# **Entities/ Models**

## **Product.java**

**package** com.sportyshoes.model;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.persistence.CascadeType;

**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;

@Entity

@Table(name = "product")

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

**public** **class** Product {

@Id

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

**private** **int** productId;

**private** String productName;

**private** **int** productPrice;

**private** String category;

**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** **int** getProductPrice() {

**return** productPrice;

}

**public** **void** setProductPrice(**int** productPrice) {

**this**.productPrice = productPrice;

}

**public** String getCategory() {

**return** category;

}

**public** **void** setCategory(String category) {

**this**.category = 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";

}

}

## **User.java**

**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.Data;

@Data

@Entity

@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;

**public** User() {}

**public** User(**int** userId, String userName, String userEmail, String userPassword, List<Product> products) {

**this**.userId = userId;

**this**.userName = userName;

**this**.userEmail = userEmail;

**this**.userPassword = userPassword;

**this**.products = products;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getUserEmail() {

**return** userEmail;

}

**public** **void** setUserEmail(String userEmail) {

**this**.userEmail = userEmail;

}

**public** String getUserPassword() {

**return** userPassword;

}

**public** **void** setUserPassword(String userPassword) {

**this**.userPassword = userPassword;

}

**public** **int** getUserId() {

**return** userId;

}

**public** **void** setUserId(**int** userId) {

**this**.userId = userId;

}

**public** List<Product> getProducts() {

**return** products;

}

**public** **void** setProducts(List<Product> products) {

**this**.products = products;

}

@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) {

**this**.products.add(product);

}

@Override

**public** String toString() {

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

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

}

}

## **PurchaseReport.java**

**package** com.sportyshoes.model;

**import** java.util.Date;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Temporal;

**import** javax.persistence.TemporalType;

@Entity

**public** **class** PurchaseReport {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

**private** **int** id;

**private** String categoryOfProduct;

**private** String productName;

**private** **int** priceOfTheProduct;

**private** String userWhoBoughtTheProduct;

**private** String userEmailBoughtTheProduct;

@Temporal(TemporalType.DATE)

**private** Date dateOfProductPurchase;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getCategoryOfProduct() {

**return** categoryOfProduct;

}

**public** **void** setCategoryOfProduct(String categoryOfProduct) {

**this**.categoryOfProduct = categoryOfProduct;

}

**public** String getProductName() {

**return** productName;

}

**public** **void** setProductName(String productName) {

**this**.productName = productName;

}

**public** **int** getPriceOfTheProduct() {

**return** priceOfTheProduct;

}

**public** **void** setPriceOfTheProduct(**int** priceOfTheProduct) {

**this**.priceOfTheProduct = priceOfTheProduct;

}

**public** String getUserWhoBoughtTheProduct() {

**return** userWhoBoughtTheProduct;

}

**public** **void** setUserWhoBoughtTheProduct(String userWhoBoughtTheProduct) {

**this**.userWhoBoughtTheProduct = userWhoBoughtTheProduct;

}

**public** String getUserEmailBoughtTheProduct() {

**return** userEmailBoughtTheProduct;

}

**public** **void** setUserEmailBoughtTheProduct(String userEmailBoughtTheProduct) {

**this**.userEmailBoughtTheProduct = userEmailBoughtTheProduct;

}

**public** Date getDateOfProductPurchase() {

**return** dateOfProductPurchase;

}

**public** **void** setDateOfProductPurchase(Date dateOfProductPurchase) {

**this**.dateOfProductPurchase = dateOfProductPurchase;

}

**protected** PurchaseReport() {}

**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;

}

}

# **Repositories**

## **ProductRepostitory.java**

**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

**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);

}

## **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);

}

## **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);

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

List<PurchaseReport> findAllByDate(Date date);

}

# **Services**

## **ProductService.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.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);

}

}

## **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 {

@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;

}

}

## **PurchaseReportService.java**

**package** com.sportyshoes.service;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**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) {

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;

}

}

# **Controllers**

## **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;

**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) {

**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());

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!";

}

}

## **AdminController.java**

**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;

@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) {

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***);

}

**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***);

}

**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***);

}

@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***);

}

}

# **Configuration**

## **SportyShoesSecurityConfiguration.java**

**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();

}

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

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

}

}

# **Application**

## **SportyShoesApplication.java**

**package** com.sportyshoes;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SportyShoes1Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SportyShoes1Application.**class**, args);

}

}