**Source code-kitchen story**

**Git link**

[**https://github.com/jyothiarunkr/simplilearn-phase4-kitchen-story.git**](https://github.com/jyothiarunkr/simplilearn-phase4-kitchen-story.git)

**Dr. Jyothi N.M.**

**ApplicationContext.java**

package com.touhid.onlineshop.config;

import java.io.IOException;

import java.util.Properties;

import javax.persistence.EntityManagerFactory;

import javax.sql.DataSource;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.env.Environment;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.jpa.JpaTransactionManager;

import org.springframework.orm.jpa.JpaVendorAdapter;

import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

import org.springframework.orm.jpa.vendor.Database;

import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;

import org.springframework.transaction.annotation.EnableTransactionManagement;

import org.springframework.web.multipart.commons.CommonsMultipartResolver;

/\*

\* @EnableTransactionManagement tells Spring that classes with the @Transactional annotation should be wrapped with the Transactional Aspect. With this the @Transactional is now ready to be used.

\*/

@EnableJpaRepositories(basePackages = { "com.touhid.onlineshop.dao" })

@EnableTransactionManagement(proxyTargetClass=true) // EnableTransactionManagement annotation

@Configuration

//@EnableSpringDataWebSupport//enable for pagable

public class ApplicationContext {

@Autowired

private Environment environment;

@Bean

public DataSource dataSource() {

DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(environment.getProperty("jdbc.driverClass"));

dataSource.setUrl(environment.getProperty("jdbc.url"));

dataSource.setUsername(environment.getProperty("jdbc.username"));

dataSource.setPassword(environment.getProperty("jdbc.password"));

return dataSource;

}

@Bean

public JpaTransactionManager transactionManager(EntityManagerFactory entityManagerFactory) {

JpaTransactionManager jpaTransactionManager = new JpaTransactionManager();

jpaTransactionManager.setEntityManagerFactory(entityManagerFactory);

return jpaTransactionManager;

}

@Bean

public JpaVendorAdapter jpaVendorAdapter() {

HibernateJpaVendorAdapter jpaVendorAdapter = new HibernateJpaVendorAdapter();

jpaVendorAdapter.setDatabase(Database.MYSQL);

jpaVendorAdapter.setShowSql(true);

return jpaVendorAdapter;

}

@Bean

public LocalContainerEntityManagerFactoryBean entityManagerFactory() {

LocalContainerEntityManagerFactoryBean entityManager = new LocalContainerEntityManagerFactoryBean();

entityManager.setDataSource(dataSource());

entityManager.setJpaVendorAdapter(jpaVendorAdapter());

entityManager.setPackagesToScan("com.touhid.onlineshop.model");

Properties jpaProperties = new Properties();

jpaProperties.setProperty("hibernate.hbm2ddl.auto", "update");

entityManager.setJpaProperties(jpaProperties);

//System.err.println("CALLED 5-----");

return entityManager;

}

@Bean(name = "multipartResolver")

public CommonsMultipartResolver getResolver() throws IOException {

CommonsMultipartResolver resolver = new CommonsMultipartResolver();

// Set the maximum allowed size (in bytes) for each individual file.

resolver.setMaxUploadSizePerFile(5242880);// 5MB

// You may also set other available properties.

return resolver;

