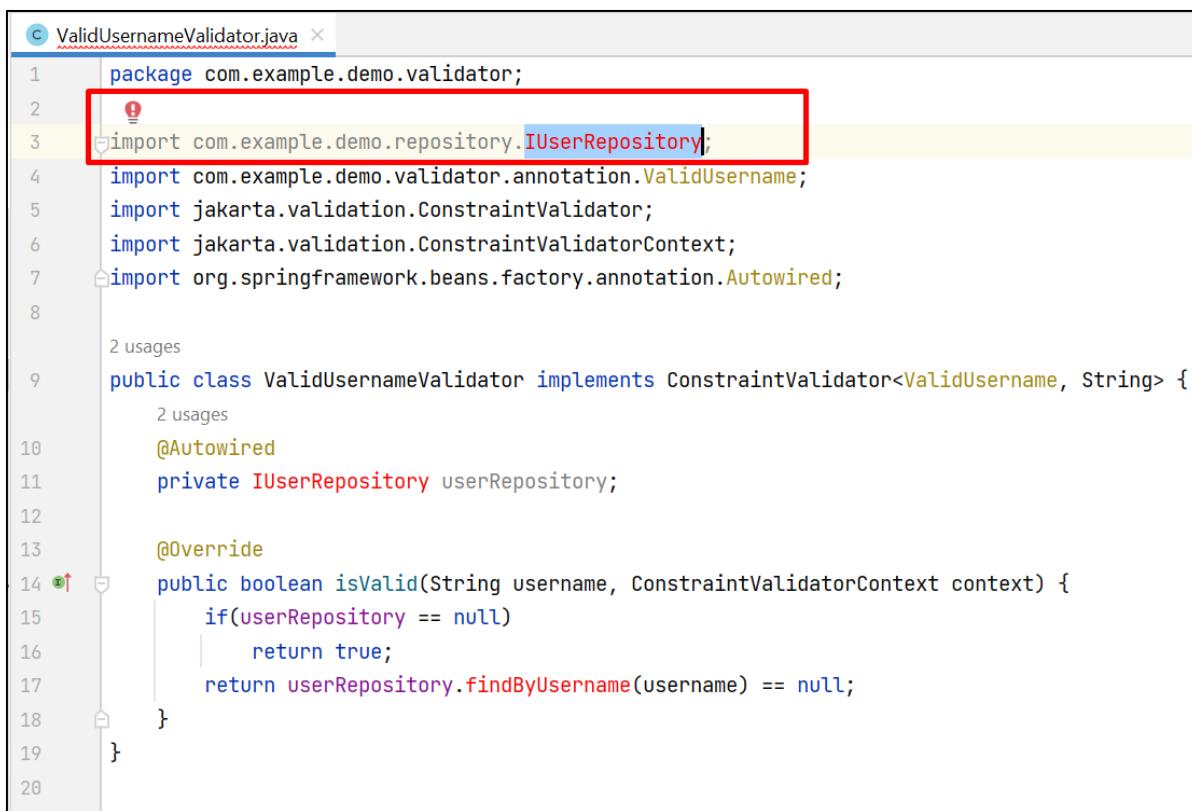
```
1 package com.example.demo.validator.annotation;
2
3 import com.example.demo.validator.ValidUsernameValidator;
4 import jakarta.validation.Constraint;
5 import jakarta.validation.Payload;
6
7 import java.lang.annotation.Retention;
8 import java.lang.annotation.Target;
9
10 import static java.lang.annotation.ElementType.FIELD;
11 import static java.lang.annotation.ElementType.TYPE;
12 import static java.lang.annotation.RetentionPolicy.RUNTIME;
13
14 @Target({TYPE, FIELD})
15 @Retention(RUNTIME)
16 @Constraint(validatedBy = ValidUsernameValidator.class)
17 public @interface ValidUsername {
18     String message() default "Username already exists";
19     Class<?>[] groups() default {};
20     Class<? extends Payload>[] payload() default {};
21 }
```

Hình 102. Thêm nội dung cho class ValidUsername.java

Trong đoạn code trong file **ValidUsername.java** có thông báo thiếu class **ValidUsernameValidator**.

Nếu chúng ta tiến hành bổ sung thêm class **ValidUsernameValidator.java** vào thư mục **validator** theo đường dẫn sau

**src/main/java/com/example/demo/validator**



```
ValidUsernameValidator.java
1 package com.example.demo.validator;
2
3 import com.example.demo.repository.IUserRepository; // Red box highlights this line
4 import com.example.demo.validator.annotation.ValidUsername;
5 import jakarta.validation.ConstraintValidator;
6 import jakarta.validation.ConstraintValidatorContext;
7 import org.springframework.beans.factory.annotation.Autowired;
8
9 public class ValidUsernameValidator implements ConstraintValidator<ValidUsername, String> {
10     @Autowired
11     private IUserRepository userRepository;
12
13     @Override
14     public boolean isValid(String username, ConstraintValidatorContext context) {
15         if(userRepository == null)
16             return true;
17         return userRepository.findByUsername(username) == null;
18     }
19 }
20
```

Hình 103. Tạo thêm class ValidUsernameValidator.java

**IUserRepository** là một interface trong Java, thường được sử dụng trong kiến trúc phần mềm định hướng đối tượng (OOP) để truy xuất và lưu trữ dữ liệu người dùng.

Tại 1 file tên **IuserRepository.java** trong thư mục theo đường dẫn  
**src/main/java/com/example/demo/repository**



```
1 package com.example.demo.repository;
2
3 import com.example.demo.entity.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.stereotype.Repository;
7
8
9 @Repository
10 public interface I UserRepository extends JpaRepository<User, Long> {
11     @Query("SELECT u FROM User u WHERE u.username = ?1")
12     User findByUsername(String username);
13 }
```

Hình 104. Tạo 1 file tên IuserRepository.java

Tiến hành chỉnh sửa tiếp tại file Book trong thư mục **entity** theo thư mục **src/main/java/com/example/demo/entity**.

Cụ thể, thêm thuộc tính user vào lớp com.example.demo.entity.Book bởi vì trong lớp com.example.demo.entity.User, một thuộc tính books đã được định nghĩa với **mappedBy** là user. Điều này cho biết rằng, quan hệ giữa hai lớp này là quan hệ một-nhiều, trong đó một đối tượng User có thể có nhiều đối tượng Book tương ứng.

```

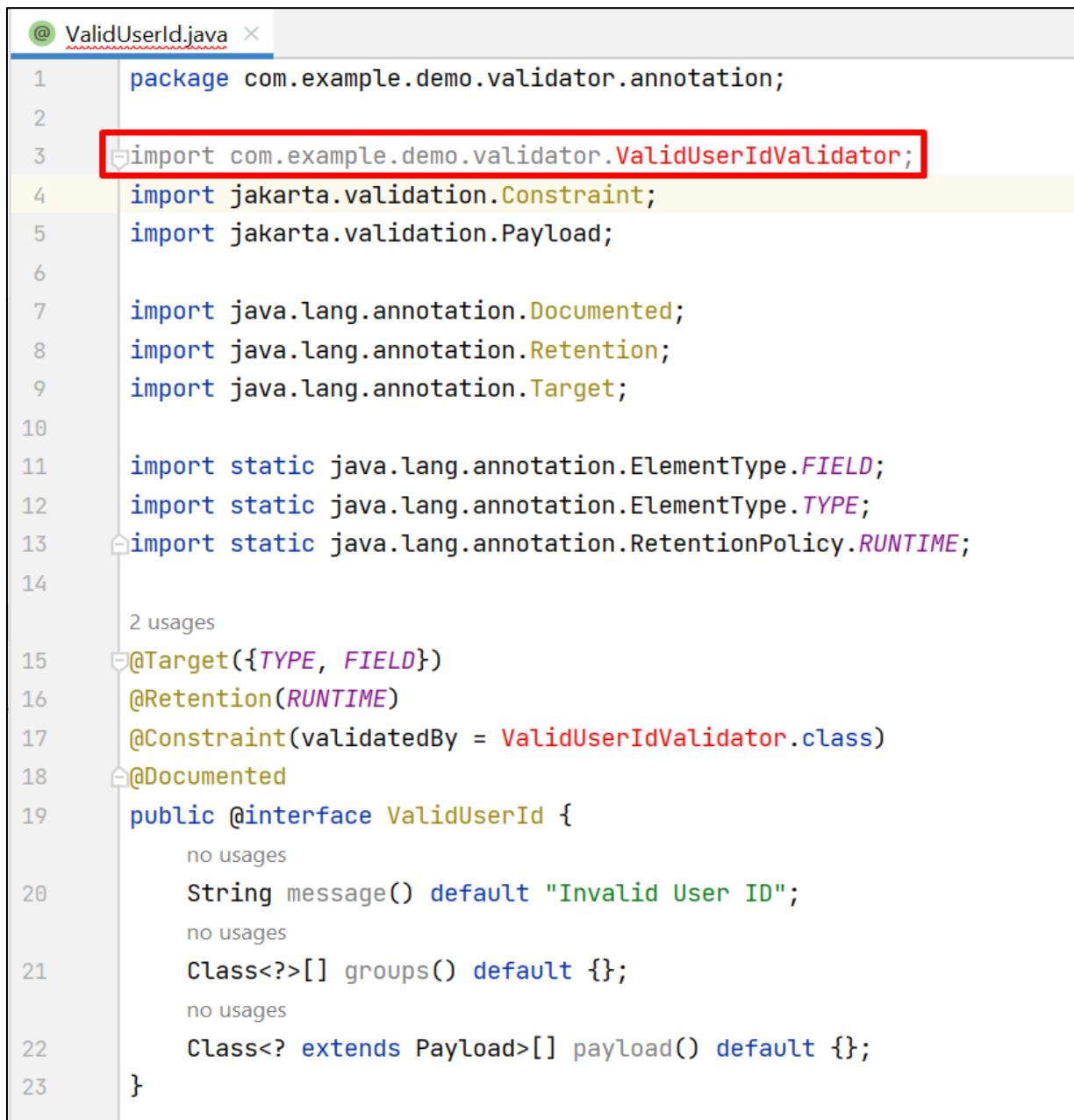
Book.java ×
1 package com.example.demo.entity;
2
3 import com.example.demo.validator.annotation.ValidCategoryId;
4 import com.example.demo.validator.annotation.ValidUserId;
5 import jakarta.persistence.*;
6 import jakarta.validation.constraints.*;
7 import lombok.Data;
8
9 @Data
10 @Entity
11 @Table(name = "book")
12 public class Book {
13     no usages
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Long id;
17
18     no usages
19     @Column(name = "title", length = 50)
20     @Size(max = 50, message = "Title must be less than 50 characters")
21     @NotNull(message = "Title must not be null")
22     private String title;
23
24     no usages
25     @Column(name = "author", length = 50)
26     @Size(max = 50, message = "Author must be less than 50 characters")
27     private String author;
28
29     no usages
30     @Column(name = "price")
31     @NotNull(message = "Price is required")
32     @Positive(message = "Price must be greater than 0")
33     private Double price;
34
35     no usages
36     @ManyToOne
37     @JoinColumn(name = "category_id", referencedColumnName = "id")
38     @ValidCategoryId
39     private Category category;
40
41     no usages
42     @ManyToOne
43     @JoinColumn(name = "user_id", referencedColumnName = "id")
44     @ValidUserId
45     private User user;
}

```

Hình 105. Thêm thuộc tính user vào Book

Tiếp theo ta cần thêm 1 file tên là **ValidUserId.java** vào thư mục annotation theo đường dẫn

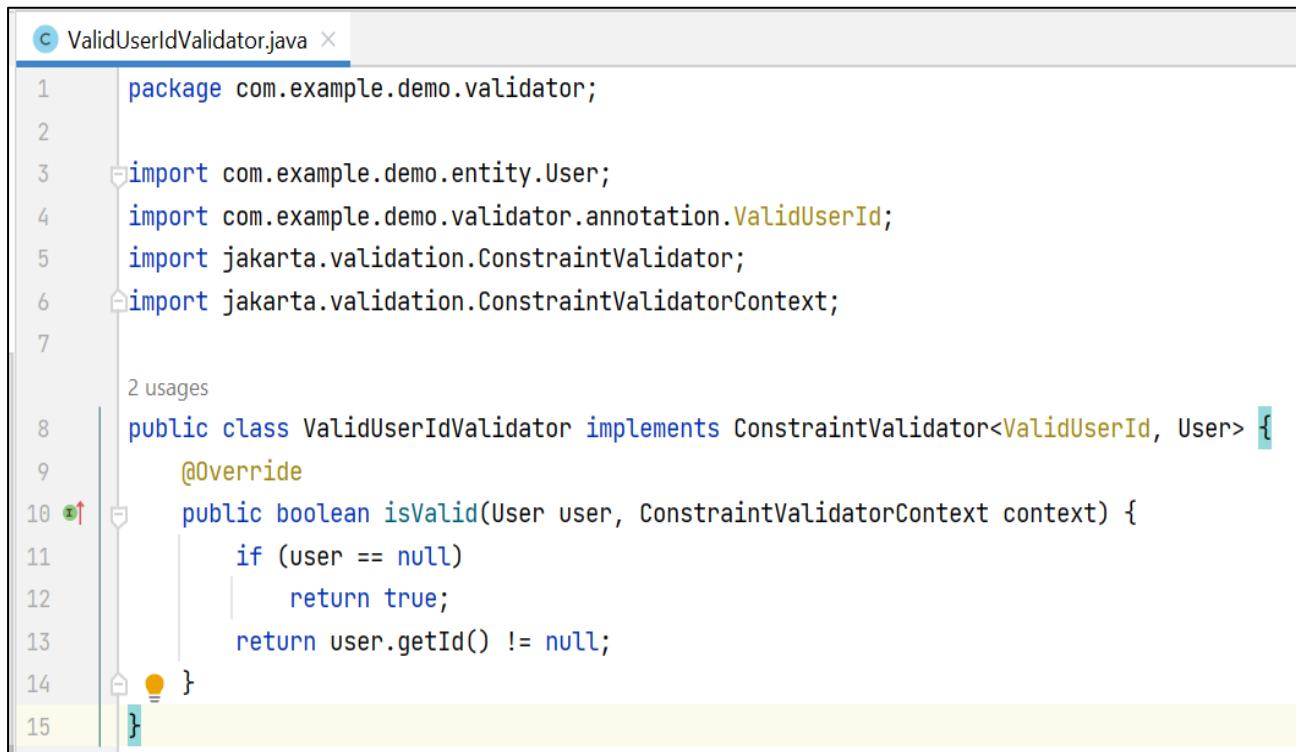
**src/main/java/com/example/demo/validator/annotation**



```
1 package com.example.demo.validator.annotation;
2
3 import com.example.demo.validator.ValidUserIdValidator;
4 import jakarta.validation.Constraint;
5 import jakarta.validation.Payload;
6
7 import java.lang.annotation.Documented;
8 import java.lang.annotation.Retention;
9 import java.lang.annotation.Target;
10
11 import static java.lang.annotation.ElementType.FIELD;
12 import static java.lang.annotation.ElementType.TYPE;
13 import static java.lang.annotation.RetentionPolicy.RUNTIME;
14
15 @Target({TYPE, FIELD})
16 @Retention(RUNTIME)
17 @Constraint(validatedBy = ValidUserIdValidator.class)
18 @Documented
19 public @interface ValidUserId {
20     String message() default "Invalid User ID";
21     Class<?>[] groups() default {};
22     Class<? extends Payload>[] payload() default {};
23 }
```

Hình 106. thêm file ValidUserId.java

Tiếp tục, tạo thêm 1 file tên là ValidUserIdValidator.java tại thư mục validator **src/main/java/com/example/demo/validator**



```

1 package com.example.demo.validator;
2
3 import com.example.demo.entity.User;
4 import com.example.demo.validator.annotation.ValidUserId;
5 import jakarta.validation.ConstraintValidator;
6 import jakarta.validation.ConstraintValidatorContext;
7
8 public class ValidUserIdValidator implements ConstraintValidator<ValidUserId, User> {
9     @Override
10    public boolean isValid(User user, ConstraintValidatorContext context) {
11        if (user == null)
12            return true;
13        return user.getId() != null;
14    }
15}

```

Hình 107. ValidUserIdValidator.java tại thư mục validator

Kiểm tra lại các class và interface đã thêm, build lại project (nhấn Ctrl + F9)

Mở **phpMyAdmin** bằng cách truy cập <http://localhost/phpmyadmin/>

Kiểm tra Database sẽ xuất hiện thêm các table đã thêm tương ứng User.

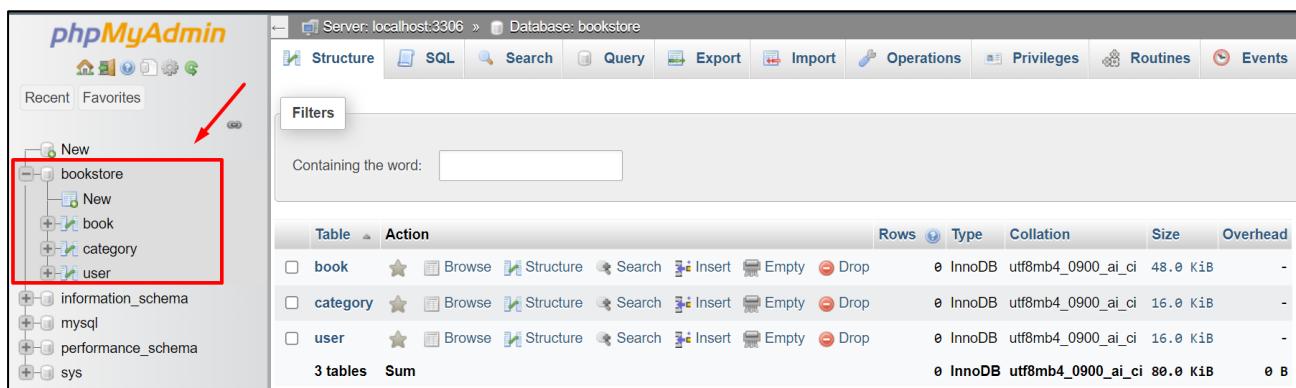


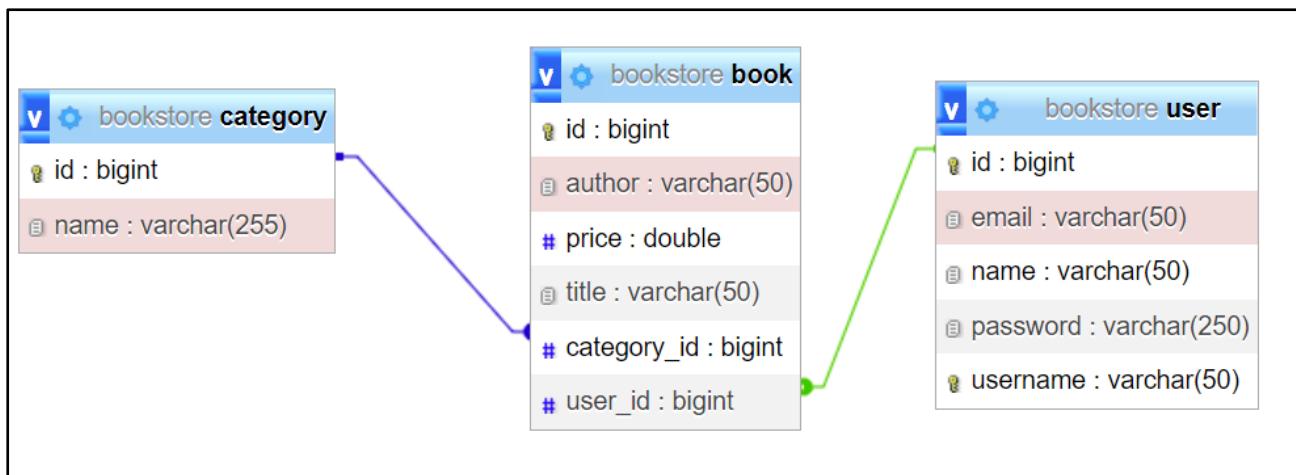
Table	Action	Rows	Type	Collation	Size	Overhead
book	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_0900_ai_ci	48.0 KiB	-
category	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_0900_ai_ci	16.0 KiB	-
user	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_0900_ai_ci	16.0 KiB	-

Hình 108. Kiểm tra database sau khi thêm table User

The screenshot shows the phpMyAdmin interface for the 'bookstore' database. On the left, the schema tree is visible with a red box labeled '1' around the 'bookstore' node. At the top right, there's a 'More' dropdown menu with a red box labeled '2'. Below it, under the 'Triggers' section, there's a 'Designer' link with a red box labeled '3'.

Table	Action	Rows	Type	Collation	Size	Overhead
book	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_0900_ai_ci	48.0 Kib	-
category	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-
user	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_0900_ai_ci	16.0 Kib	-

Hình 109. Mở Designer và xem kết quả



Hình 110. Sơ đồ Designer của database bookstore

Tiếp theo, tạo thêm class mới tên là **UserService.java** trong thư mục services theo đường dẫn **src/main/java/com/example/demo/services**.

