

On Minh Quan

ITITWE23009

## Web App Dev Lab 7 practice report

**List all products:** [http:// 127.0.0.1:8081/products](http://127.0.0.1:8081/products)

ID	Code	Name	Price	Quantity	Category	Actions
1	P001	Laptop Dell XPS 13	\$1299.99	10	Electronics	<button>Edit</button> <button>Delete</button>
2	P002	iPhone 15 Pro	\$999.99	25	Electronics	<button>Edit</button> <button>Delete</button>
3	P003	Office Chair	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>

Browser sends GET /products.

@GetMapping without a path on a controller annotated with  
@RequestMapping("/products") handles this in listProducts(Model model).

Controller calls productService.getAllProducts(), which calls productRepository.findAll() to load all Product entities from the products table.

The result List<Product> is put into the model as attribute "products".

Method returns "product-list", so Thymeleaf renders templates/product-list.html, which loops over \${products} and displays them

**Add new product:** [http:// 127.0.0.1:8081/products/new](http://127.0.0.1:8081/products/new)

The image consists of two screenshots of a web application, likely built with Spring Boot and Thymeleaf.

**Screenshot 1: Add New Product Form**

This screenshot shows a modal dialog titled "+ Add New Product". It contains the following fields:

- Product Code \***: An input field with placeholder text "Enter product code (e.g., P001)".
- Product Name \***: An input field with placeholder text "Enter product name".
- Price (\$)**: An input field with placeholder text "0.00".
- Quantity \***: An input field with placeholder text "0".
- Category \***: A dropdown menu with placeholder text "Select category".
- Description**: A text area with placeholder text "Enter product description (optional)".

**Screenshot 2: Product Management System**

This screenshot shows a dashboard titled "Product Management System" featuring a product list and a success message.

**Success Message:** "Product updated successfully!"

**Add New Product Button:** A blue button with a plus sign and "Add New Product" text.

**Search Bar:** A search bar with placeholder text "Search products..." and a "Search" button.

**Product Table:** A table listing four products:

ID	Code	Name	Price	Quantity	Category	Actions
1	P001	Laptop Dell XPS 13	\$1299.99	10	Electronics	<button>Edit</button> <button>Delete</button>
2	P002	iPhone 15 Pro	\$999.99	25	Electronics	<button>Edit</button> <button>Delete</button>
3	P003	Office Chair	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
4	P004	soap	\$12.50	20	Other	<button>Edit</button> <button>Delete</button>

Browser sends GET /products/new

@GetMapping("/new") in ProductController handles this with showNewForm(Model model)

Controller creates new Product() and adds it to the model as "product"

Returns "product-form", so product-form.html is rendered; the form fields are bound with th:object="\${product}" and th:field="\*{...}"

When the user submits, the form posts to /products/save (handled by @PostMapping("/save") to actually insert/update)

Edit product (ID=1): <http://127.0.0.1:8081/products/edit/1>

Product Code \*  
P001

Product Name \*  
Laptop Dell XPS 13

Price (\$) \*  
1299.99

Quantity \*  
10

Category \*  
Electronics

Description  
High-performance laptop

Product updated successfully!

ID	Code	Name	Price	Quantity	Category	Actions
1	P001	Laptop Dell XPS 13	\$1599.99	10	Electronics	<button>Edit</button> <button>Delete</button>
2	P002	iPhone 15 Pro	\$999.99	25	Electronics	<button>Edit</button> <button>Delete</button>
3	P003	Office Chair	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
4	P004	soap	\$12.50	20	Other	<button>Edit</button> <button>Delete</button>

