

## LAB 7

### FEATURE CODE FLOWS

#### 1. LIST PRODUCTS FEATURE

##### Flow:

1. User accesses <http://localhost:8080/products>
2. ProductController.listProducts() method invoked (@GetMapping)
3. Calls productService.getAllProducts()
4. ProductService calls productRepository.findAll()
5. Spring Data JPA executes: SELECT \* FROM products
6. Results added to Model: model.addAttribute("products", products)
7. Returns "product-list" view name
8. Thymeleaf renders product-list.html with data
9. Table displays all products using th:each loop

##### Code Snippet (Controller):

```
@GetMapping  
public String listProducts(Model model) {  
    List<Product> products = productService.getAllProducts();  
    model.addAttribute("products", products);  
    return "product-list";  
}
```

##### View Rendering (Thymeleaf):

```
<tr th:each="product : ${products}">  
    <td th:text="${product.id}"></td>  
    <td th:text="${product.name}"></td>  
    <td th:text="${product.price}"></td>  
</tr>
```

Product Management System						
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	Samsung Galaxy S24	\$899.99	20	Electronics	<button>Edit</button> <button>Delete</button>
4	P004	Office Chair Ergonomic	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
5	P005	Standing Desk	\$399.99	15	Furniture	<button>Edit</button> <button>Delete</button>

## 2. CREATE PRODUCT FEATURE

### Flow:

1. User clicks “Add New Product” button
2. GET /products/new - Shows empty form - ProductController.showNewForm() creates new Product object - Returns product-form.html view
3. User fills form and clicks Submit
4. POST /products/save - Processes form data - @ModelAttribute binds form data to Product object - Calls productService.saveProduct(product) - Repository executes INSERT statement - RedirectAttributes adds success message - Redirects to /products (PRG pattern)
5. Success message displayed on list page

### Code Snippet (Controller):

```

@GetMapping("/new")
public String showNewForm(Model model) {
    Product product = new Product();
    model.addAttribute("product", product);
    return "product-form";
}

@PostMapping("/save")
public String saveProduct(@ModelAttribute("product") Product product,
    RedirectAttributes redirectAttributes) {
    productService.saveProduct(product);
    redirectAttributes.addFlashAttribute("message", "Product added successfully!");
    return "redirect:/products";
}

```

## Form Binding (Thymeleaf):

```
<form th:action="@{/products/save}" th:object="${product}" method="post">
    <input type="text" th:field="*{name}" />
    <input type="number" th:field="*{price}" />
</form>
```

### Add New Product

Product Code \*

Product Name \*  Please fill out this field.

Price (\$) \*

Quantity \*

Category \*

Description

 Save Product  Cancel

The screenshot shows a web application titled "Product Management System". At the top, there is a green success message box containing the text "Product updated successfully!". Below the message is a navigation bar with a "Add New Product" button, a search input field with placeholder text "Search products...", and a "Search" button. The main content area displays a table of products. The table has columns: ID, Code, Name, Price, Quantity, Category, and Actions. The "Actions" column contains two buttons for each row: a green "Edit" button and a red "Delete" button. The data in the table is as follows:

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	Samsung Galaxy S24	\$899.99	20	Electronics	<button>Edit</button> <button>Delete</button>
4	P004	Office Chair Ergonomic	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
5	P005	Standing Desk	\$399.99	15	Furniture	<button>Edit</button> <button>Delete</button>
7	P006	Ghe Cong Thai Hoc	\$6996.00	1	Furniture	<button>Edit</button> <button>Delete</button>

### 3. UPDATE PRODUCT FEATURE

#### Flow:

1. User clicks Edit button on product row
2. GET /products/edit/{id} - Shows pre-filled form - @PathVariable extracts id from URL - productService.getProductById(id) returns Optional - If found: adds product to model, returns form view - If not found: redirects with error message
3. Form displays existing product data (th:field auto-populates)
4. User modifies data and submits
5. POST /products/save - Same endpoint as create - If product.id exists: Spring Data JPA performs UPDATE - If product.id is null: Spring Data JPA performs INSERT
6. Redirect to list with success message