Lớp **UserService** có mục đích cung cấp các phương thức và chức năng để quản lý người dùng và các vai trò của họ trong hệ thống.

```

1 package com.example.demo.services;
2
3 import com.example.demo.entity.User;
4 import com.example.demo.repository.IUserRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 @Service
9 public class UserService {
10
11     @Autowired
12     private IUserRepository userRepository;
13
14     public void save(User user) {
15         userRepository.save(user);
16     }
17 }

```

Hình 111. Thêm class lớp UserService

Tạo file **UserController.java** đặt tại  
**src/main/java/com.example.demo/controller**

```

package com.example.demo.controller;

import com.example.demo.entity.User;
import com.example.demo.services.UserService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

@Controller
public class UserController {
    @Autowired
    private UserService userService;
}

```

```
@GetMapping("/login")
public String login() {
    return "user/login";
}

@GetMapping("/register")
public String register(Model model) {
    model.addAttribute("user", new User());
    return "user/register";
}

@PostMapping("/register")
public String register(@Valid @ModelAttribute("user") User user,
BindingResult bindingResult, Model model) {
    if (bindingResult.hasErrors()) {
        List<FieldError> errors = bindingResult.getFieldErrors();
        for (FieldError error : errors) {
            model.addAttribute(error.getField() + "_error",
error.getDefaultMessage());
        }
        return "user/register";
    }
    user.setPassword(new
BCryptPasswordEncoder().encode(user.getPassword()));
    userService.save(user);
    return "redirect:/login";
}
}
```

Hình 112.Tạo file UserController.java

Tạo thư mục **Utils** đặt tại **src/main/java/com.example.demo**

Tạo file **SecurityConfig.java** đặt tại **src/main/java/com.example.demo/Utils** dán đoạn code bên dưới vào.

**SecurityConfig** cung cấp các cấu hình bảo mật cho ứng dụng web. Cụ thể:

- + Phương thức **userDetailsService()** trả về một CustomUserDetailService, được sử dụng để cung cấp thông tin chi tiết về người dùng.
- + Phương thức **passwordEncoder()** trả về một BCryptPasswordEncoder, được sử dụng để mã hóa mật khẩu của người dùng.
- + Phương thức **authenticationProvider()** trả về một DaoAuthenticationProvider, được sử dụng để định nghĩa cách xác thực người dùng.
- + Phương thức **securityFilterChain(HttpSecurity http)** trả về một SecurityFilterChain, được sử dụng để cấu hình các hành vi bảo mật trong ứng dụng.

```
package com.example.demo.utils;

import com.example.demo.services.CustomUserDetailsService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

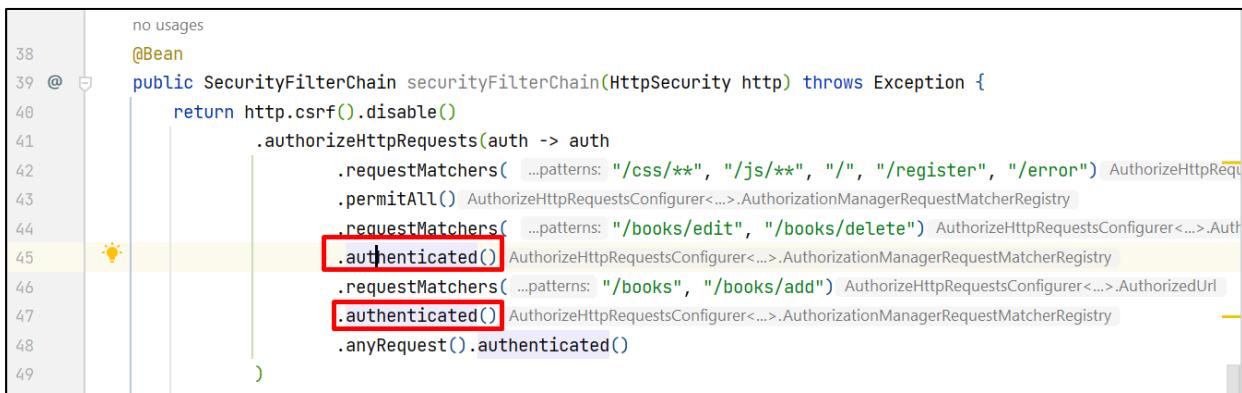
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {

    @Bean
    public UserDetailsService userDetailsService() {
        return new CustomUserDetailService();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

```
@Bean
public DaoAuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
    auth.setUserDetailsService(userDetailsService());
    auth.setPasswordEncoder(passwordEncoder());
    return auth;
}

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
    return http.csrf().disable()
        .authorizeHttpRequests(auth -> auth
            .requestMatchers( "/css/**", "/js/**", "/", "/register",
"/error")
            .permitAll()
            .requestMatchers( "/books/edit", "/books/delete")
            .authenticated()
            .requestMatchers("/books", "/books/add")
            .authenticated()
            .anyRequest().authenticated()
        )
        .logout(logout -> logout.logoutUrl("/logout")
            .logoutSuccessUrl("/login")
            .deleteCookies("JSESSIONID")
            .invalidateHttpSession(true)
            .clearAuthentication(true)
            .permitAll()
        )
        .formLogin(formLogin -> formLogin.loginPage("/login")
            .loginProcessingUrl("/login")
            .defaultSuccessUrl("/")
            .permitAll()
        )
        .rememberMe(rememberMe -> rememberMe.key("uniqueAndSecret"))
            .tokenValiditySeconds(86400)
            .userDetailsService(userDetailsService())
        )
        .exceptionHandling(exceptionHandling ->
            exceptionHandling.accessDeniedPage("/403"))
        .build();
}
}
```



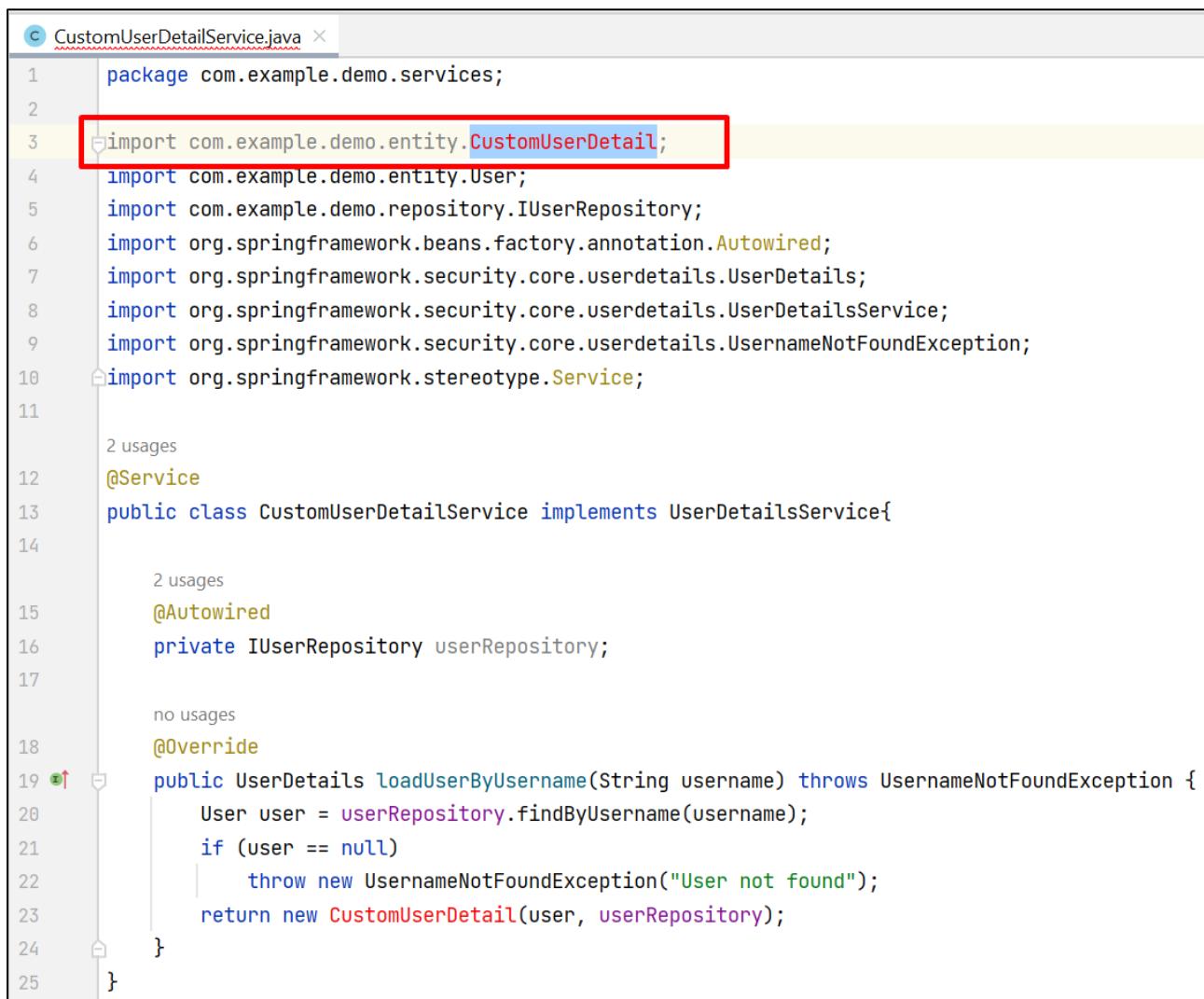
```
38 no usages
39 @
40     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
41         return http.csrf().disable()
42             .authorizeHttpRequests(auth -> auth
43                 .requestMatchers( ...patterns: "/css/**", "/js/**", "/", "/register", "/error") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
44                 .permitAll() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
45                 .requestMatchers( ...patterns: "/books/edit", "/books/delete") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
46                 .authenticated() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
47                 .requestMatchers( ...patterns: "/books", "/books/add") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
48                 .authenticated() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
49             .anyRequest().authenticated()
```

Hình 113. Giải thích authenticated() là gì?

### Tìm hiểu và giải thích, authenticated() là gì ?

Tiếp theo, tạo file **CustomUserDetailsService.java** đặt tại  
**src/main/java/com.example.demo/services**

Class **CustomUserDetailsService** là một implementation của interface **UserDetailsService**, cung cấp phương thức **loadUserByUsername** để tìm kiếm thông tin người dùng theo tên đăng nhập.

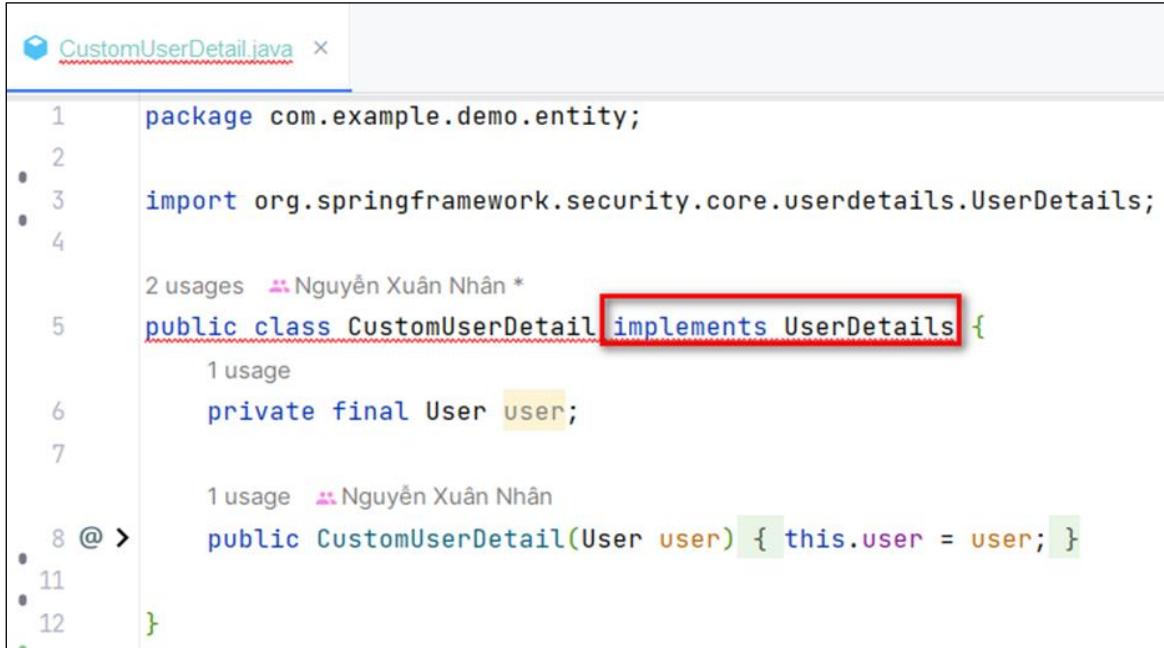


```
CustomUserDetailService.java
1 package com.example.demo.services;
2
3 import com.example.demo.entity.CustomUserDetail; // Line 3 is highlighted with a red box
4 import com.example.demo.entity.User;
5 import com.example.demo.repository.IUserRepository;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.security.core.userdetails.UserDetails;
8 import org.springframework.security.core.userdetails.UserDetailsService;
9 import org.springframework.security.core.userdetails.UsernameNotFoundException;
10 import org.springframework.stereotype.Service;
11
12 2 usages
13 @Service
14
15 2 usages
16 @Autowired
17 private IUserRepository userRepository;
18
19 no usages
20 @Override
21 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
22     User user = userRepository.findByUsername(username);
23     if (user == null)
24         throw new UsernameNotFoundException("User not found");
25     return new CustomUserDetail(user, userRepository);
26 }
```

Hình 114. tạo file CustomUserDetailService.java

Bên trên file **CustomUserDetailsService.java** ở dòng 3 có thông báo lỗi. Là do chưa add file **CustomUserDetail.java**

Tạo file **CustomUserDetail.java** đặt tại  
**src/main/java/com.example.demo/entity**

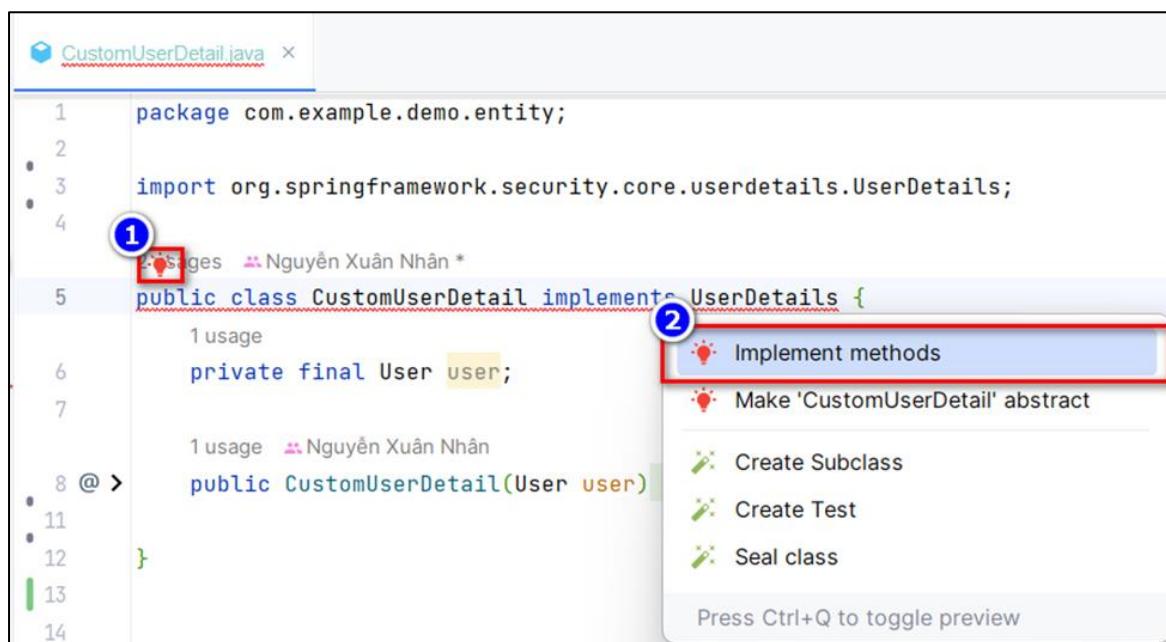


```
CustomUserDetail.java x

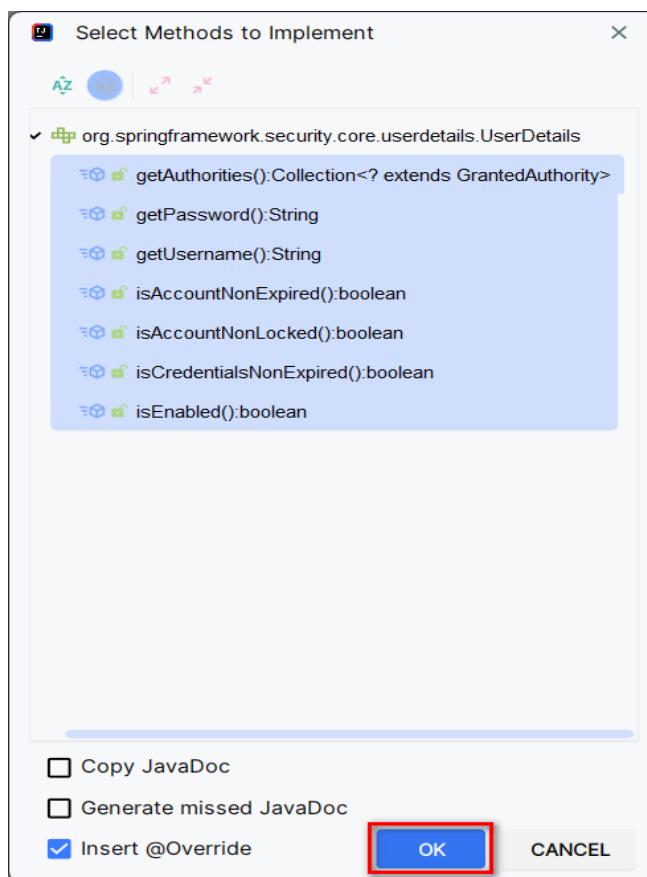
1 package com.example.demo.entity;
2
3 import org.springframework.security.core.userdetails.UserDetails;
4
5 2 usages  ↗ Nguyễn Xuân Nhân *
6 public class CustomUserDetail implements UserDetails {
7     1 usage
8     private final User user;
9
10    1 usage  ↗ Nguyễn Xuân Nhân
11    public CustomUserDetail(User user) { this.user = user; }
12 }
```

Hình 115. Tạo file *CustomUserDetail.java*

Nhấn tổ hợp phím **ALT + ENTER** hoặc nhấn vào **nút bóng đèn màu đỏ** để có thể implement các phương thức của **UserDetails**.



Hình 116. Implement methods



Hình 117. implement các phương thức của UserDetails

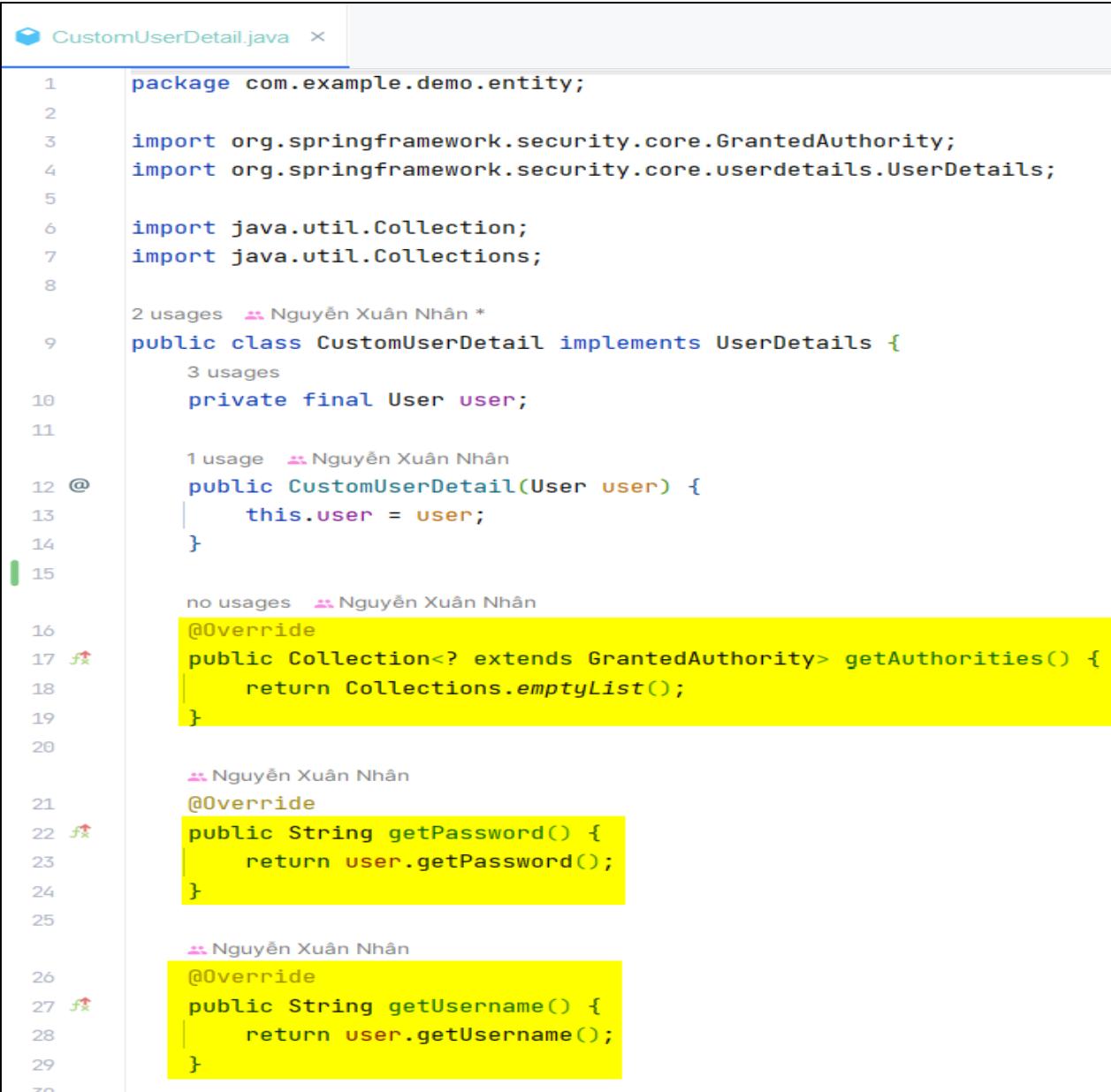
```

1 package com.example.demo.entity;
2
3 import org.springframework.security.core.GrantedAuthority;
4 import org.springframework.security.core.userdetails.UserDetails;
5
6 import java.util.Collection;
7
8 * 2 usages  Nguyen Xuân Nhân *
9 public class CustomUserDetail implements UserDetails {
10     1 usage
11     private final User user;
12
13     1 usage  Nguyen Xuân Nhân
14     public CustomUserDetail(User user) { this.user = user; }
15
16     no usages  Nguyen Xuân Nhân *
17     private final User user;
18
19     1 usage  Nguyen Xuân Nhân
20     public CustomUserDetail(User user) { this.user = user; }
21
22     no usages  Nguyen Xuân Nhân *
23     @Override
24     public Collection<? extends GrantedAuthority> getAuthorities()
25         return null;
26
27     1 usage  Nguyen Xuân Nhân *
28     @Override
29     public String getPassword() {
30         return null;
31
32     1 usage  Nguyen Xuân Nhân *
33     @Override
34     public String getUsername() {
35         return null;
36
37     1 usage  Nguyen Xuân Nhân *
38     @Override
39     public boolean isAccountNonExpired() {
40         return false;
41
42     1 usage  Nguyen Xuân Nhân *
43     @Override
44     public boolean isAccountNonLocked() {
45         return false;
46
47     1 usage  Nguyen Xuân Nhân *
48     @Override
49     public boolean isCredentialsNonExpired() {
50         return false;
51
52     1 usage  Nguyen Xuân Nhân *
53     @Override
54     public boolean isEnabled() {
55         return false;
56
57     }
58
59 }

```

Hình 118. file CustomUserDetail.java

Tiến hành chỉnh sửa các phương thức CustomUserDetail.java theo như ảnh bên dưới.



