

**International University**

School of Computer Science and Engineering

# **Web Application Development**

## **Laboratory**

### **IT093IU**

### **Lab #7**

**Submitted by**

**Nguyễn Hồng Ngọc Hân - ITCSIU22229**

## EXERCISE 5: ADVANCED SEARCH

### Task 5.1: Multi-Criteria Search

### Task 5.2: Category Filter

### Task 5.3: Search with Pagination

The screenshot shows a web browser window titled "Product Management" at the URL "localhost:8080/products". The main content area is titled "Product Management System" with a logo. At the top left is a blue button "+ Add New Product". To its right is a search bar with placeholder "Search products..." and a blue "Search" button. Below these are two input fields: "Product Name" (with placeholder "Contains...") and "Category" (set to "All Categories"). To the right of these are "Min Price" (0.00) and "Max Price" (0.00) input fields. A blue "Apply Filters" button is to the right of the price fields, and a grey "Reset" button is below it. Below the filter section is a table with the following data:

ID	Code	Name	Price	Quantity	Category	Actions
2	P002	iPhone 15 Pro	\$999,99	25	Electronics	<span>Edit</span> <span>Delete</span>
3	P003	Office Chair	\$199,99	50	Furniture	<span>Edit</span> <span>Delete</span>
4	P004	Ipad Air 3	\$450,00	25	Electronics	<span>Edit</span> <span>Delete</span>

The screenshot shows a web browser window titled "Product Management" at the URL "localhost:8080/products?sortBy=&sortDir=asc&name=&category=Electronics&minPrice=&maxPrice=200". The main content area is titled "Product Management System" with a logo. At the top left is a blue button "+ Add New Product". To its right is a search bar with placeholder "Search products..." and a blue "Search" button. Below these are two input fields: "Product Name" (with placeholder "Contains...") and "Category" (set to "Electronics"). To the right of these are "Min Price" (0.00) and "Max Price" (200) input fields. A blue "Apply Filters" button is to the right of the price fields, and a grey "Reset" button is below it. Below the filter section is a table with the following data:

ID	Code	Name	Price	Quantity	Category	Actions
5	P005	Ipad Air 3	\$105,50	32	Electronics	<span>Edit</span> <span>Delete</span>

1. Browser sends the GET request to /products.
2. Controller Receives Request: ProductController.listProducts method is invoked. It extracts parameters and constructs the Sort object (Sort.by("price"). ascending()).
3. Controller Calls Service (Search & Sort): Controller calls ProductService.searchProductsAdvanced(name, category, minPrice, maxPrice, sort).
4. Service Builds Query: ProductServiceImpl.searchProductsAdvanced uses the Specification interface to build dynamic Predicate objects for filtering (e.g., cb.like(root.get("name"), "%phone%"), cb.equal(root.get("category"), "Electronics")).
5. Service Calls Repository: Service calls productRepository.findAll(Specification, Sort).
6. JPA Executes Query: Spring Data JPA executes a single query combining the WHERE clause (from Specification) and the ORDER BY clause (from Sort).
7. Controller Adds Data: Controller receives the filtered and sorted List<Product>. It adds the results (products) and all search/sort parameters to the Model.
8. Returns view name "product-list".
9. Thymeleaf Renders HTML: Thymeleaf renders the page, showing the results and maintaining form/sorting state.
10. Response sent to browser.

## EXERCISE 6: VALIDATION

### Task 6.1: Add Validation Annotations

### Task 6.2: Add Validation in Controller

### Task 6.3: Display Validation Errors

The screenshot shows a web browser window titled "Product Form" with the URL "localhost:8080/products/save". The main content is a modal dialog titled "Edit Product". The form contains the following fields:

- Product Code \***: A002 (highlighted with a red border)
- Product Name \***: iPhone 15 Pro
- Price (\$) \***: 999.99
- Quantity \***: 25
- Category \***: Electronics
- Description**: Latest iPhone model

At the bottom of the modal are two buttons: "Save Product" and "Cancel". The browser's address bar and various icons are visible at the top and bottom of the screen.

1. Browser Sends Request: Browser sends a POST request with form data to /products/save.

2. Controller Receives & Validates: ProductController.saveProduct method is invoked. Spring attempts to bind form data to the @Valid Product product object.
3. Bean Validation Check: Spring triggers the validation constraints on Product.java (@NotBlank, @DecimalMin, etc.). Errors are recorded in BindingResult.
4. Controller Handles Error: If result.hasErrors() is TRUE. Controller returns the view "product-form".
5. Thymeleaf Renders Errors: product-form.html displays the input data along with the specific error messages using th:errors.
6. Controller Handles Success: If result.hasErrors() is FALSE. Controller calls productService.saveProduct(product).
7. Service & Repository Persist: ProductServiceImpl calls productRepository.save(product), persisting the data to the database.
8. Controller Redirects: Controller adds a success message to RedirectAttributes and issues a redirect to /products.