Browser sends GET /products/edit/1

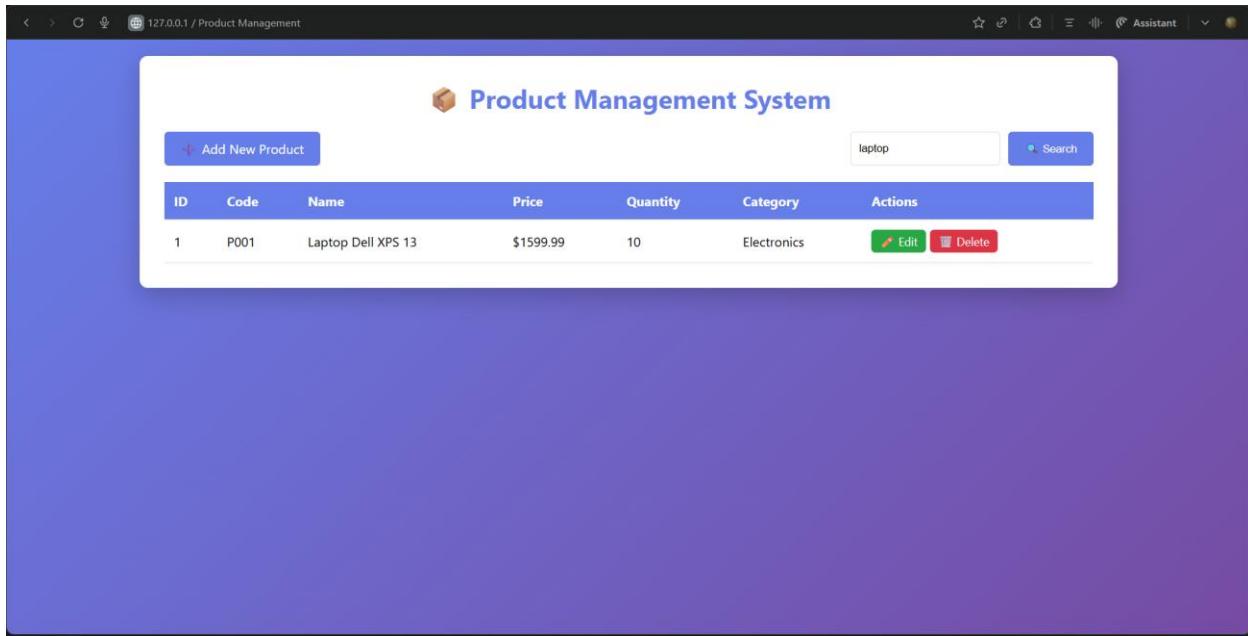
@GetMapping("/edit/{id}") handles this, binding 1 into @PathVariable Long id

Controller calls productService.getProductById(id), which calls productRepository.findById(id) and returns an Optional<Product>

If product exists:

- Put that Product into the model as "product" and return "product-form"
- The form shows existing values

**Search products: http:// 127.0.0.1:8081/products/search?keyword=laptop**



Browser sends GET /products/search?keyword=laptop, usually from the search form on the list page.

@GetMapping("/search") handles this, binding keyword from the query string via @RequestParam("keyword") String keyword.

Controller calls productService.searchProducts(keyword), which calls productRepository.findByNameContaining(keyword) (a Spring Data JPA derived query) to get matching products

Add the resulting List<Product> to model as "products" and the keyword back to the model to refill the search box

Return "product-list" so the same list page is reused to display only matching products

Delete product (ID=1): <http://127.0.0.1:8081/products/delete/1>

The screenshot shows a JavaFX application window titled "Product Management System". A modal dialog box is centered on the screen, displaying the message "127.0.0.1:8081 says" and "Are you sure you want to delete this product?". Below the dialog, the main content area shows a table of products. The table has columns: ID, Code, Name, Price, Quantity, Category, and Actions. There are three rows of data:

ID	Code	Name	Price	Quantity	Category	Actions
2	P002	iPhone 15 Pro	\$999.99	25	Electronics	<button>Edit</button> <button>Delete</button>
3	P003	Office Chair	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
4	P004	soap	\$12.50	20	Other	<button>Edit</button> <button>Delete</button>

A green flash message at the top of the table area says "Product deleted successfully!".

Browser sends GET /products/delete/1

@GetMapping("/delete/{id}") handles this, binding 1 into @PathVariable Long id

Controller calls productService.deleteProduct(id), which calls  
productRepository.deleteById(id) to remove the row from the products table

On success, add a flash "message" like "Product deleted successfully!"; on error, add an "error" message

Return "redirect:/products" so the browser is redirected to the list page and sees the updated list plus the flash message