E-COMMERCE WEBSITE FOR SPORTY SHOES

Write up

SAMEER KHAN
https://github.com/SameerKhan0411/SportyShoes.git

Concepts used in project:

O   **Eclipse** is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.

O   **Java** is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

O   **SQL** is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

O   **Maven** is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

Pom.xml

```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>2.5.4</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.sportyshoes</groupId>
    <artifactId>Sporty_Shoes</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Sporty_Shoes</name>
    <description>Sporty Shoes Prototype Application</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
```

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
```

```xml
                    <scope>test</scope>
            </dependency>
            <dependency>
                    <groupId>io.springfox</groupId>
                    <artifactId>springfox-boot-starter</artifactId>
                    <version>3.0.0</version>
            </dependency>
            <dependency>
                    <groupId>io.springfox</groupId>
                    <artifactId>springfox-swagger-ui</artifactId>
                    <version>3.0.0</version>
            </dependency>
    </dependencies>
    <build>
            <plugins>
                    <plugin>
                            <groupId>org.springframework.boot</groupId>
                            <artifactId>spring-boot-maven-plugin</artifactId>
                            <configuration>
                                    <excludes>
                                            <exclude>
                                                    <groupId>org.projectlombok</groupId>
                                                    <artifactId>lombok</artifactId>
                                            </exclude>
                                    </excludes>
                            </configuration>
                    </plugin>
```

```
        </plugins>

    </build>

</project>
```

Configuration

SprotyShoesSecurityConfiguration

package com.sportyshoes.configuration;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

public class SprotyShoesSecurityConfiguration extends WebSecurityConfigurerAdapter {

        @Override

        protected void configure(HttpSecurity http) throws Exception {

                http.authorizeRequests().antMatchers(HttpMethod.GET, "/admin/**").hasRole("ADMIN")

                .antMatchers("/users/**").permitAll().and().httpBasic();

                http.csrf().disable();

        }

```java
    @Override

    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

        auth.inMemoryAuthentication().withUser("admin").password("{noop}admin").roles("ADMIN");

    }

}
```

SwaggerConfiguration

```java
package com.sportyshoes.configuration;


import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.RequestHandlerSelectors;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

@EnableSwagger2

@Configuration

public class SwaggerConfiguration {

    @Bean

    public Docket api() {

        return new
Docket(DocumentationType.SWAGGER_2).select().apis(RequestHandlerSelectors.any()).build();

    }


}
```

Controller

AdminController

package com.sportyshoes.controller;

import java.text.ParseException;

import java.util.List;

import java.util.Optional;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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.RequestBody;

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

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

import com.sportyshoes.model.Product;

import com.sportyshoes.model.PurchaseReport;

import com.sportyshoes.model.User;

import com.sportyshoes.service.ProductService;

import com.sportyshoes.service.PurchaseReportService;

import com.sportyshoes.service.UserService;

```java
@RestController
@RequestMapping("/admin")
public class AdminController {

        @Autowired
        ProductService productService;

        @Autowired
        UserService userService;

        @Autowired
        private PurchaseReportService purchaseReportService;

        @GetMapping("/products")
        public ResponseEntity<List<Product>> getAllProducts() {
                List<Product> allProducts = productService.getAllProducts();
                if (allProducts.isEmpty()) {
                        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
                }
                ResponseEntity<List<Product>> responseEntity = new
ResponseEntity<List<Product>>(allProducts, HttpStatus.OK);
                return responseEntity;
        }

        @GetMapping("/products/categorize/{category}")
        public ResponseEntity<List<Product>>
getAllProductsBasedOnCategory(@PathVariable("category") String category) {
```

```java
            System.out.println("Category to look for -> " + category);

            List<Product> allProductsBasedOnCategory =
productService.getAllProductBasedOnCatogary(category);

            if (allProductsBasedOnCategory.isEmpty()) {

                    return new ResponseEntity<>(HttpStatus.NO_CONTENT);

            }

            ResponseEntity<List<Product>> responseEntity = new
ResponseEntity<List<Product>>(allProductsBasedOnCategory,

                            HttpStatus.OK);

            return responseEntity;

    }


    @PostMapping("/products")

    public ResponseEntity<Product> addProduct(@RequestBody Product product) {

            Product temp = productService.addProduct(product);

            if (temp == null) {

                    return new ResponseEntity<Product>(HttpStatus.BAD_REQUEST);

            }

            return new ResponseEntity<Product>(temp, HttpStatus.OK);