}

}

**SecurityConfiguration.java**

**package com.touhid.onlineshop.config;**

**import javax.sql.DataSource;**

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

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

**import org.springframework.context.annotation.Configuration;**

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

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

**import org.springframework.security.config.annotation.web.builders.WebSecurity;**

**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.crypto.password.PasswordEncoder;**

**import org.springframework.security.web.authentication.SavedRequestAwareAuthenticationSuccessHandler;**

**import org.springframework.security.web.authentication.rememberme.JdbcTokenRepositoryImpl;**

**import org.springframework.security.web.authentication.rememberme.PersistentTokenRepository;**

**import org.springframework.security.web.csrf.CookieCsrfTokenRepository;**

**@Configuration**

**@EnableWebSecurity**

**public class SecurityConfiguration extends WebSecurityConfigurerAdapter{**

**@Autowired**

**private DataSource dataSource;**

**@Autowired**

**public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {**

**auth.jdbcAuthentication().dataSource(dataSource)**

**.authoritiesByUsernameQuery("SELECT username, authorityType FROM authorities WHERE username = ?")**

**.usersByUsernameQuery("SELECT username, password, enabled FROM users WHERE username = ?")**

**.passwordEncoder(passwordEncoder());**

**}**

**@Override**

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

**http.authorizeRequests()**

**.antMatchers("/admin\*\*").access("hasRole('ADMIN')")**

**.antMatchers("/customer/\*\*").access("hasRole('USER')")**

**.and().logout().logoutSuccessUrl("/login?logout")**

**.and().formLogin().loginPage("/login").successHandler(savedRequestAwareAuthenticationSuccessHandler())**

**.loginProcessingUrl( "/j\_spring\_security\_check" )**

**.failureUrl("/login?error")**

**.usernameParameter("username")**

**.passwordParameter("password")**

**.defaultSuccessUrl("/").and()**

**.csrf().csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse()).and()**

**.rememberMe().key("remember-me").rememberMeParameter("remember-me").rememberMeCookieName("remember-me").tokenRepository(persistentTokenRepository()).tokenValiditySeconds(1209600);**

**}**

**@Override**

**public void configure(WebSecurity web) throws Exception {**

**web.ignoring().antMatchers("/resources/\*\*");//ignore resource file,can access**

**}**

**//Remember me bean setting**

**public PersistentTokenRepository persistentTokenRepository(){**

**JdbcTokenRepositoryImpl jdbcTokenRepositoryImpl=new JdbcTokenRepositoryImpl();**

**jdbcTokenRepositoryImpl.setDataSource(dataSource);**

**return jdbcTokenRepositoryImpl;**

**}**

**@Bean**

**public SavedRequestAwareAuthenticationSuccessHandler savedRequestAwareAuthenticationSuccessHandler(){**

**SavedRequestAwareAuthenticationSuccessHandler authenticationSuccessHandler=new SavedRequestAwareAuthenticationSuccessHandler();**

**return authenticationSuccessHandler;**

**}**

**@Bean**

**public PasswordEncoder passwordEncoder() {**

**return new BCryptPasswordEncoder();**

**}**

**}**

**WebApplicationInitializor.java**

**package com.touhid.onlineshop.config;**

**import javax.servlet.ServletContext;**

**import javax.servlet.ServletException;**

**import javax.servlet.ServletRegistration;**

**import org.springframework.web.context.ContextLoaderListener;**

**import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;**

**import org.springframework.web.filter.DelegatingFilterProxy;**

**import org.springframework.web.servlet.DispatcherServlet;**

**public class WebApplicationInitializer implements org.springframework.web.WebApplicationInitializer{**

**@Override**

**public void onStartup(ServletContext servletContext) throws ServletException {**

**AnnotationConfigWebApplicationContext rootContext=new AnnotationConfigWebApplicationContext();**

**rootContext.register(WebConfig.class);**

**servletContext.addListener(new ContextLoaderListener(rootContext));**

**//it will delegate to a bean named springSecurityFilterChain, which is defined implicitly by your use of the Spring Security XML namespace.**

**//HttpServletRequest, and sends it through to the filter springSecurityFilterChain. This filter is a composite of numerous filters that deal with**

