

SIMPLILEARN

ASSESSMENT -3

SPORTYSHOES.COM


















































In this file you can see the source code of SportyShoes.com which is written in java with STS 4.

[Edwin A](#)
[Source Code](#)

• PROJECT STRUCTURE

SportyShoes

- ▼  src/main/java
 - >  com.sporty.shoes.store
 - >  com.sporty.shoes.store.config
 - >  com.sporty.shoes.store.controller
 - >  com.sporty.shoes.store.domain
 - >  com.sporty.shoes.store.domain.security
 - >  com.sporty.shoes.store.dto
 - >  com.sporty.shoes.store.form
 - >  com.sporty.shoes.store.repository
 - >  com.sporty.shoes.store.service
 - >  com.sporty.shoes.store.service.impl
 - >  com.sporty.shoes.store.type
 - >  utility
- ▼  src/main/resources
 - >  META-INF
 - >  static
 - ▼  templates
 - >  common
 -  addArticle.html
 -  adminHome.html
 -  articleDetail.html
 -  articleList.html
 -  checkout.html
 -  editArticle.html
 -  error.html
 -  index.html
 -  myAccount.html
 -  myAddress.html
 -  myOrders.html
 -  myProfile.html
 -  orderDetails.html
 -  orderSubmitted.html
 -  shoppingCart.html
 -  store.html
 -  userList.html
 -  application.properties
- >  src/test/java
- >  src/test/resources
- >  JRE System Library [JavaSE-1.8]
- >  Maven Dependencies
- >  bin
- >  src
- >  target
-  mvnw
-  mvnw.cmd
-  pom.xml
-  README.md

• STOREAPPLICATION.JAVA

```
package com.sporty.shoes.store;
```

```
import org.springframework.boot.SpringApplication;
```

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

```
import org.springframework.cache.annotation.EnableCaching;
```

```
@SpringBootApplication
```

```
@EnableCaching
```

```
public class StoreApplication {
```

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

```
        SpringApplication.run(StoreApplication.class, args);
```

```
    }
```

```
}
```

• STOREAPPSTARTUPRUNNER.JAVA

```
package com.sporty.shoes.store;
```

```
import java.util.Arrays;
```

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

```
import org.springframework.boot.CommandLineRunner;
```

```
import org.springframework.stereotype.Component;
```

```
import com.sporty.shoes.store.service.UserService;
```

```
@Component
```

```
public class StoreAppStartupRunner implements CommandLineRunner{
```

```
    @Autowired
```

```
    private UserService userService;
```

```
    @Override
```

```
    public void run(String... args) throws Exception {
```

```
        userService.createUser("admin", "admin", "admin@admin.com", Arrays.asList("ROLE_USER",  
"ROLE_ADMIN"));
```

```
    }
```

```
}
```

• SECURITYCONFIG.JAVA

package com.sporty.shoes.store.config;

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

import org.springframework.context.annotation.Configuration;

import

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

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

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

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

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

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

import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

import com.sporty.shoes.store.service.impl.UserSecurityService;

import utility.SecurityUtility;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = **true**)

public class SecurityConfig **extends** WebSecurityConfigurerAdapter {

 @Autowired

private UserSecurityService userSecurityService;

private BCryptPasswordEncoder passwordEncoder() {

return SecurityUtility.passwordEncoder();

 }

private static final String[] **PUBLIC_MATCHERS** = { "/css/**", "/js/**", "/image/**", "/", "/new-user",
"/login",
 "/store", "/article-detail" };

 @Override

protected void configure(HttpSecurity http) **throws** Exception {

 http.authorizeRequests().antMatchers(**PUBLIC_MATCHERS**).permitAll().antMatchers("/article/**").h
asRole("ADMIN")

 .anyRequest().authenticated();

 http.csrf().disable().cors().disable().formLogin().failureUrl("/login?error").loginPage("/login").permitAl
l()

 .and().logout().logoutRequestMatcher(**new**

AntPathRequestMatcher("/logout")).logoutSuccessUrl("/?logout")

 .deleteCookies("remember-

me").permitAll().and().rememberMe().key("aSecretKey");

```

    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userSecurityService).passwordEncoder(passwordEncoder());
    }
}

```

• USER.JAVA

```
package com.sporty.shoes.store.domain;
```

```
import java.util.Collection;
import java.util.HashSet;
import java.util.Set;
```

```
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.NamedAttributeNode;
import javax.persistence.NamedEntityGraph;
import javax.persistence.NamedSubgraph;
import javax.persistence.OneToOne;
import javax.persistence.OneToOne;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotNull;
```

```
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
```

```
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.sporty.shoes.store.domain.security.Authority;
import com.sporty.shoes.store.domain.security.UserRole;
```

```
@NamedEntityGraph(name = "UserComplete", attributeNodes = {
    @NamedAttributeNode(value = "userRoles", subgraph = "role-subgraph" ), subgraphs = {
        @NamedSubgraph(name = "role-subgraph", attributeNodes = {
            @NamedAttributeNode("role") }) })
@Entity
@SuppressWarnings("serial")
public class User implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

@Column(name = "id", nullable = false, updatable = false)
private Long id;
@NotNull
private String username;
private String password;
private String firstName;
private String lastName;
@NotNull
@email
private String email;
@OneToOne(cascade = CascadeType.ALL, orphanRemoval = true)
@JoinColumn(name = "address_id")
private Address address;
@OneToMany(mappedBy = "user", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
@JsonIgnore
private Set<UserRole> userRoles = new HashSet<>();

public User() {
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    Set<GrantedAuthority> authorites = new HashSet<>();
    userRoles.forEach(userRole -> authorites.add(new Authority(userRole.getRole().getName())));
    return authorites;
}

@Override
public String toString() {
    return getClass().getSimpleName() + "[id=" + id + "]" + "[username=" + username + "]" +
        "[password=" + password
            + "]" + "[email=" + email + "]";
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

```

```
@Override
public boolean isEnabled() {
    return true;
}

public Long getId() {
    return id;
}

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

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 getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getEmail() {
    return email;
}
```

```

    public void setEmail(String email) {
        this.email = email;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public Set<UserRole> getUserRoles() {
        return userRoles;
    }

    public void setUserRoles(Set<UserRole> userRoles) {
        this.userRoles = userRoles;
    }
}

```

• SIZE.JAVA

```
package com.sporty.shoes.store.domain;
```

```

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.ManyToOne;

```

```
@Entity
```

```
public class Size implements Comparable<Size> {
```

```

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "article_id")
    private Article article;
    private String value;

    public Size() {
    }
}

```



```

    public Size(String value, Article article) {
        this.value = value;
        this.article = article;
    }

    public Long getId() {
        return id;
    }

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

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    public Article getArticle() {
        return article;
    }

    public void setArticle(Article article) {
        this.article = article;
    }

    @Override
    public int compareTo(Size s) {
        return this.value.compareTo(s.getValue());
    }
}

```

● SHOPPINGCART.JAVA

```
package com.sporty.shoes.store.domain;
```

```
import java.math.BigDecimal;
```

```
import java.util.List;
```

```
public class ShoppingCart {
```

```
    private List<CartItem> cartItems;
```

```
    public ShoppingCart(List<CartItem> cartItems) {
        this.cartItems = cartItems;
    }

```

```

    }

    public BigDecimal getGrandTotal() {
        BigDecimal cartTotal = new BigDecimal(0);
        for (CartItem item : this.cartItems) {
            cartTotal = cartTotal.add(item.getSubtotal());
        }
        return cartTotal;
    }

    public boolean isEmpty() {
        return cartItems.isEmpty();
    }

    public void removeCartItem(CartItem cartItem) {
        cartItems.removeIf(item -> item.getId() == cartItem.getId());
    }

    public void clearItems() {
        cartItems.clear();
    }

    public CartItem findCartItemByArticleAndSize(Long id, String size) {
        for (CartItem item : this.cartItems) {
            if (item.getArticle().getId().equals(id) && item.getSize().equals(size)) {
                return item;
            }
        }
        return null;
    }

    public int getItemCount() {
        return this.cartItems.size();
    }

    public List<CartItem> getCartItems() {
        return cartItems;
    }

    public void setCartItems(List<CartItem> cartItems) {
        this.cartItems = cartItems;
    }
}

