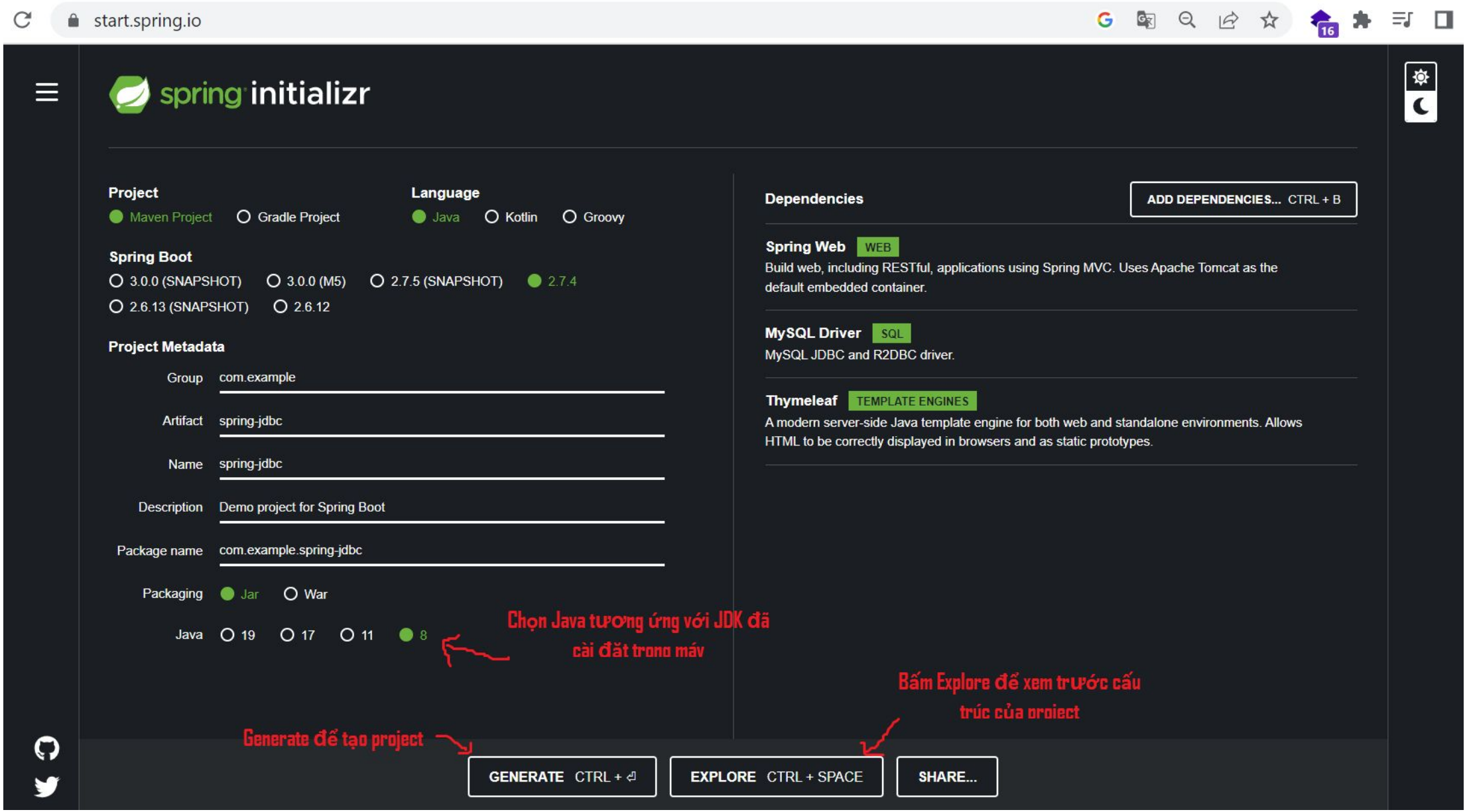