```
CustomUserDetail.java  ×

1 package com.example.demo.entity;
2
3 import org.springframework.security.core.GrantedAuthority;
4 import org.springframework.security.core.userdetails.UserDetails;
5
6 import java.util.Collection;
7 import java.util.Collections;
8
9 2 usages  ↗ Nguyễn Xuân Nhân *
10 public class CustomUserDetail implements UserDetails {
11     3 usages
12     private final User user;
13
14     1 usage  ↗ Nguyễn Xuân Nhân
15     public CustomUserDetail(User user) {
16         1 usage  ↗ Nguyễn Xuân Nhân
17         this.user = user;
18     }
19
20     no usages  ↗ Nguyễn Xuân Nhân
21     @Override
22     public Collection<? extends GrantedAuthority> getAuthorities() {
23         return Collections.emptyList();
24     }
25
26     ↗ Nguyễn Xuân Nhân
27     @Override
28     public String getPassword() {
29         return user.getPassword();
30     }
31
32     ↗ Nguyễn Xuân Nhân
33     @Override
34     public String getUsername() {
35         return user.getUsername();
36     }
37 }
```

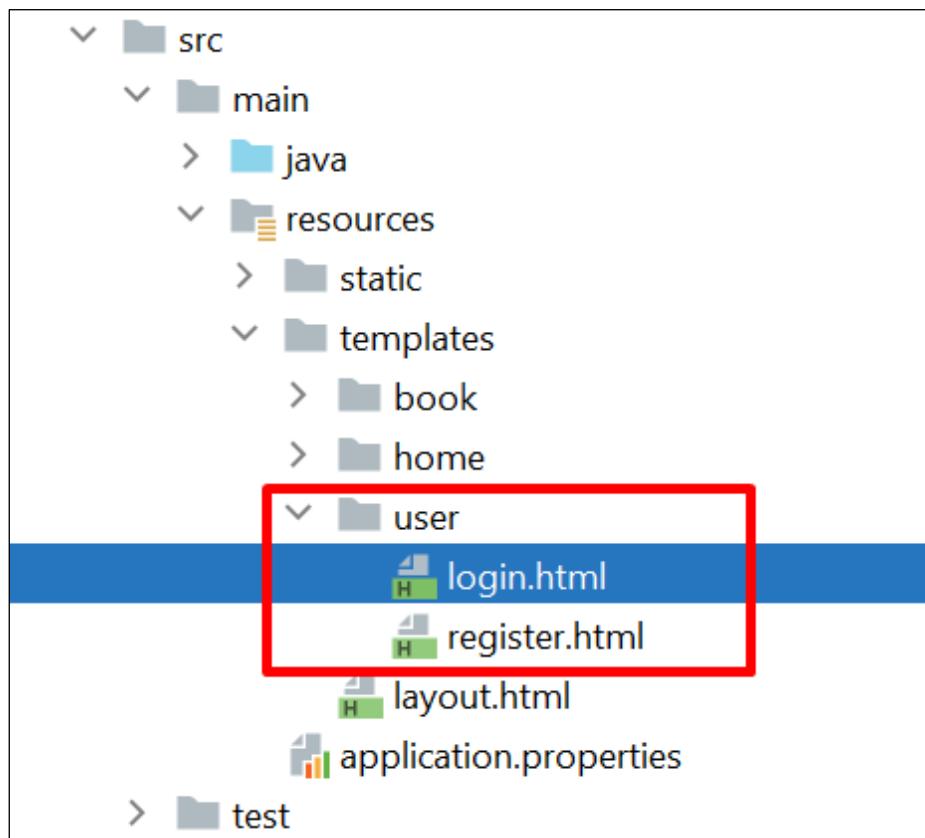
Hình 119. chỉnh sửa các phương thức CustomUserDetail.java

```
31      @Override
32  ↗    public boolean isAccountNonExpired() {
33      |      return true;
34  ↗    }
35
36          no usages ↗ Nguyễn Xuân Nhân
37  ↗    @Override
38  ↗    public boolean isAccountNonLocked() {
39  ↗    |      return true;
39  ↗    }
40
41          no usages ↗ Nguyễn Xuân Nhân
41  ↗    @Override
42  ↗    public boolean isAccountNonLocked() {
43  ↗    |      return true;
43  ↗    }
44
45          no usages ↗ Nguyễn Xuân Nhân
41  ↗    @Override
42  ↗    public boolean isCredentialsNonExpired() {
43  ↗    |      return true;
44  ↗    }
45
46          ↗ Nguyễn Xuân Nhân
46  ↗    @Override
47  ↗    public boolean isEnabled() {
48  ↗    |      return true;
49  ↗    }
50 }
```

Hình 120. chỉnh sửa các phương thức CustomUserDetail.java

## 6.2 Thiết kế giao diện Login và tạo tài khoản mới

- ⊕ Tạo thư mục **user** tại đường dẫn  
`src/main/java/com.example.demo/resources/templates`
- ⊕ Tạo 2 file model **login.html** và **register.html** đặt tại thư mục  
`src/main/java/com.example.demo/resources/templates/user`



Hình 121. Thêm 2 file login.html và register.html

## Thêm nội dung code cho file **login.html**

```

login.html ×

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Login</title>
8      <th:block th:replace="~{layout :: link-css}"></th:block>
9  </head>
10 <body>
11 <th:block th:replace="~{layout :: header}"></th:block>
12 <div class="container">
13     <div class="row justify-content-center">
14         <div class="col-lg-6 col-md-8">
15             <h3 class="card-title text-center">Login</h3>
16             <form th:action="@{/login}" method="post">
17                 <fieldset>
18                     <legend>Please login</legend>
19                     <div th:if="${param.error}" class="alert alert-danger">
20                         Invalid username and password.
21                     </div>
22                     <div th:if="${param.logout}" class="alert alert-success">
23                         You have been logged out.
24                     </div>
25                     <div class="mb-3">
26                         <label for="username" class="form-label">Username:</label>
27                         <input type="text" required class="form-control" id="username" name="username">
28                     </div>
29                     <div class="mb-3">
30                         <label for="password" class="form-label">Password:</label>
31                         <input type="password" required class="form-control" id="password" name="password">
32                     </div>
33                     <div class="mb-3">
34                         <input type="checkbox" name="remember-me" id="remember-me">
35                         <label for="remember-me">Remember me</label>
36                     </div>
37                     <div class="d-grid gap-2 form-action">
38                         <button type="submit" class="btn btn-primary">Login</button>
39                         <a class="text-primary text-center" href="/register">Don't have account. Sign up?</a>
40                     </div>
41                 </fieldset>
42             </form>
43         </div>
44     </div>
45 <th:block th:replace="~{layout :: footer}"></th:block>
46 </body>
47 </html>

```

Hình 122. Thêm nội dung cho file **login.html**

## Thêm nội dung code cho file register.html

```
register.html <hr>
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Sign Up</title>
8      <th:block th:replace="~{layout :: link-css}"></th:block>
9  </head>
10 <body>
11 <th:block th:replace="~{layout :: header}"></th:block>
12 <div class="container">
13     <div class="row justify-content-center">
14         <div class="col-lg-6 col-md-8">
15             <h3 class="card-title text-center">Sign Up</h3>
16             <form th:action="@{/register}" th:object="${user}" method="post">
17                 <div class="mb-3">
18                     <label for="name" class="form-label">First Name:</label><span class="text-danger">*</span>
19                     <input type="text" class="form-control" th:field="*{name}" id="name">
20                     <span class="text-danger" th:if="#{fields.hasErrors('name')}" th:errors="*{name}"></span>
21                 </div>
22                 <div class="mb-3">
23                     <label for="email" class="form-label">Email:</label>
24                     <input type="email" class="form-control" th:field="*{email}" id="email">
25                     <span class="text-danger" th:if="#{fields.hasErrors('email')}" th:errors="*{email}"></span>
26                 </div>
27                 <div class="mb-3">
28                     <label for="username" class="form-label">Username:</label><span class="text-danger">*</span>
29                     <input type="text" class="form-control" id="username" th:field="*{username}">
30                     <span class="text-danger" th:if="#{fields.hasErrors('username')}" th:errors="*{username}"></span>
31                 </div>
32                 <div class="mb-3">
33                     <label for="password" class="form-label">Password:</label><span class="text-danger">*</span>
34                     <input type="password" class="form-control" th:field="*{password}" id="password">
35                     <span class="text-danger" th:if="#{fields.hasErrors('password')}" th:errors="*{password}"></span>
36                 </div>
37                 <div class="d-grid gap-2">
38                     <button type="submit" class="btn btn-info">Sign up</button>
39                 </div>
40             </form>
41         </div>
42     </div>
43 <th:block th:replace="~{layout :: footer}"></th:block>
44 </body>
45 </html>
```

Hình 123. Thêm nội dung cho file register.html

## Chỉnh sửa file **layout.html**

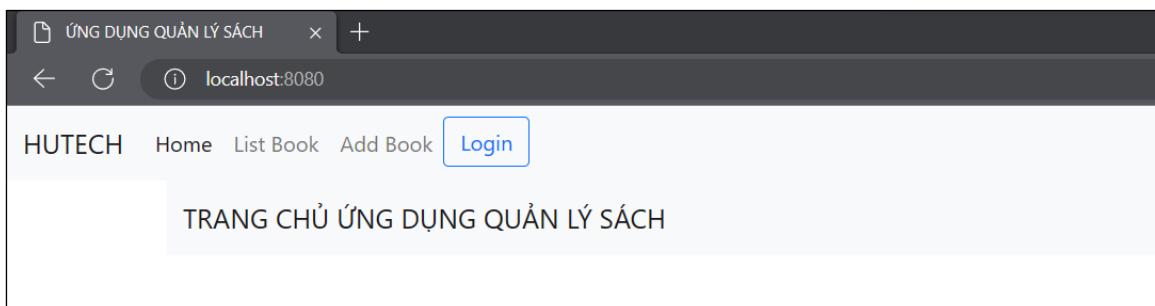


```

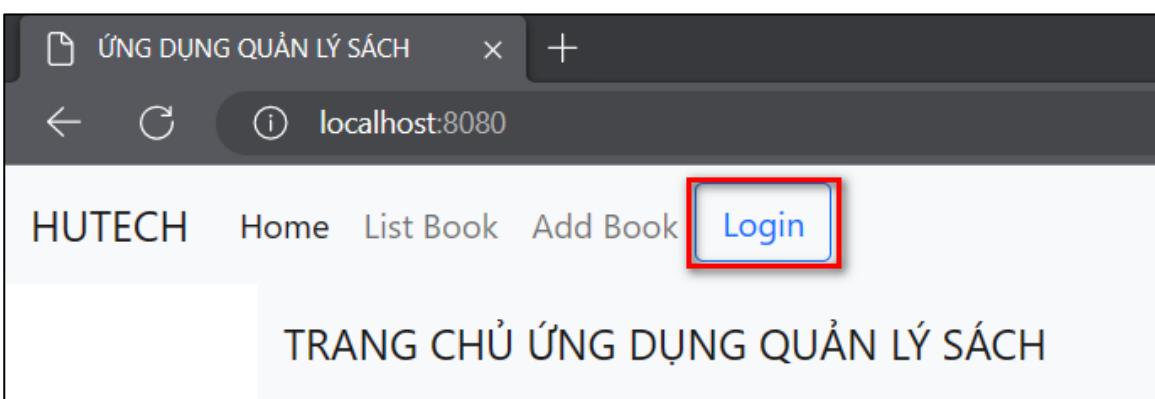
1 <!DOCTYPE html>
2 <html
3     xmlns:th="http://www.thymeleaf.org"
4     xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4"
5     lang="en">
6 <head>
7     <meta charset="UTF-8">
8     <title>My App</title>
9     <link th:fragment="link-css" rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
10 </head>
11 <body>
12 <header th:fragment="header">
13     <nav class="navbar navbar-expand-lg navbar-light bg-light">
14         <div class="container-fluid">
15             <a class="navbar-brand" href="/">HUTECH</a>
16             <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
17                     data-bs-target="#navbarSupportedContent"
18                     aria-controls="navbarSupportedContent"
19                     aria-expanded="false" aria-label="Toggle navigation">
20                 <span class="navbar-toggler-icon"></span>
21             </button>
22             <div class="collapse navbar-collapse" id="navbarSupportedContent">
23                 <ul class="navbar-nav me-auto mb-2 mb-lg-0">
24                     <li class="nav-item">
25                         <a class="nav-link active" aria-current="page" href="/">Home</a>
26                     </li>
27                     <li class="nav-item"><a class="nav-link" href="/books">List Book</a></li>
28                     <li class="nav-item"><a class="nav-link" href="/books/add">Add Book</a></li>
29                     <li sec:authorize="isAuthenticated()">
30                         <form th:action="@{/logout}" method="post">
31                             <button class="btn btn-outline-danger" type="submit">Logout</button>
32                         </form>
33                     </li>
34                     <li sec:authorize="!isAuthenticated()">
35                         <a class="btn btn-outline-primary" href="/login">Login</a>
36                     </li>
37                 </ul>
38             </div>
39         </div>
40     </nav>
41 </header>
42
43 <footer th:fragment="footer">
44     <script th:src="@{/js/bootstrap.min.js}"></script>
45 </footer>
46 </body>
47 </html>

```

Hình 124. Chỉnh sửa lại file layout.html

**Build** và chạy ứng dụng tại địa chỉ **localhost:8080***Hình 125. Giao diện web chưa đăng nhập*

Kiểm tra bằng cách tạo tài khoản. Theo các bước sau:

A screenshot of a login form titled "Login". The form has fields for "Username" and "Password", a "Remember me" checkbox, and a large blue "Login" button at the bottom. Below the "Login" button is a link "Don't have account. Sign up?" which is highlighted with a red box.*Hình 126. Tạo tài khoản mới*

Kiểm tra **valid dữ liệu** bằng cách không nhập và ấn nút **sign up**

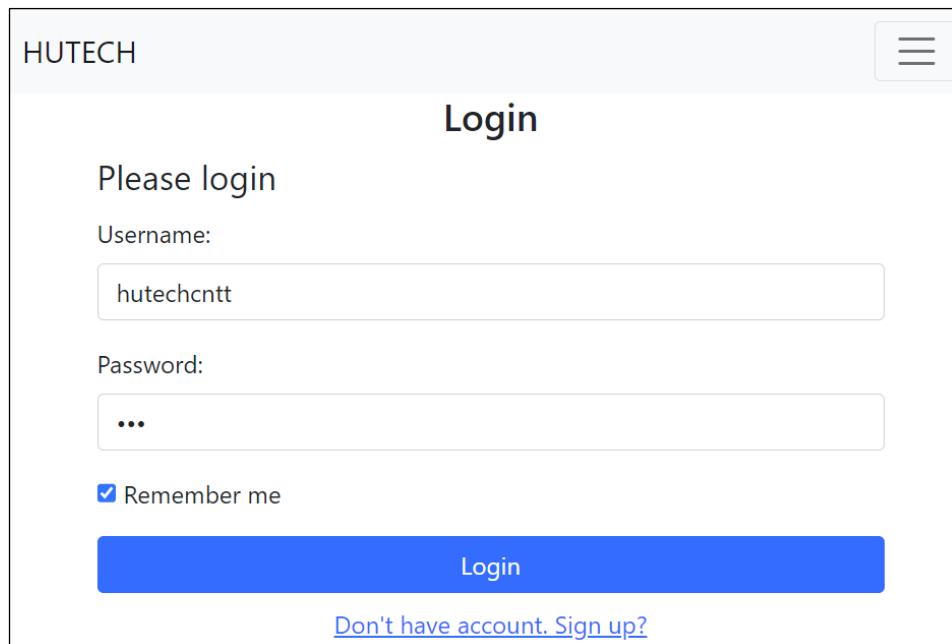
The screenshot shows a 'Sign Up' form with four input fields: First Name, Email, Username, and Password. Each field has a red error message below it indicating it is required. The 'First Name' field contains 'Your name is required'. The 'Email' field contains 'Email is required'. The 'Username' field contains 'Username is required'. The 'Password' field contains 'Password is required'. A blue 'Sign up' button is at the bottom.

Hình 127. Kiểm tra valid dữ liệu khi không điền dữ liệu

Tiến hành đăng ký thử tài khoản làm mẫu:

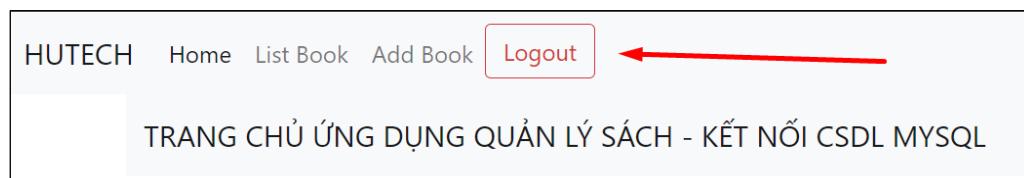
The screenshot shows a 'Sign Up' form with four input fields: First Name, Email, Username, and Password. All fields contain sample data: 'Nguyễn Văn A' in the First Name field, 'demo@gmail.com' in the Email field, 'hutechcntt' in the Username field, and '\*\*\*' in the Password field. A blue 'Sign up' button is at the bottom.

Hình 128. đăng ký thử tài khoản mẫu



Hình 129. đăng nhập bằng tài khoản vừa tạo

Đăng nhập thành công, nút **Login** chuyển sang trạng thái **Logout**.



Hình 130. đăng nhập thành công, xuất hiện nút Logout

Recent Favorites

New  
bookstore  
New  
book  
category  
role  
**user**    

SELECT \* FROM `user`

Profiling | Edit inline | Edit | Explain SQL | Create PHP code | Refresh

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	id	email	name	password	username
<input type="checkbox"/>	1	demo@gmail.com	Nguyễn Văn A	\$2a\$10\$IZko6Q26dbjBHgq9nvpM8egpSY4qjSzAH8c.91ETnEJ...	hutechcnit

Check all With selected:  Edit  Copy  Delete  Export

Show all | Number of rows: 25 Filter rows: Search this table

Hình 131. Kiểm tra thông tin tài khoản đã có trong Database

## TÓM TẮT

Trong Java Spring Boot, User và Role có vai trò quan trọng trong bảo mật ứng dụng và phân quyền người dùng.

User là đại diện cho người dùng trong hệ thống. Mỗi người dùng sẽ có một tài khoản riêng, được lưu trữ trong cơ sở dữ liệu. Thông tin người dùng bao gồm tên đăng nhập, mật khẩu đã được mã hóa, và các thuộc tính khác như tên, địa chỉ, email, số điện thoại, v.v.

Role là đại diện cho vai trò hoặc quyền hạn của người dùng trong hệ thống. Ví dụ, có thể có các vai trò như Admin, User, Manager, v.v. Mỗi vai trò sẽ được cấu hình để chỉ định các quyền hạn tương ứng với nó. Các quyền hạn này có thể được sử dụng để giới hạn truy cập vào các chức năng, tính năng hoặc tài nguyên khác nhau trong hệ thống.

Trong ứng dụng Spring Boot, thông tin người dùng và vai trò thường được lưu trữ trong cơ sở dữ liệu và được sử dụng bởi Spring Security để xác thực và phân quyền người dùng.

Trong Java Spring Boot, để tạo tài khoản và đăng nhập, bạn có thể sử dụng Spring Security để bảo mật ứng dụng và xác thực người dùng. Để làm điều này, bạn cần cấu hình SecurityConfig để chỉ định các bộ xác thực, phân quyền và các cấu hình khác cho ứng dụng.

## CÂU HỎI ÔNG TẬP

1. Làm thế nào để thêm một người dùng mới vào hệ thống trong Java Spring Boot?
2. Làm thế nào để bảo mật thông tin người dùng trong Java Spring Boot, đảm bảo rằng mật khẩu được mã hóa và không bị đánh cắp hoặc lộ ra ngoài?

# BÀI 7 PHÂN QUYỀN NGƯỜI DÙNG CHO TÀI KHOẢN ĐĂNG NHẬP

## **Bài này giúp người học nắm được các nội dung sau:**

Sau khi học xong phân quyền trong Java Spring Boot, người học sẽ nắm được các kiến thức và kỹ năng cơ bản sau đây:

- ✓ Hiểu về cách hoạt động của phân quyền trong ứng dụng web.
- ✓ Biết cách triển khai phân quyền người dùng và quản trị viên trong Java Spring Boot.
- ✓ Có khả năng sử dụng các công cụ và thư viện như Spring Security để thực hiện phân quyền trong ứng dụng của mình.
- ✓ Hiểu về cách quản lý quyền truy cập và kiểm soát truy cập của người dùng vào các tính năng cụ thể của ứng dụng.

## **Mô tả chức năng:**

Trong Java Spring Boot, phân quyền người dùng và quản trị viên là một phần quan trọng để đảm bảo tính bảo mật cho ứng dụng web của bạn. Các chức năng của phân quyền người dùng và quản trị viên trong Java Spring Boot được mô tả như sau:

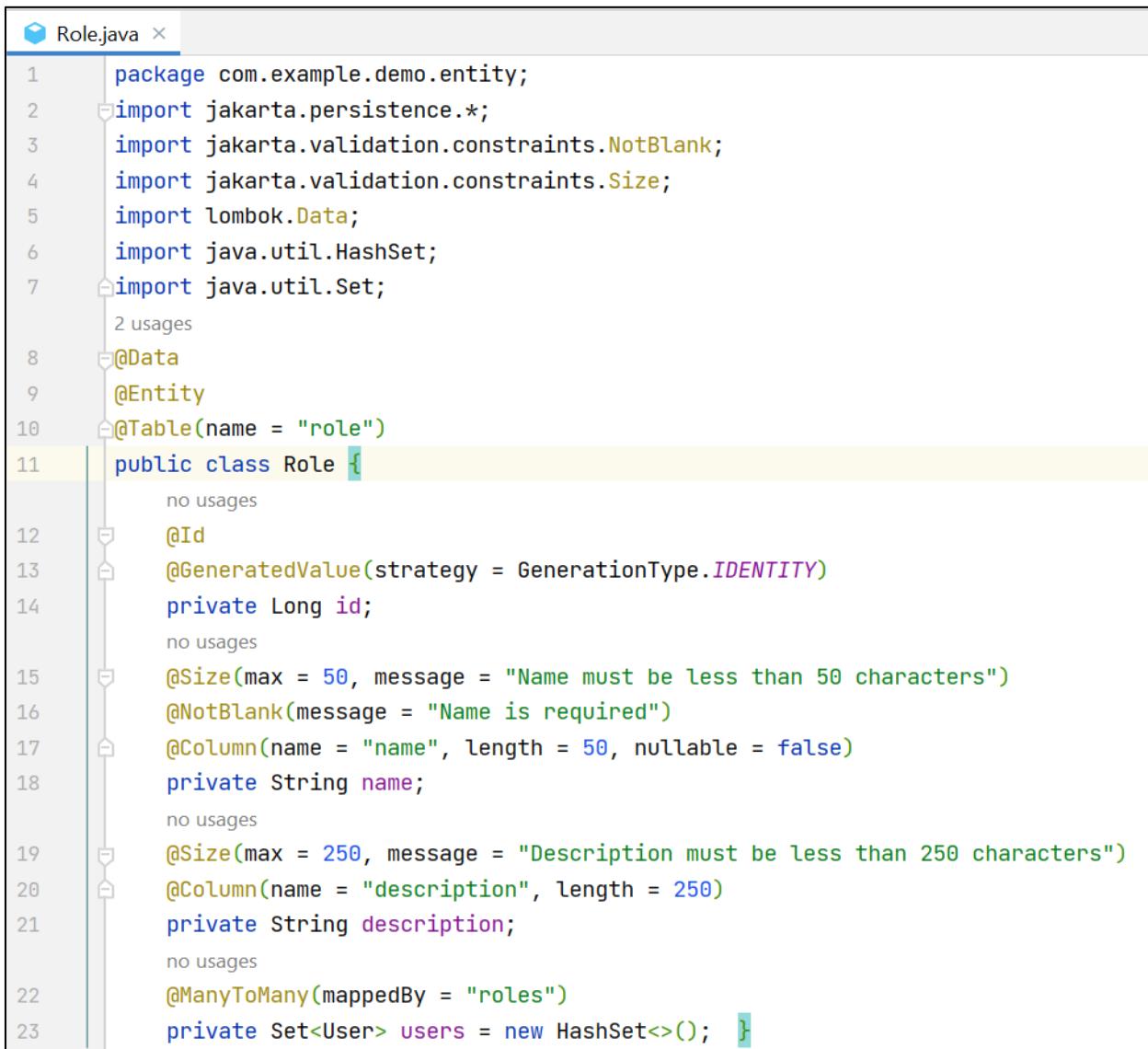
**Phân quyền người dùng (user):** Người dùng được phân quyền sẽ có quyền truy cập vào các tài nguyên cần thiết để sử dụng ứng dụng web. Các tài nguyên này có thể là các trang web, tính năng, thông tin người dùng và các chức năng khác.

**Phân quyền quản trị viên (admin):** Quản trị viên là người sở hữu quyền kiểm soát tất cả các hoạt động của ứng dụng. Quản trị viên có thể xem và chỉnh sửa các thông tin của người dùng, quản lý các tài nguyên và các chức năng khác liên quan đến ứng dụng.

## 7.1 Thêm table và dữ liệu vào table Role trong CSDL

### ➤ Thêm table Role trên CSDL

Tạo class **Role.java** trong thư mục **Entity** trên đường dẫn  
**src/main/java/com.example.demo/entity**