## EXERCISE 7: SORTING & FILTERING

### Task 7.1: Add Sorting

### Task 7.2: Filter by Category

### Task 7.3: Combined Sorting and Filtering

The screenshot shows a web browser window titled "Product Management System" with a purple header. The main content area displays a table of products. At the top left is a button "+ Add New Product". To the right is a search bar with placeholder "Search products..." and a "Search" button. Below the search bar is an "Advanced Filter" section with fields for "Product Name" (with dropdown "Contains..."), "Category" (dropdown "All Categories"), "Min Price" (text input "0.00"), "Max Price" (text input "0.00"), "Apply Filters" (button), and "Reset" (button). The table has columns: ID, Code, Name, Price ↓, Quantity, Category, and Actions. The data in the table is:

ID	Code	Name	Price ↓	Quantity	Category	Actions
2	P002	iPhone 15 Pro	\$999,99	25	Electronics	<span>Edit</span> <span>Delete</span>
4	P004	Ipad Air 3	\$450,00	25	Electronics	<span>Edit</span> <span>Delete</span>
3	P003	Office Chair	\$199,99	50	Furniture	<span>Edit</span> <span>Delete</span>
6	P006	Ipad 6	\$155,90	24	Electronics	<span>Edit</span> <span>Delete</span>
5	P005	Ipad Air 3	\$105,50	32	Electronics	<span>Edit</span> <span>Delete</span>

The screenshot shows a web-based Product Management System. At the top, there's a navigation bar with tabs for 'Product Management' and 'Dashboard'. Below the navigation is a search bar with placeholder 'Search products...' and a 'Search' button. To the left of the main content area is a sidebar with a blue header containing a product icon and the text 'Product Management System'. On the right side of the sidebar is a 'Relaunch to update' button. The main content area features a title 'Advanced Filter' with input fields for 'Product Name' (with a dropdown placeholder 'Contains...'), 'Category' (set to 'All Categories'), 'Min Price' (0.00), 'Max Price' (0.00), and buttons for 'Apply Filters' and 'Reset'. Below the filter section is a table with the following data:

ID	Code	Name	Price	Quantity	Category	Actions
5	P005	Ipad Air 3	\$105,50	32	Electronics	<a href="#">Edit</a> <a href="#">Delete</a>
6	P006	Ipad 6	\$155,90	24	Electronics	<a href="#">Edit</a> <a href="#">Delete</a>
3	P003	Office Chair	\$199,99	50	Furniture	<a href="#">Edit</a> <a href="#">Delete</a>
4	P004	Ipad Air 3	\$450,00	25	Electronics	<a href="#">Edit</a> <a href="#">Delete</a>
2	P002	iPhone 15 Pro	\$999,99	25	Electronics	<a href="#">Edit</a> <a href="#">Delete</a>

### Explain:

The sorting mechanism in the product-management system is implemented entirely via server-side rendering, ensuring robust and secure sorting logic. This functionality is embedded directly within the anchor tags () inside the table headers ( ) in the product-list.html file using Thymeleaf's URL expression syntax (@{...}). When a user clicks a column header (e.g., Name or Price), the system triggers a new HTTP GET request to /products, carrying the crucial parameters: sortBy (the field to sort) and sortDir (the sorting direction). The core logic involves state toggling: it uses a ternary operation to check the current direction and automatically reverse it (from 'asc' to 'desc' and vice versa) for the new URL, allowing users to cycle the sort order with repeated clicks. Furthermore, this mechanism implements combined sorting and filtering by ensuring that all existing filter parameters (name, category, minPrice, maxPrice) are maintained in the new URL, guaranteeing that the re-sorted list remains accurately filtered. |

## EXERCISE 8: STATISTICS DASHBOARD

### Task 8.1: Add Statistics Methods

### Task 8.2: Create Dashboard Controller

### Task 8.3: Create Dashboard View

The screenshot shows a web browser window titled "Executive Dashboard" at the URL "localhost:8080/dashboard". The dashboard features several cards and tables:

- Total Products:** 3
- Total Inventory Value:** \$46249,25
- Avg. Price:** \$549,99
- Low Stock Items:** 0
- Inventory by Category:** Electronics: 2, Furniture: 1
- Low Stock Alerts (< 10):** No items listed.
- Recently Added Products:**

ID	Name	Price	Category
4	Ipad Air 3	\$450.00	Electronics
3	Office Chair	\$199.99	Furniture

1. Browser sends a GET request to /dashboard.
2. Controller Receives Request DashboardController.showDashboard method is invoked.
3. Controller Calls Service (Stats) : Controller makes multiple calls to the service to fetch metrics: getTotalCount(), getTotalInventoryValue(), getAveragePrice(), getAllCategories(), getLowStockProducts(10), and getRecentProducts().
4. Service Calls Repository (Queries): ProductServiceImpl calls the corresponding specialized query methods in ProductRepository (e.g., calculateTotalValue(), findLowStockProducts(), findTop5ByOrderByIdDesc()). It also loops through all categories to call countByCategory(cat) and build the categoryStats Map.
5. Controller Adds Data: Controller receives the computed metrics and the categoryStats Map, and adds all of them to the Model.
6. Returns view name "dashboard".
7. Thymeleaf Renders HTML: Thymeleaf renders the page, displaying the metrics in stat cards and data tables.
8. Response sent to browser.