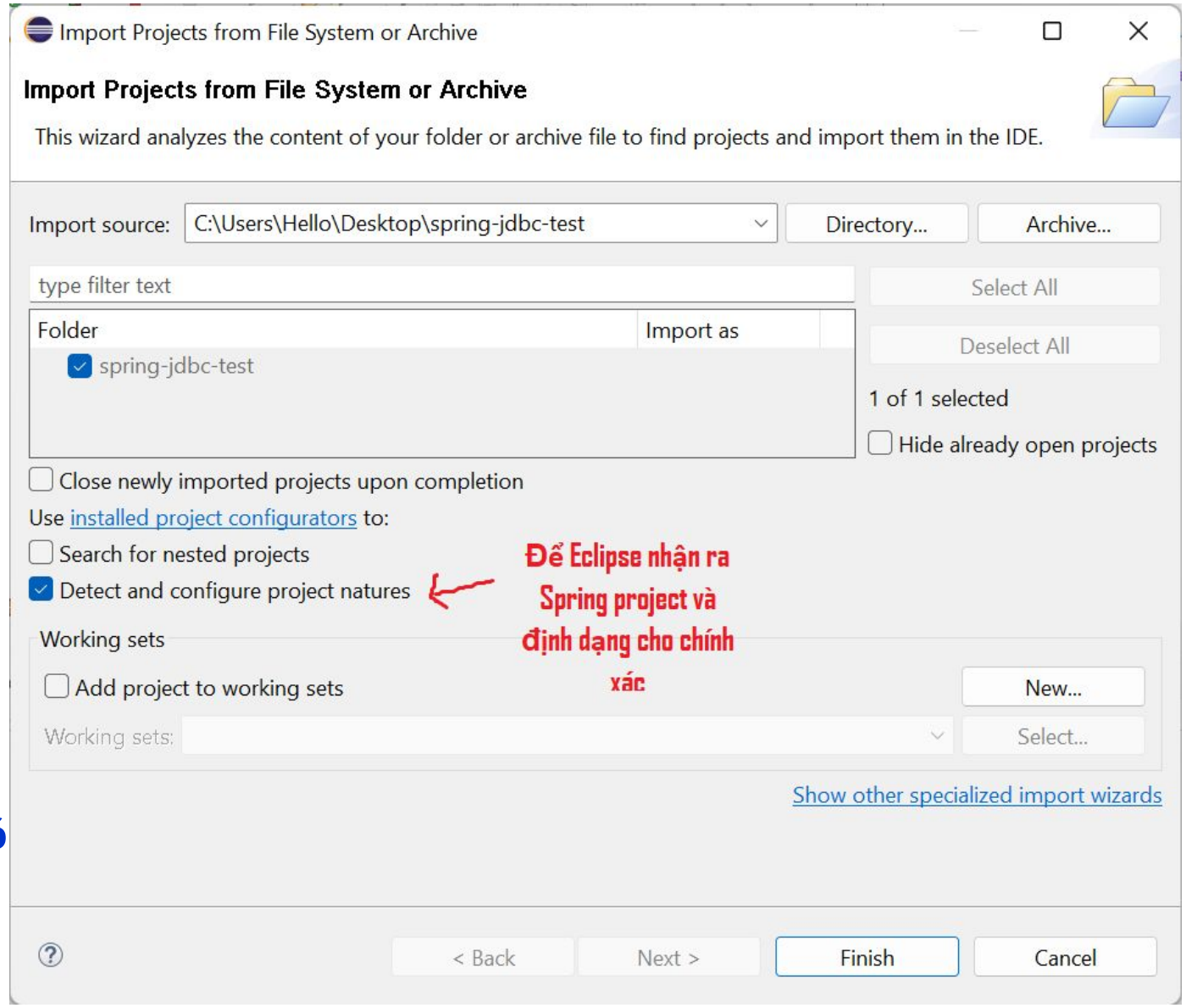
JDBC + Thymeleaf

