Sporty Shoes.

(An E-Commerce Website.)

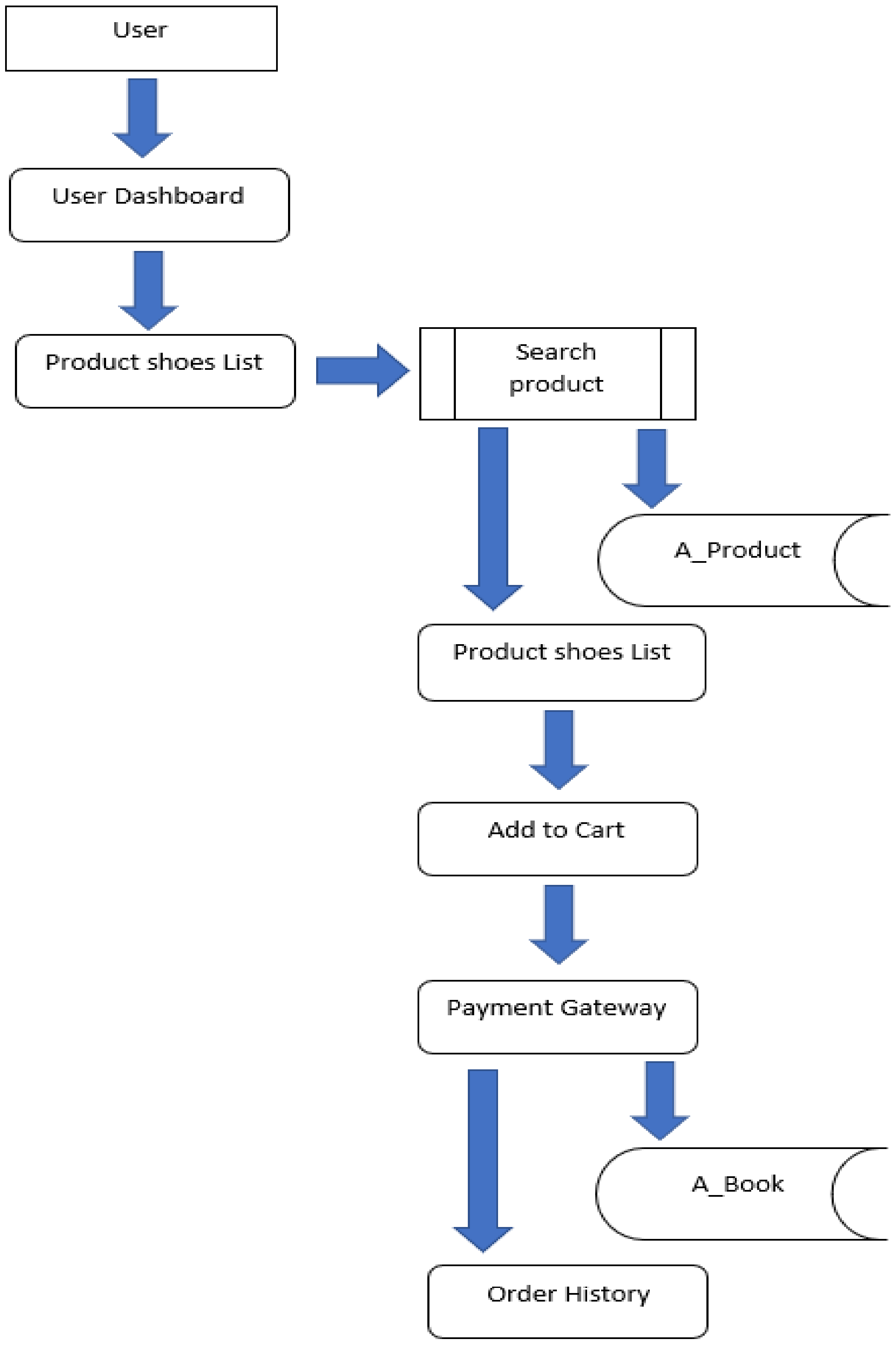
(Source Code)

- Version History:

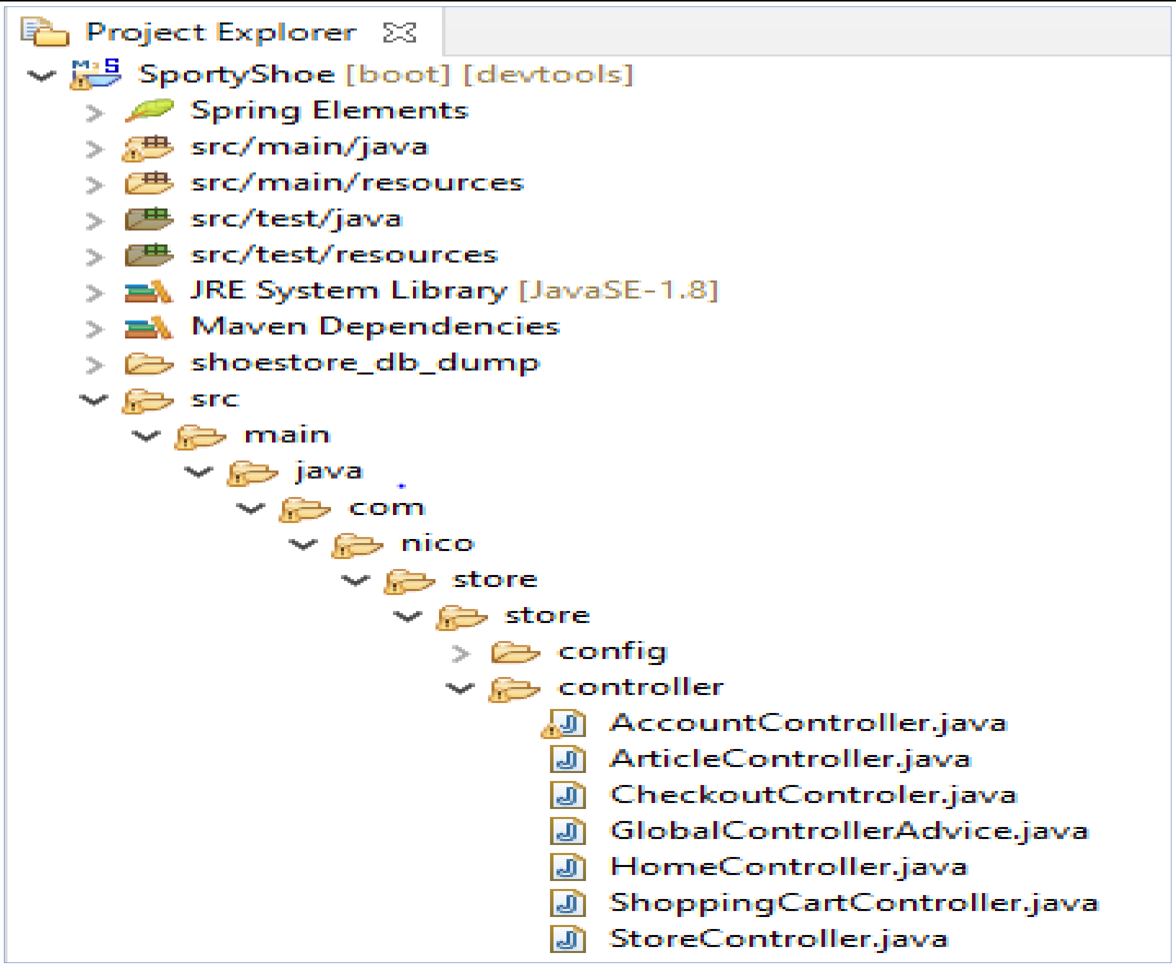
1.	**Author Name**	**Tushar Khillare.**
2.	**Purpose**	Source Code.
3.	**Date**	09 November 2021
4.	**Version**	1.0
5.	**Contact**	*tusharkhillare2015@gmail.com*

