Ecommerce Project – Phase 3

Sachin Davis Mundassery

Source Code:

1. Phase3EcommerceProjectApplication.java

```
package com.ecommerce;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.transaction.annotation.EnableTransactionManagement;


@SpringBootApplication

@EnableTransactionManagement

public class Phase3EcommerceProjectApplication {


        public static void main(String[] args) {

                SpringApplication.run(Phase3EcommerceProjectApplication.class, args);




        }


}
```

CONTROLLER FILES

2. **AdminController.java**

```
package com.ecommerce.controller;


import java.util.List;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.ecommerce.models.Admin;
import com.ecommerce.service.AdminService;

@RestController
public class AdminController {

	@Autowired
	private AdminService service;

	@GetMapping("/admins")
	public List<Admin> findAllAdmins()
	{
		return service.getAdmins();
	}

	@PutMapping("/updateAdmin")
	public Admin updateAdmin(@RequestBody Admin admin)
	{
		return service.updateAdmin(admin);
	}
}
```

3. ProductController.java

```
package com.ecommerce.controller;

import java.util.List;
import java.util.Optional;
import org.springframework.data.domain.Sort;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
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.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import com.ecommerce.models.Products;
import com.ecommerce.repository.ProductRepository;
import com.ecommerce.service.ProductService;

@RestController
@RequestMapping(path = "/ecommerce")
public class ProductController {

    @Autowired
    private ProductRepository productRepository;

    @Autowired
    private ProductService productService;

    @PostMapping(path = "/addProduct")
    public @ResponseBody String addProduct(@RequestBody Products product) {
        productRepository.save(product);
        return "Data saved successfully";
    }

    @GetMapping("getProductById/{productId}")
    public Optional<Products> getbookId(@PathVariable int productId) {
        return productRepository.findById(productId);
    }

    @GetMapping("getAllProducts")
    public List<Products> getAllProducts() {
        List<Products> product = (List<Products>) productRepository.findAll();
        return product;
    }

    @GetMapping("getProductByCategory/{category}")
    public List<Products> getProductByCategory(@PathVariable String category) {
        List<Products> product = (List<Products>)
productService.findAllProductsByCategory(category);
        return (List<Products>) product;
    }

    @GetMapping("getProductBySeason/{season}")
    public List<Products> getProductBySeason(@PathVariable String season) {
        List<Products> product = (List<Products>)
productService.findAllProductsBySeason(season);
        return (List<Products>) product;
    }

    @GetMapping("getProductByColor/{color}")
    public List<Products> getProductByColor(@PathVariable String color) {
        List<Products> product = (List<Products>)
productService.findAllProductsByColor(color);
        return (List<Products>) product;
    }

    @GetMapping("getProductByBrand/{brand}")
    public List<Products> getProductByBrand(@PathVariable String brand) {
        List<Products> product = (List<Products>)
productService.findAllProductsByBrand(brand);
        return (List<Products>) product;
    }

    @GetMapping("getProductByPrice/{priceOrder}")
    public List<Products> getProductByPrice(@PathVariable String priceOrder) {
        List<Products> product;
        if (priceOrder.equals("asc")) {
            product = (List<Products>) productRepository.findAll(Sort.by("price"));
        } else {
            product = (List<Products>)
productRepository.findAll(Sort.by("price").descending());
        }

        return  product;
    }
    @GetMapping("getProductByDate/{dateOrder}")
    public List<Products> getProductByDate(@PathVariable String dateOrder) {
        List<Products> product;
        if (dateOrder.equals("asc")) {
            product = (List<Products>)
productRepository.findAll(Sort.by("createdDate"));
        } else {
            product = (List<Products>)
productRepository.findAll(Sort.by("createdDate").descending());
```

```
        }

        return product;
    }

    @PutMapping("updateProductById/{productId}")
    public Products updateProductById(@PathVariable int productId, @RequestBody Products newProduct) {
        return productService.updateProductById(productId, newProduct);
    }

    @DeleteMapping("deleteProductById/{productId}")
    public String deleteAProductById(@PathVariable int productId) {
        productRepository.deleteById(productId);
        return "Deleted the product with ID : " + productId;
    }

    @DeleteMapping("deleteAllProducts")
    public String deleteAppProducts() {
        productRepository.deleteAll();
        return "Deleted all the products";
    }

    // Search?
}
```