**//different parts of the authentication/authorization process**

**DelegatingFilterProxy filterProxy=new DelegatingFilterProxy("springSecurityFilterChain");**

**DispatcherServlet dispatcherServlet=new DispatcherServlet(rootContext);**

**dispatcherServlet.setThrowExceptionIfNoHandlerFound(true);**

**ServletRegistration.Dynamic registration=servletContext.addServlet("dispatcherServlet", dispatcherServlet);**

**//You are specifying here that requests to all URLs (/\*) will go through the DelegatingFilterProxy filter**

**servletContext.addFilter("springSecurityFilterChain", filterProxy).addMappingForUrlPatterns(null, false, "/\*");**

**registration.setLoadOnStartup(1);**

**registration.addMapping("/");**

**}**

**}**

**WebConfig.java**

package com.touhid.onlineshop.config;

import java.util.List;

import java.util.Locale;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

import org.springframework.context.annotation.Import;

import org.springframework.context.annotation.PropertySource;

import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;

import org.springframework.context.support.ResourceBundleMessageSource;

import org.springframework.security.web.bind.support.AuthenticationPrincipalArgumentResolver;

import org.springframework.web.method.support.HandlerMethodArgumentResolver;

import org.springframework.web.servlet.LocaleResolver;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.config.annotation.InterceptorRegistry;

import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;

import org.springframework.web.servlet.i18n.SessionLocaleResolver;

import org.springframework.web.servlet.view.InternalResourceViewResolver;

import com.touhid.onlineshop.validator.ProductFormValidator;

@EnableAspectJAutoProxy

@EnableWebMvc

@Configuration

@ComponentScan(basePackages="com.touhid.onlineshop")

@Import({ApplicationContext.class,SecurityConfiguration.class,WebFlowConfig.class})

@PropertySource("classpath:mysql-connection.properties")

public class WebConfig extends WebMvcConfigurerAdapter{

@Bean

public InternalResourceViewResolver internalResourceViewResolver(){

InternalResourceViewResolver viewResolver=new InternalResourceViewResolver();

viewResolver.setPrefix("/WEB-INF/views/");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

@Override

public void addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/resources/\*\*").addResourceLocations("/WEB-INF/resources/");

}

@Bean

public PropertySourcesPlaceholderConfigurer propertySourcesPlaceholderConfigurer(){

return new PropertySourcesPlaceholderConfigurer();

}

@Override

public void addArgumentResolvers(List<HandlerMethodArgumentResolver> argumentResolvers) {

argumentResolvers.add(resolver());

}

@Bean

public AuthenticationPrincipalArgumentResolver resolver(){

return new AuthenticationPrincipalArgumentResolver();

}

@Bean

public ResourceBundleMessageSource messageSource(){

ResourceBundleMessageSource source=new ResourceBundleMessageSource();

source.setBasename("i18n/messages");

source.setDefaultEncoding("UTF-8");

return source;

}

@Bean

public ProductFormValidator productFormValidator(){

return new ProductFormValidator();

}

@Bean

public LocaleResolver localeResolver(){

SessionLocaleResolver resolver=new SessionLocaleResolver();

resolver.setDefaultLocale(Locale.ENGLISH);

return resolver;

}

@Override

public void addInterceptors(InterceptorRegistry registry) {

LocaleChangeInterceptor localeChangeInterceptor=new LocaleChangeInterceptor();

localeChangeInterceptor.setParamName("lang");

registry.addInterceptor(localeChangeInterceptor);