Contents.

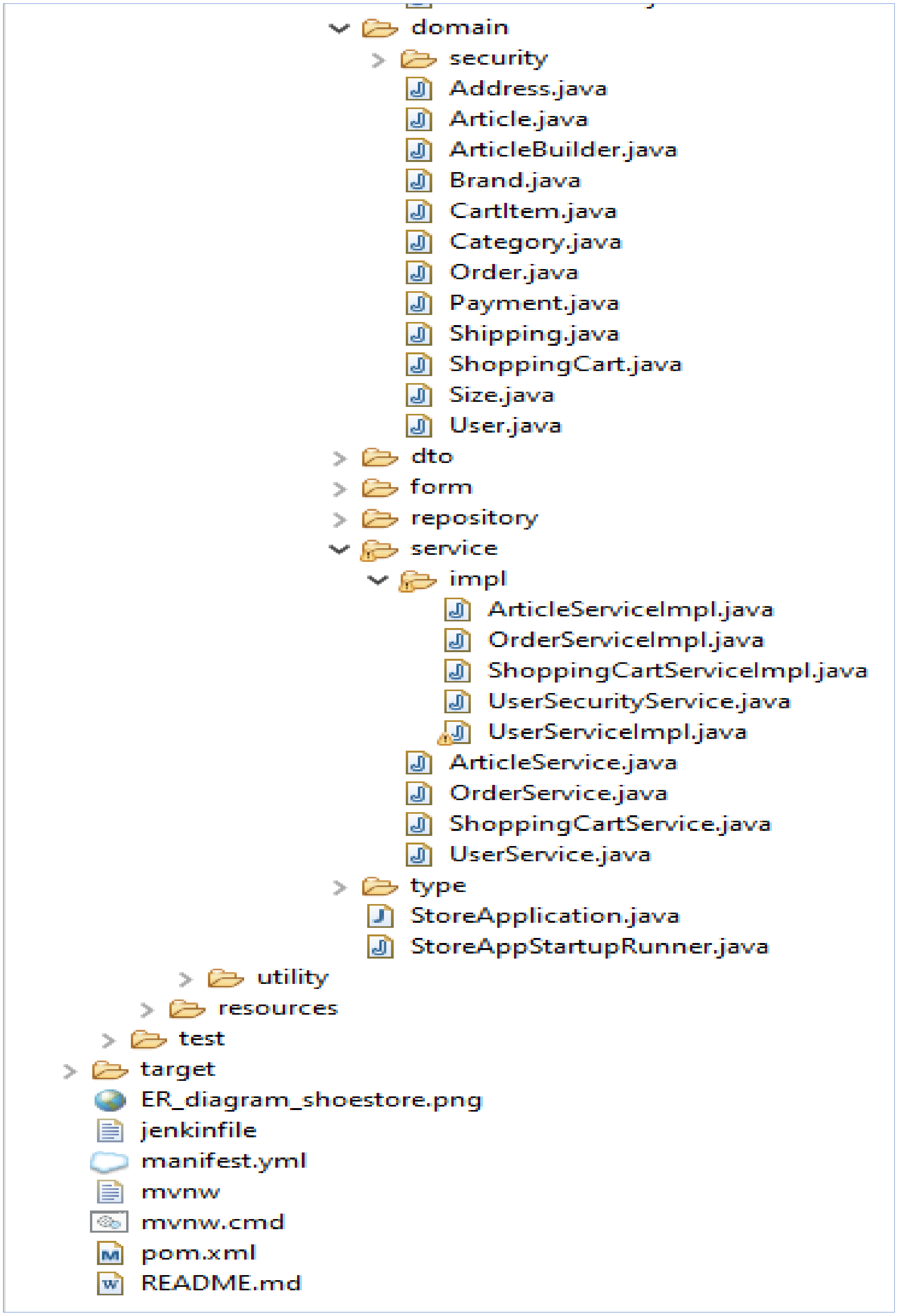
1. User Flow Chart Diagram.



2. Folder Structure.



```
Project Explorer
SportyShoe [boot] [devtools]
    Spring Elements
    src/main/java
    src/main/resources
    src/test/java
    src/test/resources
    JRE System Library [JavaSE-1.8]
    Maven Dependencies
    shoestore_db_dump
    src
        main
            java
                com
                    nico
                        store
                            store
                                config
                                controller
                                    AccountController.java
                                    ArticleController.java
                                    CheckoutControler.java
                                    GlobalControllerAdvice.java
                                    HomeController.java
                                    ShoppingCartController.java
                                    StoreController.java
```



```
        v domain
            > security
              Address.java
              Article.java
              ArticleBuilder.java
              Brand.java
              CartItem.java
              Category.java
              Order.java
              Payment.java
              Shipping.java
              ShoppingCart.java
              Size.java
              User.java
        > dto
        > form
        > repository
        v service
            v impl
                  ArticleServiceImpl.java
                  OrderServiceImpl.java
                  ShoppingCartServiceImpl.java
                  UserSecurityService.java
                  UserServiceImpl.java
              ArticleService.java
              OrderService.java
              ShoppingCartService.java
              UserService.java
        > type
          StoreApplication.java
          StoreAppStartupRunner.java
    > utility
  > resources
> test
> target
  ER_diagram_shoestore.png
  jenkinfile
  manifest.yml
  mvnw
  mvnw.cmd
  pom.xml
  README.md
```

3.Project Files.