1. Spring Boot
2. Vẫn sử dụng sở dữ liệu Book
3. Get Books, Get Book
4. Post, Put Book

Vào https://start.spring.io/ để tạo nhanh Spring project

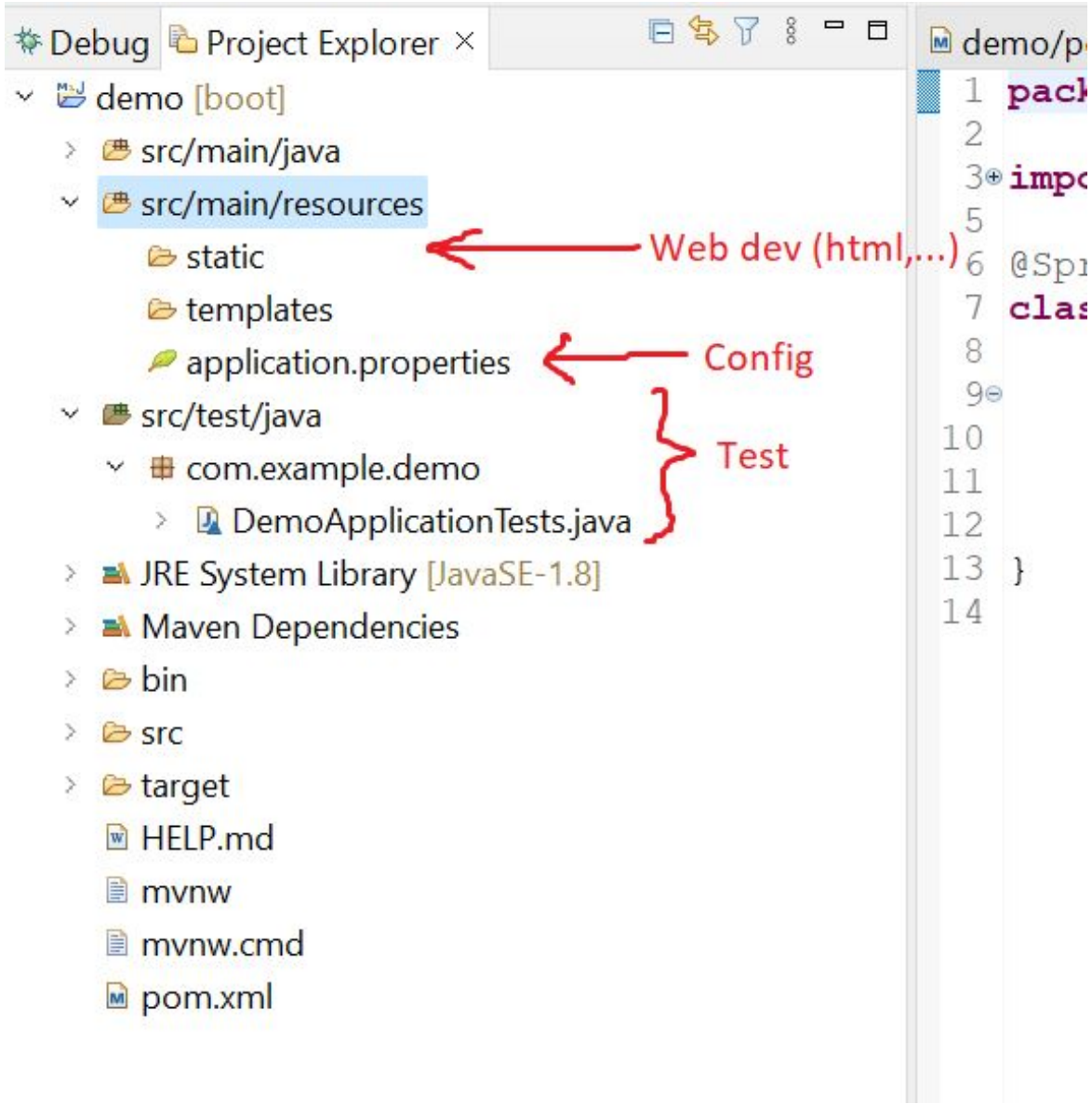


Sau khi Download project từ start.spring.io, giải nén rồi mở project trên eclipse =>



Sau khi bấm Finish, Eclipse có thể sẽ mất vài giây để định dạng lại project cho đúng

Cấu trúc của Spring Project



- Phần lớn code java cho backend sẽ được viết ở src/main/java
- src/main/resources chứa file html,css,js trả về cho frontend