}

}

**WebFlowConfig.java**

**package com.touhid.onlineshop.config;**

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

**import org.springframework.context.annotation.Configuration;**

**import org.springframework.webflow.config.AbstractFlowConfiguration;**

**import org.springframework.webflow.definition.registry.FlowDefinitionRegistry;**

**import org.springframework.webflow.executor.FlowExecutor;**

**import org.springframework.webflow.mvc.servlet.FlowHandlerAdapter;**

**import org.springframework.webflow.mvc.servlet.FlowHandlerMapping;**

**@Configuration**

**public class WebFlowConfig extends AbstractFlowConfiguration {**

**// A flow registry’s job is to load flow definitions and make them available**

**// to the flow executor**

**@Bean**

**public FlowDefinitionRegistry flowRegistry() {**

**return getFlowDefinitionRegistryBuilder().setBasePath("/WEB-INF/flows")**

**.addFlowLocation("/checkout/checkout-flow.xml", "checkout").build();**

**}**

**/\***

**\* the flow executor is responsible for creating and executing flows,**

**\*/**

**@Bean**

**public FlowExecutor flowExecutor() {**

**return getFlowExecutorBuilder(flowRegistry()).build();**

**}**

**/\***

**\* Whereas the FlowHandlerMapping’s job is to direct flow requests to Spring**

**\* Web Flow, it’s the job of a FlowHandlerAdapter to answer that call. A**

**\* FlowHandlerAdapter is equivalent to a Spring MVC controller in that it**

**\* handles requests coming in for a flow and processes those requests**

**\*/**

**/\***

**\* This handler adapter is the bridge between DispatcherServlet and Spring**

**\* Web Flow. It handles flow requests and manipulates the flow based on**

**\* those requests. Here, it’s wired with a reference to the flow executor to**

**\* execute the flows for which it handles requests**

**\*/**

**@Bean**

**public FlowHandlerAdapter flowHandlerAdapter() {**

**FlowHandlerAdapter handlerAdapter = new FlowHandlerAdapter();**

**handlerAdapter.setFlowExecutor(flowExecutor());**

**return handlerAdapter;**

**}**

**/\***

**\* A DispatcherServlet typically dispatches requests to controllers. But for**

**\* flows, you need a FlowHandlerMapping to help DispatcherServlet know that**

**\* it should send flow requests to Spring Web Flow.**

**\*/**

**@Bean**

**public FlowHandlerMapping flowHandlerMapping() {**

**FlowHandlerMapping handlerMapping = new FlowHandlerMapping();**

**/\***

**\* As you can see, the FlowHandlerMapping is wired with a reference to**

**\* the flow registry so it knows when a request’s URL maps to a flow.**

**\* For example, if you have a flow whose ID is pizza, then**

**\* FlowHandlerMapping will know to map a request to that flow if the**

**\* request’s URL pattern (relative to the application context path) is**

**\* /pizza.**

**\*/**

**handlerMapping.setFlowRegistry(flowRegistry());**

**return handlerMapping;**

**}**

**}**

**CartController.java**

**package com.touhid.onlineshop.controller;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

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

**import org.springframework.security.core.annotation.AuthenticationPrincipal;**

**import org.springframework.security.core.userdetails.User;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.ui.Model;**

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

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

**import org.springframework.web.servlet.HandlerExceptionResolver;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.touhid.onlineshop.exception.CustomError;**

**import com.touhid.onlineshop.model.Customer;**

**import com.touhid.onlineshop.service.CustomerService;**

**@Controller**

**@RequestMapping("/customer/cart")**

**public class CartController implements HandlerExceptionResolver{**

**@Autowired**

**private CustomerService customerService;**

**@RequestMapping**

**public String getCart(@AuthenticationPrincipal User activeUser){**

**Customer customer = customerService.findCustomerByUsername(activeUser.getUsername());**

**long cartId = customer.getCart().getCartId();**

**return "redirect:/customer/cart/" + cartId;**

**}**

**@RequestMapping("/{cartId}")**

**public String getCartRedirect(@PathVariable (value = "cartId") long cartId, Model model){**

**model.addAttribute("cartId", cartId);**

**return "cart";**

**}**

**public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler,**

**Exception ex) {**

**ModelAndView modelAndView=new ModelAndView();**

**CustomError error=new CustomError();**

**error.setMessage("Your request is not valid.Please Enter a valid request.");**

**modelAndView.addObject("customError", error);**

**modelAndView.setViewName("error\_page");**

**return modelAndView;**

**}**

**}**

**CartItemController.java**

**package com.touhid.onlineshop.controller;**

**import java.util.List;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

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

**import org.springframework.http.HttpStatus;**

**import org.springframework.security.core.userdetails.User;**

**import org.springframework.security.web.bind.annotation.AuthenticationPrincipal;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ExceptionHandler;**

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

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

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

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

**import org.springframework.web.bind.annotation.ResponseStatus;**

**import org.springframework.web.servlet.HandlerExceptionResolver;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.touhid.onlineshop.exception.CustomError;**