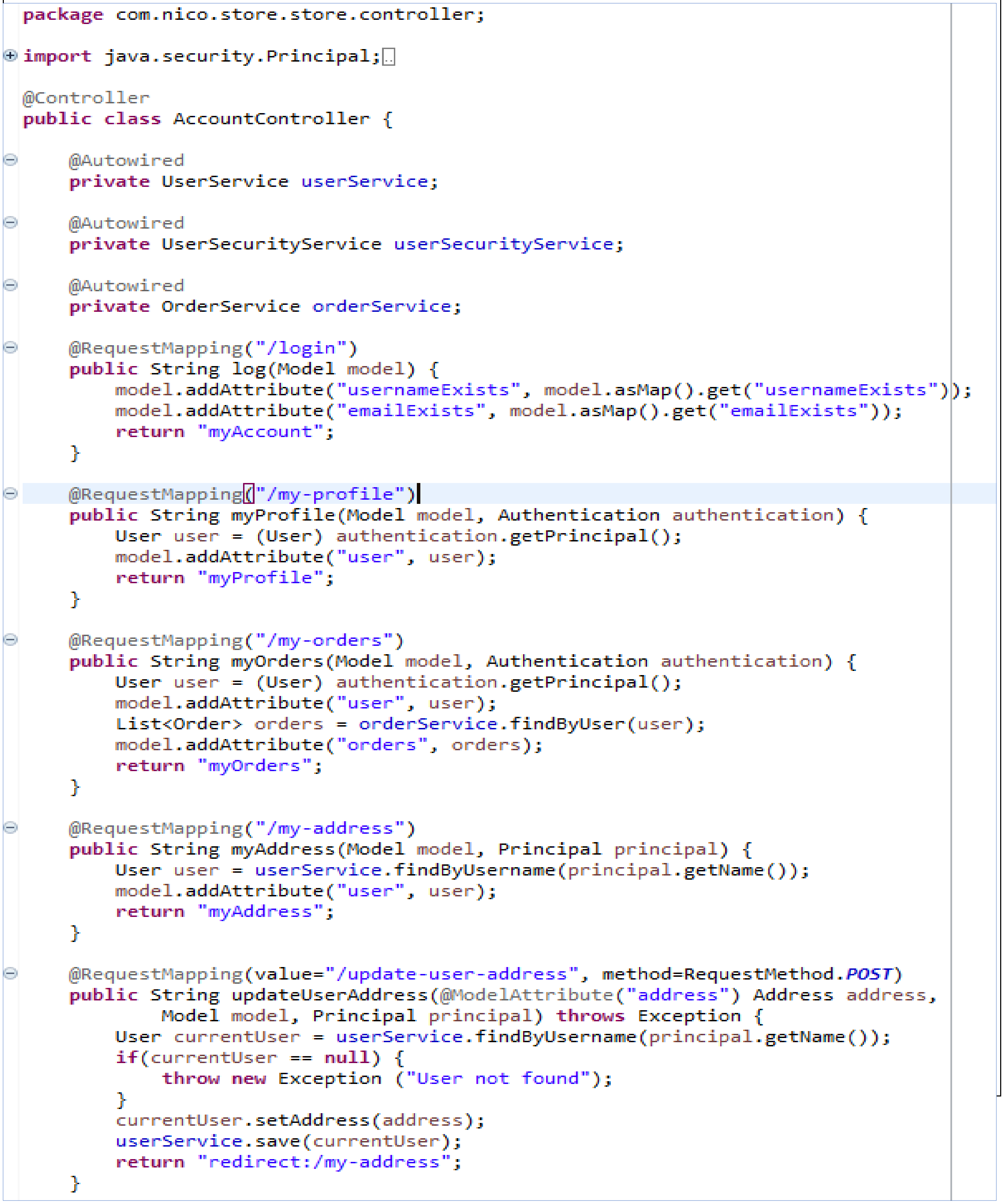
Rest of the files are present in the given below GitHub link kindly refer this…

Note: For Database ER-Diagram for the reference kindly refer the file ER_diagram_shoestore.png. All database dump file is available inside folder shoestore_db_dump

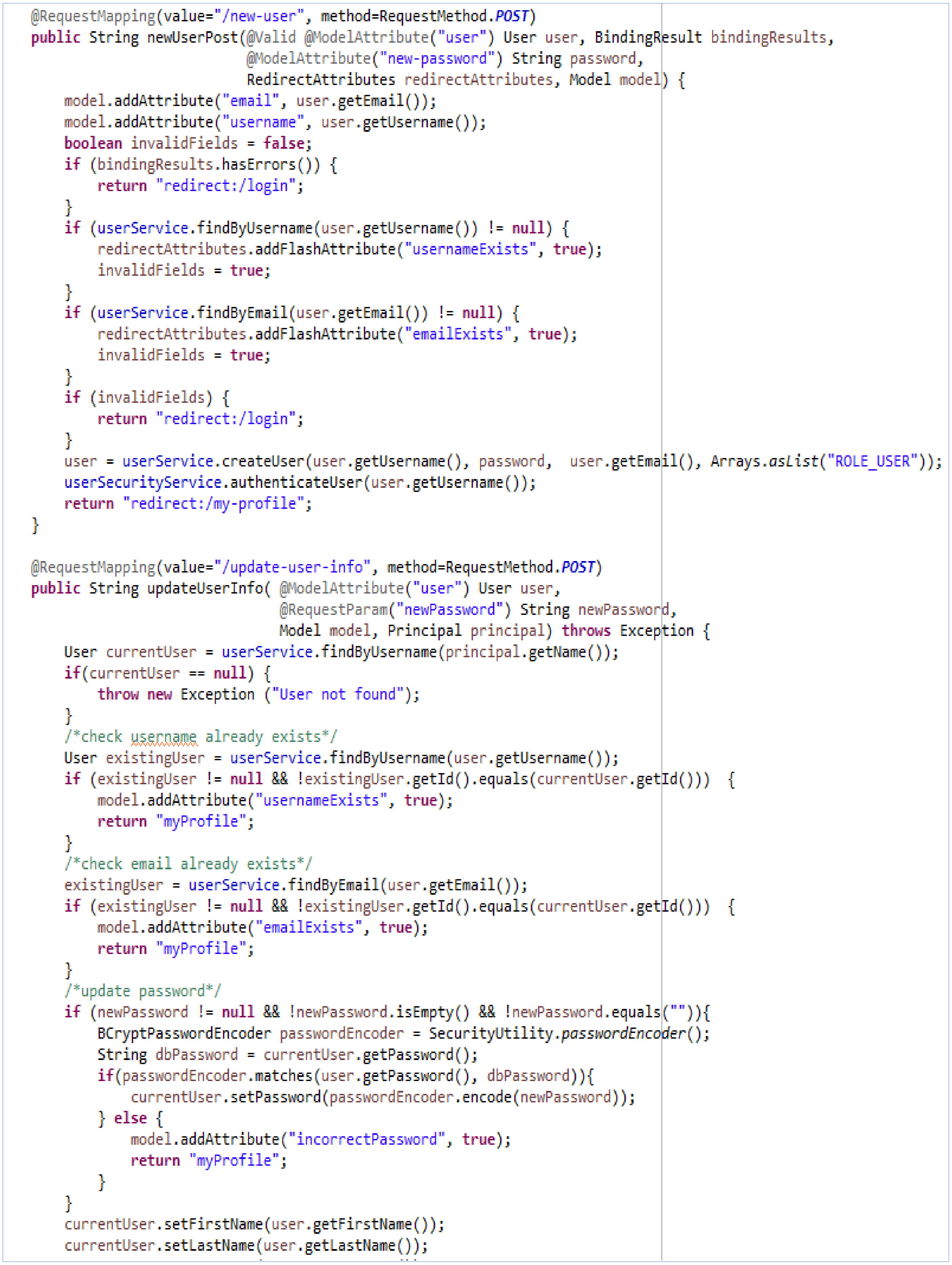
Repository Name.	SportyShoes (1.0).
GitHub Link.	https://github.com/tuskhillare/SportyShoes-1.0-

1.Controller.

