E-Commerce Product Catalogue

```
package com.example.demo;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication

public class EcommerceCatalogApplication {


	public static void main(String[] args) {

		SpringApplication.run(EcommerceCatalogApplication.class, args);

	}

}


package com.example.demo.controller;


import java.util.List;


import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestMapping;


import com.example.demo.entity.Product;

import com.example.demo.service.ProductService;
import jakarta.validation.Valid;

@Controller
@RequestMapping("/products")
public class ProductController {

    private final ProductService productService;

    public ProductController(ProductService productService) {
        this.productService = productService;
    }

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

    @GetMapping("/{id}")
    public String getProductById(@PathVariable Long id, Model model) {
        Product product = productService.getProductById(id);
        model.addAttribute("product", product);
        return "product-detail";
    }

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

    @PostMapping
    public String addProduct(@Valid @ModelAttribute Product product, BindingResult result) {
        if (result.hasErrors()) {
            return "add-product";
        }
        productService.addProduct(product);
        return "redirect:/products";
    }

    @GetMapping("/{id}/edit")
    public String showUpdateForm(@PathVariable Long id, Model model) {
        Product product = productService.getProductById(id);
        model.addAttribute("product", product);
        return "update-product";
    }

    @PutMapping("/{id}")
    public String updateProduct(@PathVariable Long id, @Valid @ModelAttribute Product product,
BindingResult result) {
        if (result.hasErrors()) {
            return "update-product";
        }
        productService.updateProduct(id, product);
        return "redirect:/products";
    }

    @DeleteMapping("/{id}")
    public String deleteProduct(@PathVariable Long id) {
        productService.deleteProduct(id);
        return "redirect:/products";
    }

//    @ExceptionHandler(Exception.class)
//    public String handleError(Exception e, Model model) {
//        model.addAttribute("errorMessage", e.getMessage());
//        return "errorPage"; // Return a custom error page
//    }
}

package com.example.demo.entity;

@jakarta.persistence.Entity
public class Product {

    @jakarta.persistence.Id
    @jakarta.persistence.GeneratedValue(strategy = jakarta.persistence.GenerationType.IDENTITY)
    private Long id;

    @jakarta.validation.constraints.NotBlank(message = "Product name is required")
    private String name;

    private String description;

    @jakarta.validation.constraints.Positive(message = "Price must be greater than 0")
    private double price;

    @jakarta.validation.constraints.Positive(message = "Stock must be greater than 0")
    private int stockQuantity;

    private String category;
```

```
// Default Constructor (required by JPA)
public Product() {
}

// Getters
public Long getId() {
   return id;
}

public String getName() {
   return name;
}

public String getDescription() {
   return description;
}

public double getPrice() {
   return price;
}

public int getStockQuantity() {
   return stockQuantity;
}

public String getCategory() {
   return category;
}

// Setters
```

```
    public void setId(Long id) {
        this.id = id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void setStockQuantity(int stockQuantity) {
        this.stockQuantity = stockQuantity;
    }

    public void setCategory(String category) {
        this.category = category;
    }
}

package com.example.demo.repository;

import com.example.demo.entity.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;
```

```java
public interface ProductRepository extends JpaRepository<Product, Long> {
    List<Product> findByNameContaining(String name);
    List<Product> findByCategory(String category);
}

package com.example.demo.service;

import com.example.demo.entity.Product;
import com.example.demo.repository.ProductRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ProductService {

    private final ProductRepository productRepository;

    public ProductService(ProductRepository productRepository) {
        this.productRepository = productRepository;
    }

    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    public Product getProductById(Long id) {
        return productRepository.findById(id).orElseThrow(() -> new RuntimeException("Product not found"));
    }
    public Product addProduct(Product product) {
        return productRepository.save(product);
    }

    public Product updateProduct(Long id, Product productDetails) {
        Product product = getProductById(id);
        product.setName(productDetails.getName());
        product.setDescription(productDetails.getDescription());
        product.setPrice(productDetails.getPrice());
        product.setStockQuantity(productDetails.getStockQuantity());
        product.setCategory(productDetails.getCategory());
        return productRepository.save(product);
    }

    public void deleteProduct(Long id) {
        productRepository.deleteById(id);
    }

    public List<Product> searchProductsByName(String name) {
        return productRepository.findByNameContaining(name);
    }

    public List<Product> searchProductsByCategory(String category) {
        return productRepository.findByCategory(category);
    }
}
```

```
spring.application.name=ecommerce-catalog
```

```
# H2 Database settings (useful for testing)

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true


logging.level.org.springframework=DEBUG

logging.level.root=DEBUG

logging.file.name=logs/spring-boot-app.log

logging.level.org.apache=DEBUG


server.error.whitelabel.enabled=false

server.error.include-message=always
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>


    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>3.3.4</version>

        <relativePath/>

    </parent>
```

```xml
<groupId>com.example</groupId>

<artifactId>ecommerce-catalog</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ecommerce-catalog</name>

<description>Demo project for Spring Boot</description>


<properties>

   <java.version>17</java.version>

</properties>


<dependencies>

   <!-- Spring Boot Data JPA Dependency -->

   <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter-data-jpa</artifactId>

   </dependency>


   <!-- Spring Boot Web Starter -->

   <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter-web</artifactId>

   </dependency>


   <!-- Spring Boot Thymeleaf Starter -->

   <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter-thymeleaf</artifactId>

   </dependency>


   <!-- Spring Boot Validation Starter -->

   <dependency>
```