**import com.touhid.onlineshop.model.Cart;**

**import com.touhid.onlineshop.model.CartItem;**

**import com.touhid.onlineshop.model.Customer;**

**import com.touhid.onlineshop.model.Product;**

**import com.touhid.onlineshop.service.CartItemService;**

**import com.touhid.onlineshop.service.CartService;**

**import com.touhid.onlineshop.service.CustomerService;**

**import com.touhid.onlineshop.service.ProductService;**

**@Controller**

**@RequestMapping("/rest/cart")**

**public class CartItemController implements HandlerExceptionResolver{**

**@Autowired**

**CartService cartService;**

**@Autowired**

**private CustomerService customerService;**

**@Autowired**

**ProductService productService;**

**@Autowired**

**CartItemService cartItemService;**

**//GET CART FOR REST SERVICE**

**@RequestMapping("/{cartId}")**

**public @ResponseBody Cart getCartById(@PathVariable("cartId")int cartId){**

**return cartService.getCartById(cartId);**

**}**

**//ADD PRODUCT IN THE CART**

**@RequestMapping(value="/add/{productId}",method=RequestMethod.PUT)**

**@ResponseStatus(value=HttpStatus.NO\_CONTENT)**

**public void addItem(@PathVariable("productId")int productId,@AuthenticationPrincipal User activeUser){**

**Customer customer=customerService.findCustomerByUsername(activeUser.getUsername());**

**Cart cart=customer.getCart();**

**Product product=productService.getProductById(productId);**

**List<CartItem> cartItems=cart.getCartItems();**

**//IF PRODUCT ALREDAY EXIST IN CART THEN INCRESE ITS QUANTITY**

**for(int i=0;i<cartItems.size();i++){**

**if (product.getProductId()==cartItems.get(i).getProduct().getProductId()) {**

**CartItem cartItem=cartItems.get(i);**

**cartItem.setQuantity(cartItem.getQuantity()+1);**

**cartItem.setTotalPrice(product.getProductPrice()\*cartItem.getQuantity());**

**cartItemService.addCartItem(cartItem);**

**return;**

**}**

**}**

**//IF PRODUCT IS NEW**

**CartItem cartItem=new CartItem();**

**cartItem.setProduct(product);**

**cartItem.setQuantity(1);**

**cartItem.setTotalPrice(product.getProductPrice()\*cartItem.getQuantity());**

**cartItem.setCart(cart);**

**cartItemService.addCartItem(cartItem);**

**}**

**//REMOVE A CARTITEM FROM CART**

**@RequestMapping(value="/remove/{productId}",method=RequestMethod.PUT)**

**@ResponseStatus(value=HttpStatus.OK)**

**public void removeItem(@PathVariable("productId") int productId){**

**System.out.println("--------------hahah----------------");**

**CartItem cartItem=cartItemService.getCartItemByProduct(productId);**

**System.out.println("C:"+cartItem.getTotalPrice());**

**System.out.println(cartItem);**

**cartItemService.deleteCartItem(cartItem);**

**System.out.println("c:hpoihpoifihihitih-----------");**

**}**

**//REMOVE CART/CLEAR CART**

**@RequestMapping(value="/{cartId}",method=RequestMethod.DELETE)**

**@ResponseStatus(value=HttpStatus.OK)**

**public void clearCart(@PathVariable(value="cartId")int cartId){**

**Cart cart=cartService.getCartById(cartId);**

**cartItemService.removeAllCartItems(cart);**

**}**

**//EXCEPTION HANDLING**

**@ExceptionHandler(IllegalArgumentException.class)**

**@ResponseStatus(value=HttpStatus.INTERNAL\_SERVER\_ERROR,reason="")**

**public void handleClientErrors(Exception e){}**

**@ExceptionHandler(Exception.class)**

**@ResponseStatus(value=HttpStatus.BAD\_REQUEST,reason="Illegal request,please verify your payload")**

**public void handleServerErrors(Exception e){}**

**public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler,**

**Exception ex) {**

**ModelAndView modelAndView=new ModelAndView();**

**CustomError error=new CustomError();**

**if (ex instanceof IllegalArgumentException) {**

**error.setMessage("Internal server error");**

**modelAndView.addObject("customError", error);**

**modelAndView.setViewName("error\_page");**

**return modelAndView;**

**}**

**error.setMessage("Your request is not valid.Please Enter a valid request.");**

**modelAndView.addObject("customError", error);**

**modelAndView.setViewName("error\_page");**

**return modelAndView;**

**}**

**}**

**HomeController.java**

**package com.touhid.onlineshop.controller;**

**import java.util.Date;**

**import javax.servlet.http.Cookie;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

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