File pom.xml thừa kế từ Maven, dùng để quản lý dependencies

Servlet vs Spring Controller

```java
@WebServlet("/test")
public class TestServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletExcep
        RequestDispatcher rd = req.getRequestDispatcher("test.html");
        rd.forward(req, resp);
    }
}
```

Cả 2 đều là Controller trả về trang test.html

```java
@Controller
public class TestController {
    @GetMapping("/test")
    public String test() {
        return "test";
    }
}
```

Khai báo đâu là Apl sử dụng phương thức (Get, Post, Put,...) nào tùy ý, không cần sử dụng hàm thừa kế từ Class cha như Servlet

File test.html của Spring project đặt tại folder **templates** trong src/main/resources

Ví dụ trang web quản lý sách
Vẫn sử dụng sở dữ liệu book từ buổi trước

Xây dựng ứng dụng nhập sách trong thư viện với các yêu cầu:

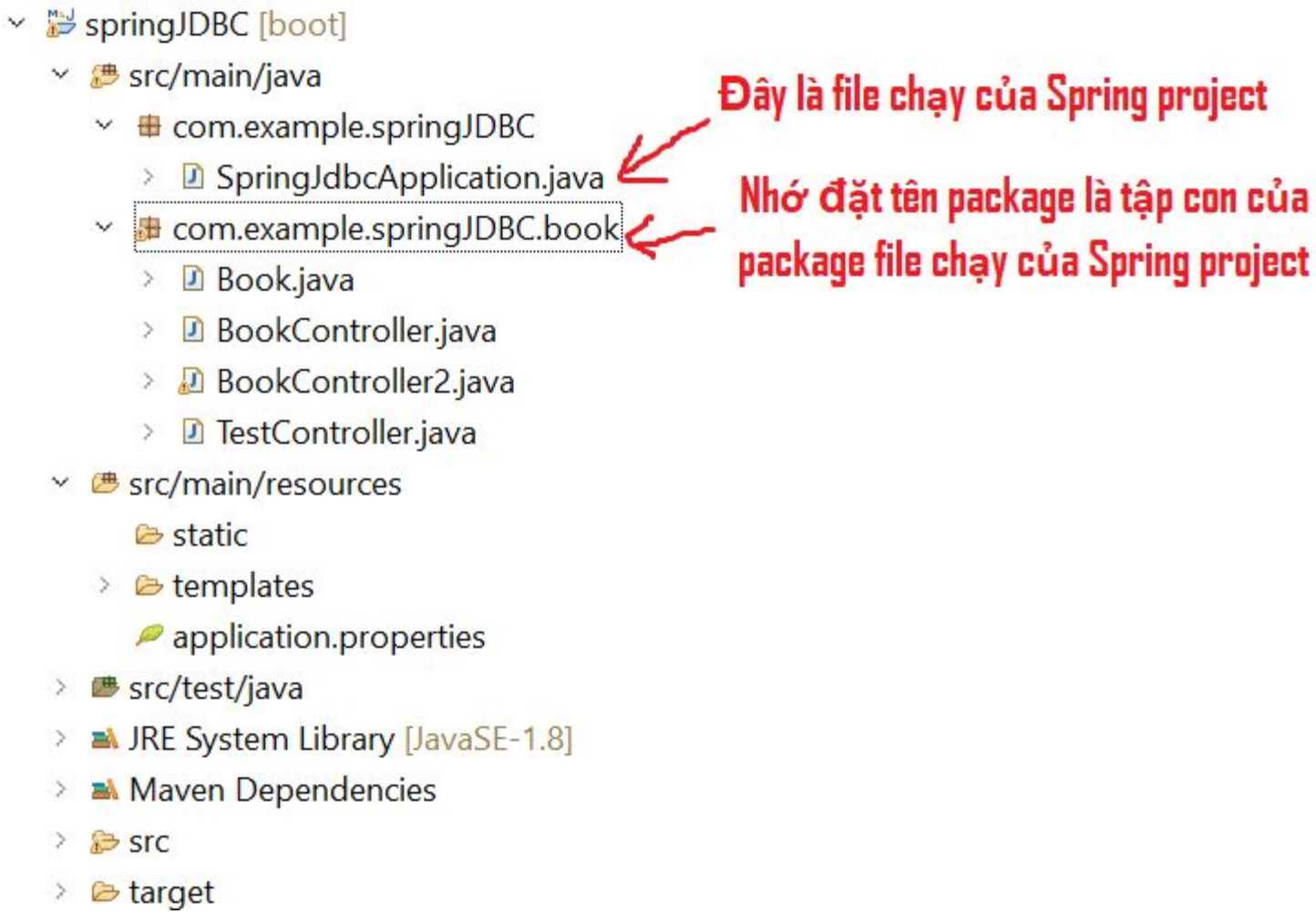
1. Tạo schema có tên là "jdbc_demo"
2. Tạo CSDL gồm 1 bảng có cấu trúc như sau:

Book

STT	Tên trường	Kiểu dữ liệu
1	bookcode	INT
2	title	TEXT
3	author	TEXT
4	category	TEXT
5	approved	INT

3. Thêm (INSERT) vào CSDL 2 dòng dữ liệu ví dụ tự chọn.
4. Đọc hết dữ liệu trong bảng Book (SELECT).
5. Cập nhật (UPDATE) 1 trong 2 dòng dữ liệu trên giá trị "category" và "approved".
6. Xóa (DELETE) 1 trong 2 dòng dữ liệu trên

Tạo package mới và bắt đầu viết các class Java



Có nhiều hình thức sắp xếp cấu trúc project, theo tài liệu của Spring thì cấu trúc bên dưới được khuyên dùng

```
com
    +- example
        +- myapplication
            +- Application.java
            +- customer
                +- Customer.java
                +- CustomerController.java
                +- CustomerService.java
                +- CustomerRepository.java
            +- order
                +- order.java
                +- OrderController.java
                +- OrderService.java
                +- OrderRepository. java
```

Cấu trúc cho trang web đặt hàng

Tất cả các Service về book sẽ sử dụng 1 class Book Model

Book Model (Book.java)

```
package com.example.springJDBC.book;

public class Book {
    private int bookcode;
    private String title;
    private String author;
    private String category;
    private boolean approved;

    // Tự generate constructors, getters and setters bên dưới


}
```

Tạo class BookController.java để truy cập CSDL Book đã tạo bên trên

```
package com.example.springJDBC.book;

import org.springframework.stereotype.Controller;

@Controller
public class BookController{

    //get all books ở đây
    //get 1 book ở đây
    //post ở đây
    //put ở đây
    //delete ở đây

}
```