```

● SHIPPING.JAVA

```
package com.sporty.shoes.store.domain;
```

```
import javax.persistence.CascadeType;
```

```

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;

@Entity
public class Shipping {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String receiver;
    @OneToOne(cascade = CascadeType.ALL, orphanRemoval = true)
    @JoinColumn(name = "address_id")
    private Address address;
    @OneToOne
    private Order order;

    public Long getId() {
        return id;
    }

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

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public String getReceiver() {
        return receiver;
    }

    public void setReceiver(String receiver) {
        this.receiver = receiver;
    }

    public Order getOrder() {
        return order;
    }

    public void setOrder(Order order) {

```

```
        this.order = order;
    }
}
```

● PAYMENT.JAVA

```
package com.sporty.shoes.store.domain;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
```

```
@Entity
```

```
public class Payment {
```

```
    @Id
```

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

```
    private Long id;
```

```
    private String type;
```

```
    private String cardName;
```

```
    private String cardNumber;
```

```
    private int expiryMonth;
```

```
    private int expiryYear;
```

```
    private int cvc;
```

```
    private String holderName;
```

```
    @OneToOne
```

```
    private Order order;
```

```
    public Long getId() {
        return id;
    }
```

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

```
    public Order getOrder() {
        return order;
    }
```

```
    public void setOrder(Order order) {
        this.order = order;
    }
```

```
    public String getType() {
```

```

        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getCardName() {
        return cardName;
    }

    public void setCardName(String cardName) {
        this.cardName = cardName;
    }

    public String getCardNumber() {
        return cardNumber;
    }

    public void setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
    }

    public int getExpiryMonth() {
        return expiryMonth;
    }

    public void setExpiryMonth(int expiryMonth) {
        this.expiryMonth = expiryMonth;
    }

    public int getExpiryYear() {
        return expiryYear;
    }

    public void setExpiryYear(int expiryYear) {
        this.expiryYear = expiryYear;
    }

    public int getCvc() {
        return cvc;
    }

    public void setCvc(int cvc) {
        this.cvc = cvc;
    }

    public String getHolderName() {
        return holderName;
    }

```

```

    }

    public void setHolderName(String holderName) {
        this.holderName = holderName;
    }
}

```

• ORDER.JAVA

```
package com.sporty.shoes.store.domain;
```

```
import java.math.BigDecimal;
import java.util.Date;
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.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
```

```
@Entity
@Table(name="user_order")
public class Order {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private Date orderDate;
    private Date shippingDate;
    private String orderStatus;
    private BigDecimal orderTotal;
```

```
    @OneToMany(mappedBy="order", cascade=CascadeType.ALL, fetch = FetchType.LAZY)
    private List<CartItem> cartItems;
```

```
    @OneToOne(cascade=CascadeType.ALL, fetch = FetchType.LAZY)
    private Shipping shipping;
```

```
    @OneToOne(cascade=CascadeType.ALL, fetch = FetchType.LAZY)
    private Payment payment;
```

```
    @ManyToOne(fetch = FetchType.LAZY)
```

```
private User user;

public Long getId() {
    return id;
}

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

public Date getOrderDate() {
    return orderDate;
}

public void setOrderDate(Date orderDate) {
    this.orderDate = orderDate;
}

public Date getShippingDate() {
    return shippingDate;
}

public void setShippingDate(Date shippingDate) {
    this.shippingDate = shippingDate;
}

public String getOrderStatus() {
    return orderStatus;
}

public void setOrderStatus(String orderStatus) {
    this.orderStatus = orderStatus;
}

public BigDecimal getOrderTotal() {
    return orderTotal;
}

public void setOrderTotal(BigDecimal orderTotal) {
    this.orderTotal = orderTotal;
}

public List<CartItem> getCartItems() {
    return cartItems;
}

public void setCartItems(List<CartItem> cartItems) {
    this.cartItems = cartItems;
}
```

```

    public Shipping getShipping() {
        return shipping;
    }

    public void setShipping(Shipping shipping) {
        this.shipping = shipping;
    }

    public Payment getPayment() {
        return payment;
    }

    public void setPayment(Payment payment) {
        this.payment = payment;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}

```

• CATEGORY.JAVA

```
package com.sporty.shoes.store.domain;
```

```

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

```

```
@Entity
```

```
public class Category {
```

```
    @Id
```

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

```
    private Long id;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "article_id")
```

```
    private Article article;
```

```
    private String name;
```



```

    public Category() {
    }

    public Category(String name, Article article) {
        this.name = name;
        this.article = article;
    }

    public Long getId() {
        return id;
    }

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

    public Article getArticle() {
        return article;
    }

    public void setArticle(Article article) {
        this.article = article;
    }

    public String getName() {
        return name;
    }

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

```

● CARTITEM.JAVA

```

package com.sporty.shoes.store.domain;

import java.math.BigDecimal;

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.ManyToOne;
import javax.persistence.OneToOne;

```

@Entity

public class CartItem {

 @Id

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

private Long id;

private int qty;

private String size;

 @OneToOne

 @JoinColumn(name = "article_id")

private Article article;

 @ManyToOne(fetch = FetchType.**LAZY**)

 @JoinColumn(name = "user_id")

private User user;

 @ManyToOne

 @JoinColumn(name = "order_id")

private Order order;

public CartItem() {

 }

public boolean canUpdateQty(Integer qty) {

return qty == **null** || qty <= 0 || **this**.getArticle().hasStock(qty);

 }

public BigDecimal getSubtotal() {

return new BigDecimal(article.getPrice()).multiply(**new** BigDecimal(qty));

 }

public void addQuantity(**int** qty) {

if (qty > 0) {

this.qty = **this**.qty + qty;

 }

 }

public boolean hasSameSizeThan(String size2) {

return this.size.equals(size2);

 }

public Long getId() {

return id;

 }

public void setId(Long id) {

this.id = id;

 }

```

    public int getQty() {
        return qty;
    }

    public void setQty(int qty) {
        this.qty = qty;
    }

    public Article getArticle() {
        return article;
    }

    public void setArticle(Article article) {
        this.article = article;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Order getOrder() {
        return order;
    }

    public void setOrder(Order order) {
        this.order = order;
    }

    public String getSize() {
        return size;
    }

    public void setSize(String size) {
        this.size = size;
    }
}

```

● BRAND.JAVA

```

package com.sporty.shoes.store.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

```

```

import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Brand {

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

    @ManyToOne
    @JoinColumn(name = "article_id")
    private Article article;

    private String name;

    public Brand() {
    }

    public Brand(String name, Article article) {
        this.name = name;
        this.article = article;
    }

    public Long getId() {
        return id;
    }

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

    public Article getArticle() {
        return article;
    }

    public void setArticle(Article article) {
        this.article = article;
    }

    public String getName() {
        return name;
    }

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

```

● ARTICLEBUILDER.JAVA

```
package com.sporty.shoes.store.domain;
```

```
import java.util.HashSet;
```

```
import java.util.List;
```

```
import java.util.Set;
```

```
public class ArticleBuilder {
```

```
    private String title;
```

```
    private int stock;
```

```
    private double price;
```

```
    private String picture;
```

```
    private List<String> sizes;
```

```
    private List<String> categories;
```

```
    private List<String> brands;
```

```
    public ArticleBuilder() {  
    }
```

```
    public ArticleBuilder withTitle(String title) {  
        this.title = title;  
        return this;  
    }
```

```
    public ArticleBuilder stockAvailable(int stock) {  
        this.stock = stock;  
        return this;  
    }
```

```
    public ArticleBuilder withPrice(double price) {  
        this.price = price;  
        return this;  
    }
```

```
    public ArticleBuilder imageLink(String picture) {  
        this.picture = picture;  
        return this;  
    }
```

```
    public ArticleBuilder sizesAvailable(List<String> sizes) {  
        this.sizes = sizes;  
        return this;  
    }
```

```
    public ArticleBuilder ofCategories(List<String> categories) {  
        this.categories = categories;  
        return this;  
    }
```

```

    }

    public ArticleBuilder ofBrand(List<String> brands) {
        this.brands = brands;
        return this;
    }

    public Article build() {
        Article article = new Article();
        article.setTitle(this.title);
        article.setPrice(this.price);
        article.setStock(this.stock);
        article.setPicture(this.picture);

        if (this.sizes != null && !this.sizes.isEmpty()) {
            Set<Size> sizeElements = new HashSet<>();
            for (String val : this.sizes) {
                sizeElements.add(new Size(val, article));
            }
            article.setSizes(sizeElements);
        }

        if (this.categories != null && !this.categories.isEmpty()) {
            Set<Category> catElements = new HashSet<>();
            for (String val : this.categories) {
                catElements.add(new Category(val, article));
            }
            article.setCategories(catElements);
        }

        if (this.brands != null && !this.brands.isEmpty()) {
            Set<Brand> brandlements = new HashSet<>();
            for (String val : this.brands) {
                brandlements.add(new Brand(val, article));
            }
            article.setBrands(brandlements);
        }

        return article;
    }
}