4. ProductOrderController

```
package com.ecommerce.controller;
```

```
import java.util.List;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.ecommerce.models.ProductOrder;
import com.ecommerce.service.ProductOrderMySQLDAO;



@RestController
@RequestMapping(path = "/ecommerce")
public class ProductOrderController {

	@Autowired
	ProductOrderMySQLDAO pDAO;

	@GetMapping(path ="/orders")
	public List<ProductOrder> getPurchaseOrders() {



		List<ProductOrder> results = pDAO.getAllPurchaseOrders();
		System.out.println(results.size()+"--------------------------------");

		return results;
	}
```

```
            @GetMapping("/orderFilter")
   public List<ProductOrder> getOrders( @RequestBody Map<String, String> rMap) throws Exception
{
       if (rMap == null) {
           return (List<ProductOrder>) ResponseEntity
                   .status(HttpStatus.OK)
                   .body(pDAO.getAllPurchaseOrders());
       } else {
           String date = rMap.get("date");
           String category = rMap.get("category");
           return (pDAO.getFilteredOrders(date, category));
       }
   }
}
```

5. UserController

```
package com.ecommerce.controller;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
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.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import com.ecommerce.models.Products;
import com.ecommerce.models.User;
import com.ecommerce.repository.UserRepository;
import com.ecommerce.service.UserService;


@RestController
@RequestMapping(path = "/ecommerce")
public class UserController {

	@Autowired
	private UserRepository userRepository;

//	@Autowired
//	private UserService userService;

	@PostMapping(path = "/addUser")
	public @ResponseBody String addProduct(@RequestBody User user) {
		System.out.println("Inside Add User ------------------------------" + user.getUserId());
		userRepository.save(user);
		return "Data saved successfully";
	}

	@GetMapping("getUserById/{userId}")
	public Optional<User> getbookId(@PathVariable int userId)
	{
		System.out.println("Inside Get product by id ------------------------------" + userId);
```

```
        return userRepository.findById(userId);
    }



    @GetMapping("getAllUsers")
    public List<User> getAllUsers()
    {
        List<User> user=(List<User>) userRepository.findAll();
        return user;
    }

    @DeleteMapping("deleteUserById/{userId}")
    public String deleteUserById(@PathVariable int userId) {
        userRepository.deleteById(userId);
        return "Deleted the user with ID : "+ userId;
    }

    @DeleteMapping("deleteAllUsers")
    public String deleteAllUsers() {
        userRepository.deleteAll();
        return "Deleted all the users";
    }
}
```

MODELS

6. Admin.java

```
package com.ecommerce.models;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="admin")
public class Admin {

	@Id
	@Column(name="adminId")
	@GeneratedValue(strategy = GenerationType.AUTO)
	private String adminId;

	private String password;

	public String getAdminId() {
		return adminId;
	}

	public void setAdminId(String adminId) {
		this.adminId = adminId;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}
}
```

7. ProductOrder.java

```
package com.ecommerce.models;

public class ProductOrder {

        private User user;
        private Products product;
        public User getUser() {
                return user;
        }
        public void setUser(User user) {
                this.user = user;
        }
        public Products getProduct() {
                return product;
        }
        public void setProduct(Products product) {
                this.product = product;
        }

}
```