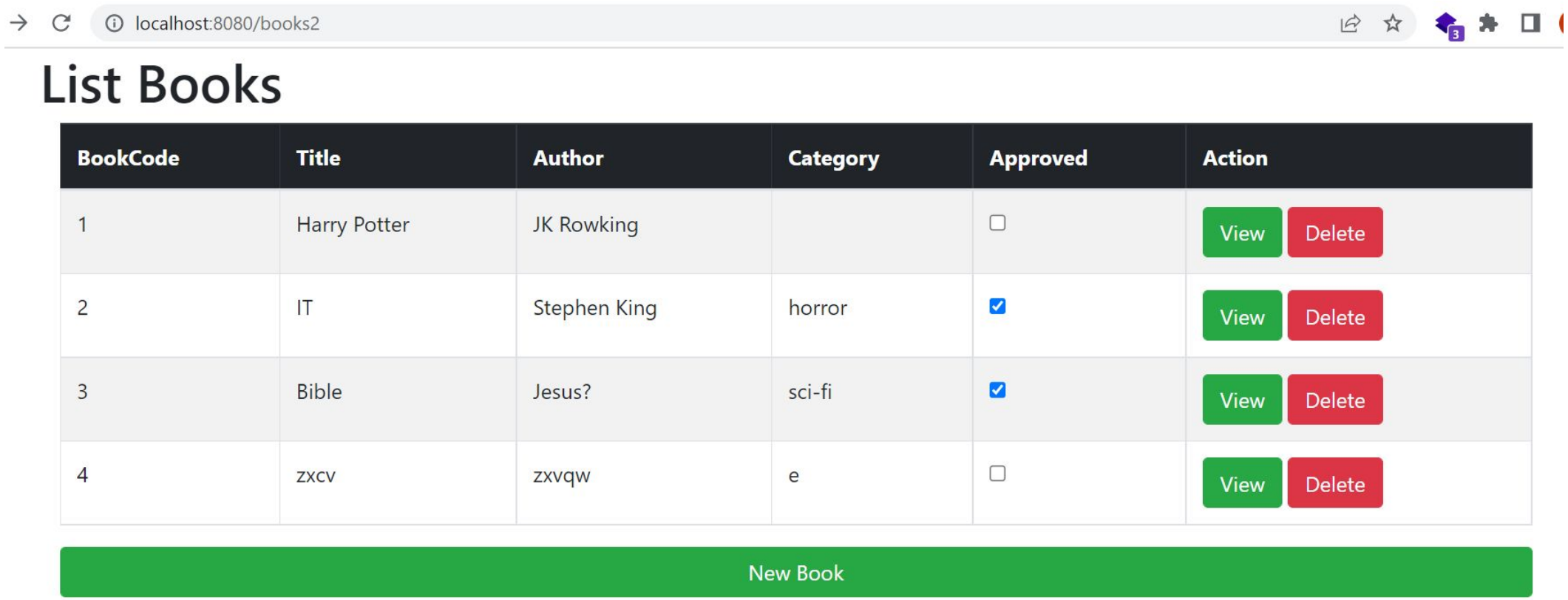
Bên dưới là hàm getBooks để lấy tất cả những giá trị sách

```
@GetMapping("/books2")
    public String getBooks(Model model) throws IOException {
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;
        List<Book> books = new ArrayList<Book>();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "password");
            statement = connection.createStatement();
            resultSet = statement.executeQuery("select * from book");
            while (resultSet.next()) {
                int bookcode = resultSet.getInt("bookcode");
                String title = resultSet.getString("title");
                String author = resultSet.getString("author");
```

Hàm trên trỏ về file “books.html” với dãy giá trị các quyển sách đặt trong attribute “books” => Ta tạo file books.html trong folder template để hiển thị các giá trị sách