```

• ARTICLE.JAVA

```
package com.sporty.shoes.store.domain;
```

```
import java.util.Set;
```

```
import javax.persistence.CascadeType;
```

```
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
```

```
@Entity
```

```
public class Article {
```

```
    @Id
```

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

```
    private Long id;
```

```
    private String title;
```

```
    private int stock;
```

```
    private double price;
```

```
    private String picture;
```

```
    @OneToMany(mappedBy = "article", cascade = CascadeType.ALL, orphanRemoval = true)
```

```
    private Set<Size> sizes;
```

```
    @OneToMany(mappedBy = "article", cascade = CascadeType.ALL, orphanRemoval = true)
```

```
    private Set<Brand> brands;
```

```
    @OneToMany(mappedBy = "article", cascade = CascadeType.ALL, orphanRemoval = true)
```

```
    private Set<Category> categories;
```

```
    public Article() {
```

```
    }
```

```
    public boolean hasStock(int amount) {
```

```
        return (this.getStock() > 0) && (amount <= this.getStock());
```

```
    }
```

```
    public void decreaseStock(int amount) {
```

```
        this.stock -= amount;
```

```
    }
```

```
    public void addSize(Size size) {
```

```
        sizes.add(size);
```

```
        size.setArticle(this);
```

```
    }
```

```
    public void removeSize(Size size) {
```

```
        sizes.remove(size);
```

```
        size.setArticle(null);
```

```
    }
```

```
    public void addCategory(Category category) {
```

```
        categories.add(category);
```

```
        category.setArticle(this);
```

```

}

public void removeCategory(Category category) {
    categories.remove(category);
    category.setArticle(null);
}

public void addSize(Brand brand) {
    brands.add(brand);
    brand.setArticle(this);
}

public void removeSize(Brand brand) {
    brands.remove(brand);
    brand.setArticle(null);
}

public Long getId() {
    return id;
}

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

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public double getPrice() {
    return price;
}

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

public int getStock() {
    return stock;
}

public void setStock(int stock) {
    this.stock = stock;
}

```



```

    public Set<Size> getSizes() {
        return sizes;
    }

    public void setSizes(Set<Size> sizes) {
        this.sizes = sizes;
    }

    public Set<Brand> getBrands() {
        return brands;
    }

    public void setBrands(Set<Brand> brands) {
        this.brands = brands;
    }

    public Set<Category> getCategories() {
        return categories;
    }

    public void setCategories(Set<Category> categories) {
        this.categories = categories;
    }

    public String getPicture() {
        return picture;
    }

    public void setPicture(String picture) {
        this.picture = picture;
    }
}

```

● ADDRESS.JAVA

```

package com.sporty.shoes.store.domain;

```

```

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

```

```

@Entity

```

```

public class Address {

```

```

    @Id

```

```

    @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

    private Long id;

```

```

    private String streetAddress;

```

```

private String city;
private String country;
private String zipCode;

public Long getId() {
    return id;
}
public void setId(Long id) {
    this.id = id;
}
public String getStreetAddress() {
    return streetAddress;
}
public void setStreetAddress(String streetAddress) {
    this.streetAddress = streetAddress;
}
public String getCity() {
    return city;
}
public void setCity(String city) {
    this.city = city;
}
public String getCountry() {
    return country;
}
public void setCountry(String country) {
    this.country = country;
}
public String getZipCode() {
    return zipCode;
}
public void setZipCode(String zipCode) {
    this.zipCode = zipCode;
}
}

```

• USERROLE.JAVA

```
package com.sporty.shoes.store.domain.security;
```

```

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.ManyToOne;
import javax.persistence.Table;

```

```

import com.sporty.shoes.store.domain.User;

@Entity
@Table(name = "user_role")
public class UserRole {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userRoleId;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    private User user;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "role_id")
    private Role role;

    public UserRole() {

    }

    public UserRole(User user, Role role) {
        this.user = user;
        this.role = role;
    }

    public Long getUserRoleId() {
        return userRoleId;
    }

    public void setUserRoleId(Long userRoleId) {
        this.userRoleId = userRoleId;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }
}

```

```
}  
}
```

• ROLE.JAVA

```
package com.sporty.shoes.store.domain.security;
```

```
import java.util.HashSet;
```

```
import java.util.Set;
```

```
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.OneToMany;
```

```
@Entity
```

```
public class Role {
```

```
    @Id
```

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

```
    private Long roleId;
```

```
    @Column(unique = true)
```

```
    private String name;
```

```
    @OneToMany(mappedBy = "role", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
```

```
    private Set<UserRole> userRoles = new HashSet<>();
```

```
    public Role() {
```

```
    }
```

```
    public Long getRoleId() {
```

```
        return roleId;
```

```
    }
```

```
    public void setRoleId(Long roleId) {
```

```
        this.roleId = roleId;
```

```
    }
```

```
    public String getName() {
```

```
        return name;
```

```
    }
```

```
    public void setName(String name) {
```

```
        this.name = name;
```

```

    }

    public Set<UserRole> getUserRoles() {
        return userRoles;
    }

    public void setUserRoles(Set<UserRole> userRoles) {
        this.userRoles = userRoles;
    }
}

```

● AUTHORITY.JAVA

```

package com.sporty.shoes.store.domain.security;

import org.springframework.security.core.GrantedAuthority;

@SuppressWarnings("serial")
public class Authority implements GrantedAuthority {

    private final String authority;

    public Authority(String authority) {
        this.authority = authority;
    }

    public String getAuthority() {
        return authority;
    }
}

```

● USERCONTROLLER.JAVA

```

package com.sporty.shoes.store.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import com.sporty.shoes.store.service.UserService;

@Controller
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping("/userdata")

```

```

    public String userList(Model model) {

        model.addAttribute("listUsers", userService.getAllUsers());
        return "userList";
    }
}

```

• STORECONTROLLER.JAVA

```

package com.sporty.shoes.store.controller;

```

```

import javax.websocket.server.PathParam;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;

```

```

import com.sporty.shoes.store.domain.Article;
import com.sporty.shoes.store.form.ArticleFilterForm;
import com.sporty.shoes.store.service.ArticleService;
import com.sporty.shoes.store.type.SortFilter;

```

```

@Controller

```

```

public class StoreController {

```

```

    @Autowired

```

```

    private ArticleService articleService;

```

```

    @RequestMapping("/store")

```

```

    public String store(@ModelAttribute("filters") ArticleFilterForm filters, Model model) {
        Integer page = filters.getPage();
        int pagenumber = (page == null || page <= 0) ? 0 : page - 1;
        SortFilter sortFilter = new SortFilter(filters.getSort());
        Page<Article> pageresult = articleService.findArticlesByCriteria(
            PageRequest.of(pagenumber, 9, sortFilter.getSortType()), filters.getPricelow(),
            filters.getPricehigh(),
            filters.getSize(), filters.getCategory(), filters.getBrand(), filters.getSearch());
        model.addAttribute("allCategories", articleService.getAllCategories());
        model.addAttribute("allBrands", articleService.getAllBrands());
        model.addAttribute("allSizes", articleService.getAllSizes());
        model.addAttribute("articles", pageresult.getContent());
        model.addAttribute("totalitems", pageresult.getTotalElements());
        model.addAttribute("itemsperpage", 9);
        return "store";
    }
}

```

```

    }

    @RequestMapping("/article-detail")
    public String articleDetail(@PathParam("id") Long id, Model model) {
        Article article = articleService.findArticleById(id);
        model.addAttribute("article", article);
        model.addAttribute("notEnoughStock", model.asMap().get("notEnoughStock"));
        model.addAttribute("addArticleSuccess", model.asMap().get("addArticleSuccess"));
        return "articleDetail";
    }
}