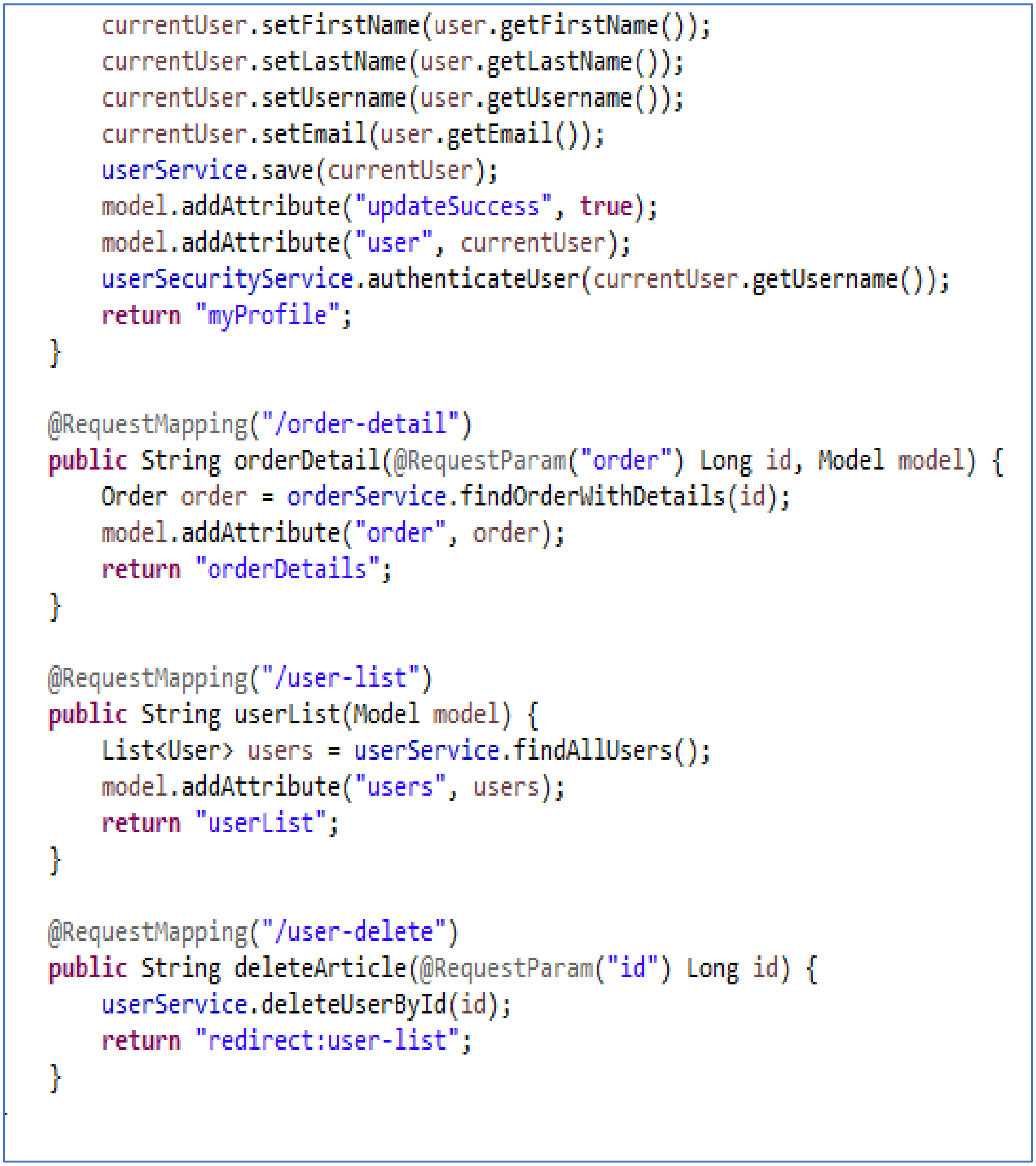
a. AccountController.java



```java
package com.nico.store.store.controller;

import java.security.Principal;

@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";
    }
```