```

Role.java
1 package com.example.demo.entity;
2 import jakarta.persistence.*;
3 import jakarta.validation.constraints.NotBlank;
4 import jakarta.validation.constraints.Size;
5 import lombok.Data;
6 import java.util.HashSet;
7 import java.util.Set;
8
9 @Data
10 @Entity
11 @Table(name = "role")
12 public class Role {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16
17     @Size(max = 50, message = "Name must be less than 50 characters")
18     @NotBlank(message = "Name is required")
19     @Column(name = "name", length = 50, nullable = false)
20     private String name;
21
22     @Size(max = 250, message = "Description must be less than 250 characters")
23     @Column(name = "description", length = 250)
24     private String description;
25
26     @ManyToMany(mappedBy = "roles")
27     private Set<User> users = new HashSet<>();
}

```

Hình 132. tạo thêm class role

Tiếp theo, chỉnh sửa thêm nội dung Role.java như ảnh bên dưới, quan hệ n-n (many-to-many) giữa đối tượng User và đối tượng Role trong cơ sở dữ liệu.



```
Role.java x User.java x
29     @NotBlank(message = "Password is required")
30     private String password;
31
32     no usages
33     @Column(name = "email", length = 50)
34     @Size(max = 50, message = "Email must be less than 50 characters")
35     private String email;
36
37     no usages
38     @Column(name = "name", length = 50, nullable = false)
39     @Size(max = 50, message = "Your name must be less than 50 characters")
40     @NotBlank(message = "Your name is required")
41     private String name;
42
43     no usages
44     @ManyToMany
45     @JoinTable(name = "user_role",
46                 joinColumns = @JoinColumn(name = "user_id"),
47                 inverseJoinColumns = @JoinColumn(name = "role_id"))
48     private Set<Role> roles = new HashSet<>();
49
50 }
```

Hình 133. Bổ sung thêm đoạn code nối user và role

Cụ thể, thuộc tính roles trong đối tượng User được đánh dấu với **@ManyToMany** để biểu thị mối quan hệ n-n với đối tượng Role.

Chú thích **@JoinTable** được sử dụng để chỉ định tên bảng liên kết (join table) để lưu trữ các bản ghi quan hệ giữa các đối tượng User và Role trong cơ sở dữ liệu. Các thuộc tính **joinColumns** và **inverseJoinColumns** được sử dụng để chỉ định tên của các cột khóa ngoại tham chiếu đến bảng User và Role.

Kiểm tra lại các class và interface đã thêm, build lại project (nhấn Ctrl + F9)

Mở **phpMyAdmin** bằng cách truy cập <http://localhost/phpmyadmin/>

Kiểm tra **Database** sẽ xuất hiện thêm table Role đã thêm.

The screenshot shows the phpMyAdmin interface with the following details:

- Left Panel (Schemas):** Shows the database structure. A red box highlights the **bookstore** schema, which contains tables: **book**, **category**, **role**, **user**, and **user\_role**.
- Top Bar:** Shows the server as **localhost:3306** and the database as **bookstore**. Navigation tabs include Structure, SQL, Search, Query, Export, Import, and a gear icon.
- Filters:** A search bar labeled "Containing the word:".
- Table List:** Shows the following tables with their respective actions:
  - book**: Browse, Structure, Search, Insert, Empty, Drop
  - category**: Browse, Structure, Search, Insert, Empty, Drop
  - role**: Browse, Structure, Search, Insert, Empty, Drop (highlighted with a red box)
  - user**: Browse, Structure, Search, Insert, Empty, Drop
  - user\_role**: Browse, Structure, Search, Insert, Empty, Drop (highlighted with a red box)
- Total:** 5 tables

Hình 134. Database sẽ xuất table Role và user\_role

User.java lớp Java để xác định thông tin người dùng và quyền truy cập.Thêm role **ADMIN** và **USER** vào bảng **role**

			<b>id</b>	<b>description</b>	<b>name</b>
<input type="checkbox"/>		Edit		Delete	1 <i>NULL</i> ADMIN
<input type="checkbox"/>		Edit		Delete	2 <i>NULL</i> USER

Hình 135. Thêm dữ liệu vào table role

Tạo 2 tài khoản như bên dưới để thử, ta sẽ gán tài khoản **DEMO1** thành quyền **ADMIN**, tài khoản **DEMO2** là quyền **USER**.

	<b>id</b>	<b>email</b>	<b>name</b>	<b>password</b>	<b>username</b>
<input type="checkbox"/>	1	demo@gmail.com	Nguyễn Văn A	\$2a\$10\$e/7E1/A0oW6GZQDQE0/l6Og87VVK6KFcj/J3Ablo8O...	DEMO1
<input type="checkbox"/>	2	demo2@gmail.com	NGuyễn Văn B	\$2a\$10\$JLbb8J.ySbMd6T0mxSM9debRaRtGO1.uyzZH2ftqcbF...	DEMO2

Hình 136. xem id của user

phpMyAdmin

Server: localhost:3306 » Database: bookstore » Table: user\_role

Browse Structure SQL Search Insert Export Import Privileges

Column Type Function Null Value

user\_id bigint

role\_id bigint

Điền giá trị là id của user và role

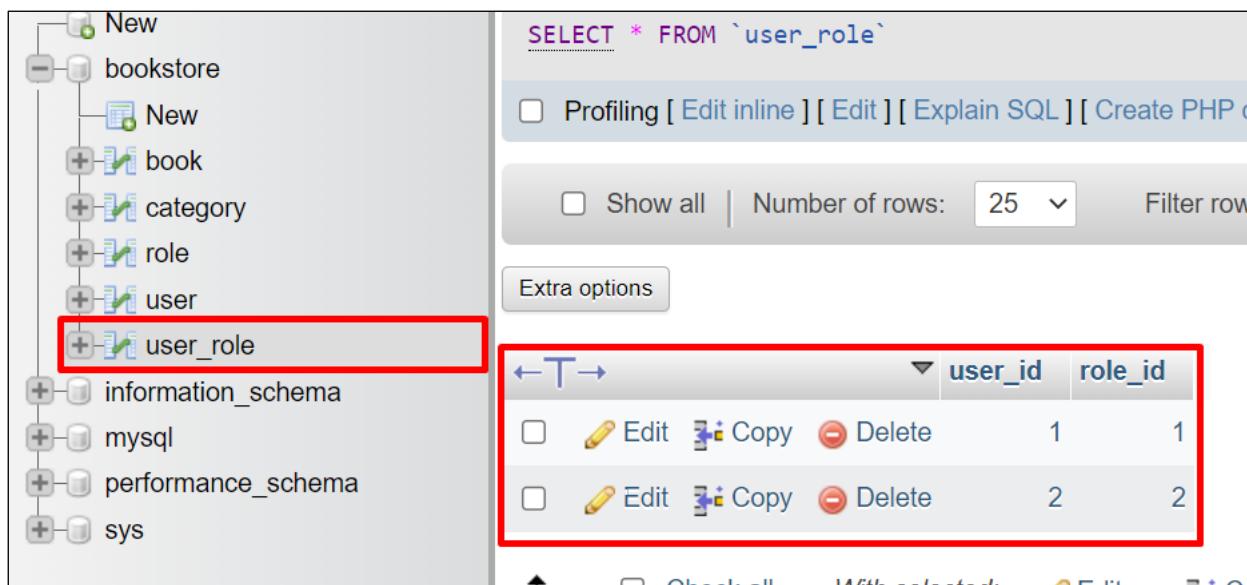
1 2 3

Recent Favorites

New bookstore New book category role user user\_role information\_schema

Ignore

Hình 137. điền id của user và role vào bảng user\_role



Hình 138. dữ liệu bảng user\_role vừa set quyền

Thiết lập mặc định khi tạo tài khoản sẽ có role mặc định là “**USER**”

Tạo file **IRoleRepository.java** đặt tại thư mục repository theo đường dẫn **src/main/java/com.example.demo/repository**.

khai báo interface **IRoleRepository** để truy vấn và quản lý dữ liệu trong cơ sở dữ liệu liên quan đến đối tượng **Role**.

```

1 package com.example.demo.repository;
2
3 import com.example.demo.entity.Role;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.stereotype.Repository;
7
8 no usages
9 @Repository
10 public interface IRoleRepository extends JpaRepository<Role, Long> {
11     no usages
12     @Query("SELECT r.id FROM Role r WHERE r.name = ?1")
13     Long getRoleIdByName(String roleName);
14 }

```

Hình 139. khai báo interface IroleRepository

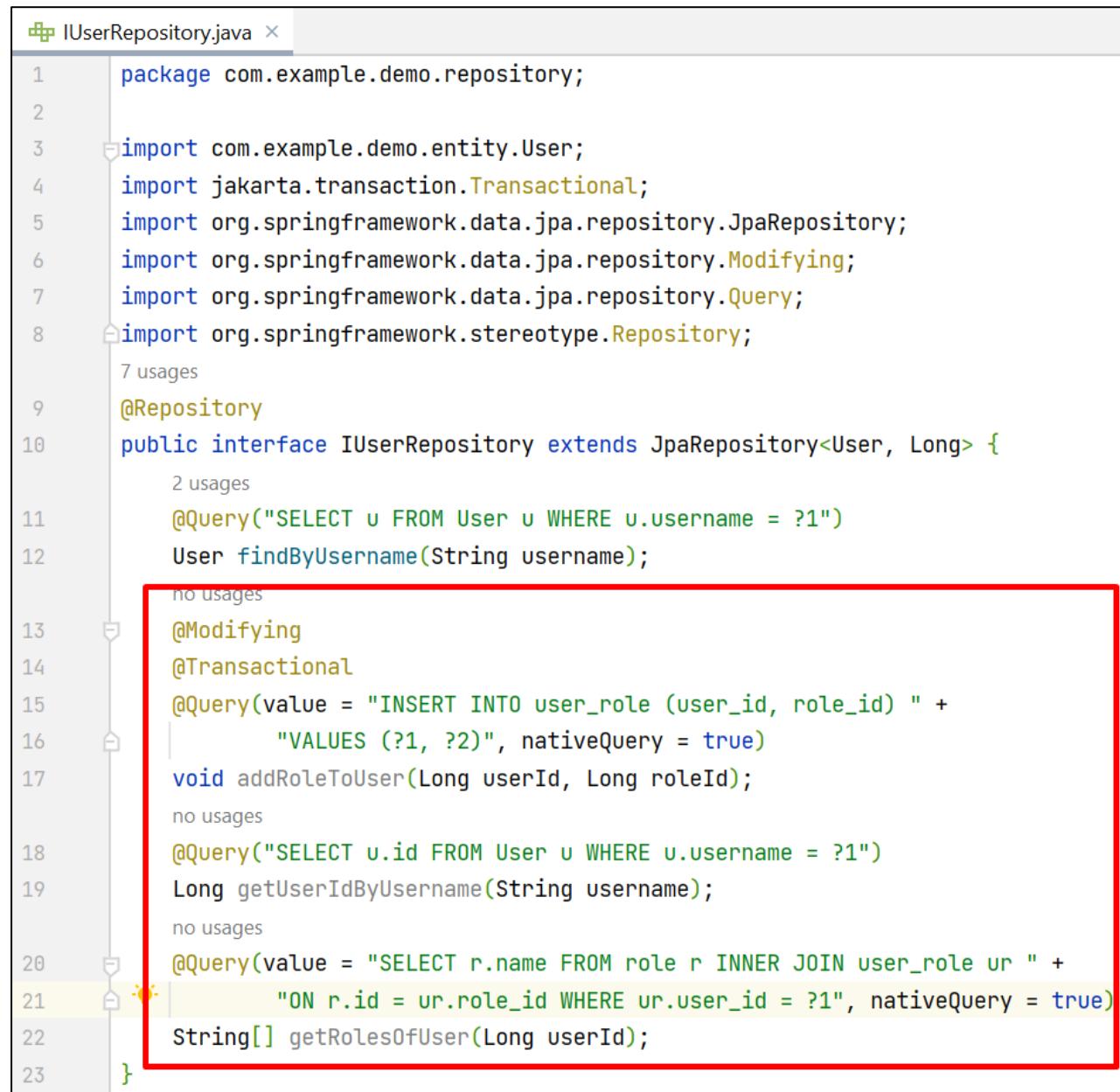
Tại file **IUserRepository.java** đặt tại thư mục repository theo đường dẫn **src/main/java/com.example.demo/repository** ta bổ sung thêm các phương thức sau:

Phương thức **addRoleToUser** được định nghĩa để thêm một quyền (Role) cho một người dùng (User). Câu truy vấn được sử dụng trong phương thức này là "**INSERT INTO user\_role (user\_id, role\_id) VALUES (?1, ?2)**". Câu truy vấn này sẽ thêm một bản ghi mới vào bảng user\_role với các giá trị user\_id và role\_id được truyền vào.

Phương thức **getUserIdByUsername** được định nghĩa để trả về ID của một người dùng dựa trên tên đăng nhập của họ. Câu truy vấn được sử dụng trong phương thức này là "**SELECT u.id FROM User u WHERE u.username = ?1**". Câu truy vấn này sẽ trả về ID của đối tượng User có tên đăng nhập trùng với tham số đầu vào ?1.

Phương thức **getRolesOfUser** được định nghĩa để trả về tên của các quyền của một người dùng dựa trên ID của họ. Câu truy vấn được sử dụng trong phương thức này là "**SELECT r.name FROM role r INNER JOIN user\_role ur ON r.id =**

**ur.role\_id WHERE ur.user\_id = ?1".** Câu truy vấn này sẽ trả về một mảng các tên của các đối tượng Role liên quan đến đối tượng User có ID trùng với tham số đầu vào ?1.