```

• SHOPPINGCARTCONTROLLER.JAVA

```
package com.sporty.shoes.store.controller;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

```

```

import com.sporty.shoes.store.domain.Article;
import com.sporty.shoes.store.domain.CartItem;
import com.sporty.shoes.store.domain.ShoppingCart;
import com.sporty.shoes.store.domain.User;
import com.sporty.shoes.store.service.ArticleService;
import com.sporty.shoes.store.service.ShoppingCartService;

```

```

@Controller
@RequestMapping("/shopping-cart")
public class ShoppingCartController {

```

```

    @Autowired
    private ArticleService articleService;

```

```

    @Autowired
    private ShoppingCartService shoppingCartService;

```

```

    @RequestMapping("/cart")
    public String shoppingCart(Model model, Authentication authentication) {
        User user = (User) authentication.getPrincipal();
        ShoppingCart shoppingCart = shoppingCartService.getShoppingCart(user);
        model.addAttribute("cartItemList", shoppingCart.getCartItems());
        model.addAttribute("shoppingCart", shoppingCart);
        return "shoppingCart";
    }
}

```

```

    }

    @RequestMapping("/add-item")
    public String addItem(@ModelAttribute("article") Article article, @RequestParam("qty") String qty,
        @RequestParam("size") String size, RedirectAttributes attributes, Model model,
        Authentication authentication) {
        article = articleService.findArticleById(article.getId());
        if (!article.hasStock(Integer.parseInt(qty))) {
            attributes.addFlashAttribute("notEnoughStock", true);
            return "redirect:/article-detail?id=" + article.getId();
        }
        User user = (User) authentication.getPrincipal();
        shoppingCartService.addArticleToShoppingCart(article, user, Integer.parseInt(qty), size);
        attributes.addFlashAttribute("addArticleSuccess", true);
        return "redirect:/article-detail?id=" + article.getId();
    }

    @RequestMapping("/update-item")
    public String updateItemQuantity(@RequestParam("id") Long cartItemId, @RequestParam("qty")
Integer qty,
        Model model) {
        CartItem cartItem = shoppingCartService.findCartItemById(cartItemId);
        if (cartItem.canUpdateQty(qty)) {
            shoppingCartService.updateCartItem(cartItem, qty);
        }
        return "redirect:/shopping-cart/cart";
    }

    @RequestMapping("/remove-item")
    public String removeItem(@RequestParam("id") Long id) {
        shoppingCartService.removeCartItem(shoppingCartService.findCartItemById(id));
        return "redirect:/shopping-cart/cart";
    }
}

```

• HomeController.java

```

package com.sporty.shoes.store.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import com.sporty.shoes.store.domain.Article;
import com.sporty.shoes.store.service.ArticleService;

```


@Controller

```
public class HomeController {  
  
    @Autowired  
    private ArticleService articleService;  
  
    @RequestMapping("/")  
    public String index(Model model) {  
        List<Article> articles = articleService.findFirstArticles();  
        model.addAttribute("articles", articles);  
        return "index";  
    }  
}
```

● GLOBALCONTROLLERADVICE.JAVA

```
package com.sporty.shoes.store.controller;
```

```
import java.util.Date;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.propertyeditors.StringTrimmerEditor;  
import org.springframework.core.annotation.AnnotationUtils;  
import org.springframework.security.authentication.AnonymousAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.WebDataBinder;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.annotation.InitBinder;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.ResponseStatus;  
import org.springframework.web.servlet.ModelAndView;
```

```
import com.sporty.shoes.store.domain.User;
```

```
import com.sporty.shoes.store.service.ShoppingCartService;
```

@ControllerAdvice

```
public class GlobalControllerAdvice {  
  
    public static final String DEFAULT_ERROR_VIEW = "error";  
  
    @Autowired  
    private ShoppingCartService shoppingCartService;  
  
    @InitBinder
```

```

    public void initBinder(WebDataBinder dataBinder) {
        StringTrimmerEditor stringTrimmerEditor = new StringTrimmerEditor(true);
        dataBinder.registerCustomEditor(String.class, stringTrimmerEditor);
    }

    @ModelAttribute
    public void populateModel(Model model) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (auth != null && auth.isAuthenticated() && !(auth instanceof
AnonymousAuthenticationToken)) {
            User user = (User) auth.getPrincipal();
            if (user != null) {
                model.addAttribute("shoppingCartItemNumber",
shoppingCartService.getItemsNumber(user));
            }
        } else {
            model.addAttribute("shoppingCartItemNumber", 0);
        }
    }

    @ExceptionHandler(value = Exception.class)
    public ModelAndView defaultErrorHandler(HttpServletRequest req, Exception e) throws Exception {
        if (AnnotationUtils.findAnnotation(e.getClass(), ResponseStatus.class) != null)
            throw e;
        ModelAndView mav = new ModelAndView();
        mav.addObject("timestamp", new Date(System.currentTimeMillis()));
        mav.addObject("path", req.getRequestURL());
        mav.addObject("message", e.getMessage());
        mav.setViewName(DEFAULT_ERROR_VIEW);
        return mav;
    }
}

```

• CHECKOUTCONTROLLER.JAVA

```

package com.sporty.shoes.store.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.sporty.shoes.store.domain.Address;
import com.sporty.shoes.store.domain.Order;

```

```

import com.sporty.shoes.store.domain.Payment;
import com.sporty.shoes.store.domain.Shipping;
import com.sporty.shoes.store.domain.ShoppingCart;
import com.sporty.shoes.store.domain.User;
import com.sporty.shoes.store.service.OrderService;
import com.sporty.shoes.store.service.ShoppingCartService;

@Controller
public class CheckoutController {

    @Autowired
    private ShoppingCartService shoppingCartService;

    @Autowired
    private OrderService orderService;

    @RequestMapping("/checkout")
    public String checkout(@RequestParam(value = "missingRequiredField", required = false) boolean
missingRequiredField,
        Model model, Authentication authentication) {
        User user = (User) authentication.getPrincipal();
        ShoppingCart shoppingCart = shoppingCartService.getShoppingCart(user);
        if (shoppingCart.isEmpty()) {
            model.addAttribute("emptyCart", true);
            return "redirect:/shopping-cart/cart";
        }
        model.addAttribute("cartItemList", shoppingCart.getCartItems());
        model.addAttribute("shoppingCart", shoppingCart);
        if (missingRequiredField) {
            model.addAttribute("missingRequiredField", true);
        }
        return "checkout";
    }

    @RequestMapping(value = "/checkout", method = RequestMethod.POST)
    public String placeOrder(@ModelAttribute("shipping") Shipping shipping,
@ModelAttribute("address") Address address,
        @ModelAttribute("payment") Payment payment, RedirectAttributes
redirectAttributes,
        Authentication authentication) {
        User user = (User) authentication.getPrincipal();
        ShoppingCart shoppingCart = shoppingCartService.getShoppingCart(user);
        if (!shoppingCart.isEmpty()) {
            shipping.setAddress(address);
            Order order = orderService.createOrder(shoppingCart, shipping, payment, user);
            redirectAttributes.addFlashAttribute("order", order);
        }
        return "redirect:/order-submitted";
    }
}

```

```

    @RequestMapping(value = "/order-submitted", method = RequestMethod.GET)
    public String orderSubmitted(Model model) {
        Order order = (Order) model.asMap().get("order");
        if (order == null) {
            return "redirect:/";
        }
        model.addAttribute("order", order);
        return "orderSubmitted";
    }
}

```

• ARTICLECONTROLLER.JAVA

```
package com.sporty.shoes.store.controller;
```

```
import java.util.Arrays;
```

```
import java.util.List;
```

```
import javax.servlet.http.HttpServletRequest;
```

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

```
import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.Model;
```