#### Code Snippet (Controller):

```

@GetMapping("/edit/{id}")
public String showEditForm(@PathVariable Long id, Model model,
    RedirectAttributes redirectAttributes) {
    return productService.getProductById(id)
        .map(product -> {
            model.addAttribute("product", product);
            return "product-form";
        })
        .orElseGet(() -> {
            redirectAttributes.addFlashAttribute("error", "Product not found");
        });
}

```

```
        return "redirect:/products";
    });
}
```

**Key Concept:** Spring Data JPA's save() method:  
- Checks if entity ID exists  
- If ID exists: UPDATE query  
- If ID is null: INSERT query  
- Single method handles both create and update

The screenshot shows a modal dialog titled "Edit Product". The form contains the following fields:

- Product Code \*: P006
- Product Name \*: Ghe Cong Thai Hoc
- Price (\$) \*: 6996.00
- Quantity \*: 1
- Category \*: Furniture
- Description: Sieu Thai Cong

At the bottom of the form are two buttons: "Save Product" and "Cancel".

The screenshot shows a web application titled "Product Management System". At the top, there is a green banner with the text "Product updated successfully!". Below the banner is a header bar with a "Add New Product" button, a search input field, and a "Search" button. The main content area is a table with the following data:

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	Samsung Galaxy S24	\$899.99	20	Electronics	<button>Edit</button> <button>Delete</button>
4	P004	Office Chair Ergonomic	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
5	P005	Standing Desk	\$399.99	15	Furniture	<button>Edit</button> <button>Delete</button>
7	P006	Ghe Cong Thai Hoc	\$6996.00	1	Furniture	<button>Edit</button> <button>Delete</button>

## 4. DELETE PRODUCT FEATURE

### Flow:

1. User clicks Delete button (with confirmation dialog)
2. JavaScript confirms: “Are you sure?”
3. GET /products/delete/{id} - @PathVariable Long id extracts product ID - Calls productService.deleteProduct(id) - Service calls productRepository.deleteById(id) - Spring Data JPA executes: DELETE FROM products WHERE id = ?
4. Try-catch handles errors gracefully
5. Flash message confirms deletion
6. Redirects back to product list

### Code Snippet (Controller):

```
@GetMapping("/delete/{id}")
public String deleteProduct(@PathVariable Long id,
                           RedirectAttributes redirectAttributes) {
    try {
        productService.deleteProduct(id);
        redirectAttributes.addFlashAttribute("message", "Product deleted successfully!");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("error", "Error deleting product");
    }
    return "redirect:/products";
}
```

### Client-Side Confirmation (HTML):

```
<a th:href="@{/products/delete/{id}(id=${product.id})}"  
    onclick="return confirm('Are you sure?')">Delete</a>
```

A screenshot of a web-based product management system. At the top, there is a navigation bar with a logo and some text. Below the navigation bar, a success message 'Product updated successfully!' is displayed in a green bar. In the center, there is a table listing products with columns for ID, Code, Name, Price, Quantity, Category, and Actions. The table contains six rows of data. Overlaid on the page is a dark modal dialog box with the title 'localhost:8089 says' and the question 'Are you sure you want to delete this product?'. It has two buttons: 'OK' and 'Cancel'. The background of the page is slightly blurred.

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	Samsung Galaxy S24	\$899.99	20	Electronics	<button>Edit</button> <button>Delete</button>
4	P004	Office Chair Ergonomic	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
5	P005	Standing Desk	\$399.99	15	Furniture	<button>Edit</button> <button>Delete</button>
7	P006	Ghe Cong Thai Hoc	\$6996.00	1	Furniture	<button>Edit</button> <button>Delete</button>

A screenshot of the same web-based product management system. The interface is identical to the first screenshot, including the navigation bar, success message, table, and modal dialog. However, the modal dialog now displays the message 'Product deleted successfully!' instead of asking for confirmation. The background of the page is slightly blurred.

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	Samsung Galaxy S24	\$899.99	20	Electronics	<button>Edit</button> <button>Delete</button>
4	P004	Office Chair Ergonomic	\$199.99	50	Furniture	<button>Edit</button> <button>Delete</button>
5	P005	Standing Desk	\$399.99	15	Furniture	<button>Edit</button> <button>Delete</button>