8. Products.java

```
package com.ecommerce.models;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;


@Entity

@Table(name="product")

public class Products {


        @Id

        @GeneratedValue(strategy = GenerationType.AUTO)

        private int productId;


        private String productName;

        private String season;

        private String brand;

        private String category;

        @Column(name="price")

        private int price;

        private String color;


        @Column(name="created_date")

        private String createdDate;

        private int discount;

        private int quantity;


        public int getProductId() {

                return productId;

        }

        public void setProductId(int productId) {

                this.productId = productId;
}
public String getProductName() {
		return productName;
}
public void setProductName(String productName) {
		this.productName = productName;
}
public String getSeason() {
		return season;
}
public void setSeason(String season) {
		this.season = season;
}
public String getBrand() {
		return brand;
}
public void setBrand(String brand) {
		this.brand = brand;
}
public String getCategory() {
		return category;
}
public void setCategory(String category) {
		this.category = category;
}
public int getPrice() {
		return price;
}
public void setPrice(int price) {
		this.price = price;
}
```

```
	public String getColor() {
		return color;
	}
	public void setColor(String color) {
		this.color = color;
	}
	public String getCreatedDate() {
		return createdDate;
	}
	public void setCreatedDate(String createdDate) {
		this.createdDate = createdDate;
	}
	public int getDiscount() {
		return discount;
	}
	public void setDiscount(int discount) {
		this.discount = discount;
	}
	public int getQuantity() {
		return quantity;
	}
	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}

}
```

9. User.java

```
package com.ecommerce.models;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="user")
public class User {
        @Id
        private int userId;

        private String username;
        private String password;
        private String email;
        private String phone_number;
        public int getUserId() {
                return userId;
        }
        public void setUserId(int userId) {
                this.userId = userId;
        }
        public String getUsername() {
                return username;
        }
        public void setUsername(String username) {
                this.username = username;
        }
        public String getPassword() {
                return password;
        }
        public void setPassword(String password) {
                this.password = password;
        }
```

```
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPhone_number() {
		return phone_number;
	}
	public void setPhone_number(String phone_number) {
		this.phone_number = phone_number;
	}


}
```

10. ProductOrderRepository.java

```
package com.ecommerce.repository;

import org.springframework.data.repository.CrudRepository;
import com.ecommerce.models.ProductOrder;
import com.ecommerce.models.Products;

//public interface ProductOrderRepository extends CrudRepository<ProductOrder, Integer> {
//}
public interface ProductOrderRepository extends CrudRepository<Products, Integer> {
}
```

11. AdminRepository.java

```
package com.ecommerce.repository;
```

```
import org.springframework.data.repository.CrudRepository;

import com.ecommerce.models.Admin;



public interface AdminRepository extends CrudRepository<Admin, String> {



}
```

12. ProductRepository.java

```
package com.ecommerce.repository;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
import com.ecommerce.models.Products;
import org.springframework.data.domain.Sort;

public interface ProductRepository extends CrudRepository<Products, Integer> {
	Iterable<Products> findAll(Sort sort);

	public List<Products> findAllProductsByCategory(String category);

	public List<Products> findAllProductsBySeason(String season);

	public List<Products> findAllProductsByColor(String color);

	public List<Products> findAllProductsByBrand(String brand);
//	public List<Products> findAllProductsByPrice(String priceOrder);
}
```

13. UserRepository.java

```
package com.ecommerce.repository;

import org.springframework.data.repository.CrudRepository;
import com.ecommerce.models.User;

public interface UserRepository extends CrudRepository<User, Integer>{

}
```