```

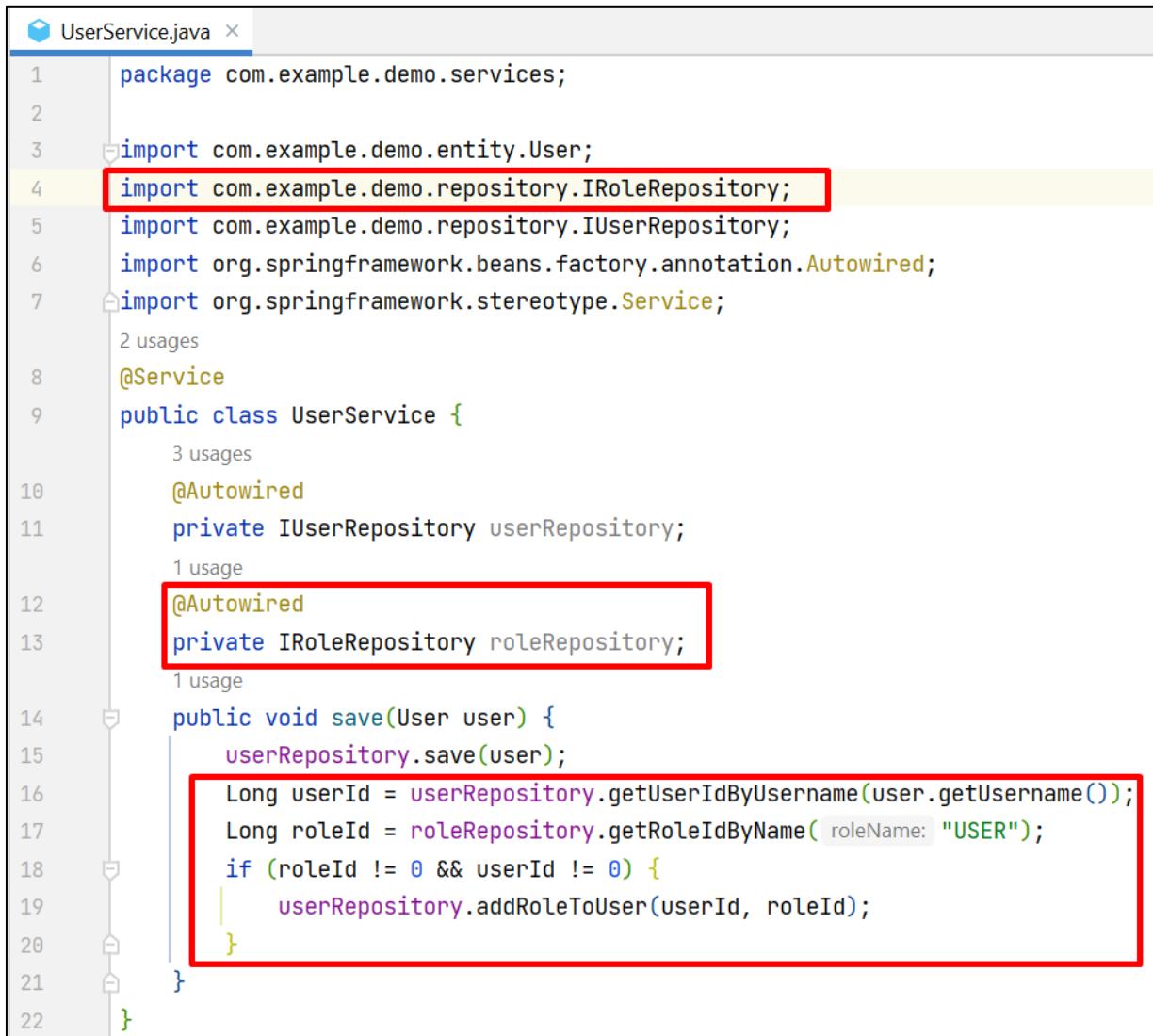
1 package com.example.demo.repository;
2
3 import com.example.demo.entity.User;
4 import jakarta.transaction.Transactional;
5 import org.springframework.data.jpa.repository.JpaRepository;
6 import org.springframework.data.jpa.repository.Modifying;
7 import org.springframework.data.jpa.repository.Query;
8 import org.springframework.stereotype.Repository;
9
10 @Repository
11 public interface IUserRepository extends JpaRepository<User, Long> {
12
13     @Modifying
14     @Transactional
15     @Query(value = "INSERT INTO user_role (user_id, role_id) " +
16             "VALUES (?1, ?2)", nativeQuery = true)
17     void addRoleToUser(Long userId, Long roleId);
18
19     @Query("SELECT u.id FROM User u WHERE u.username = ?1")
20     Long getUserIdByUsername(String username);
21     @Query(value = "SELECT r.name FROM role r INNER JOIN user_role ur " +
22             "ON r.id = ur.role_id WHERE ur.user_id = ?1", nativeQuery = true)
23     String[] getRolesOfUser(Long userId);
}

```

Hình 140. Bổ sung thêm các phương thức cho IUserRepository

Tại file **UserService.java** đặt **src/main/java/com/example/demo/services/** ta thay đổi phương thức thêm User như sau:

Bổ sung thêm nội dung như đoạn code bên dưới. Mục đích đoạn code này sử dụng các phương thức được định nghĩa trong các đối tượng **userRepository** và **roleRepository** để thêm một quyền cho một người dùng trong hệ thống.



```
UserService.java
1 package com.example.demo.services;
2
3 import com.example.demo.entity.User;
4 import com.example.demo.repository.IRoleRepository; // Line 4 highlighted with a red box
5 import com.example.demo.repository.IUserRepository;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 @Service
10 public class UserService {
11
12     @Autowired
13     private IUserRepository userRepository;
14
15     @Autowired
16     private IRoleRepository roleRepository; // Line 13 highlighted with a red box
17
18     public void save(User user) {
19         Long userId = userRepository.getUserIdByUsername(user.getUsername());
20         Long roleId = roleRepository.getRoleIdByName(roleName: "USER");
21
22         if (roleId != 0 & userId != 0) {
23             userRepository.addRoleToUser(userId, roleId);
24         }
25     }
26 }
```

Hình 141. Bổ sung đoạn code UserService.java để thêm quyền

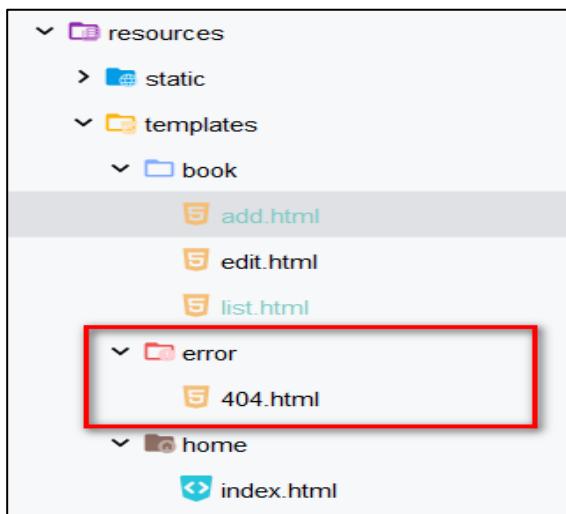
## 7.2 Tạo thông báo lỗi khi chuyển trang

### ➤ Thiết kế giao diện 404.html

Tạo 1 thư mục mới tên **error** trong thư mục **templates**. Tạo file **404.html** tại thư đường dẫn:

**src/main/java/com.example.demo/resources/templates/error**

**404.html** là một tệp HTML được sử dụng để hiển thị trang lỗi "404 Not Found". Khi một trình duyệt web yêu cầu một trang không tồn tại trên máy chủ, máy chủ sẽ trả về một mã trạng thái HTTP 404 và trình duyệt web sẽ hiển thị nội dung của tệp 404.html.



Hình 142. Thư mục và file 404.html được tạo

```
1  <!DOCTYPE html>
2  <html lang="en" xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Not Found</title>
8      <th:block th:replace="~{layout :: link-css}"></th:block>
9  </head>
10 <body>
11     <th:block th:replace="~{layout :: header}"></th:block>
12     <div class="d-flex align-items-center justify-content-center vh-25">
13         <div class="text-center">
14             <h1 class="display-1 fw-bold">404</h1>
15             <p class="fs-3"> <span class="text-danger">Opps!</span> Page is not exists.</p>
16             <p class="lead">
17                 The page you are looking for might have been removed.
18             </p>
19             <a href="/" class="btn btn-primary">Back to homepage</a>
20         </div>
21     </div>
22     <th:block th:replace="~{layout :: footer}"></th:block>
23 </body>
24 </html>
```

Hình 143. file 404.html được tạo ra trong thư mục error

## ➤ Cấu hình lại application.properties

Tìm đến đường dẫn chứa **src/main/resources/application.properties** tiến hành thêm đoạn **server.error.path=/error**.

**server.error.path=/error** là một cấu hình trong tệp application.properties. Khi người dùng truy cập đến một trang không tồn tại trên ứng dụng, hoặc có lỗi xảy ra trong quá trình xử lý yêu cầu của người dùng, Spring Boot sẽ tự động chuyển hướng đến đường dẫn được chỉ định trong cấu hình server.error.path để hiển thị thông báo lỗi cho người dùng.



```

</> application.properties x

1 # Database connection properties
2 spring.datasource.url=jdbc:mysql://localhost:3306/bookstore
3 spring.datasource.username=root
4 spring.datasource.password=
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6
7 # Hibernate properties
8 spring.jpa.show-sql=true
9 spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
11 spring.jpa.open-in-view=false
12
13 server.error.path=/error

```

Hình 144. Cấu hình lại file application.properties

Tạo file **CustomErrorController.java** đặt tại  
**src/main/java/com.example.demo**

Phương thức **handleError** xác định loại lỗi của yêu cầu bằng cách kiểm tra giá trị của thuộc tính **ERROR\_STATUS\_CODE** trong **HttpServletRequest**. Nếu lỗi là lỗi 404, nó sẽ trả về tên của tệp HTML hoặc template của trang lỗi 404. Nếu không, nó sẽ trả về null.

```
1 package com.example.demo.controller;
2
3     import jakarta.servlet.RequestDispatcher;
4     import jakarta.servlet.http.HttpServletRequest;
5     import org.springframework.boot.web.servlet.error.ErrorController;
6     import org.springframework.stereotype.Controller;
7     import org.springframework.web.bind.annotation.GetMapping;
8     import org.springframework.web.bind.annotation.RequestMapping;
9
10    import java.util.Optional;
11
12    /**
13     * Nguyễn Xuân Nhân
14     */
15    @Controller
16    @RequestMapping("/error")
17    public class CustomErrorController implements ErrorController {
18        /**
19         * Nguyễn Xuân Nhân
20         */
21        @GetMapping
22        public String handleError(HttpServletRequest request) {
23            return Optional.ofNullable(request.getAttribute(RequestDispatcher.ERROR_STATUS_CODE))
24                .filter(status → Integer.parseInt(status.toString()) = 404)
25                .map(status → "error/404") Optional<String>
26                .orElse( other: null);
27        }
28    }
```

Hình 145. Tạo file CustomErrorController.java

Tại **IuserRepository.java** ta thêm phương thức lấy ra tất cả các role của user hiện hành

## 7.3 Gán quyền cho User và Role

Tại tệp **CustomUserDetails.java** ta thêm và thay đổi các phương thức sau:

```
CustomUserDetail.java
import java.util.HashSet;
import java.util.Set;

public class CustomUserDetail implements UserDetails {
    private final User user;
    private final IRepository userRepository;

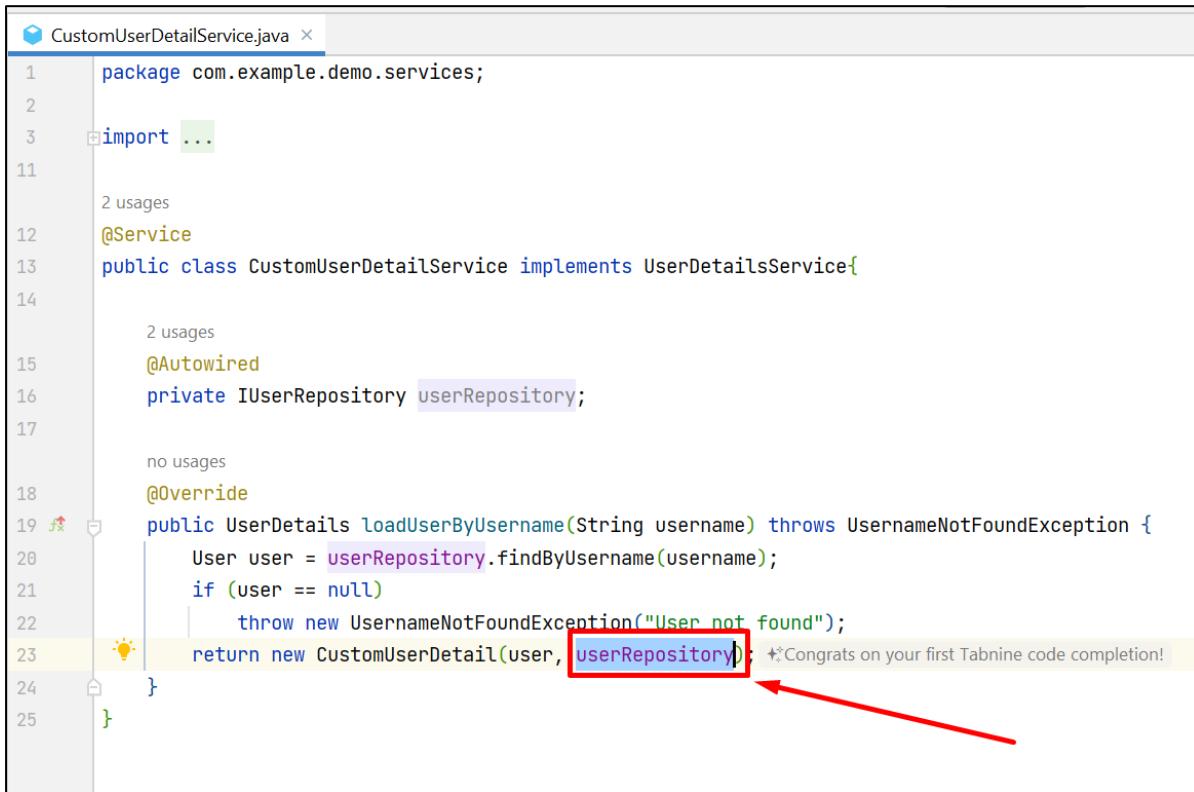
    public CustomUserDetail(User user, IRepository userRepository) {
        this.user = user;
        this.userRepository = userRepository;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        Set<SimpleGrantedAuthority> authorities = new HashSet<>();
        String[] roles = userRepository.getRolesOfUser(user.getId());
        for (String role : roles) {
            authorities.add(new SimpleGrantedAuthority(role));
        }
        return authorities;
    }

    @Override
    public String getPassword() { return user.getPassword(); }
}
```

Hình 146. Thay đổi các phương thức trong CustomUserDetail

Tại tệp **CustomUserDetailsService.java** ta thay đổi các phương thức sau:



```

1 package com.example.demo.services;
2
3 import ...
11
12 2 usages
13 @Service
14 public class CustomUserDetailsService implements UserDetailsService{
15
16     2 usages
17     @Autowired
18     private IUserRepository userRepository;
19
20     no usages
21     @Override
22     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
23         User user = userRepository.findByUsername(username);
24         if (user == null)
25             throw new UsernameNotFoundException("User not found");
26         return new CustomUserDetail(user, userRepository);
27     }
28 }

```

Hình 147. truyền userRepository vào CustomUserDetail

Lý do phải truyền **userRepository** vào **CustomUserDetail** là để đảm bảo rằng CustomUserDetail có thể sử dụng userRepository để thực hiện các truy vấn tùy ý khác vào cơ sở dữ liệu.

Tại tệp **SecurityConfig.java** ta tiến hành phân quyền truy cập resource tại **SecurityFilterChain**.

Resource	Quyền
/books/edit", "/books/delete"	ADMIN
"/books", "/books/add"	ADMIN, USER

Hình 148. phân quyền theo admin và user



```

38     no usages
39     @Bean
40     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
41         return http.csrf().disable()
42             .authorizeHttpRequests(auth -> auth
43                 .requestMatchers( ...patterns: "/css/**", "/js/**", "/", "/register", "/error") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
44                 .permitAll() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
45                 .requestMatchers( ...patterns: "/books/edit", "/books/delete") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
46                 .hasAnyAuthority( ...authorities: "ADMIN") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
47                 .requestMatchers( ...patterns: "/books", "/books/add") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
48                 .hasAnyAuthority( ...authorities: "ADMIN", "USER") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
49                 .anyRequest().authenticated())
50
51             .logout(logout -> logout.logoutUrl("/logout")
52                 .logoutSuccessUrl("/login")
53                 .deleteCookies( ...cookieNamesToClear: "JSESSIONID")
54                 .invalidateHttpSession(true)
55                 .clearAuthentication(true)
56                 .permitAll())
57
58             .formLogin(formLogin -> formLogin.loginPage("/login")
59                 .loginProcessingUrl("/login")
60                 .defaultSuccessUrl("/"))
61         )
62     )
63 }

```

Hình 149. tiến hành phân quyền truy cập resource tại SecurityFilterChain

## 7.4 ĐIỀU CHỈNH PHÂN QUYỀN TẠI GIAO DIỆN

### list.html

Tiến hành phân quyền tại file **list.html** tại

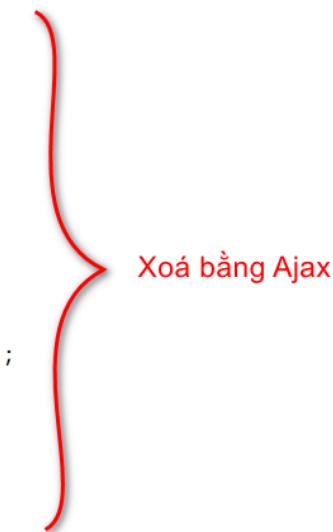
**src/main/java/com.example.demo/resources/templates/book**

```
list.html x

1 <!DOCTYPE html>
2 <html
3     xmlns:th="http://www.thymeleaf.org"
4     xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4"
5     lang="en">
6 <head>
7     <meta charset="UTF-8">
8     <title>My Book List</title>
9     <th:block th:replace="~{layout :: link-css}"></th:block>
10 </head>
11 <body>
12 <th:block th:replace="~{layout :: header}"></th:block>
13 <div class="container">
14     <table class="table">
15         <tr>
16             <th>ID</th>
17             <th>Title</th>
18             <th>Author</th>
19             <th>Price</th>
20             <th>Category</th>
21             <th sec:authorize="hasAnyAuthority('AD')">Action</th>
22         </tr>
23         <tr th:each="book : ${books}">
24             <td th:text="${book.id}"></td>
25             <td th:text="${book.title}"></td>
26             <td th:text="${book.author}"></td>
27             <td th:text="${book.price}"></td>
28             <td th:text="${book.category != null ? book.category.name : 'N/A'}"></td>
29             <td sec:authorize="hasAnyAuthority('AD')">
30                 <a th:href="@{/books/edit(id=__${book.id}__)}" class="text-info">Edit</a> | 
31                 <a th:href="@{/books/delete(id=__${book.id}__)}" 
32                     onclick="deleteBook(this); return false;" class="text-danger">Delete</a>
33             </td>
34         </tr>
35     </table>
36 </div>
```

Hình 150. phân quyền tại file list.html