    }


    @GetMapping("/products/{productId}")

    public ResponseEntity<Product> getProductById(@PathVariable("productId") int id) {

            Optional<Product> product = productService.getProductById(id);

            if (!product.isPresent()) {

                    return new ResponseEntity<Product>(HttpStatus.NO_CONTENT);

            }
```

```java
            return new ResponseEntity<Product>(product.get(), HttpStatus.OK);
    }


    @DeleteMapping("/products/{productId}")
    public ResponseEntity<HttpStatus> deleteById(@PathVariable("productId") int id) {
            productService.deleteProductById(id);
            return new ResponseEntity<>(HttpStatus.OK);
    }


    @GetMapping("/users")
    public ResponseEntity<List<User>> getAllSignedUpUsers() {
            List<User> allSignedUpUsers = userService.allSignedUpUsers();
            if (allSignedUpUsers.isEmpty()) {
                    return new ResponseEntity<List<User>>(HttpStatus.NO_CONTENT);
            }
            return new ResponseEntity<List<User>>(allSignedUpUsers, HttpStatus.OK);


    }


    @GetMapping("/users/{userName}")
    public ResponseEntity<User> getSignedUpUser(@PathVariable String userName) {
            Optional<User> signedUpUser =
userService.getSignedUpUserByName(userName);
            if (!signedUpUser.isPresent()) {
                    return new ResponseEntity<User>(HttpStatus.NOT_FOUND);
            }
```

```java
			return new ResponseEntity<User>(signedUpUser.get(), HttpStatus.OK);

	}


	@GetMapping("/purchasereport")

	public ResponseEntity<List<PurchaseReport>> getPurchaseReport() {

			List<PurchaseReport> purchaseReport =
purchaseReportService.getAllPurchaseReport();

			if (purchaseReport.isEmpty()) {

					return new
ResponseEntity<List<PurchaseReport>>(HttpStatus.NO_CONTENT);

			}

			return new ResponseEntity<List<PurchaseReport>>(purchaseReport,
HttpStatus.OK);


	}


	@GetMapping("/purchasereport/category/{category}")

	public ResponseEntity<List<PurchaseReport>>
getPurchaseReportBasedOnCategory(@PathVariable String category) {

			List<PurchaseReport> purchaseReportBasedOnCategory =
purchaseReportService.getPurchaseReportBasedOnCategory(category);

			if (purchaseReportBasedOnCategory.isEmpty()) {

					return new
ResponseEntity<List<PurchaseReport>>(HttpStatus.NO_CONTENT);

			}

			return new
ResponseEntity<List<PurchaseReport>>(purchaseReportBasedOnCategory, HttpStatus.OK);


	}
```

```java
    @GetMapping("/purchasereport/date/{date}")
    public ResponseEntity<List<PurchaseReport>>
getPurchaseReportBasedOnDate(@PathVariable String date) throws ParseException {

        System.out.println("Date from url is : " + date);

        List<PurchaseReport> purchaseReportBasedOnCategory =
purchaseReportService.getPurchaseReportBasedOnDate(date);

        if (purchaseReportBasedOnCategory.isEmpty()) {

            return new
ResponseEntity<List<PurchaseReport>>(HttpStatus.NO_CONTENT);

        }

        return new
ResponseEntity<List<PurchaseReport>>(purchaseReportBasedOnCategory, HttpStatus.OK);


    }
}
```

UserController

```java
package com.sportyshoes.controller;



import java.security.SecureRandom;

import java.util.Date;

import java.util.Optional;



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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.transaction.annotation.Transactional;

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

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

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

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

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

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


import com.sportyshoes.model.Product;

import com.sportyshoes.model.User;

import com.sportyshoes.service.ProductService;

import com.sportyshoes.service.PurchaseReportService;

import com.sportyshoes.service.UserService;


@RestController
@RequestMapping("/users")
public class UserController {

        @Autowired
        private UserService userService;

        @Autowired
        private ProductService productService;

        @Autowired
        private PurchaseReportService purchaseReportService;

        @PostMapping("/signup")
        public @ResponseBody String register(@RequestBody(required = false) User user) {
```

```java
            if (user == null) {

                        return "Enter Valid User Details - User details should not be Null";

                }else if(user.getUserName() == null || user.getUserPassword()== null ||
user.getUserEmail() == null) {

                        return "Enter Valid User Details - All the fields(Name, Password, Email)
are mandatory";