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

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

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

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

```
import com.sporty.shoes.store.domain.Article;
```

```
import com.sporty.shoes.store.domain.ArticleBuilder;
```

```
import com.sporty.shoes.store.domain.Brand;
```

```
import com.sporty.shoes.store.domain.Category;
```

```
import com.sporty.shoes.store.domain.Size;
```

```
import com.sporty.shoes.store.service.ArticleService;
```

```
@Controller
```

```
@RequestMapping("/article")
```

```
public class ArticleController {
```

```
    @Autowired
```

```
    private ArticleService articleService;
```

```
    @RequestMapping("/add")
```

```
    public String addArticle(Model model) {
```

```
        Article article = new Article();
```

```
        model.addAttribute("article", article);
```

```
        model.addAttribute("allSizes", articleService.getAllSizes());
```

```
        model.addAttribute("allBrands", articleService.getAllBrands());
```

```

        model.addAttribute("allCategories", articleService.getAllCategories());
        return "addArticle";
    }

    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public String addArticlePost(@ModelAttribute("article") Article article, HttpServletRequest request) {
        Article newArticle = new
ArticleBuilder().withTitle(article.getTitle()).stockAvailable(article.getStock())
                .withPrice(article.getPrice()).imageLink(article.getPicture())
                .sizesAvailable(Arrays.asList(request.getParameter("size").split("\\s*,\\s*")))

.ofCategories(Arrays.asList(request.getParameter("category").split("\\s*,\\s*")))

.ofBrand(Arrays.asList(request.getParameter("brand").split("\\s*,\\s*"))).build();
        articleService.saveArticle(newArticle);
        return "redirect:article-list";
    }

    @RequestMapping("/article-list")
    public String articleList(Model model) {
        List<Article> articles = articleService.findAllArticles();
        model.addAttribute("articles", articles);
        return "articleList";
    }

    @RequestMapping("/edit")
    public String editArticle(@RequestParam("id") Long id, Model model) {
        Article article = articleService.findArticleById(id);
        String preselectedSizes = "";
        for (Size size : article.getSizes()) {
            preselectedSizes += (size.getValue() + ",");
        }
        String preselectedBrands = "";
        for (Brand brand : article.getBrands()) {
            preselectedBrands += (brand.getName() + ",");
        }
        String preselectedCategories = "";
        for (Category category : article.getCategories()) {
            preselectedCategories += (category.getName() + ",");
        }
        model.addAttribute("article", article);
        model.addAttribute("preselectedSizes", preselectedSizes);
        model.addAttribute("preselectedBrands", preselectedBrands);
        model.addAttribute("preselectedCategories", preselectedCategories);
        model.addAttribute("allSizes", articleService.getAllSizes());
        model.addAttribute("allBrands", articleService.getAllBrands());
        model.addAttribute("allCategories", articleService.getAllCategories());
        return "editArticle";
    }
}

```

```

    @RequestMapping(value = "/edit", method = RequestMethod.POST)
    public String editArticlePost(@ModelAttribute("article") Article article, HttpServletRequest request) {
        Article newArticle = new
ArticleBuilder().withTitle(article.getTitle()).stockAvailable(article.getStock())
                .withPrice(article.getPrice()).imageLink(article.getPicture())
                .sizesAvailable(Arrays.asList(request.getParameter("size").split("\\s*,\\s*")))

        .ofCategories(Arrays.asList(request.getParameter("category").split("\\s*,\\s*")))

        .ofBrand(Arrays.asList(request.getParameter("brand").split("\\s*,\\s*"))).build();
        newArticle.setId(article.getId());
        articleService.saveArticle(newArticle);
        return "redirect:article-list";
    }

    @RequestMapping("/delete")
    public String deleteArticle(@RequestParam("id") Long id) {
        articleService.deleteArticleById(id);
        return "redirect:article-list";
    }
}

```

● ACCOUNTCONTROLLER.JAVA

```

package com.sporty.shoes.store.controller;

```

```

import java.security.Principal;
import java.util.Arrays;
import java.util.List;

```

```

import javax.validation.Valid;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

```

```

import com.sporty.shoes.store.domain.Address;
import com.sporty.shoes.store.domain.Order;
import com.sporty.shoes.store.domain.User;
import com.sporty.shoes.store.service.OrderService;

```

```

import com.sporty.shoes.store.service.UserService;
import com.sporty.shoes.store.service.impl.UserSecurityService;

import utility.SecurityUtility;

@Controller
public class AccountController {

    @Autowired
    private UserService userService;

    @Autowired
    private UserSecurityService userSecurityService;

    @Autowired
    private OrderService orderService;

    @RequestMapping("/login")
    public String log(Model model) {
        model.addAttribute("usernameExists", model.asMap().get("usernameExists"));
        model.addAttribute("emailExists", model.asMap().get("emailExists"));
        return "myAccount";
    }

    @RequestMapping("/my-profile")
    public String myProfile(Model model, Authentication authentication) {
        User user = (User) authentication.getPrincipal();
        model.addAttribute("user", user);
        return "myProfile";
    }

    @RequestMapping("/my-orders")
    public String myOrders(Model model, Authentication authentication) {
        User user = (User) authentication.getPrincipal();
        model.addAttribute("user", user);
        List<Order> orders = orderService.findByUser(user);
        model.addAttribute("orders", orders);
        return "myOrders";
    }

    @RequestMapping("/my-address")
    public String myAddress(Model model, Principal principal) {
        User user = userService.findByUsername(principal.getName());
        model.addAttribute("user", user);
        return "myAddress";
    }

    @RequestMapping(value = "/update-user-address", method = RequestMethod.POST)

```

```

    public String updateUserAddress(@ModelAttribute("address") Address address, Model model,
Principal principal)
        throws Exception {
    User currentUser = userService.findByUsername(principal.getName());
    if (currentUser == null) {
        throw new Exception("User not found");
    }
    currentUser.setAddress(address);
    userService.save(currentUser);
    return "redirect:/my-address";
}

@RequestMapping(value = "/new-user", method = RequestMethod.POST)
public String newUserPost(@Valid @ModelAttribute("user") User user, BindingResult bindingResults,
    @ModelAttribute("new-password") String password, RedirectAttributes
redirectAttributes, Model model) {
    model.addAttribute("email", user.getEmail());
    model.addAttribute("username", user.getUsername());
    boolean invalidFields = false;
    if (bindingResults.hasErrors()) {
        return "redirect:/login";
    }
    if (userService.findByUsername(user.getUsername()) != null) {
        redirectAttributes.addFlashAttribute("usernameExists", true);
        invalidFields = true;
    }
    if (userService.findByEmail(user.getEmail()) != null) {
        redirectAttributes.addFlashAttribute("emailExists", true);
        invalidFields = true;
    }
    if (invalidFields) {
        return "redirect:/login";
    }
    user = userService.createUser(user.getUsername(), password, user.getEmail(),
Arrays.asList("ROLE_USER"));
    userSecurityService.authenticateUser(user.getUsername());
    return "redirect:/my-profile";
}

@RequestMapping(value = "/update-user-info", method = RequestMethod.POST)
public String updateUserInfo(@ModelAttribute("user") User user, @RequestParam("newPassword")
String newPassword,
    Model model, Principal principal) throws Exception {
    User currentUser = userService.findByUsername(principal.getName());
    if (currentUser == null) {
        throw new Exception("User not found");
    }
    /* check username already exists */
    User existingUser = userService.findByUsername(user.getUsername());

```



```

        if (existingUser != null && !existingUser.getId().equals(currentUser.getId())) {
            model.addAttribute("usernameExists", true);
            return "myProfile";
        }
        /* check email already exists */
        existingUser = userService.findByEmail(user.getEmail());
        if (existingUser != null && !existingUser.getId().equals(currentUser.getId())) {
            model.addAttribute("emailExists", true);
            return "myProfile";
        }
        /* update password */
        if (newPassword != null && !newPassword.isEmpty() && !newPassword.equals("")) {
            BCryptPasswordEncoder passwordEncoder = SecurityUtility.passwordEncoder();
            String dbPassword = currentUser.getPassword();
            if (passwordEncoder.matches(user.getPassword(), dbPassword)) {
                currentUser.setPassword(passwordEncoder.encode(newPassword));
            } else {
                model.addAttribute("incorrectPassword", true);
                return "myProfile";
            }
        }
        currentUser.setFirstName(user.getFirstName());
        currentUser.setLastName(user.getLastName());
        currentUser.setUsername(user.getUsername());
        currentUser.setEmail(user.getEmail());
        userService.save(currentUser);
        model.addAttribute("updateSuccess", true);
        model.addAttribute("user", currentUser);
        userSecurityService.authenticateUser(currentUser.getUsername());
        return "myProfile";
    }

    @RequestMapping("/order-detail")
    public String orderDetail(@RequestParam("order") Long id, Model model) {
        Order order = orderService.findOrderWithDetails(id);
        model.addAttribute("order", order);
        return "orderDetails";
    }
}