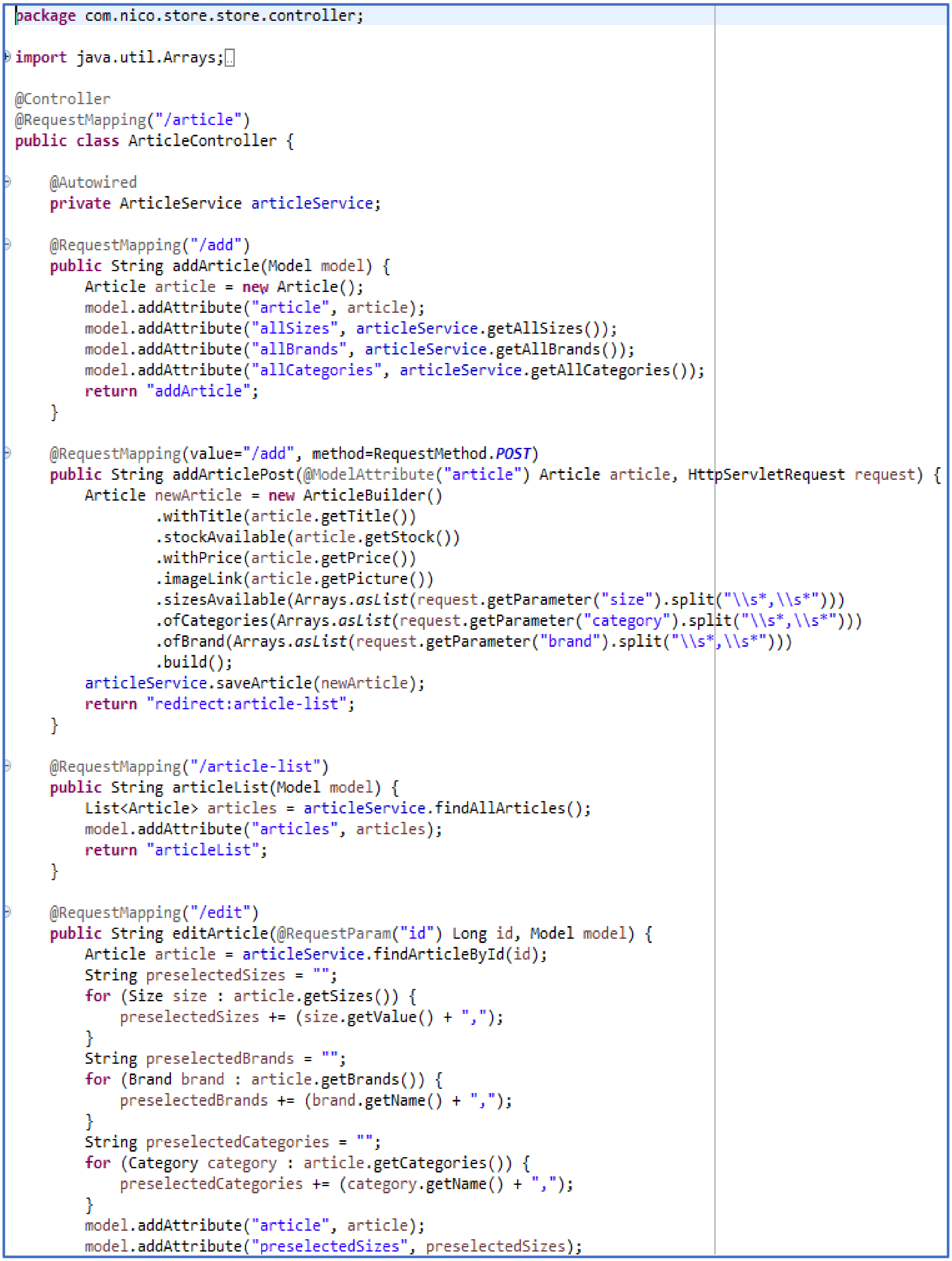


```
@RequestMapping(value="/new-user", method=RequestMethod.POST)
public String newUserPost(@Valid @ModelAttribute("user") User user, BindingResult bindingResults,
                          @ModelAttribute("new-password") String password,
                          RedirectAttributes redirectAttributes, 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());
```

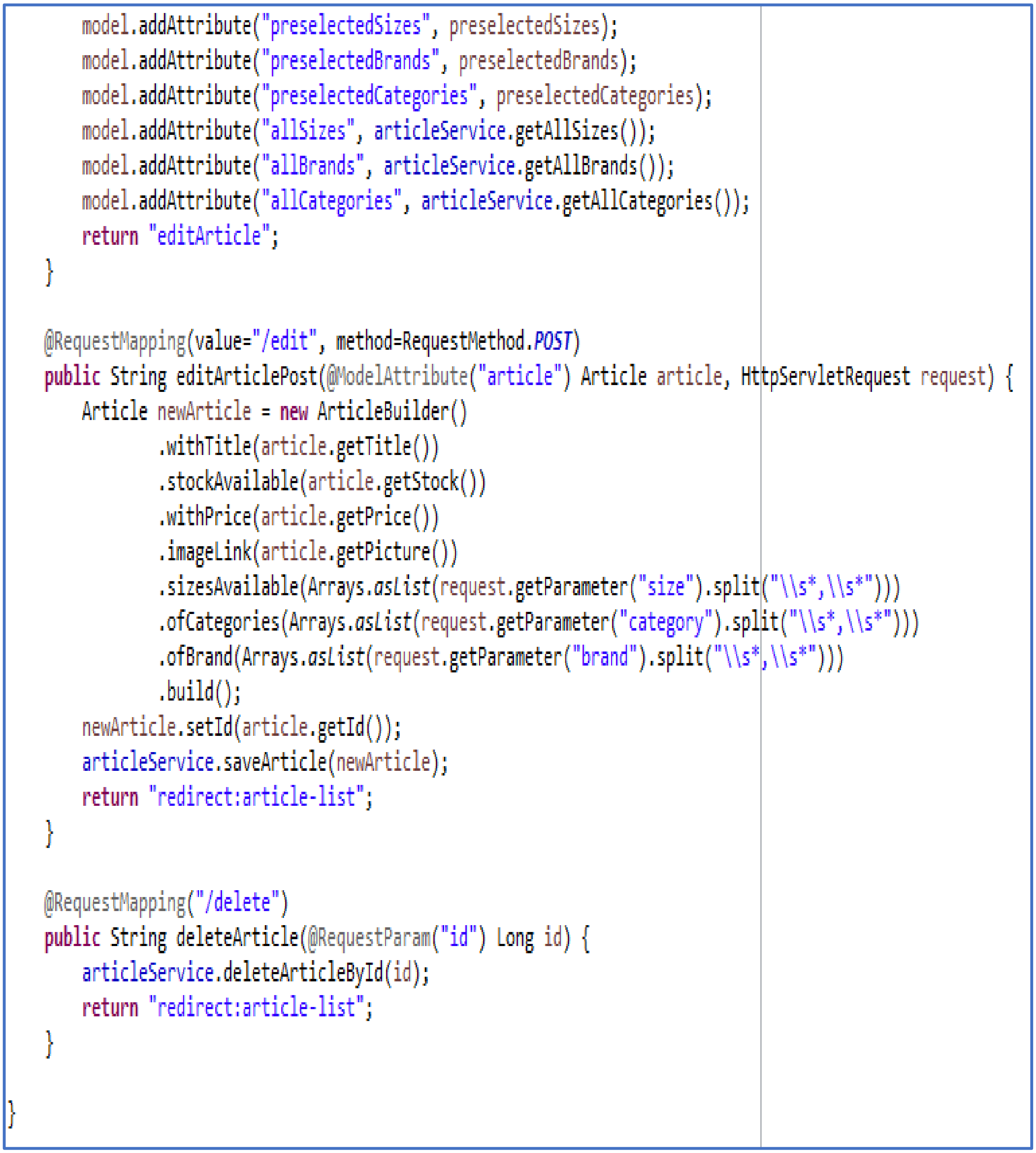


```java
        currentUser.setFirstName(user.getFirstName());
        currentUser.setLastName(user.getLastName());
        currentUser.setUsername(user.getUsername());
        currentUser.setEmail(user.getEmail());
        userService.save(currentUser);
        model.addAttribute("updateSuccess", true);
        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";
    }

    @RequestMapping("/user-list")
    public String userList(Model model) {
        List<User> users = userService.findAllUsers();
        model.addAttribute("users", users);
        return "userList";
    }

    @RequestMapping("/user-delete")
    public String deleteArticle(@RequestParam("id") Long id) {
        userService.deleteUserById(id);
        return "redirect:user-list";
    }
```