```
37 <script th:src="@{/js/jquery-3.5.1.min.js}"></script>
38 <script th:inline="javascript">
39     1 usage  ✎ Nguyễn Xuân Nhân
40     function deleteBook(link) {
41         if (confirm('Are you sure?')) {
42             $.ajax({
43                 url: $(link).attr('href'),
44                 type: 'DELETE',
45                 success: result => {
46                     if (!result.success) {
47                         alert(result.message);
48                     } else {
49                         $(link).parent().parent().remove();
50                     }
51                 });
52             }
53         }
54     </script>
55 <th:block th:replace="~{layout :: footer}"></th:block>
56 </body>
57 </html>
```



Hình 151. Xoá sách bằng Ajax

## 7.5 Kiểm tra kết quả phân quyền

Các bạn kiểm tra bằng cách, đăng nhập bằng tài khoản **admin** sẽ xuất hiện cột **action** có thêm chức năng **Edit** và **Delete**. Như ảnh bên dưới.

ID	Title	Author	Price	Category	Action
3	Lập trình web	Khoa công nghệ thông tin	24000.0	Giáo dục	<a href="#">Edit</a>   <a href="#">Delete</a>
4	Kỹ thuật lập trình	Khoa công nghệ thông tin	43000.0	Giáo dục	<a href="#">Edit</a>   <a href="#">Delete</a>
5	Cách nghĩ để thành công	Napoleon Hill	168000.0	Giáo dục	<a href="#">Edit</a>   <a href="#">Delete</a>

Hình 152. Giao diện xem với quyền ADMIN

Đăng nhập bằng tài khoản user bình thường thì chỉ có thể xem List Book và Add Book.

ID	Title	Author	Price	Category
3	Lập trình web	Khoa công nghệ thông tin	24000.0	Giáo dục
4	Kỹ thuật lập trình	Khoa công nghệ thông tin	43000.0	Giáo dục
5	Cách nghĩ để thành công	Napoleon Hill	168000.0	Giáo dục

Hình 153. Giao diện xem với quyền USER

## TÓM TẮT

Sau khi học xong phân quyền 2 quyền user và admin trong Java Spring Boot, bạn sẽ có khả năng triển khai một hệ thống phân quyền bảo mật và an toàn trong ứng dụng web, đồng thời hiểu rõ vai trò và trách nhiệm của người dùng và quản trị viên trong việc sử dụng ứng dụng web.

## CÂU HỎI ÔNG TẬP

1. Phân quyền trong lập trình ứng dụng web là gì?
2. Spring Security là gì và nó được sử dụng để làm gì trong phân quyền?
3. Tại sao phân quyền là một thành phần quan trọng trong bảo mật ứng dụng web?

# BÀI 8 XÂY DỰNG CÁC TÍNH NĂNG API CƠ BẢN-APPLICATION PROGRAMMING INTERFACE

**Bài này giúp người học nắm được các nội dung sau:**

Sau khi học xong phần quyền trong Java Spring Boot, người học sẽ nắm được các kiến thức và kỹ năng cơ bản sau đây:

- ✓ Thực hiện được thao tác lấy danh sách Books bằng API
- ✓ Thực hiện được thao tác thêm sách bằng API
- ✓ Thực hiện được thao tác xoá sách bằng API
- ✓ Thực hiện được thao tác sửa sách bằng API

## Mô tả chức năng:

Chức năng lấy danh sách, thêm, xoá, sửa sách bằng API trong Java Spring Boot bao gồm:

- Lấy danh sách sách: Tạo API endpoint GET để truy vấn và trả về danh sách sách từ cơ sở dữ liệu.
- Thêm sách mới: Tạo API endpoint POST để đọc dữ liệu sách từ yêu cầu và lưu vào cơ sở dữ liệu.
- Xoá sách: Tạo API endpoint DELETE để xoá sách dựa trên mã sách hoặc ID sách được cung cấp.
- Sửa sách: Tạo API endpoint PUT để cập nhật thông tin sách dựa trên mã sách hoặc ID sách và dữ liệu cập nhật từ yêu cầu.

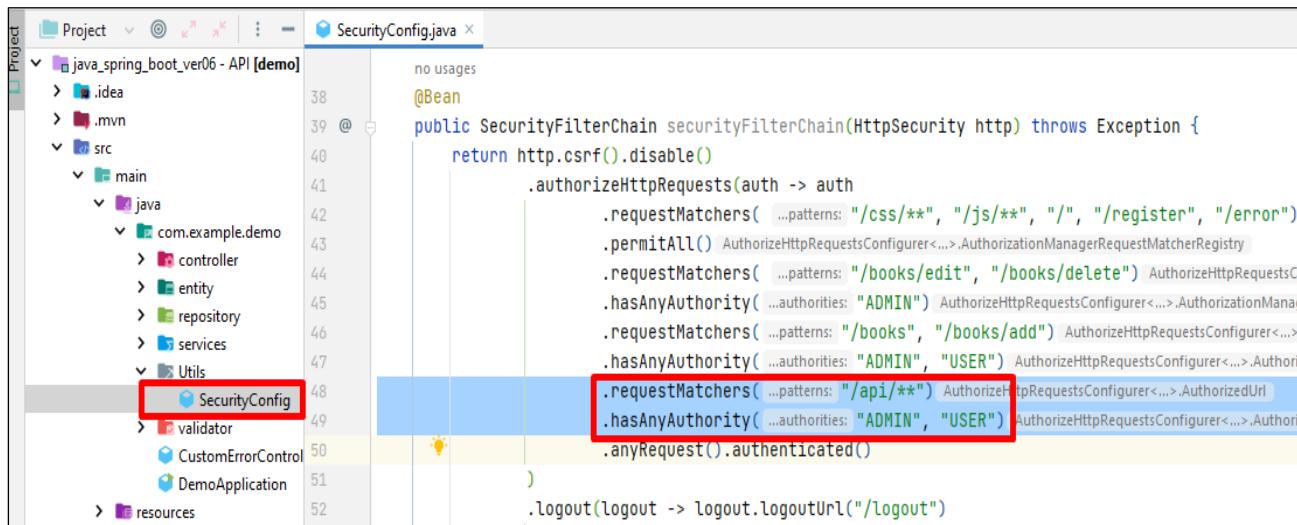
Các chức năng này có thể được triển khai bằng Spring Boot và Spring Data JPA để tương tác với cơ sở dữ liệu, và có thể bảo mật bằng Spring Security để xác thực và phân quyền truy cập.

## 8.1 Thêm authorize cho API

Authorize trong API là quá trình xác thực người dùng và cấp quyền truy cập cho các yêu cầu API. Điều này đồng nghĩa với việc chỉ những người dùng được ủy quyền mới có thể truy cập và thực thi các hoạt động trong API.

Tìm đến và mở file **SecurityConfig.java** theo đường dẫn trong thư mục `src/main/java/com/example/demo/Utils`.

Bổ sung đoạn code như bên dưới vào.



```

@Project
Project: java_spring_boot_ver06 - API [demo]
File: SecurityConfig.java
no usages
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    return http.csrf().disable()
        .authorizeHttpRequests(auth -> auth
            .requestMatchers( ...patterns: "/css/**", "/js/**", "/", "/register", "/error")
            .permitAll() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
            .requestMatchers( ...patterns: "/books/edit", "/books/delete") AuthorizeHttpRequestsC...
            .hasAnyAuthority( ...authorities: "ADMIN") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManag...
            .requestMatchers( ...patterns: "/books", "/books/add") AuthorizeHttpRequestsConfigurer<...>
            .hasAnyAuthority( ...authorities: "ADMIN", "USER") AuthorizeHttpRequestsConfigurer<...>.Authoriz...
            .requestMatchers( ...patterns: "/api/**") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
            .hasAnyAuthority( ...authorities: "ADMIN", "USER") AuthorizeHttpRequestsConfigurer<...>.Authoriz...
        ).anyRequest().authenticated()
        .logout(logout -> logout.logoutUrl("/logout"))
}

```

Hình 154. Thêm quyền authorize cho API

Định nghĩa rằng chỉ những người dùng có vai trò "ADMIN" hoặc "USER" mới được phép truy cập vào các API endpoint có đường dẫn bắt đầu là "/api/\*\*".

Dấu " \*\* " được sử dụng làm đại diện cho bất kỳ phần tử nào trong đường dẫn của URL.

## 8.2 Tạo package models chứa các Data transfer object

Package "models" là một thư mục trong dự án của bạn, được sử dụng để chứa các lớp đại diện cho các đối tượng dữ liệu cụ thể trong ứng dụng của bạn. Các lớp trong package "models" là các lớp DTO (Data Transfer Object), được sử dụng để đóng gói dữ liệu và thực hiện các thao tác liên quan đến dữ liệu, chẳng hạn như lấy, thêm, sửa, xoá, v.v... Các lớp trong package "models" thường không chứa logic xử lý nghiệp vụ, mà chỉ đơn giản là chứa các trường dữ liệu (*có thể có phương thức getter và setter*) để đại diện cho đối tượng dữ liệu trong ứng dụng của bạn. Package "models" giúp giữ gìn tính rõ ràng và phân tách logic xử lý nghiệp vụ trong ứng dụng của bạn.

Tạo một thư mục tên **dto** theo đường dẫn

**src/main/java/com/example/demo/dto**, trong thư mục **dto** tạo thêm file **BookDto.java** và thêm các thuộc tính của BookDto

```
package com.example.demo.dto;

import lombok.Data;

public class BookDto {
```

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Tree:** Shows the project structure under "java\_spring\_boot\_ver06 - API [demo]".
  - src/main/java/com.example.demo/dto/**: Contains a file named **BookDto.java**, which is highlighted with a red box.
  - Other packages like controller, entity, repository, services, Utils, SecurityConfig, validator, CustomErrorController, and DemoApplication are also listed.
- Editor:** Displays the code for **BookDto.java**. The code includes:

```
package com.example.demo.dto;

import lombok.Data;

public class BookDto {
```

no usages

```
@Data
```

```
    private Long id;
```

no usages

```
    private String title;
```

no usages

```
    private String author;
```

no usages

```
    private Double price;
```

no usages

```
    private String categoryName;
```

Hình 155. Tạo lớp BookDto

## 8.3 Tạo ApiController

Tạo thêm 1 file có tên là **ApiController** trong thư mục controller và bổ sung các phương thức như sau:

```
ApiController.java x
1 package com.example.demo.controller;
2
3 import com.example.demo.entity.Book;
4 import com.example.demo.dto.BookDto;
5 import com.example.demo.services.BookService;
6 import com.example.demo.services.CategoryService;
7 import jakarta.transaction.Transactional;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.web.bind.annotation.*;
10
11 import java.util.ArrayList;
12 import java.util.List;
13
14 import NguyenXuanNhien.*;
15
16 @RestController
17 @CrossOrigin("*")
18 @RequestMapping("/api/v1/books")
```

Chỉ định Restful Api

Thiết lập CORS

Hình 156. Chỉ định Restful API và thiết lập CORS

**@RestController**: Đánh dấu lớp là một điều khiển của Spring Boot để xử lý yêu cầu và trả về đáp ứng dưới dạng dữ liệu hoặc định dạng khác.

**@CrossOrigin("\*")**: Cho phép yêu cầu gọi từ bất kỳ nguồn nào, không bị hạn chế bởi chính sách cùng nguồn của trình duyệt.

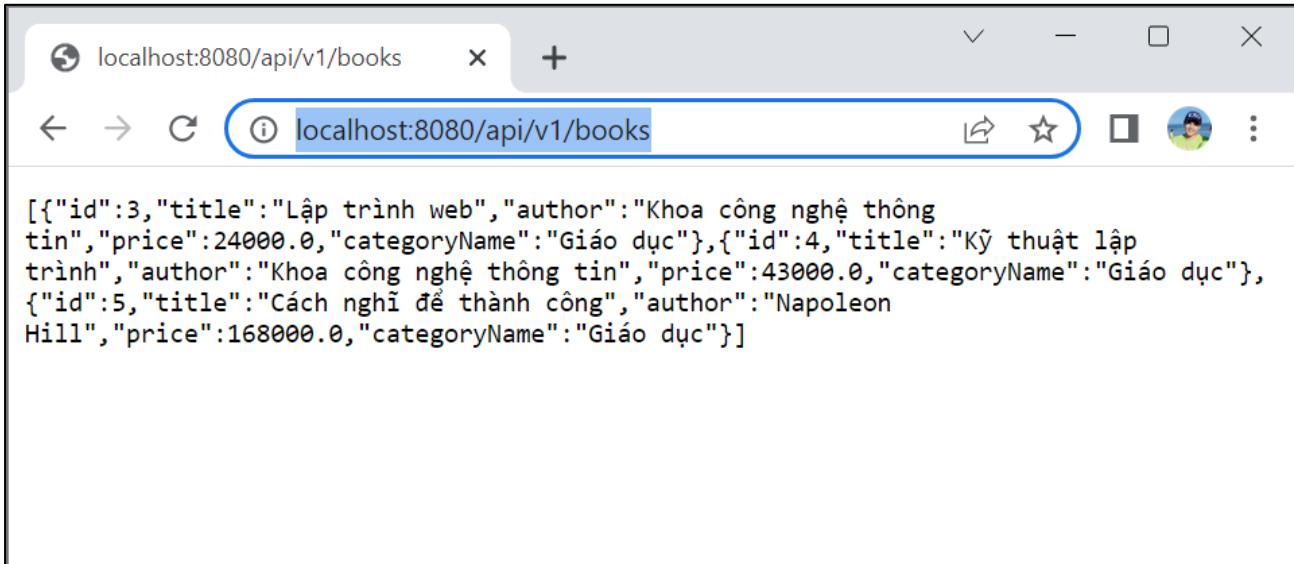
**@RequestMapping("/api/v1/books")**: Xác định đường dẫn URI mà ApiController lắng nghe để xử lý yêu cầu liên quan đến sách trong phiên bản API v1 của ứng dụng.

```

ApiController.java x
1 package com.example.demo.controller;
2
3 import com.example.demo.dto.BookDto;
4 import com.example.demo.entity.Book;
5 import com.example.demo.services.BookService;
6 import com.example.demo.services.CategoryService;
7 import jakarta.transaction.Transactional;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.web.bind.annotation.*;
10
11 import java.util.List;
12
13 /**
14 * Nguyễn Xuân Nhân *
15 */
16 @RestController
17 @CrossOrigin("*")
18 @RequestMapping("/api/v1/books")
19
20 public class ApiController {
21     @Autowired
22     private BookService bookService;
23
24     @Autowired
25     private CategoryService categoryService;
26
27     /**
28      * Nguyễn Xuân Nhân
29      */
30     private BookDto convertToBookDto(Book book) {
31         BookDto bookDTO = new BookDto();
32         bookDTO.setId(book.getId());
33         bookDTO.setTitle(book.getTitle());
34         bookDTO.setAuthor(book.getAuthor());
35         bookDTO.setPrice(book.getPrice());
36         bookDTO.setCategoryName(categoryService.getCategoryById(book.getCategory().getId()).getName());
37         return bookDTO;
38     }
39
40     /**
41      * Nguyễn Xuân Nhân
42      */
43     @GetMapping("/{id}")
44     @ResponseBody
45     public BookDto getBookById(@PathVariable Long id) {
46         return convertToBookDto(bookService.getBookById(id));
47     }
48
49     /**
50      * Nguyễn Xuân Nhân
51      */
52     @DeleteMapping("/{id}")
53     @Transactional
54     public void deleteBook(@PathVariable Long id) {
55         if (bookService.getBookById(id) != null)
56             bookService.deleteBook(id);
57     }
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
647
648
649
649
650
651
652
653
654
655
656
656
657
658
658
659
659
660
661
662
663
664
665
665
666
666
667
667
668
668
669
669
670
671
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1
```

Sau khi tạo xong các API gồm: **Lấy danh sách Books, lấy sách theo ID sách, xoá sách dựa vào ID sách**. Kết quả khi truy vấn các đường dẫn API sẽ trả về các kết quả như bên dưới.

- Gọi API lấy danh sách Books <http://localhost:8080/api/v1/books> , kết quả trả về là chuỗi JSON như ảnh bên dưới.

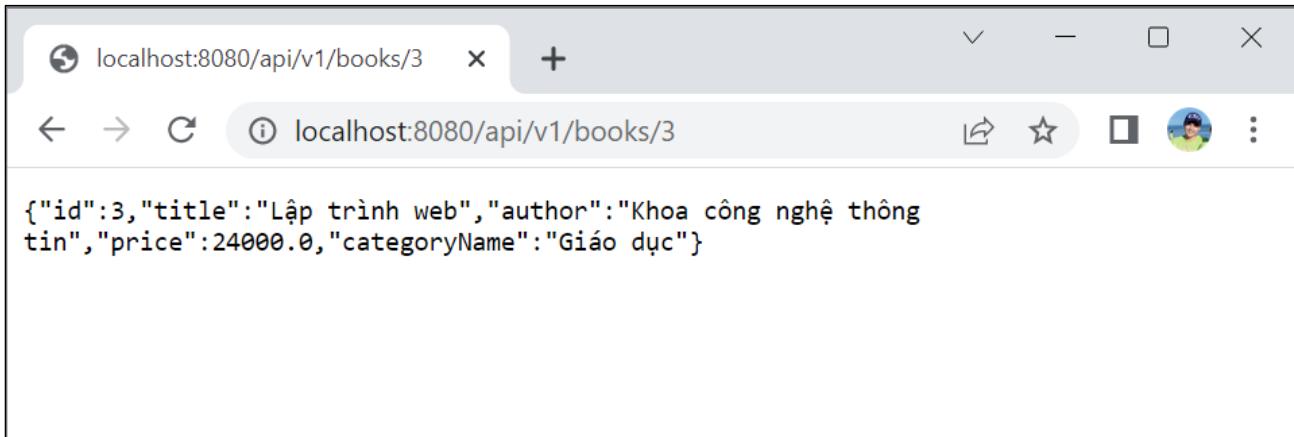


```
[{"id":3,"title":"Lập trình web","author":"Khoa công nghệ thông tin","price":24000.0,"categoryName":"Giáo dục"}, {"id":4,"title":"Kỹ thuật lập trình","author":"Khoa công nghệ thông tin","price":43000.0,"categoryName":"Giáo dục"}, {"id":5,"title":"Cách nghĩ để thành công","author":"Napoleon Hill","price":168000.0,"categoryName":"Giáo dục"}]
```

Hình 158. Gọi api lấy danh sách books

- Gọi API lấy sách Book dựa vào id của Book đó

<http://localhost:8080/api/v1/books/{id}>, ví dụ: cần tìm cuốn sách có id = 3, gọi API <http://localhost:8080/api/v1/books/3> kết quả trả về là chuỗi JSON chứa thông tin của Book có id=3 như ảnh bên dưới.



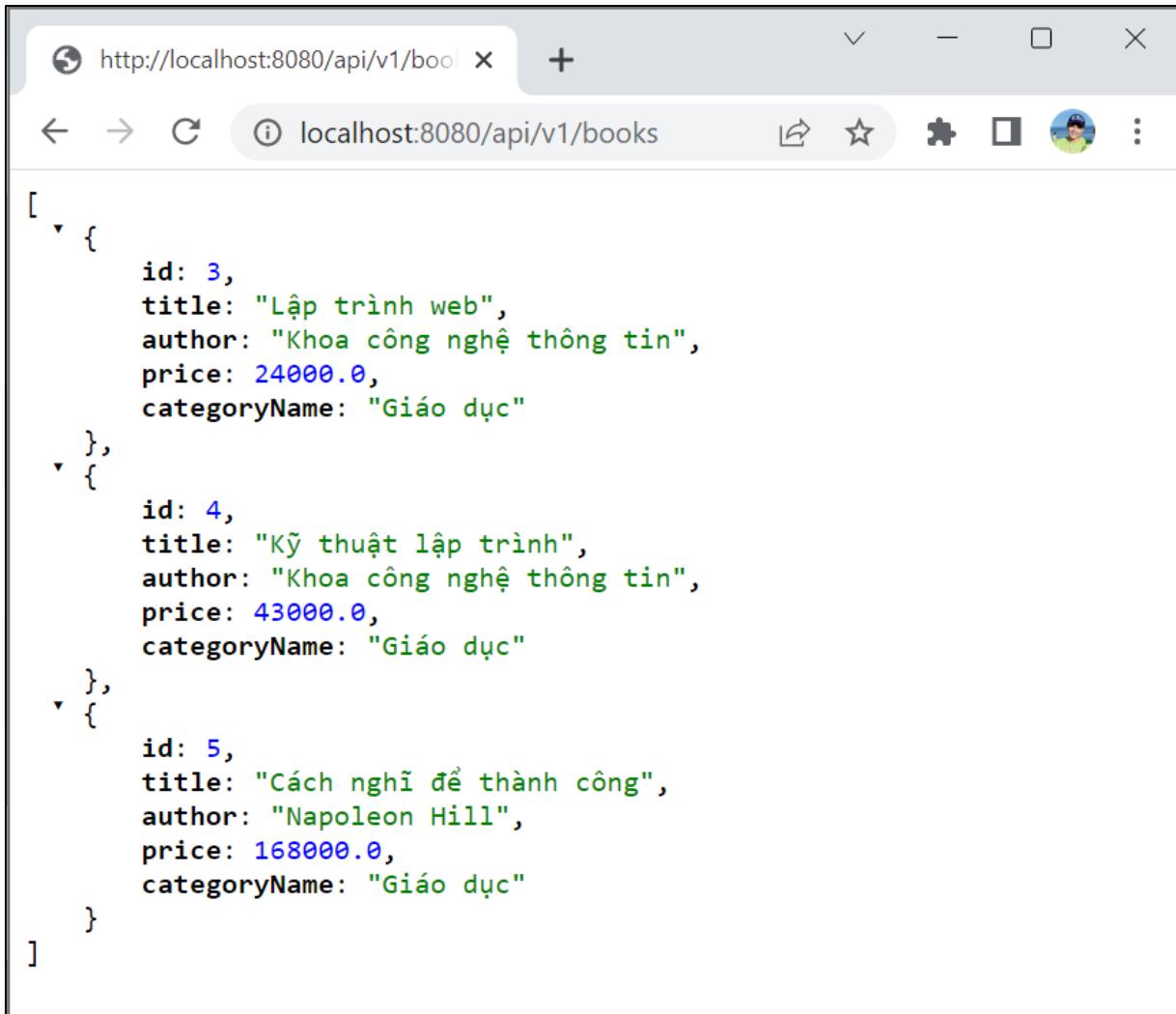
```
{"id":3,"title":"Lập trình web","author":"Khoa công nghệ thông tin","price":24000.0,"categoryName":"Giáo dục"}
```

Hình 159. Gọi api lấy thông tin sách dựa vào id

Theo mặc định, file JSON khi xem trên trình duyệt sẽ như 1 file văn bản thông thường như bên trên. Nếu dùng trình duyệt Google Chrome bạn có thể thêm tiện ích **JSONVue** để xem JSON. Bạn có thể thêm tiện ích mở rộng này thì các file JSON sẽ được hiển thị rất đẹp và trực quan hơn nhiều trên cửa sổ trình duyệt Google Chrome.

Link add extension trên GG Chrome: [JSONVue](#)

**Ví dụ** như xem kết quả hiển thị danh sách books sẽ như bên dưới.



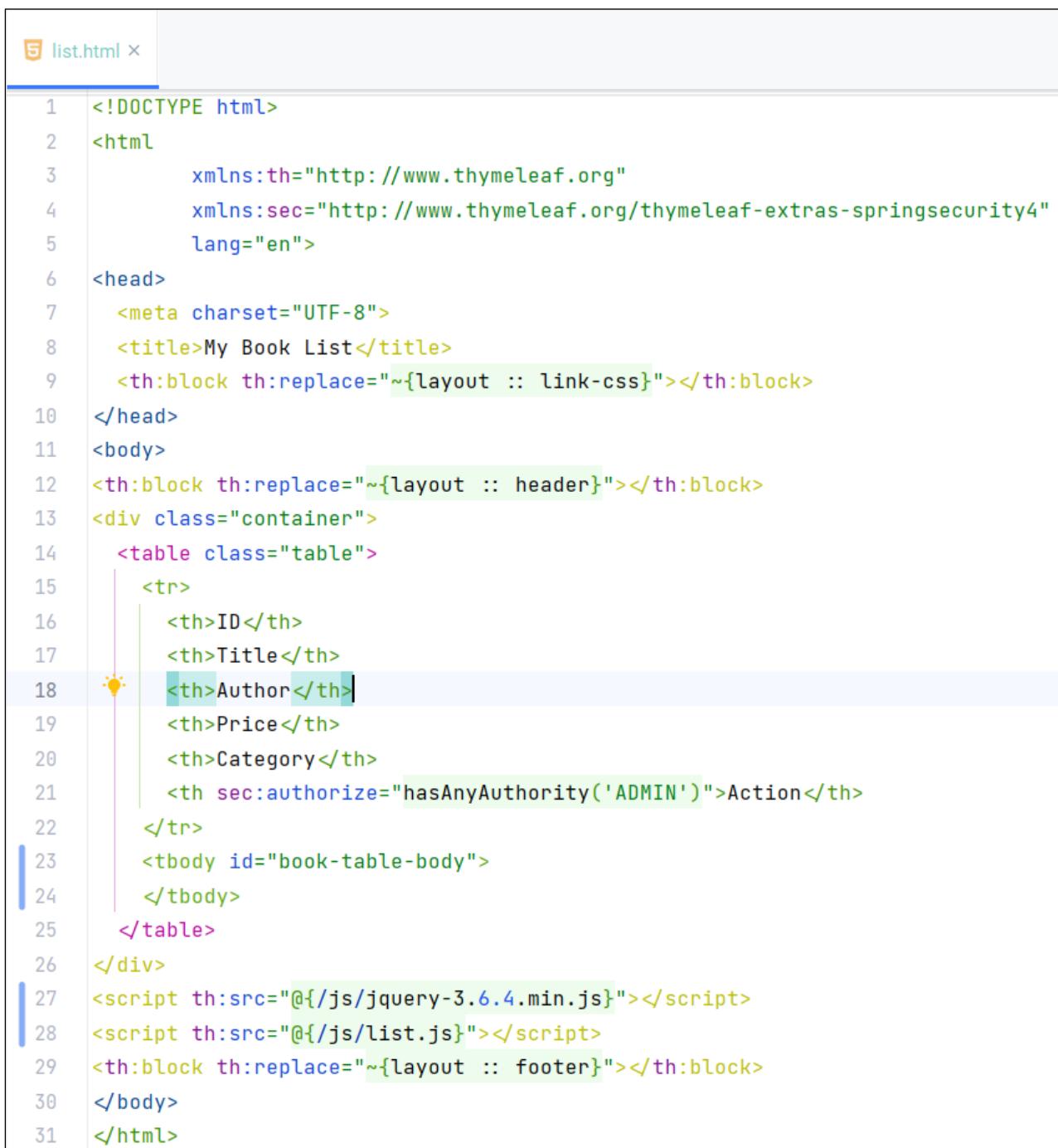
```
[  
  {  
    id: 3,  
    title: "Lập trình web",  
    author: "Khoa công nghệ thông tin",  
    price: 24000.0,  
    categoryName: "Giáo dục"  
  },  
  {  
    id: 4,  
    title: "Kỹ thuật lập trình",  
    author: "Khoa công nghệ thông tin",  
    price: 43000.0,  
    categoryName: "Giáo dục"  
  },  
  {  
    id: 5,  
    title: "Cách nghĩ để thành công",  
    author: "Napoleon Hill",  
    price: 168000.0,  
    categoryName: "Giáo dục"  
  }  
]
```

Hình 160. Thêm extend xem Json trên trình duyệt

Ngoài ra, sinh viên có thể sử dụng các công cụ khác như Postman, Insomnia, REST Client, Thunder Client, IntelliJ HTTP... để kiểm thử API

## 8.4 Chỉnh sửa List.html cho phù hợp

Chỉnh sửa giao diện **List.html** cho phù hợp với tính năng xây dựng API.