                }

                int strength = 10;

                BCryptPasswordEncoder bCryptPasswordEncoder = new
BCryptPasswordEncoder(strength, new SecureRandom());

                String encodedPassword =
bCryptPasswordEncoder.encode(user.getUserPassword());

                user.setUserPassword(encodedPassword);

                user.setUserName(user.getUserName().toLowerCase());

                userService.signUp(user);

                return "Signed Up Successfully!";

        }


        @PostMapping("/{userId}/buy/{productName}")

        @Transactional

        public @ResponseBody String buyProductByName(@PathVariable(name = "userId") int
userID,

                        @PathVariable(name = "productName") String productName) {

                Optional<Product> product = productService.getProductByName(productName);

                if (product.isPresent()) {

                        Optional<User> user = userService.getSignedUpUserById(userID);

                        if (user.isPresent()) {

                                User user2 = user.get();

                                user2.addProduct(product.get());
```

```java
                        Product product2 = product.get();

                        product2.addUser(user.get());

                        userService.saveUserWithProduct(user2);

                        productService.addProduct(product2);


        purchaseReportService.savePurchaseReport(product2.getProductName(),
product2.getCategory(),

                                        product2.getProductPrice(), user2.getUserName(),
user2.getUserEmail(), new Date());

                        return "You have successfully bought : " +
product.get().getProductName();

                    } else {

                        return "User Not Found! to buy the Product";

                    }

            }

            return "Product Not Found!";

        }
}
```

Model

Product

package com.sportyshoes.model;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.CascadeType;

```java
import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;


import com.fasterxml.jackson.annotation.JsonIgnoreProperties;


import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;


@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "product")

//Added below line to not get Infinite loop when retriving user and product details

@JsonIgnoreProperties({ "hibernateLazyInitializer", "handler", "users" })

public class Product {


        @Id

        @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```java
        private int productId;

        private String productName;

        private int productPrice;

        private String category;

        @ManyToMany(fetch = FetchType.LAZY, cascade = { CascadeType.PERSIST,
CascadeType.MERGE }, mappedBy = "products")
        private List<User> users = new ArrayList<User>();

        public void addUser(User user) {
                this.users.add(user);
        }

        @Override
        public String toString() {

                return "Custom ToString -> Product";
        }
}
```

PurchaseReport

```java
package com.sportyshoes.model;



import java.util.Date;
```

```java
import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;


import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;


@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class PurchaseReport {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
```

```java
        private String categoryOfProduct;

        private String productName;

        private int priceOfTheProduct;

        private String userWhoBoughtTheProduct;

        private String userEmailBoughtTheProduct;

        @Temporal(TemporalType.DATE)
        private Date dateOfProductPurchase;

        public PurchaseReport(String productName, String categoryOfProduct, int
priceOfTheProduct, String userWhoBoughtTheProduct, String
userEmailBoughtTheProduct, Date dateOfProductPurchase) {
                this.productName = productName;
                this.categoryOfProduct = categoryOfProduct;
                this.userWhoBoughtTheProduct = userWhoBoughtTheProduct;
                this.dateOfProductPurchase = dateOfProductPurchase;
                this.userEmailBoughtTheProduct = userEmailBoughtTheProduct;
                this.priceOfTheProduct = priceOfTheProduct;
        }

}
```

User

package com.sportyshoes.model;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Data

@Entity

```java
@Table(name = "user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int userId;

    @Column(name = "name")
    private String userName;

    @Column(name = "email")
    private String userEmail;

    @Column(name = "password")
    private String userPassword;

    @ManyToMany(fetch = FetchType.LAZY, cascade = { CascadeType.PERSIST,
    CascadeType.MERGE })
    @JoinTable(name = "USER_PRODUCT", joinColumns = @JoinColumn(name =
    "USER_ID"), inverseJoinColumns = @JoinColumn(name = "PRODUCT_ID"))
    private List<Product> products = new ArrayList<Product>();

    public User(String userName, String userEmail) {
        this.userEmail = userEmail;
        this.userName = userName;
    }

    public void addProduct(Product product) {
```

```java
                this.products.add(product);

    }


    @Override

    public String toString() {

            return "Custom ToString -> User [userId=" + userId + ", userName=" + userName
+ ", userEmail=" + userEmail + ", userPassword="

                        + userPassword + ", products=" + products + "]";

    }


}
```

Repository

ProductRepository

package com.sportyshoes.repository;

import java.util.List;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import com.sportyshoes.model.Product;

@Repository