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

        <!-- H2 Database (in-memory database) -->
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>

        <!-- Spring Boot Test Starter -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- Example Dependency -->
        <dependency>
            <groupId>com.example</groupId>
            <artifactId>demo</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    </dependencies>

    <build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <mainClass>com.example.demo.EcommerceCatalogApplication</mainClass>
            </configuration>
        </plugin>
    </plugins>
</build>

    <!-- Maven Profiles for different environments -->
    <profiles>
        <profile>
            <id>dev</id>
            <properties>
                <spring.profiles.active>dev</spring.profiles.active>
            </properties>
        </profile>
    </profiles>
    <url>http://localhost:8080</url>
</project>

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Product Form</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <h1 th:text="${product != null ? 'Edit Product' : 'Add Product'}">Add Product</h1>
        <form th:action="@{/products}" th:object="${product}" method="post" class="form-group">
            <input type="hidden" th:if="${product != null}" th:field="*{id}">
            <div class="mb-3">
                <label for="name">Product Name</label>
                <input type="text" th:field="*{name}" class="form-control" required>
            </div>
            <div class="mb-3">
                <label for="description">Description</label>
                <textarea th:field="*{description}" class="form-control" required></textarea>
            </div>
            <div class="mb-3">
                <label for="price">Price</label>
                <input type="number" th:field="*{price}" class="form-control" required>
            </div>
            <div class="mb-3">
                <label for="stockQuantity">Stock Quantity</label>
                <input type="number" th:field="*{stockQuantity}" class="form-control" required>
            </div>
            <div class="mb-3">
                <label for="category">Category</label>
                <input type="text" th:field="*{category}" class="form-control" required>
            </div>
            <div class="mb-3">
                <button type="submit" class="btn btn-success">Save</button>
            </div>
        </form>
    </div>
</body>
</html>


<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
```

```
<head>
    <title>Product Catalog</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <h1>Product Catalog</h1>
        <a th:href="@{/products/add}" class="btn btn-primary mb-2">Add New Product</a>
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Description</th>
                    <th>Price</th>
                    <th>Stock</th>
                    <th>Category</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="product : ${products}">
                    <td th:text="${product.id}">1</td>
                    <td th:text="${product.name}">Product Name</td>
                    <td th:text="${product.description}">Description</td>
                    <td th:text="${product.price}">$0.00</td>
                    <td th:text="${product.stockQuantity}">0</td>
                    <td th:text="${product.category}">Category</td>
                    <td>
                        <a th:href="@{/products/edit/{id}(id=${product.id})}" class="btn btn-
warning">Edit</a>
```

```
                        <a th:href="@{/products/delete/{id}(id=${product.id})}" class="btn btn-danger"
onclick="return confirm('Are you sure?')">Delete</a>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```