```
list.html x

1 <!DOCTYPE html>
2 <html
3     xmlns:th="http://www.thymeleaf.org"
4     xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4"
5     lang="en">
6 <head>
7     <meta charset="UTF-8">
8     <title>My Book List</title>
9     <th:block th:replace="~{layout :: link-css}"></th:block>
10 </head>
11 <body>
12 <th:block th:replace="~{layout :: header}"></th:block>
13 <div class="container">
14     <table class="table">
15         <tr>
16             <th>ID</th>
17             <th>Title</th>
18             <th>Author</th> // Cursor is here
19             <th>Price</th>
20             <th>Category</th>
21             <th sec:authorize="hasAnyAuthority('ADMIN')">Action</th>
22         </tr>
23         <tbody id="book-table-body">
24             </tbody>
25     </table>
26 </div>
27 <script th:src="@{/js/jquery-3.6.4.min.js}"></script>
28 <script th:src="@{/js/list.js}"></script>
29 <th:block th:replace="~{layout :: footer}"></th:block>
30 </body>
31 </html>
```

Hình 161. Tuỳ chỉnh lại giao diện list.html cho phù hợp

```

list.js x
1  Nguyễn Xuân Nhàn *
2  $(document).ready(function () :void { Lấy dữ liệu từ API
3      $.ajax({ url: [
4          'http://localhost:8080/api/v1/books',
5          type: 'GET',
6          dataType: 'json',
7          success: function (data) :void {
8              let trHTML :string = '';
9              $.each(data, callback: function (i, item) :void {
10                  trHTML = trHTML + `tr id="book-${item.id}"` +
11                      '<td>' + item.id + '</td>' +
12                      '<td>' + item.title + '</td>' +
13                      '<td>' + item.author + '</td>' +
14                      '<td>' + item.price + '</td>' +
15                      '<td>' + item.categoryName + '</td>' +
16                      '<td sec:authorize="hasAnyAuthority('ADMIN\\')">' +
17                      '<a href="#" + item.id + '" class="text-primary">Edit</a> | ' +
18                      '<a href="/books/" + item.id + ' class="text-danger" onclick="apiDeleteBook(' + item.id + ', this); return false;" sec:authorize="hasAnyAuthority('ADMIN\\')">Delete</a>' +
19                      '</td>' +
20                      '</tr>';
21              });
22              $('#book-table-body').append(trHTML);
23          });
24      });
25  });
26
27 usage Nguyễn Xuân Nhàn Xoá bằng API
28 function apiDeleteBook(id) :void {
29     if (confirm('Are you sure you want to delete this book?')) {
30         $.ajax({ url: [
31             'http://localhost:8080/api/v1/books/' + id,
32             type: 'DELETE',
33             success: function () :void {
34                 alert('Book deleted successfully!');
35                 $('#book-' + id).remove();
36             }
37         });
38     }
39 }

```

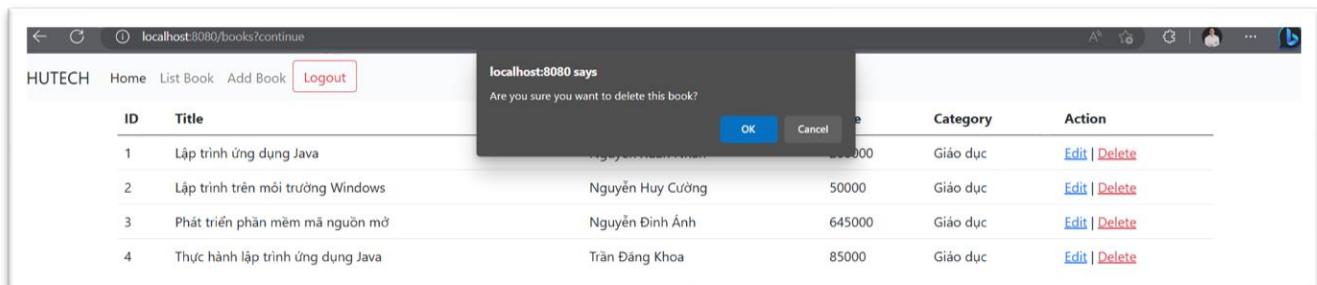
Hình 162. Các hàm lấy danh sách books và Delete book

Giao diện lấy danh sách Books sử dụng dữ liệu Json từ API khi đăng nhập bằng người dùng có quyền là USER

ID	Title	Author	Price	Category
1	Lập trình ứng dụng Java	Nguyễn Xuân Nhàn	200000	Giáo dục
2	Lập trình trên môi trường Windows	Nguyễn Huy Cường	50000	Giáo dục
3	Phát triển phần mềm mã nguồn mở	Nguyễn Đinh Ánh	645000	Giáo dục
4	Thực hành lập trình ứng dụng Java	Trần Đăng Khoa	85000	Giáo dục

Hình 163. Giao diện hiển thị danh sách Books bằng API

Giao diện Delete Book sử dụng API khi đăng nhập bằng người dùng có quyền là ADMIN



Hình 164. Giao diện xoá sách bằng API

## 8.5 Yêu cầu hoàn thành bài tập bổ sung sau

Chức năng hiển thị danh sách Books và Delete Book đã được thực hiện.

Dựa vào các nội dung đã thực hiện ở trên, sinh viên hãy thực hiện các chức năng bên dưới:

- ✓ Thực hiện thao tác thêm và sửa sách
- ✓ Thực hiện xác thực kiểm tra dữ liệu đầu vào
- ✓ **Bài tập nâng cao:** Thiết lập JWT cho việc bảo mật và phân quyền API

## TÓM TẮT

API bao gồm ba phương thức chính để thực hiện các hoạt động liên quan đến dữ liệu:

- ✓ GET: Dùng để lấy thông tin hoặc dữ liệu từ nguồn tài nguyên.
- ✓ POST: Dùng để gửi dữ liệu mới lên nguồn tài nguyên để tạo mới hoặc cập nhật thông tin.
- ✓ PUT: Dùng để cập nhật dữ liệu của nguồn tài nguyên đã tồn tại.
- ✓ DELETE: Dùng để xoá dữ liệu của nguồn tài nguyên.

Các phương thức này giúp đơn giản hóa việc truy cập và quản lý dữ liệu của API, bao gồm danh sách sách, xoá sách theo ID, và chỉnh sửa sách.

Lợi ích của việc sử dụng API cho các chức năng này bao gồm tiết kiệm thời gian và công sức, đơn giản hóa tích hợp dữ liệu và xử lý yêu cầu liên quan đến danh sách sách, đồng thời đảm bảo tính chính xác và bảo mật trong quá trình thao tác với dữ liệu sách.

## CÂU HỎI ÔNG TẬP

1. Lợi ích chính của việc sử dụng API trong phát triển ứng dụng là gì?
2. Lợi ích chính của việc sử dụng API cho các chức năng liên quan đến danh sách sách là gì?
3. API có mấy phương thức? liệt kê chi tiết ra?