```

• USERDTO.JAVA

package com.sporty.shoes.store.dto;

```

public class UserDTO {

    private String username;
    private String firstName;
    private String lastName;

```

```

    private String email;

    public UserDTO() {
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

```

• ARTICLEDTO.JAVA

```
package com.sporty.shoes.store.dto;
```

```

public class ArticleDTO {

    private Long id;
    private String title;
    private double price;
    private String picture;
}

```

```

    public ArticleDTO() {
    }

    public Long getId() {
        return id;
    }

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

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public double getPrice() {
        return price;
    }

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

    public String getPicture() {
        return picture;
    }

    public void setPicture(String picture) {
        this.picture = picture;
    }
}

```

• USERPROFILEFORM.JAVA

```
package com.sporty.shoes.store.form;
```

```

public class UserProfileForm {

}

```

• ARTICLEFILTERFORM.JAVA

```
package com.sporty.shoes.store.form;
```

```
import java.util.List;
```

```

public class ArticleFilterForm {

    private List<String> size;
    private List<String> category;
    private List<String> brand;
    private Integer pricelow;
    private Integer pricehigh;
    private String sort;
    private Integer page;
    private String search;

    public ArticleFilterForm() {
    }

    public List<String> getSize() {
        return size;
    }

    public void setSize(List<String> size) {
        this.size = size;
    }

    public List<String> getCategory() {
        return category;
    }

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

    public List<String> getBrand() {
        return brand;
    }

    public void setBrand(List<String> brand) {
        this.brand = brand;
    }

    public Integer getPricelow() {
        return pricelow;
    }

    public void setPricelow(Integer pricelow) {
        this.pricelow = pricelow;
    }

    public Integer getPricehigh() {
        return pricehigh;
    }
}

```

```

    }

    public void setPricehigh(Integer pricehigh) {
        this.pricehigh = pricehigh;
    }

    public String getSort() {
        return sort;
    }

    public void setSort(String sort) {
        this.sort = sort;
    }

    public Integer getPage() {
        return page;
    }

    public void setPage(Integer page) {
        this.page = page;
    }

    public String getSearch() {
        return search;
    }

    public void setSearch(String search) {
        this.search = search;
    }
}

```

• USERREPOSITORY.JAVA

```
package com.sporty.shoes.store.repository;
```

```
import org.springframework.data.jpa.repository.EntityGraph;
```

```
import org.springframework.data.jpa.repository.EntityGraph.EntityGraphType;
```

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

```
import com.sporty.shoes.store.domain.User;
```

```
public interface UserRepository extends CrudRepository<User, Long> {
```

```
    @EntityGraph(value = "UserComplete", type = EntityGraphType.FETCH)
    User findByUsername(String username);
```

```
    User findByEmail(String email);
```

```
}
```

• ROLEREPOSITORY.JAVA

```
package com.sporty.shoes.store.repository;

import org.springframework.data.repository.CrudRepository;

import com.sporty.shoes.store.domain.security.Role;

public interface RoleRepository extends CrudRepository<Role, Long> {

    Role findByName(String name);
}
```

• ORDERREPOSITORY.JAVA

```
package com.sporty.shoes.store.repository;

import java.util.List;

import org.springframework.data.jpa.repository.EntityGraph;
import org.springframework.data.repository.CrudRepository;

import com.sporty.shoes.store.domain.Order;
import com.sporty.shoes.store.domain.User;

public interface OrderRepository extends CrudRepository<Order, Long> {

    List<Order> findByUser(User user);

    @EntityGraph(attributePaths = { "cartItems", "payment", "shipping" })
    Order findEagerById(Long id);
}
```

• CARTITEMREPOSITORY.JAVA

```
package com.sporty.shoes.store.repository;

import java.util.List;

import org.springframework.data.jpa.repository.EntityGraph;
import org.springframework.data.repository.CrudRepository;

import com.sporty.shoes.store.domain.CartItem;
import com.sporty.shoes.store.domain.User;

public interface CartItemRepository extends CrudRepository<CartItem, Long> {

    @EntityGraph(attributePaths = { "article" })
```

```

        List<CartItem> findAllByUserAndOrderIsNull(User user);

        void deleteAllByUserAndOrderIsNull(User user);

        int countDistinctByUserAndOrderIsNull(User user);
    }

```

● ARTICLE SPECIFICATION.JAVA

```
package com
```

```
package com.sporty.shoes.store.repository;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import javax.persistence.criteria.CriteriaBuilder;
```

```
import javax.persistence.criteria.CriteriaQuery;
```

```
import javax.persistence.criteria.Join;
```

```
import javax.persistence.criteria.Predicate;
```

```
import javax.persistence.criteria.Root;
```

```
import org.springframework.data.jpa.domain.Specification;
```

```
import com.sporty.shoes.store.domain.Article;
```

```
import com.sporty.shoes.store.domain.Brand;
```

```
import com.sporty.shoes.store.domain.Category;
```

```
import com.sporty.shoes.store.domain.Size;
```

```
public class ArticleSpecification {
```

```
    private ArticleSpecification() {
    }

```

```
    @SuppressWarnings("serial")
```

```
    public static Specification<Article> filterBy(Integer priceLow, Integer priceHigh, List<String> sizes,
        List<String> categories, List<String> brands, String search) {
```

```
        return new Specification<Article>() {
```

```
            @Override
```

```
            public Predicate toPredicate(Root<Article> root, CriteriaQuery<?> query,
                CriteriaBuilder criteriaBuilder) {
```

```
                List<Predicate> predicates = new ArrayList<>();
```

```
                query.distinct(true);
```

```
                if (sizes != null && !sizes.isEmpty()) {
```

```
                    Join<Article, Size> joinSize = root.join("sizes");
```

```
                    predicates.add(criteriaBuilder.and(joinSize.get("value").in(sizes)));
```

```
                }
```

```
                if (categories != null && !categories.isEmpty()) {
```

```
                    Join<Article, Category> joinSize = root.join("categories");
```

```
                    predicates.add(criteriaBuilder.and(joinSize.get("name").in(categories)));
```

```

    }
    if (brands != null && !brands.isEmpty()) {
        Join<Article, Brand> joinSize = root.join("brands");
        predicates.add(criteriaBuilder.and(joinSize.get("name").in(brands)));
    }

    if (search != null && !search.isEmpty()) {
        predicates.add(criteriaBuilder.and(criteriaBuilder.like(root.get("title"),
"% " + search + "%")));
    }
    if (priceLow != null && priceLow >= 0) {
        predicates.add(

criteriaBuilder.and(criteriaBuilder.greaterThanOrEqualTo(root.get("price"), priceLow)));
    }
    if (priceHigh != null && priceHigh >= 0) {
        predicates

.add(criteriaBuilder.and(criteriaBuilder.lessThanOrEqualTo(root.get("price"), priceHigh)));
    }
    return criteriaBuilder.and(predicates.toArray(new
Predicate[predicates.size()]));
    }
    };
}
}

```

● ARTICLEREPOSITORY.JAVA

```
package com.sporty.shoes.store.repository;
```

```
import java.util.List;
```

```
import java.util.Optional;
```

```
import org.springframework.data.jpa.repository.EntityGraph;
```

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

```
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
```

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

```
import com.sporty.shoes.store.domain.Article;
```

```
public interface ArticleRepository extends JpaRepository<Article, Long>, JpaSpecificationExecutor<Article> {
```

```
    @EntityGraph(attributePaths = { "sizes", "categories", "brands" })
```

```
    List<Article> findAllEagerBy();
```

```
    @EntityGraph(attributePaths = { "sizes", "categories", "brands" })
```

```
    Optional<Article> findById(Long id);

```



```

@Query("SELECT DISTINCT s.value FROM Size s")
List<String> findAllSizes();

@Query("SELECT DISTINCT c.name FROM Category c")
List<String> findAllCategories();

@Query("SELECT DISTINCT b.name FROM Brand b")
List<String> findAllBrands();

}