```java
public interface ProductRepository extends JpaRepository<Product, Integer>{


        @Query(value = "select p from Product p where p.category=:category")

        List<Product> findAllByCategory(@Param("category") String category);


        @Query(value = "select p from Product p where p.productName=:name")

        Optional<Product> findByName(String name);
}
```

PurchaseReportRepository

```java
package com.sportyshoes.repository;



import java.util.Date;

import java.util.List;


import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;


import com.sportyshoes.model.PurchaseReport;


@Repository
public interface PurchaseReportRepository extends JpaRepository<PurchaseReport, Integer> {


        @Query("select pr from PurchaseReport pr where pr.categoryOfProduct=:category")

        List<PurchaseReport> findAllByCategory(String category);
```

```java
@Query("select pr from PurchaseReport pr where pr.dateOfProductPurchase=:date")

List<PurchaseReport> findAllByDate(Date date);



}
```

UserRepository

```java
package com.sportyshoes.repository;


import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;


import com.sportyshoes.model.User;


@Repository
public interface UserRepository extends JpaRepository<User, Integer> {


        @Query(value = "select u from User u where u.userName=:name")

        Optional<User> findUserByName(String name);


}
```

Service

ProductService

```java
package com.sportyshoes.service;
```

```java
import java.util.List;

import java.util.Optional;


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

import org.springframework.stereotype.Service;


import com.sportyshoes.model.Product;

import com.sportyshoes.repository.ProductRepository;


@Service
public class ProductService {


    @Autowired
    ProductRepository productRepository;


    public Product addProduct(Product product) {
        return productRepository.save(product);
    }


    public Product addProductWithUser(Product product) {
        return productRepository.save(product);
    }


    public Optional<Product> getProductById(int id) {
        Optional<Product> proOptional = productRepository.findById(id);
        return proOptional;
```

```
        }


        public Optional<Product> getProductByName(String name) {

                Optional<Product> proOptional = productRepository.findByName(name);

                return proOptional;

        }


        public List<Product> getAllProducts() {

                return productRepository.findAll();

        }


        public List<Product> getAllProductBasedOnCatogary(String category) {

                return productRepository.findAllByCategory(category);

        }


        public void deleteProductById(int prdId) {

                productRepository.deleteById(prdId);

        }


}
```

PurchaseReportService

```
package com.sportyshoes.service;



import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;
```

```java
import java.util.List;

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

import com.sportyshoes.model.PurchaseReport;
import com.sportyshoes.repository.PurchaseReportRepository;

@Service
public class PurchaseReportService {

    @Autowired
    private PurchaseReportRepository purchaseReportRepository;

    public void savePurchaseReport(String productName, String category, int productPrice,
    String userName, String userEmail, Date date) {
        PurchaseReport purchaseReport = new PurchaseReport(productName, category,
    productPrice, userName, userEmail, date);
        purchaseReportRepository.save(purchaseReport);
    }

    public List<PurchaseReport> getAllPurchaseReport() {
        List<PurchaseReport> purchaseReports = purchaseReportRepository.findAll();
        return purchaseReports;
    }

    public List<PurchaseReport> getPurchaseReportBasedOnCategory(String category) {
```

```java
            List<PurchaseReport> purchaseReports =
purchaseReportRepository.findAllByCategory(category);

        return purchaseReports;

    }


    public List<PurchaseReport> getPurchaseReportBasedOnDate(String date) throws
ParseException {

            List<PurchaseReport> purchaseReports =
purchaseReportRepository.findAllByDate(new SimpleDateFormat("yyyy-MM-dd").parse(date));

        return purchaseReports;

    }


}
```

UserService

```java
package com.sportyshoes.service;



import java.util.List;

import java.util.Optional;


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

import org.springframework.stereotype.Service;


import com.sportyshoes.model.User;

import com.sportyshoes.repository.UserRepository;


@Service

public class UserService {
```

```java
@Autowired
UserRepository userRepository;

public User signUp(User user) {
        return userRepository.save(user);
}

public User saveUserWithProduct(User user) {
        return userRepository.save(user);
}

public List<User> allSignedUpUsers() {
        return userRepository.findAll();
}

public Optional<User> getSignedUpUserByName(String name) {
        Optional<User> user = userRepository.findUserByName(name);
        return user;
}

public Optional<User> getSignedUpUserById(int id) {
        Optional<User> user = userRepository.findById(id);
        return user;
}
}
```