# PHỤ LỤC CÀI ĐẶT MÔI TRƯỜNG VÀ CÔNG CỤ CẦN THIẾT

## Cài đặt Java JDK (Java Development Kit)

Java JDK (Java Development Kit) là một bộ công cụ để phát triển các ứng dụng Java. Java JDK cung cấp một số công cụ để biên dịch và thực thi mã Java, cũng như các thư viện và tài liệu hỗ trợ. **Lưu ý:** phiên bản cài đặt là JDK20

Để cài đặt Java JDK:

Google: JAVA JDK download

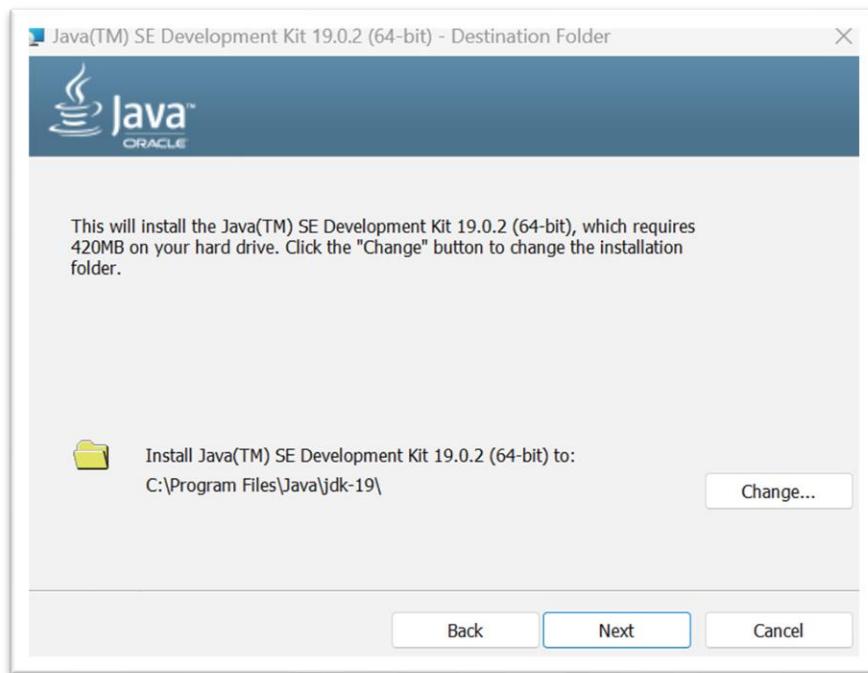
Hoặc truy cập đường dẫn sau:

<https://www.oracle.com/eg/java/technologies/downloads/>

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.		
Linux	macOS	Windows
Product/file description	File size	Download
x64 Compressed Archive	179.13 MB	<a href="https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.zip">https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.zip</a> (sha256)
x64 Installer	158.91 MB	<a href="https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.exe">https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.exe</a> (sha256)
x64 MSI Installer	157.76 MB	<a href="https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.msi">https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.msi</a> (sha256)

Hình 165. Cài đặt Java JDK X64 Installer

Cài đặt Java JDK như hướng dẫn, chọn Next..



Hình 166. Install Java SE to path

Quá trình cài đặt thành công sẽ như bên dưới.



Hình 167. Successfully installed Java JDK

## Cài đặt Laragon (Quản lý CSDL MySQL)

Laragon cung cấp tính năng quản lý CSDL (Cơ sở dữ liệu) MySQL thông qua trình quản lý cơ sở dữ liệu phpMyAdmin. Ngoài ra có thể sử dụng các phần mềm tương đương khác nhau XAMPP, WAMP, Open Server Panel...

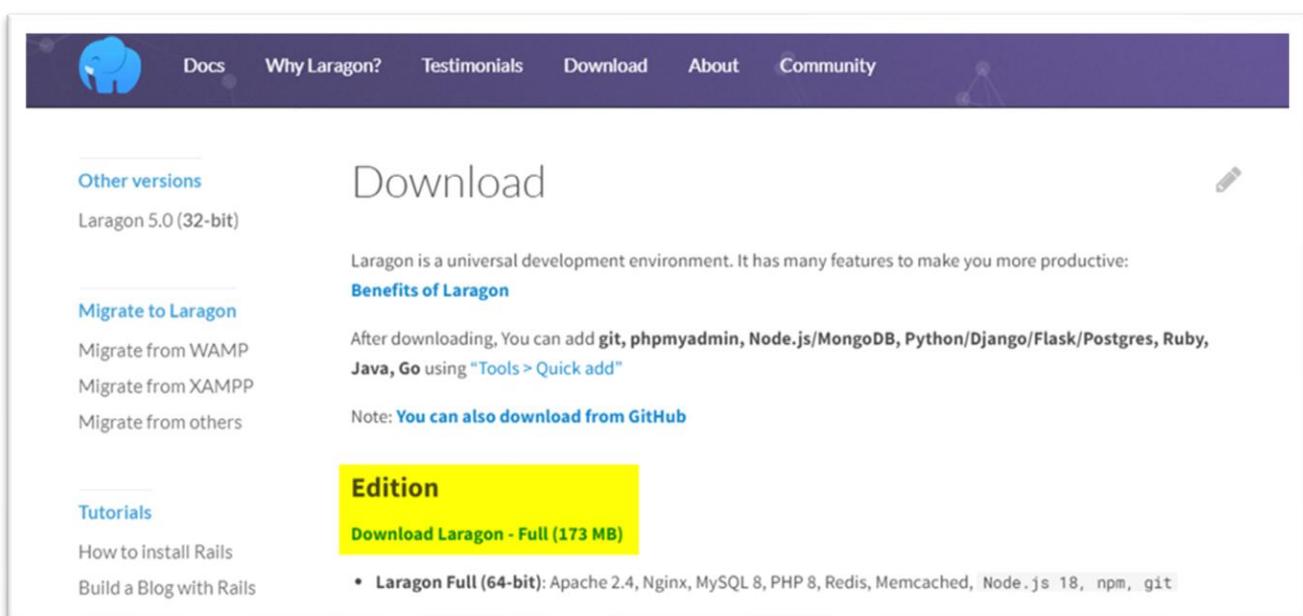
Khi bạn cài đặt và khởi chạy Laragon, nó cũng cài đặt một máy chủ MySQL đồng thời để bạn có thể tạo, quản lý và sử dụng các cơ sở dữ liệu MySQL.

Để cài đặt Laragon:

Google: download Laragon

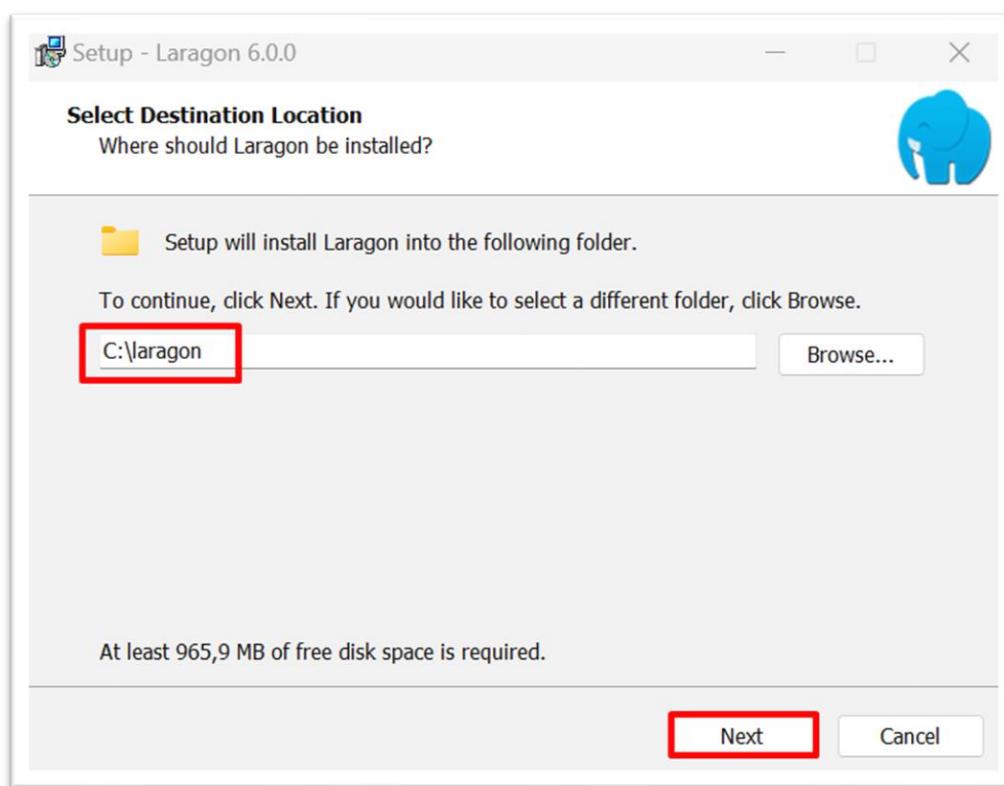
Hoặc truy cập đường dẫn sau: <https://laragon.org/download/index.html>

Tại trang Download, chọn phiên bản Download Laragon – Full (173 MB)



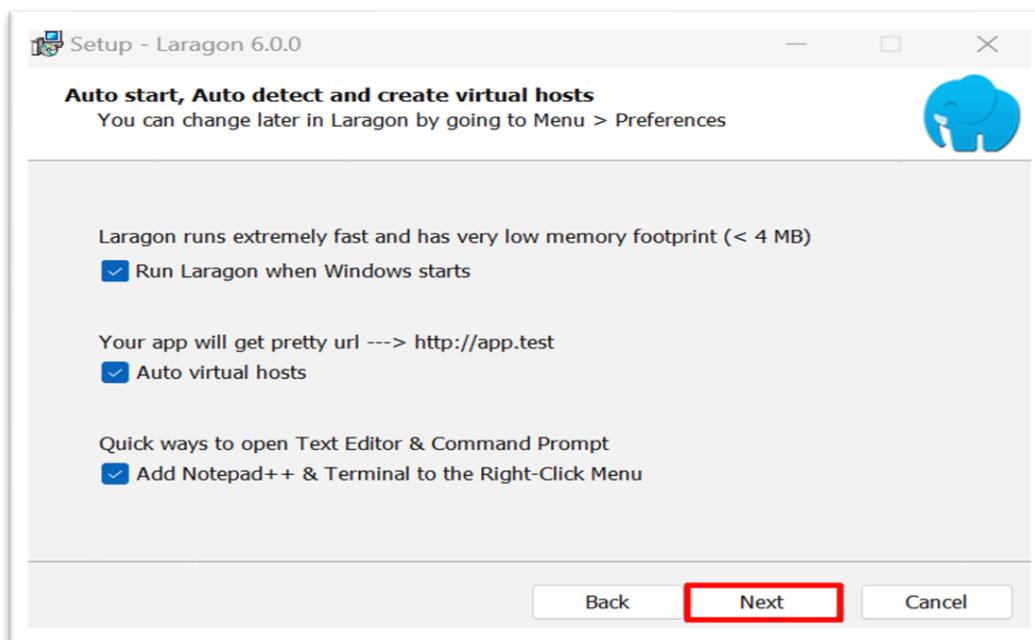
Hình 168. Download Edition Laragon - full

Tiếp tục, chọn nơi lưu trữ và nhấn Next

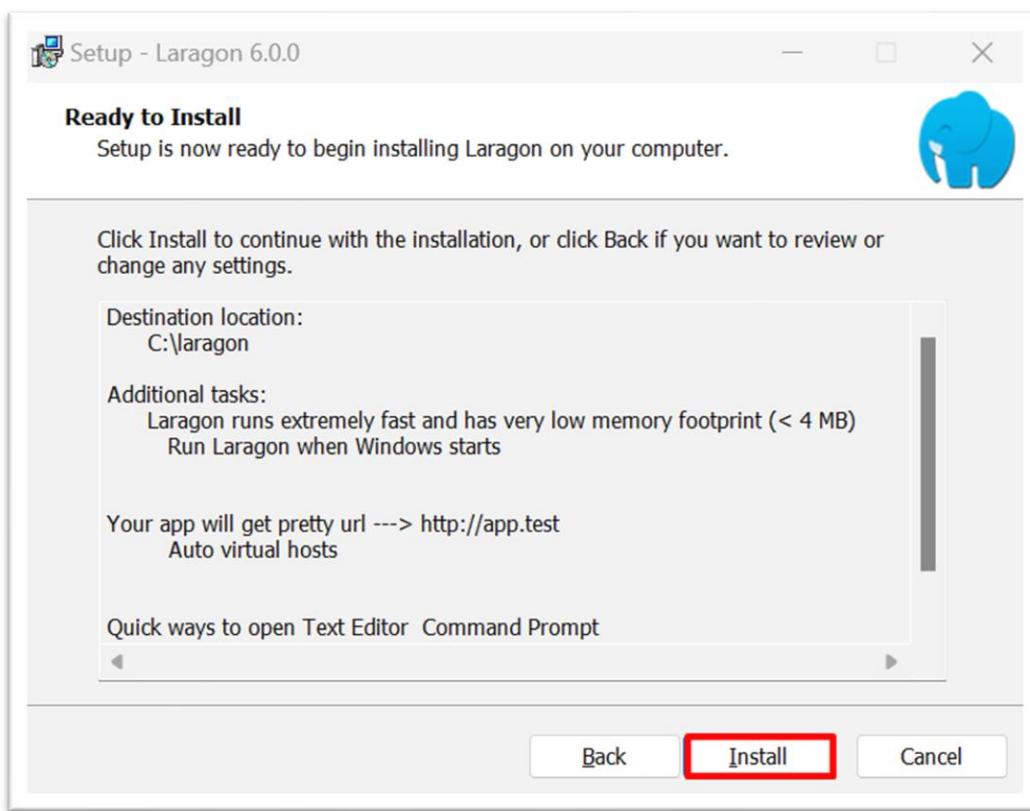


Hình 169. Setup Laragon path

Tích chọn 3 dấu tích và nhấn Next



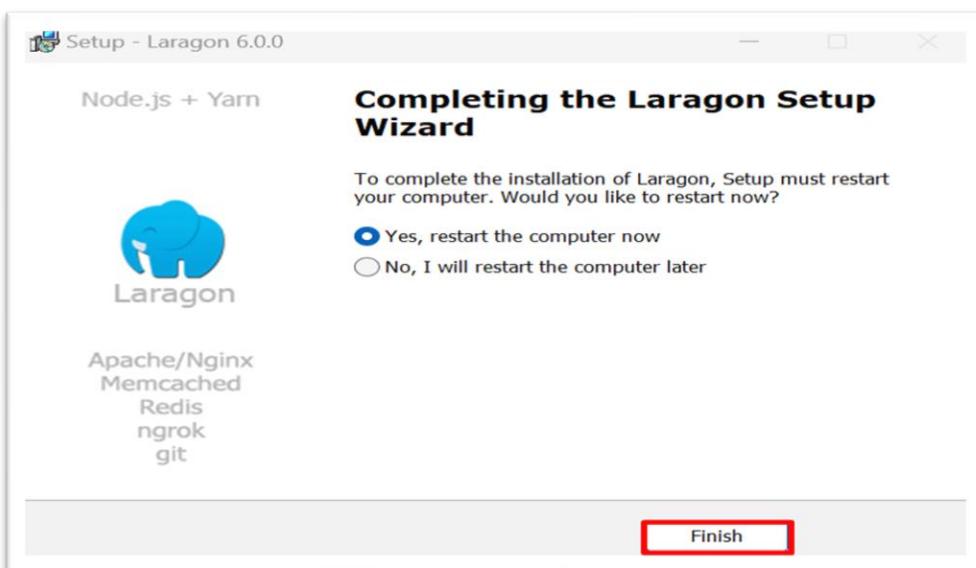
Hình 170. Setup Laragon runs extremely



Hình 171. Ready to Install Laragon

Quá trình cài đặt hoàn thành, máy tính sẽ khởi động lại.

Kiểm tra trên máy tính xuất hiện Laragon.



Hình 172. Completing the Laragon Setup

## Cài đặt IntelliJ IDEA

IntelliJ IDEA là một IDE (Integrated Development Environment) cho lập trình viên phát triển các ứng dụng sử dụng ngôn ngữ lập trình Java, Scala, Kotlin và một số ngôn ngữ khác. IntelliJ IDEA được phát triển bởi JetBrains và được coi là một trong những IDE tốt nhất cho lập trình Java.

Ngoài ra, IntelliJ IDEA còn hỗ trợ tính năng đa nền tảng, cho phép lập trình viên phát triển ứng dụng trên nhiều hệ điều hành khác nhau bao gồm Windows, MacOS và Linux.

Ngoài ra có thể sử dụng các IDE khác như: Visual Studio Code, Netbean, Eclipse, Spring Tool Suite...

Để cài đặt Laragon:

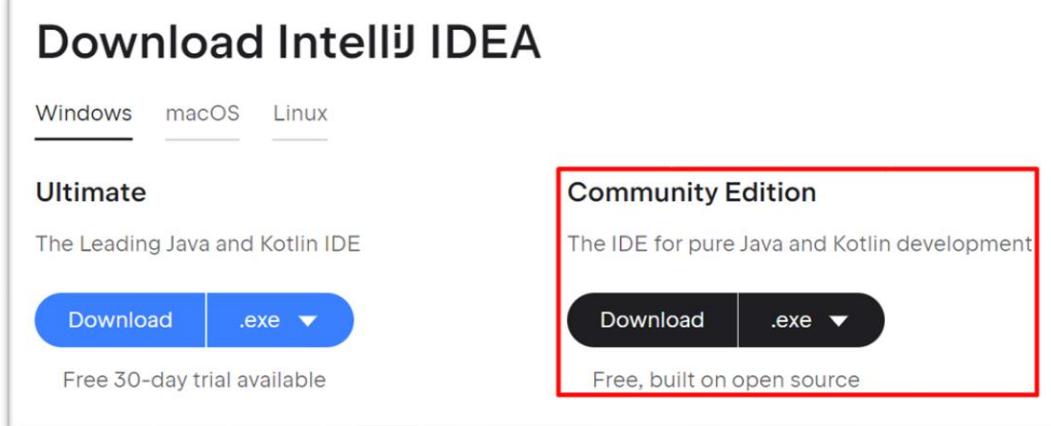
Google: Download IntelliJ IDEA Community

Hoặc truy cập đường dẫn sau:

<https://www.jetbrains.com/idea/download/#section=windows>

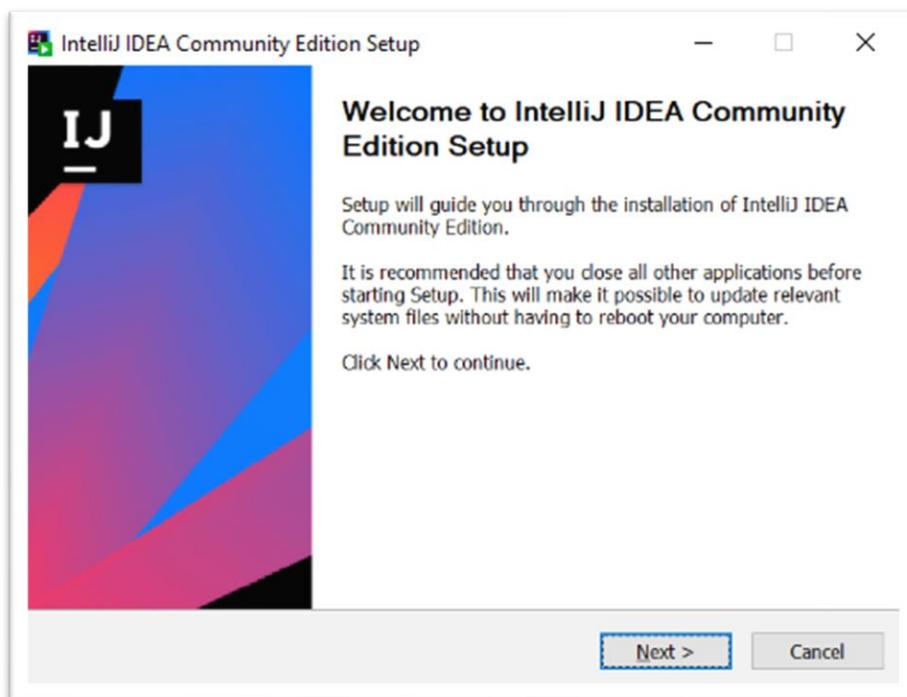
Tại trang download IntelliJ IDEA, chọn phiên bản **Community Edition** bên phải.

Phiên bản Community Edition này miễn phí cho người dùng.



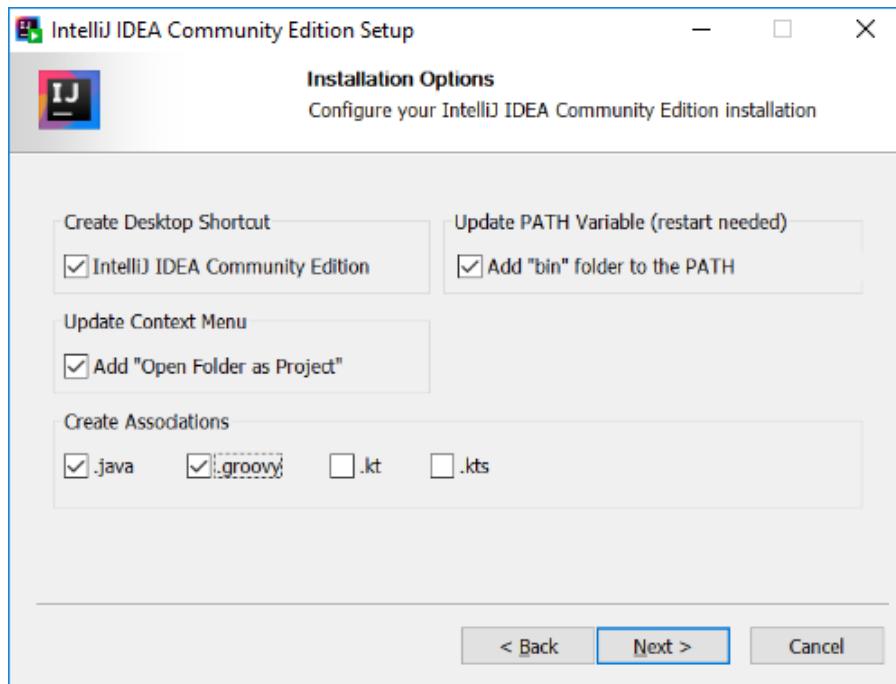
Hình 173. Download IntelliJ IDEA Community Edition

Sau khi tải về, nhấn cài đặt như hướng dẫn bên dưới. Tiếp theo, Nhấn **Next**



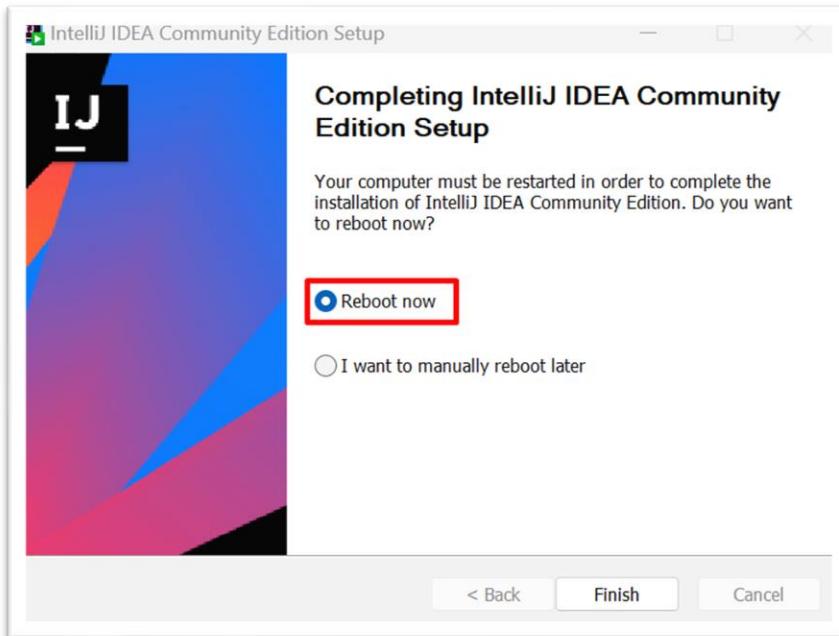
Hình 174. Next Setup IntelliJ IDEA

Ở bước này, tích chọn 4 lựa chọn như ảnh bên dưới, sau đó nhấn **Next**.



Hình 175. Installation Options IntelliJ IDEA

Tiếp theo, Tích chọn **Reboot now**, chọn **Finish** để hoàn tất cài đặt.

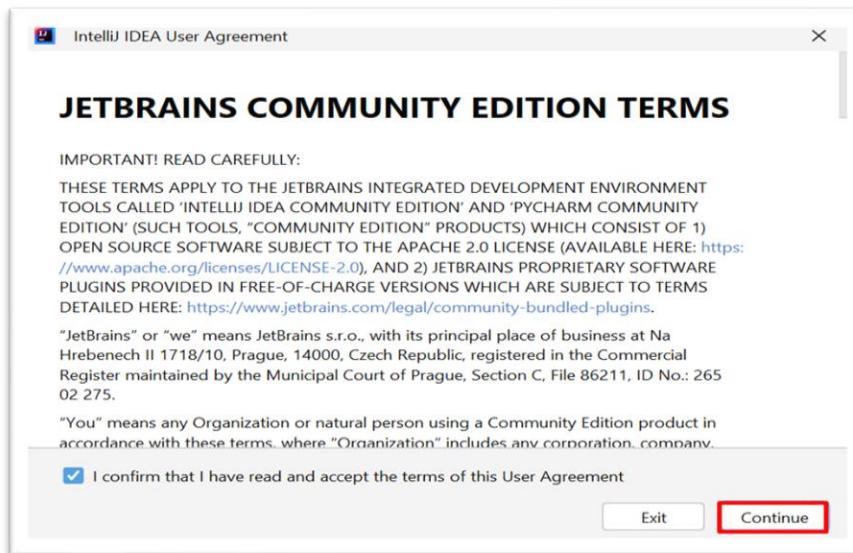


Hình 176. Completing IntelliJ IDEA Community

Sau khi hoàn tất cài đặt, tìm phần mềm trên máy và khởi động.

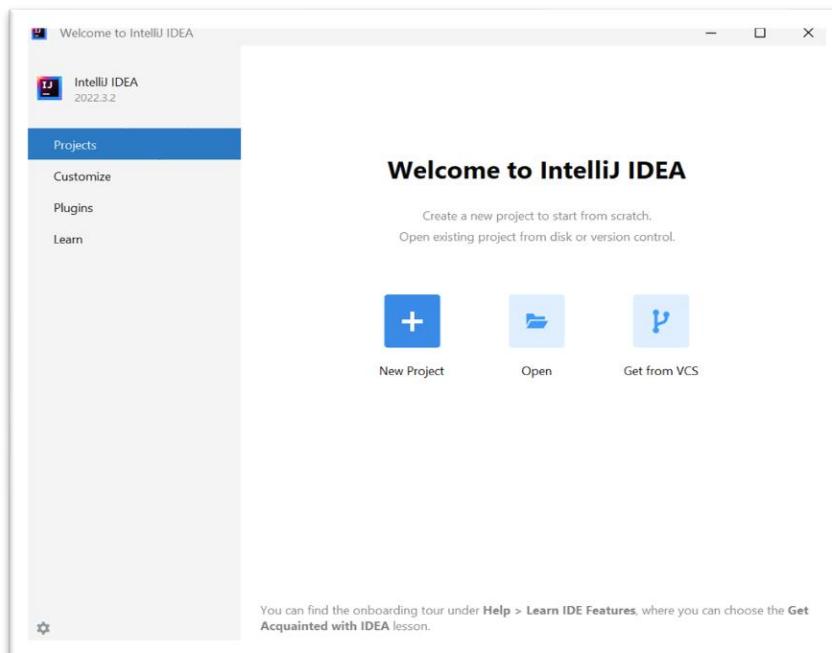


Chọn **Continue**,



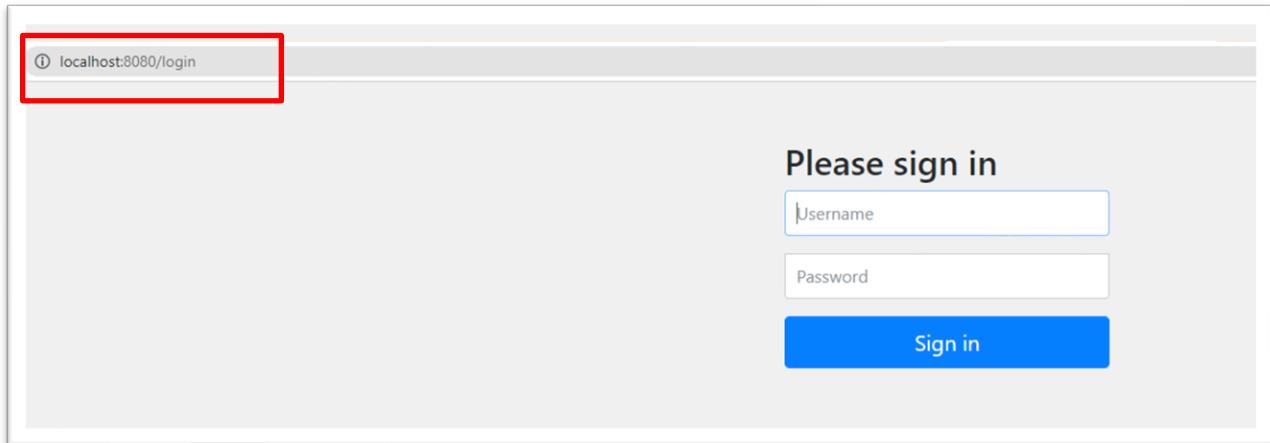
Hình 177. IntelliJ IDEA use agreement

Giao diện bắt đầu của **IntelliJ IDEA**, chúng ta sẽ quay lại nó sau.



Hình 178. Giao diện bắt đầu của IntelliJ IDEA

Sau đó, thực hiện mở trình duyệt trên máy tính, truy cập vào đường dẫn mặc định: <http://localhost:8080/>, kết quả xuất hiện giao diện đăng nhập.



Hình 34. Giao diện mặc định đăng nhập

Trang mặc định, được cấu hình khởi chạy từ **Spring Security**

Tắt **Spring Security** để thực hiện các demo phía dưới

Chỉnh sửa file **src/application.properties**, thêm phần **#Spring Security**

```
# Database connection properties
spring.datasource.url=jdbc:mysql://localhost:3306/mydatabase
spring.datasource.username=
spring.datasource.password=

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

#Spring Security
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration
```