b.ArticleController.java

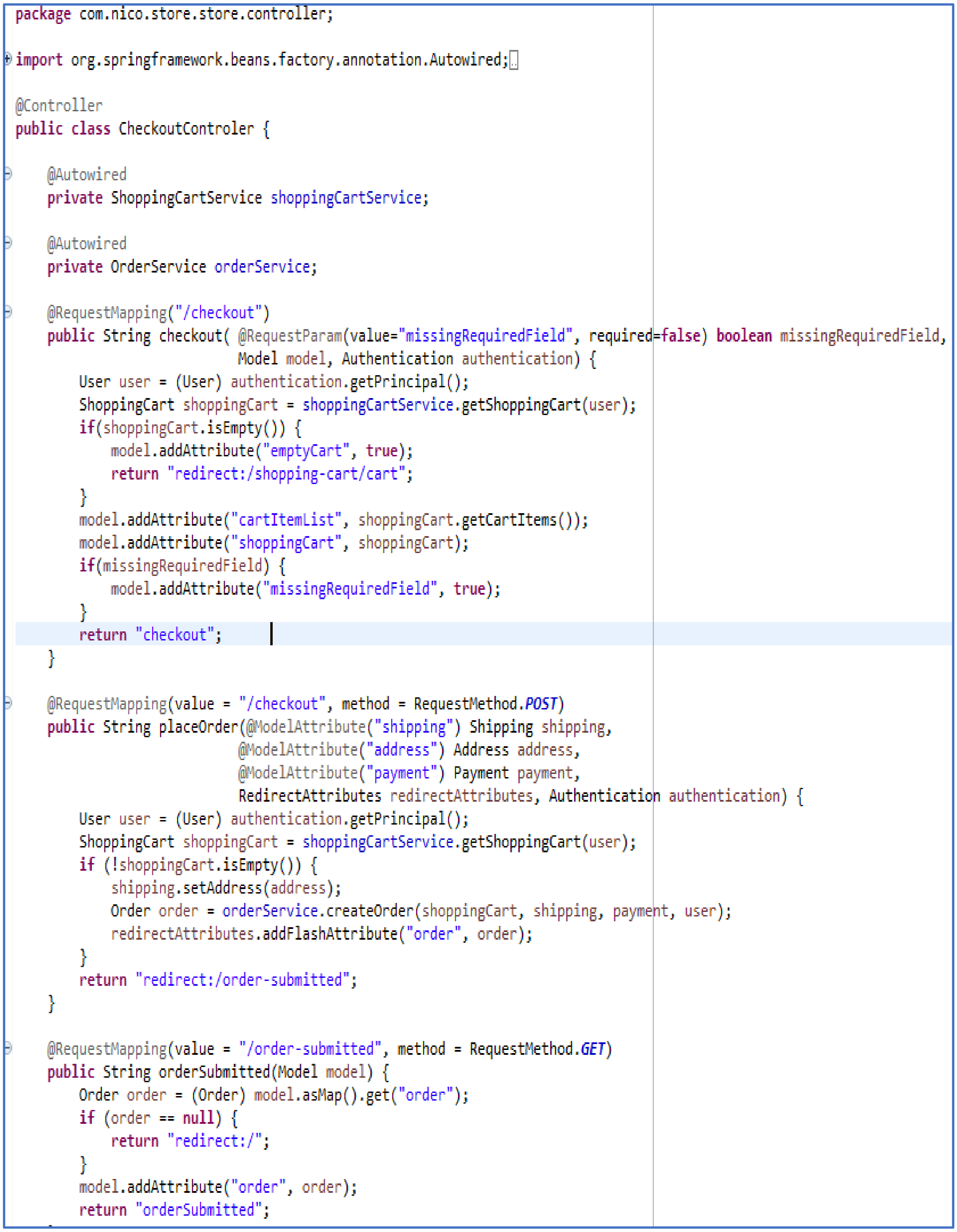


```
package com.nico.store.store.controller;

import java.util.Arrays;

@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);
```



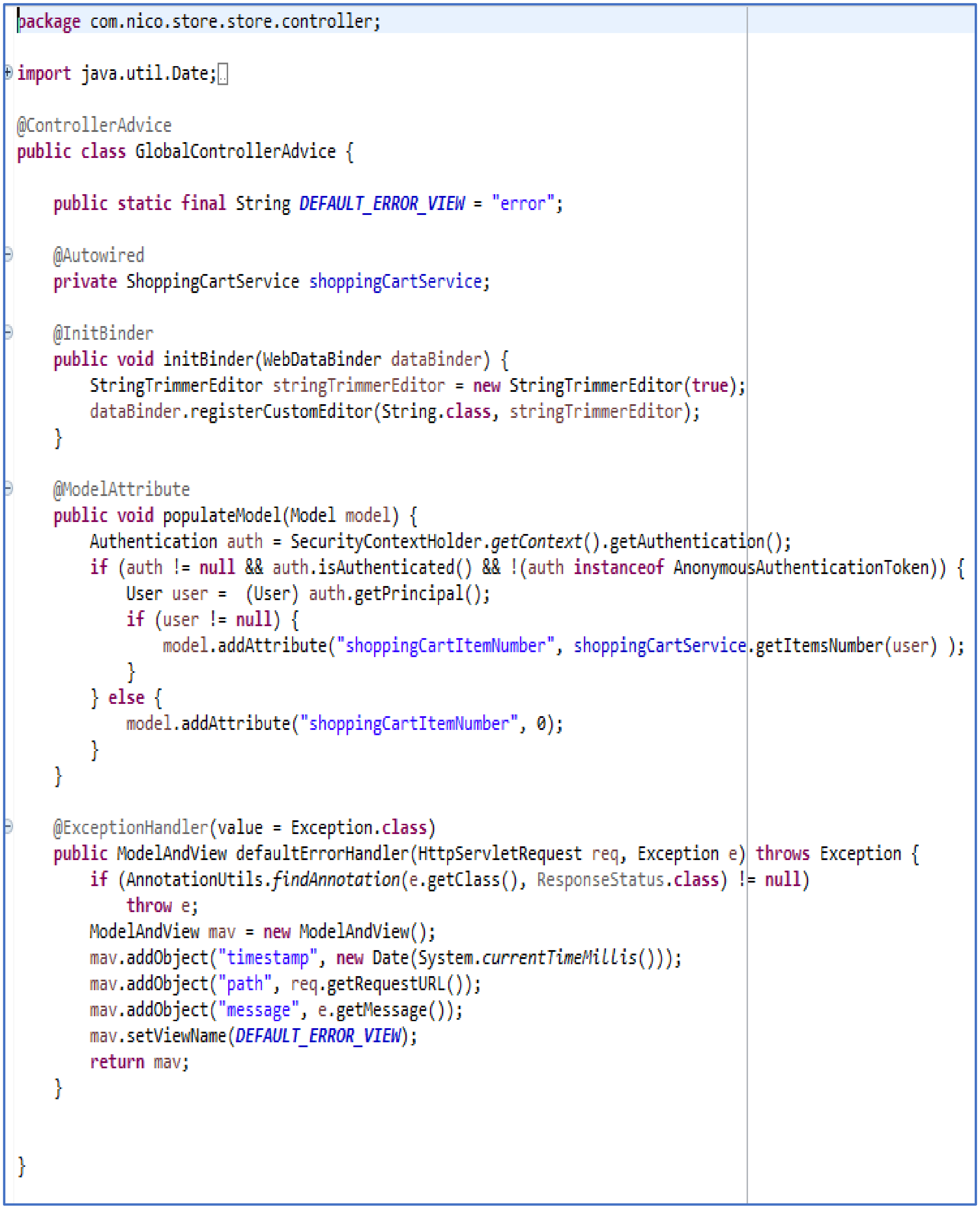
```
        model.addAttribute("preselectedSizes", preselectedSizes);
        model.addAttribute("preselectedBrands", preselectedBrands);
        model.addAttribute("preselectedCategories", preselectedCategories);
        model.addAttribute("allSizes", articleService.getAllSizes());
        model.addAttribute("allBrands", 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";
    }

}
```