14. ProductOrderMySQLDAO

```
package com.ecommerce.service;
import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import com.ecommerce.models.ProductOrder;
import com.ecommerce.models.Products;
import com.ecommerce.models.User;

@Repository
public class ProductOrderMySQLDAO {
	private final JdbcTemplate jdbcTemplate;
	public ProductOrderMySQLDAO(JdbcTemplate jdbcTemplate) {
		this.jdbcTemplate = jdbcTemplate;
	}
```

```
    public List<ProductOrder> getAllPurchaseOrders() {
        return jdbcTemplate.query("SELECT * FROM product_order", (rs, rowNum) -> {

            ProductOrder order = new ProductOrder();
            int productId = rs.getInt(2);
            int userId = rs.getInt(3);

            Products product = jdbcTemplate.queryForObject("SELECT * FROM product
WHERE product_id = ?", (rs1, rowNum1) -> {
                Products productInner = new Products();
                System.out.println(rs1.getInt(1)+"----------------Product ID-------------");
                productInner.setProductId(rs1.getInt(1));
                productInner.setBrand(rs1.getString(2));
                productInner.setCategory(rs1.getString(3));
                productInner.setColor(rs1.getString(4));
                productInner.setCreatedDate(rs1.getString(5));
                productInner.setDiscount(rs1.getInt(6));
                productInner.setPrice(rs1.getInt(7));
                productInner.setProductName(rs1.getString(8));

                productInner.setQuantity(rs1.getInt(9));
                productInner.setSeason(rs1.getString(10));
                return productInner;
            }, productId);
            User user = jdbcTemplate.queryForObject("SELECT * FROM user WHERE
user_id = ?", (rs2, rowNum2) -> {
                User u = new User();
                u.setUserId(rs2.getInt(1));
                u.setEmail(rs2.getString(2));
                u.setPassword(rs2.getString(3));
                u.setPhone_number(rs2.getString(4));
```

```
                                u.setUsername(rs2.getString(5));
                                return u;
                        }, userId);

                        order.setProduct(product);
                        order.setUser(user);
                        return order;

                });

        }

        public List<ProductOrder> getFilteredOrders(String date, String category) {
    List<ProductOrder> orders = getAllPurchaseOrders();

    if (date != null) {
      orders = orders
          .stream()
          .filter(order -> order.getProduct().getCreatedDate().toString().equals(date))
          .toList();
    }
    if (category != null) {
      orders = orders
          .stream()
          .filter(order -> order.getProduct().getCategory().equals(category))
          .toList();
    }
    return orders;
  }
}
```

15. ProductService.java

```
package com.ecommerce.service;

import org.springframework.stereotype.Service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import com.ecommerce.models.Products;
import com.ecommerce.repository.ProductRepository;

@Service
public class ProductService {

	private final ProductRepository productRepository;

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

	public Products updateProductById(int productId, Products newProduct) {
		Products oldProduct = productRepository.findById(productId).get();
		oldProduct.setProductName(newProduct.getProductName());
		oldProduct.setSeason(newProduct.getSeason());
		oldProduct.setBrand(newProduct.getBrand());
		oldProduct.setCategory(newProduct.getCategory());
		oldProduct.setPrice(newProduct.getPrice());
		oldProduct.setColor(newProduct.getColor());
```

```
                oldProduct.setCreatedDate(newProduct.getCreatedDate());
                oldProduct.setDiscount(newProduct.getDiscount());
                oldProduct.setQuantity(newProduct.getQuantity());
                productRepository.save(oldProduct);
                return newProduct;
        }

        public List<Products> findAllProductsByCategory(String category) {
                return productRepository.findAllProductsByCategory(category);
        }

        public List<Products> findAllProductsBySeason(String season) {
                return productRepository.findAllProductsBySeason(season);
        }
        public List<Products> findAllProductsByColor(String color) {
                return productRepository.findAllProductsByColor(color);
        }
        public List<Products> findAllProductsByBrand(String brand) {
                return productRepository.findAllProductsByBrand(brand);
        }

}
```

16. UserService.java

```
package com.ecommerce.service;

public class UserService {

}
```

Application Properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/ecommercedb

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.database=mysql

spring.jpa.hibernate.ddl-auto=update
```

POM.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<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>2.6.2</version>
		<relativePath /> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.ecommerce</groupId>
	<artifactId>Phase3_EcommerceProject</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>Phase3_EcommerceProject</name>
	<description>Ecommerce project for Spring Boot</description>
	<properties>
		<java.version>1.8</java.version>
	</properties>
	<dependencies>
```

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.6</version>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
        <groupId>com.microsoft.sqlserver</groupId>
        <artifactId>mssql-jdbc</artifactId>
        <scope>runtime</scope>
</dependency>

<!-- Reload automatically -->
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
</dependency>

<dependency>
```

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

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```