```

• USERSERVICE.JAVA

```

package com.sporty.shoes.store.service;

import java.util.List;

import com.sporty.shoes.store.domain.User;

public interface UserService {

    User findById(Long id);

    User findByUsername(String username);

    User findByEmail(String email);

    void save(User user);

    User createUser(String username, String email, String password, List<String> roles);

    List<User> getAllUsers();

}

```

• SHOPPINGCARTSERVICE.JAVA

```

package com.sporty.shoes.store.service;

import com.sporty.shoes.store.domain.Article;
import com.sporty.shoes.store.domain.CartItem;
import com.sporty.shoes.store.domain.ShoppingCart;
import com.sporty.shoes.store.domain.User;

public interface ShoppingCartService {

    ShoppingCart getShoppingCart(User user);

    int getItemsNumber(User user);

}

```

```

CartItem findCartItemById(Long cartItemId);

CartItem addArticleToShoppingCart(Article article, User user, int qty, String size);

void clearShoppingCart(User user);

void updateCartItem(CartItem cartItem, Integer qty);

void removeCartItem(CartItem cartItem);

}

```

• ORDERSERVICE.JAVA

```

package com.sporty.shoes.store.service;

import java.util.List;

import com.sporty.shoes.store.domain.Order;
import com.sporty.shoes.store.domain.Payment;
import com.sporty.shoes.store.domain.Shipping;
import com.sporty.shoes.store.domain.ShoppingCart;
import com.sporty.shoes.store.domain.User;

public interface OrderService {

    Order createOrder(ShoppingCart shoppingCart, Shipping shippingAddress, Payment payment, User user);

    List<Order> findByUser(User user);

    Order findOrderWithDetails(Long id);

}

```

• ORDERSERVICE.JAVA

```

package com.sporty.shoes.store.service;

import java.util.List;

import com.sporty.shoes.store.domain.Order;
import com.sporty.shoes.store.domain.Payment;
import com.sporty.shoes.store.domain.Shipping;
import com.sporty.shoes.store.domain.ShoppingCart;
import com.sporty.shoes.store.domain.User;

public interface OrderService {

```

```

    Order createOrder(ShoppingCart shoppingCart, Shipping shippingAddress, Payment payment, User
user);

    List<Order> findByUser(User user);

    Order findOrderWithDetails(Long id);
}

```

• ARTICLESERVICE.JAVA

```

package com.sporty.shoes.store.service;

```

```

import java.util.List;

```

```

import org.springframework.data.domain.Page;

```

```

import org.springframework.data.domain.Pageable;

```

```

import com.sporty.shoes.store.domain.Article;

```

```

public interface ArticleService {

```

```

    List<Article> findAllArticles();

```

```

    Page<Article> findArticlesByCriteria(Pageable pageable, Integer priceLow, Integer priceHigh,
List<String> sizes,
        List<String> categories, List<String> brands, String search);

```

```

    List<Article> findFirstArticles();

```

```

    Article findArticleById(Long id);

```

```

    Article saveArticle(Article article);

```

```

    void deleteArticleById(Long id);

```

```

    List<String> getAllSizes();

```

```

    List<String> getAllCategories();

```

```

    List<String> getAllBrands();
}

```

• USERSERVICEIMPL.JAVA

```

package com.sporty.shoes.store.service.impl;

```

```

import java.util.HashSet;

```

```

import java.util.List;

```

```

import java.util.Optional;

```

```
import java.util.Set;
```

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

```
import org.springframework.stereotype.Service;
```

```
import org.springframework.transaction.annotation.Transactional;
```

```
import com.sporty.shoes.store.domain.User;
```

```
import com.sporty.shoes.store.domain.security.Role;
```

```
import com.sporty.shoes.store.domain.security.UserRole;
```

```
import com.sporty.shoes.store.repository.RoleRepository;
```

```
import com.sporty.shoes.store.repository.UserRepository;
```

```
import com.sporty.shoes.store.service.UserService;
```

```
import utility.SecurityUtility;
```

```
@Service
```

```
public class UserServiceImpl implements UserService {
```

```
    @Autowired
```

```
    private UserRepository userRepository;
```

```
    @Autowired
```

```
    private RoleRepository roleRepository;
```

```
    @Override
```

```
    public User findById(Long id) {
```

```
        Optional<User> opt = userRepository.findById(id);
```

```
        return opt.get();
```

```
    }
```

```
    @Override
```

```
    public User findByUsername(String username) {
```

```
        return userRepository.findByUsername(username);
```

```
    }
```

```
    @Override
```

```
    public User findByEmail(String email) {
```

```
        return userRepository.findByEmail(email);
```

```
    }
```

```
    @Override
```

```
    public void save(User user) {
```

```
        userRepository.save(user);
```

```
    }
```

```
    @Override
```

```
    @Transactional
```

```
    public User createUser(String username, String password, String email, List<String> roles) {
```

```
        User user = findByUsername(username);
```

```

        if (user != null) {
            return user;
        } else {
            user = new User();
            user.setUsername(username);
            user.setPassword(SecurityUtility.passwordEncoder().encode(password));
            user.setEmail(email);
            Set<UserRole> userRoles = new HashSet<>();
            for (String rolename : roles) {
                Role role = roleRepository.findByName(rolename);
                if (role == null) {
                    role = new Role();
                    role.setName(rolename);
                    roleRepository.save(role);
                }
                userRoles.add(new UserRole(user, role));
            }
            user.setUserRoles(userRoles);
            return userRepository.save(user);
        }
    }

    @Override
    public List<User> getAllUsers() {

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

```

● USERSECURITYSERVICE.JAVA

```
package com.sporty.shoes.store.service.impl;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

```

```

import com.sporty.shoes.store.domain.User;
import com.sporty.shoes.store.repository.UserRepository;

```

```

@Service
public class UserSecurityService implements UserDetailsService {

```

```

@Autowired
private UserRepository userRepository;

@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    User user = userRepository.findByUsername(username);
    if (user == null) {
        throw new UsernameNotFoundException("Username not found");
    }
    return user;
}

public void authenticateUser(String username) {
    UserDetails userDetails = loadUserByUsername(username);
    Authentication authentication = new UsernamePasswordAuthenticationToken(userDetails,
userDetails.getPassword(),
                                userDetails.getAuthorities());
    SecurityContextHolder.getContext().setAuthentication(authentication);
}
}

```

• SHOPPINGCARTSERVICEIMPL.JAVA

```

package com.sporty.shoes.store.service.impl;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.stereotype.Service;

import com.sporty.shoes.store.domain.Article;
import com.sporty.shoes.store.domain.CartItem;
import com.sporty.shoes.store.domain.ShoppingCart;
import com.sporty.shoes.store.domain.User;
import com.sporty.shoes.store.repository.CartItemRepository;
import com.sporty.shoes.store.service.ShoppingCartService;

@Service
public class ShoppingCartServiceImpl implements ShoppingCartService {

    @Autowired
    private CartItemRepository cartItemRepository;

    @Override
    public ShoppingCart getShoppingCart(User user) {
        return new ShoppingCart(cartItemRepository.findAllByUserAndOrderIsNull(user));
    }
}

```

```

@Override
@Cacheable("itemcount")
public int getItemsNumber(User user) {
    return cartItemRepository.countDistinctByUserAndOrderIsNull(user);
}

@Override
public CartItem findCartItemById(Long cartItemId) {
    Optional<CartItem> opt = cartItemRepository.findById(cartItemId);
    return opt.get();
}

@Override
@CacheEvict(value = "itemcount", allEntries = true)
public CartItem addArticleToShoppingCart(Article article, User user, int qty, String size) {
    ShoppingCart shoppingCart = this.getShoppingCart(user);
    CartItem cartItem = shoppingCart.findCartItemByArticleAndSize(article.getId(), size);
    if (cartItem != null && cartItem.hasSameSizeThan(size)) {
        cartItem.addQuantity(qty);
        cartItem.setSize(size);
        cartItem = cartItemRepository.save(cartItem);
    } else {
        cartItem = new CartItem();
        cartItem.setUser(user);
        cartItem.setArticle(article);
        cartItem.setQty(qty);
        cartItem.setSize(size);
        cartItem = cartItemRepository.save(cartItem);
    }
    return cartItem;
}

@Override
@CacheEvict(value = "itemcount", allEntries = true)
public void removeCartItem(CartItem cartItem) {
    cartItemRepository.deleteById(cartItem.getId());
}

@Override
@CacheEvict(value = "itemcount", allEntries = true)
public void updateCartItem(CartItem cartItem, Integer qty) {
    if (qty == null || qty <= 0) {
        this.removeCartItem(cartItem);
    } else if (cartItem.getArticle().hasStock(qty)) {
        cartItem.setQty(qty);
        cartItemRepository.save(cartItem);
    }
}