c.Checkout Controler.java.



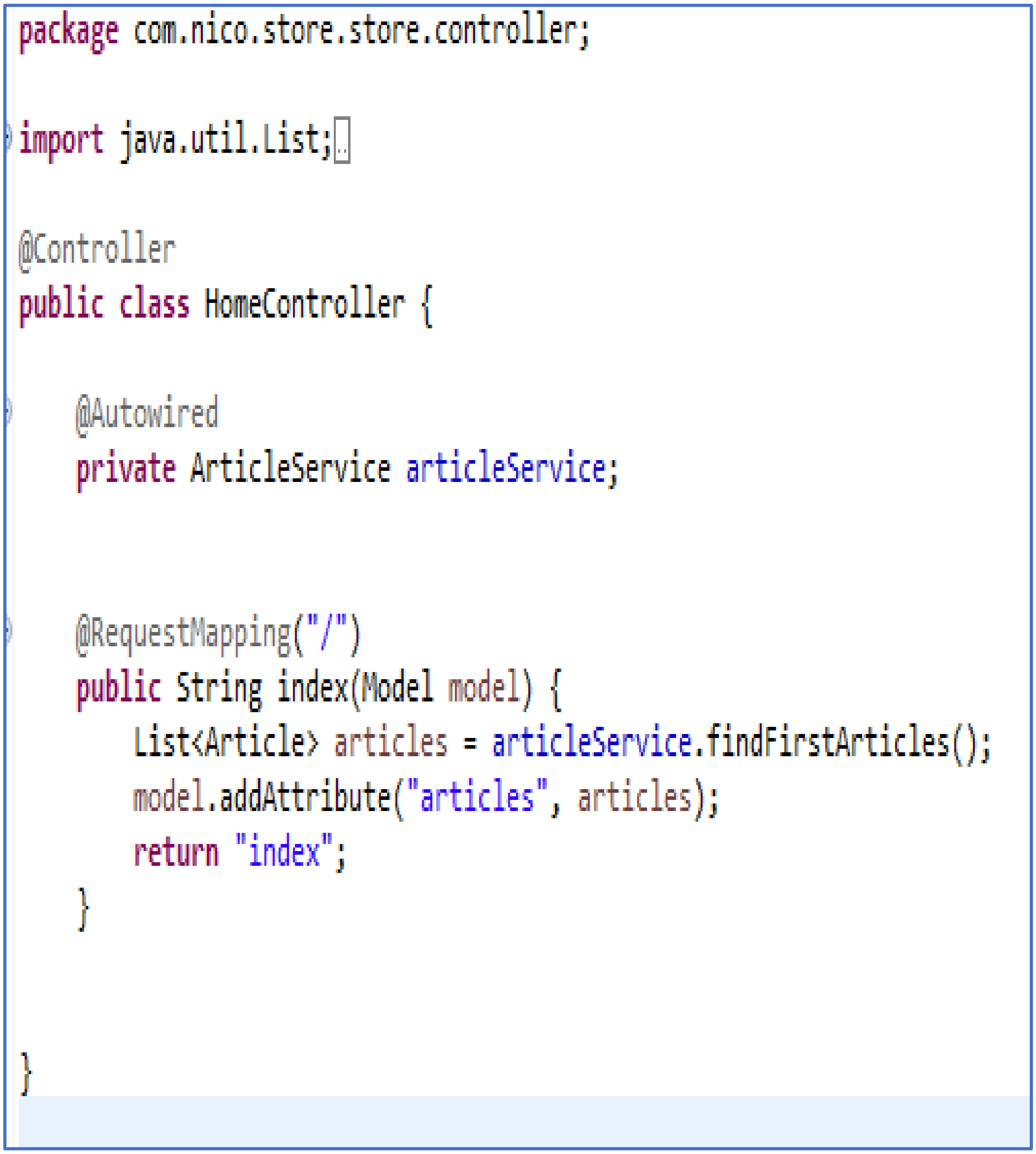
```java
package com.nico.store.store.controller;

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

@Controller
public class CheckoutControler {

    @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";
```