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Books</title>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6
JXm"
crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="row">
<h1>List Books</h1>
</div>
<table class="table table-striped table-bordered">
<thead class="table-dark">
<tr>
<th>BookCode</th>
<th>Title</th>
```

Kết quả hàm getBooks trên trả về



Bấm vào nút View hay Add Book để đưa đến trang Book Detail, Viết hàm getBook để trả về 1 giá trị Sách

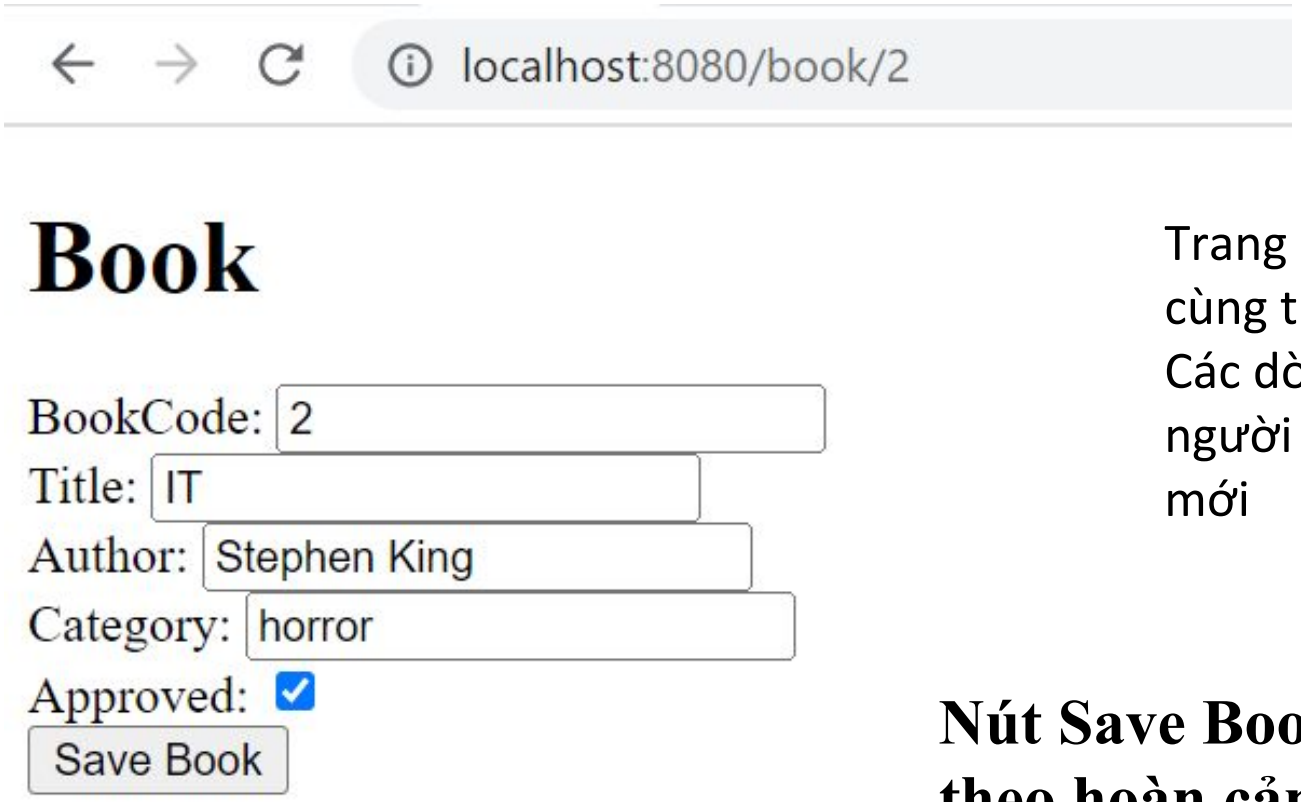
```
@GetMapping("/book/{bookcode}")
	public String getBook(Model model, @PathVariable String bookcode) {
		model.addAttribute("bookcode", bookcode);
		Connection connection = null;
		PreparedStatement ps = null;
		ResultSet result = null;

		Book book = new Book();
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"password");
			ps = connection.prepareStatement("select * from book where bookcode = ?");
			ps.setInt(1, Integer.valueOf(bookcode));
			result = ps.executeQuery();
			while (result.next()) {
				book.setBookcode(result.getInt("bookcode"));
				book.setTitle(result.getString("title"));
				book.setAuthor(result.getString("author"));
				book.setCategory(result.getString("category"));
				book.setApproved(result.getInt("approved") != 0 ? true : false);
			}
```

Trang book-detail.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Book</title>
</head>
<body>
<div class="row">
<h1>Book</h1>
</div>
<form th:object="${book}" th:action="@{save/{id}(id=${book.bookcode})}"
th:method="${bookcode} < 0 ? POST : PUT">
BookCode: <input type="number" name="bookcode" th:value="${book.bookcode}"><br />
Title: <input type="text" name="title" th:value="${book.title}"><br />
Author: <input type="text" name="author" th:value="${book.author}"><br />
Category: <input type="text" name="category" th:value="${book.category}"><br />
Approved: <input type="checkbox" name="approved" th:checked="${book.approved}"><br />
<input type="submit" value="Save Book">
</form>
</body>
</html>
```

Kết quả hàm getBook trên trả về (Sau khi bấm View)



Trang "Add Book" cũng sử dụng cùng trang book-detail.html. Các dòng ở đây sẽ là trống để người dùng điền thông tin sách mới

Nút Save Book sẽ thực hiện 2 công dụng tùy theo hoàn cảnh: sẽ Tạo mới sách nếu người dùng tạo mới; sẽ Cập Nhật sách nếu người dùng view Sách cũ