```

```

@Override
@CacheEvict(value = "itemcount", allEntries = true)
public void clearShoppingCart(User user) {
    cartItemRepository.deleteAllByUserAndOrderIsNull(user);
}
}

```

• ORDERSERVICEIMPL.JAVA

```
package com.sporty.shoes.store.service.impl;
```

```
import java.time.LocalDate;
import java.time.ZoneId;
import java.util.Date;
import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
```

```
import com.sporty.shoes.store.domain.Article;
import com.sporty.shoes.store.domain.CartItem;
import com.sporty.shoes.store.domain.Order;
import com.sporty.shoes.store.domain.Payment;
import com.sporty.shoes.store.domain.Shipping;
import com.sporty.shoes.store.domain.ShoppingCart;
import com.sporty.shoes.store.domain.User;
import com.sporty.shoes.store.repository.ArticleRepository;
import com.sporty.shoes.store.repository.CartItemRepository;
import com.sporty.shoes.store.repository.OrderRepository;
import com.sporty.shoes.store.service.OrderService;
```

```
@Service
public class OrderServiceImpl implements OrderService {
```

```
    @Autowired
    OrderRepository orderRepository;
```

```
    @Autowired
    CartItemRepository cartItemRepository;
```

```
    @Autowired
    ArticleRepository articleRepository;
```

```
    @Override
    @Transactional
    @CacheEvict(value = "itemcount", allEntries = true)
```



```

    public synchronized Order createOrder(ShoppingCart shoppingCart, Shipping shipping, Payment
payment, User user) {
        Order order = new Order();
        order.setUser(user);
        order.setPayment(payment);
        order.setShipping(shipping);
        order.setOrderTotal(shoppingCart.getGrandTotal());
        shipping.setOrder(order);
        payment.setOrder(order);
        LocalDate today = LocalDate.now();
        LocalDate estimatedDeliveryDate = today.plusDays(5);
        order.setOrderDate(Date.from(today.atStartOfDay(ZoneId.systemDefault()).toInstant()));

        order.setShippingDate(Date.from(estimatedDeliveryDate.atStartOfDay(ZoneId.systemDefault()).toIns
tant()));

        order.setOrderStatus("In Progress");

        order = orderRepository.save(order);

        List<CartItem> cartItems = shoppingCart.getCartItems();
        for (CartItem item : cartItems) {
            Article article = item.getArticle();
            article.decreaseStock(item.getQty());
            articleRepository.save(article);
            item.setOrder(order);
            cartItemRepository.save(item);
        }
        return order;
    }

    @Override
    public Order findOrderWithDetails(Long id) {
        return orderRepository.findEagerById(id);
    }

    public List<Order> findByUser(User user) {
        return orderRepository.findByUser(user);
    }
}

```

● ARTICLESERVICEIMPL.JAVA

```

package com.sporty.shoes.store.service.impl;

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

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

```

```

import org.springframework.beans.factory.annotation.Value;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

```

```

import com.sporty.shoes.store.domain.Article;
import com.sporty.shoes.store.repository.ArticleRepository;
import com.sporty.shoes.store.repository.ArticleSpecification;
import com.sporty.shoes.store.service.ArticleService;

```

```
@Service
```

```
@Transactional
```

```
public class ArticleServiceImpl implements ArticleService {
```

```
    @Autowired
```

```
    private ArticleRepository articleRepository;
```

```
    @Value("${articleservice.featured-items-number}")
```

```
    private int featuredArticlesNumber;
```

```
    @Override
```

```
    public List<Article> findAllArticles() {
        return (List<Article>) articleRepository.findAllEagerBy();
    }

```

```
    @Override
```

```
    public Page<Article> findArticlesByCriteria(Pageable pageable, Integer priceLow, Integer priceHigh,
        List<String> sizes, List<String> categories, List<String> brands, String search) {
        Page<Article> page = articleRepository.findAll(
            ArticleSpecification.filterBy(priceLow, priceHigh, sizes, categories, brands,
search), pageable);
        return page;
    }

```

```
    @Override
```

```
    public List<Article> findFirstArticles() {
        return articleRepository.findAll(PageRequest.of(0, featuredArticlesNumber)).getContent();
    }

```

```
    @Override
```

```
    public Article findArticleById(Long id) {
        Optional<Article> opt = articleRepository.findById(id);
        return opt.get();
    }

```

```

@Override
@CacheEvict(value = { "sizes", "categories", "brands" }, allEntries = true)
public Article saveArticle(Article article) {
    return articleRepository.save(article);
}

@Override
@CacheEvict(value = { "sizes", "categories", "brands" }, allEntries = true)
public void deleteArticleById(Long id) {
    articleRepository.deleteById(id);
}

@Override
@Cacheable("sizes")
public List<String> getAllSizes() {
    return articleRepository.findAllSizes();
}

@Override
@Cacheable("categories")
public List<String> getAllCategories() {
    return articleRepository.findAllCategories();
}

@Override
@Cacheable("brands")
public List<String> getAllBrands() {
    return articleRepository.findAllBrands();
}
}

```

• SORTFILTER.JAVA

```

package com.sporty.shoes.store.type;

import org.springframework.data.domain.Sort;

public class SortFilter {

    private String sortType;

    public SortFilter(String type) {
        this.sortType = type;
    }

    public Sort getSortType() {
        if (this.sortType == null) {
            return Sort.by("id").descending();
        }
    }
}

```

```

        switch (this.sortType) {
        case "priceasc":
            return Sort.by("price").ascending();
        case "pricedesc":
            return Sort.by("price").descending();
        case "alphasc":
            return Sort.by("title").ascending();
        case "alphdesc":
            return Sort.by("title").descending();
        default:
            return Sort.by("id").descending();
        }
    }
}

```

• SECURITYUTILITY.JAVA

```
package utility;
```

```
import java.security.SecureRandom;
```

```
import java.util.Random;
```

```
import org.springframework.context.annotation.Bean;
```

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

```
import org.springframework.stereotype.Component;
```

```
@Component
```

```
public class SecurityUtility {
```

```
    private static final String SALT = "salt";
```

```
    @Bean
```

```
    public static BCryptPasswordEncoder passwordEncoder() {
```

```
        return new BCryptPasswordEncoder(12, new SecureRandom(SALT.getBytes()));
```

```
    }
```

```
    @Bean
```

```
    public static String randomPassword() {
```

```
        String SALTCHARS = "ABCEFGHIJKLMNOPQRSTUVWXYZ1234567890";
```

```
        StringBuilder salt = new StringBuilder();
```

```
        Random rnd = new Random();
```

```
        while (salt.length() < 18) {
```

```
            int index = (int) (rnd.nextFloat() * SALTCHARS.length());
```

```
            salt.append(SALTCHARS.charAt(index));
```

```
        }
```

```
        String saltStr = salt.toString();
```

```

        return saltStr;
    }
}

```

● APPLICATION.PROPERTIES

spring.thymeleaf.cache=false

spring.datasource.url=\${SPRING_DATASOURCE_URL:jdbc:mysql://localhost:3306/sportyshoes?useSSL=false}

spring.datasource.username=\${SPRING_DATASOURCE_USERNAME:root}

spring.datasource.password=\${SPRING_DATASOURCE_PASSWORD:root12345}

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

spring.jpa.open-in-view=false

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect

#Custom

#number of items loaded on index

articleservice.featured-items-number=8

server.port=8081

logging.level.org.hibernate.sql=DEBUG

logging.level.org.hibernate.type=TRACE

● 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 http://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>2.1.3.RELEASE</version>
```

```
        <relativePath/> <!-- lookup parent from repository -->
```

```
    </parent>
```

```
    <groupId>com.nico.store</groupId>
```

```
    <artifactId>store</artifactId>
```

```
    <version>0.0.1-SNAPSHOT</version>
```

```
    <name>store</name>
```

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

```

<properties>
  <java.version>1.8</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
  </dependency>
  <dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity5</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context-support</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

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

</project>
```