d.GlobalControllerAdvice.java



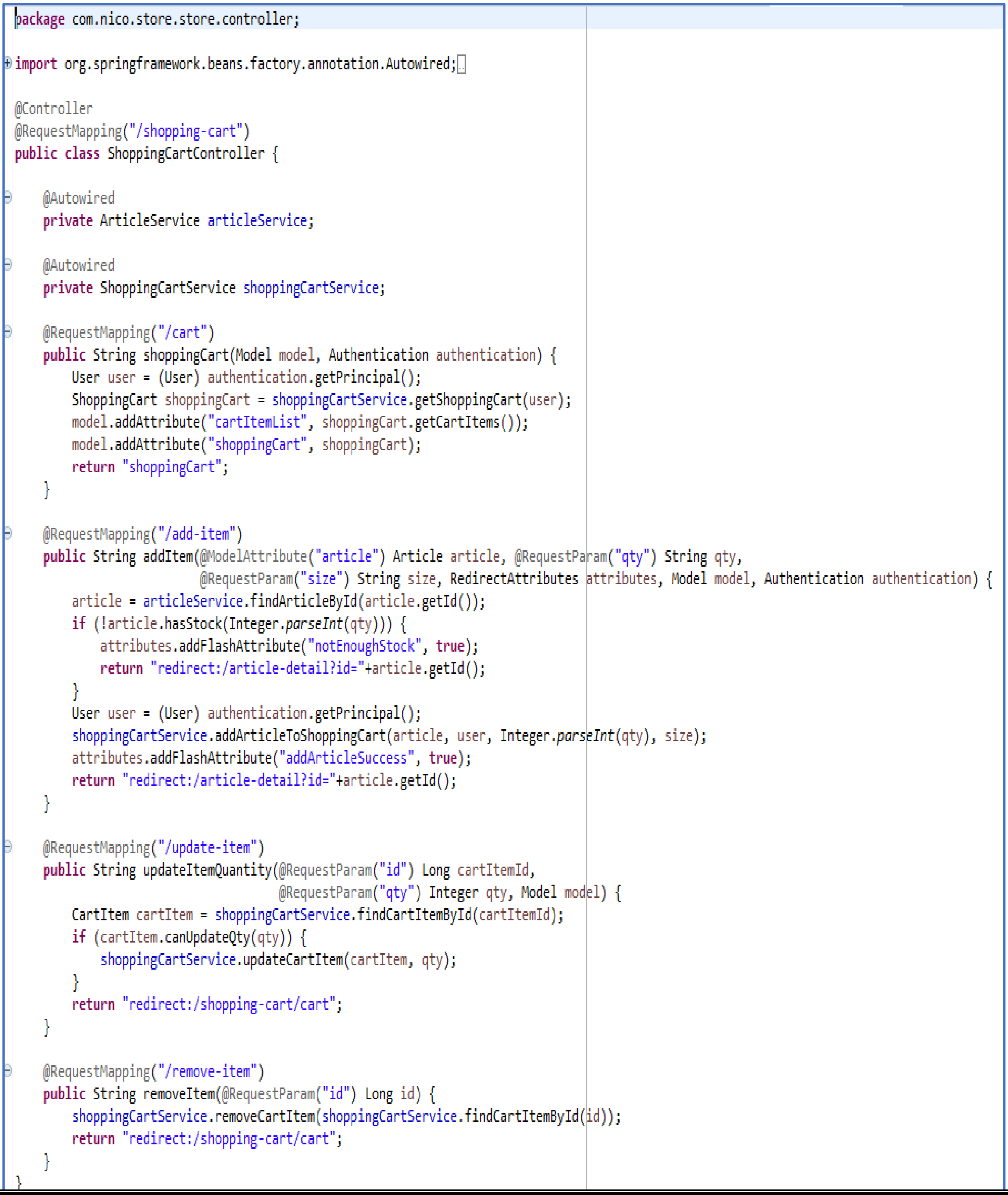
```java
package com.nico.store.store.controller;

import java.util.Date;

@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;
    }


}
```

e. HomeController.java



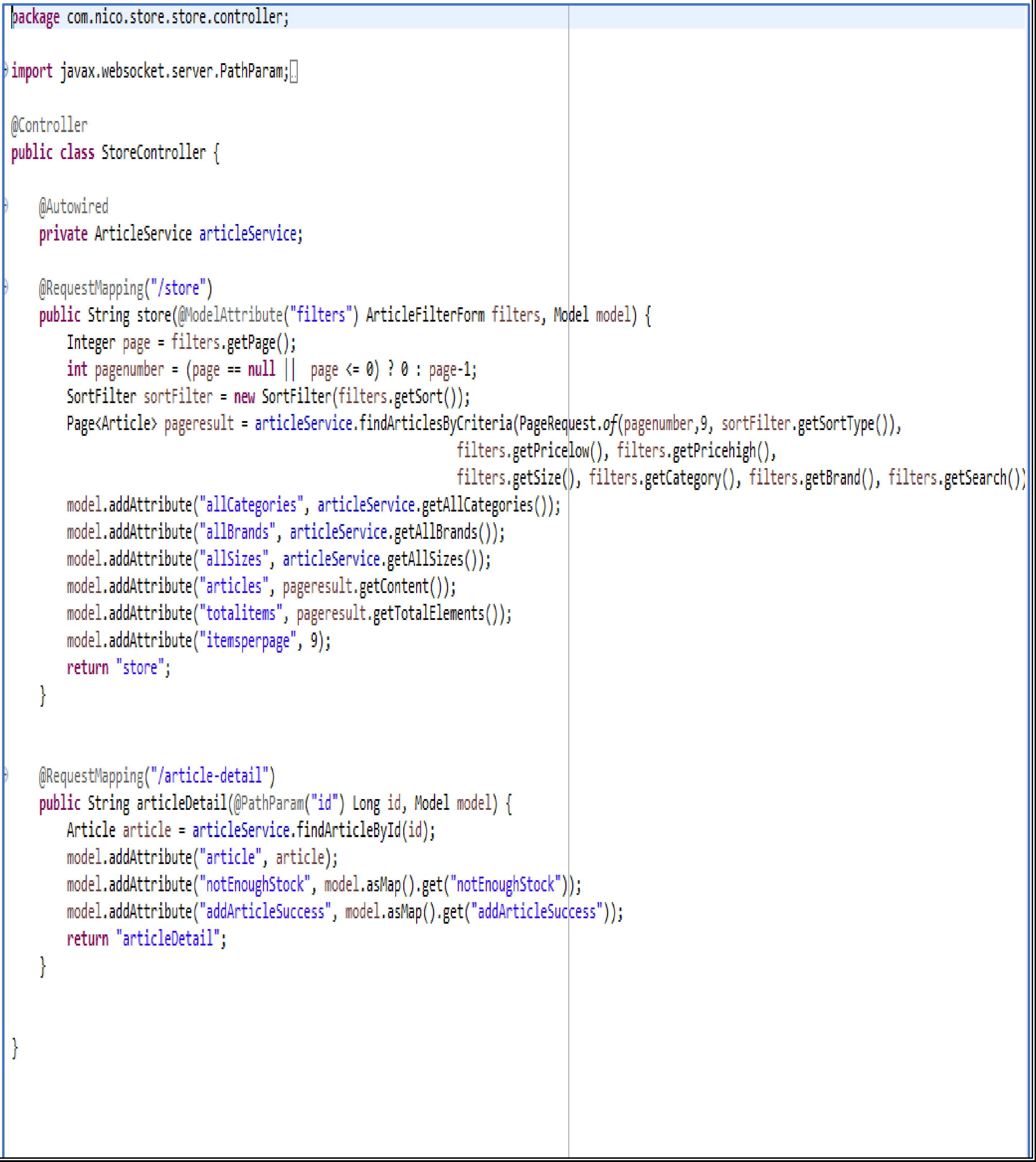
```java
package com.nico.store.store.controller;

import java.util.List;

@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";
    }


}
```

f.ShoppingCartController.java



```java
package com.nico.store.store.controller;

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

@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";
    }
}
```

g.StoreController.java



```java
package com.nico.store.store.controller;

import javax.websocket.server.PathParam;

@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";
    }


}
```

………………………Thank You………………