Bên dưới là hàm addBook để tạo mới giá trị sách trong CSDL

```
@PostMapping("/book/save/{bookcode}")
    public String addBook(Book book, @PathVariable String bookcode) {
        Connection connection = null;
        PreparedStatement ps = null;
        int result = 0;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"password");
            ps = connection.prepareStatement("INSERT INTO book VALUES (?, ?, ?, ?,
?)");
            ps.setInt(1, Integer.valueOf(book.getBookcode()));
            ps.setString(2, book.getTitle());
            ps.setString(3, book.getAuthor());
            ps.setString(4, book.getCategory());
            ps.setInt(5, book.isApproved() ? 1 : 0);
            result = ps.executeUpdate();

            ps.close();
```

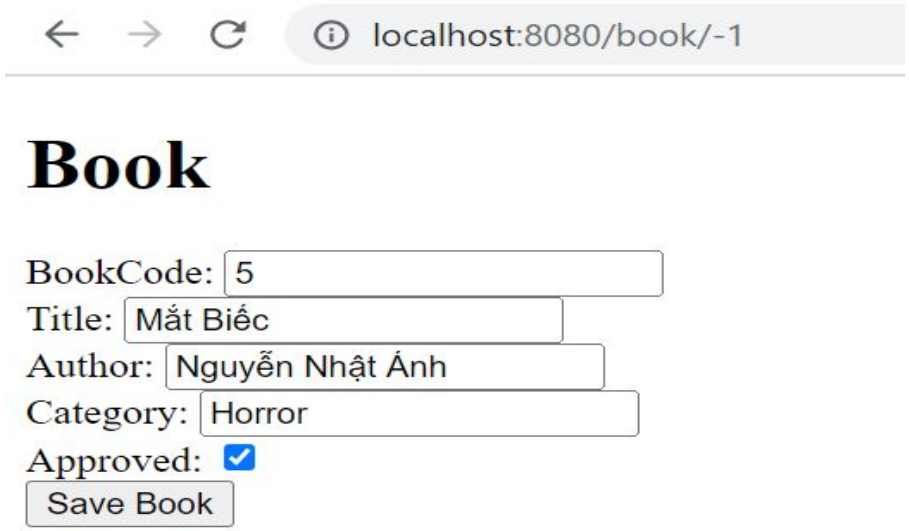
Bên dưới là hàm updateBook để cập nhật giá trị sách trong CSDL

```
@PutMapping("/book/save/{bookcode}")
    public String updateBook(Book book, @PathVariable String bookcode) {
        Connection connection = null;
        PreparedStatement ps = null;
        int result = 0;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"password");
            ps = connection.prepareStatement("UPDATE book SET
title=?,author=?,category=?,approved=? WHERE bookcode=?");
            ps.setString(1, book.getTitle());
            ps.setString(2, book.getAuthor());
            ps.setString(3, book.getCategory());
            ps.setInt(4, book.isApproved() ? 1 : 0);
            ps.setInt(5, Integer.valueOf(book.getBookcode()));
            result = ps.executeUpdate();

            ps.close();
```

Trang thêm sách



Sau khi thêm sách thành công website sẽ chuyển về trang danh sách sách ở bên trên

List Books

BookCode	Title	Author	Category	Approved	Action
1	Harry Potter	JK Rowking		☐	View Delete
2	IT	Stephen King	horror	☑	View Delete
3	Bible	Jesus?	sci-fi	☑	View Delete
4	zxcv	zxvqw	e	☐	View Delete
5	Mắt Biếc	Nguyễn Nhật Ánh	Horror	☑	View Delete

Chúng ta đã có 3 API phương thức chính (GET, PUT, POST). Các em tìm hiểu và viết nốt phương thức xóa (DETETE)

Lưu ý: Các hàm ví dụ ở trên chưa hề có bất kì 1 sự kiểm tra (validation) nào.

Luôn nhớ phải thực hiện validate dữ liệu nhận được:

- Báo lỗi khi có ô nhập dữ liệu bị để trống.
- Báo lỗi khi dữ liệu nhập vào bị sai định dang.
- Không cho phép nhập trùng dữ liệu sách đã có trong CSDL.
- Nếu có lỗi quay trở về form nhập ban đầu và hiển thị lỗi cạnh ô nhập.
- …..