**import org.springframework.security.core.Authentication;**

**import org.springframework.security.core.annotation.AuthenticationPrincipal;**

**import org.springframework.security.core.context.SecurityContextHolder;**

**import org.springframework.security.core.userdetails.User;**

**import org.springframework.security.crypto.password.PasswordEncoder;**

**import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.ui.Model;**

**import org.springframework.util.StringUtils;**

**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.HandlerExceptionResolver;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.touhid.onlineshop.exception.CustomError;**

**import com.touhid.onlineshop.model.Customer;**

**import com.touhid.onlineshop.model.CustomerContact;**

**import com.touhid.onlineshop.service.CustomerContactService;**

**import com.touhid.onlineshop.service.CustomerService;**

**@Controller**

**public class HomeController implements HandlerExceptionResolver{**

**@Autowired**

**private CustomerService customerService;**

**@Autowired**

**private CustomerContactService customerContactService;**

**@RequestMapping("/")**

**public String homePage(){**

**return "home";**

**}**

**@RequestMapping("/login")**

**public String login(**

**@RequestParam(value="error", required = false)**

**String error,**

**@RequestParam(value="logout", required = false)**

**String logout,**

**Model model){**

**if(error != null){**

**model.addAttribute("error", "Invalid username and password");**

**}**

**if (logout !=null){**

**model.addAttribute("msg", "You have been logged out successfully");**

**}**

**return "login";**

**}**

**@RequestMapping(value="/logout", method = RequestMethod.GET)**

**public String logoutPage (HttpServletRequest request, HttpServletResponse response) {**

**Authentication auth = SecurityContextHolder.getContext().getAuthentication();**

**if (auth != null){**

**new SecurityContextLogoutHandler().logout(request, response, auth);**

**}**

**cancelCookie(request, response);**

**return "redirect:/login?logout";//You can redirect wherever you want, but generally it's a good practice to show login screen again.**

**}**

**void cancelCookie(HttpServletRequest request, HttpServletResponse response)**

**{**

**String cookieName = "remember-me";**

**Cookie cookie = new Cookie(cookieName, null);**

**cookie.setMaxAge(0);**

**cookie.setPath(StringUtils.hasLength(request.getContextPath()) ? request.getContextPath() : "/");**

**response.addCookie(cookie);**

**}**

**@RequestMapping("/about")**

**public String aboutPage(){**

**return "about";**

**}**

**@RequestMapping(value="/contact",method=RequestMethod.GET)**

**public String contact(){**

**return "contact";**

**}**

**@RequestMapping(value="/contact",method=RequestMethod.POST)**

**public String contactPost(@AuthenticationPrincipal User activeUser,@RequestParam("message")String message,Model model){**

**Customer customer = customerService.findCustomerByUsername(activeUser.getUsername());**

**CustomerContact customerContact=new CustomerContact();**

**customerContact.setContactInfo(message);**

**customerContact.setCustomer(customer);**

**customerContact.setDate(new Date());**

**customerContactService.addNewMessage(customerContact);**

**model.addAttribute("msg","Message is sent to the green online shop");**

**return "contact";**

**}**

**public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler,**

**Exception ex) {**

**ModelAndView modelAndView=new ModelAndView();**

**CustomError error=new CustomError();**

**error.setMessage("Your request is not valid.Please Enter a valid request.");**

**modelAndView.addObject("customError", error);**

**modelAndView.setViewName("error\_page");**

**return modelAndView;**

**}**

**}**

**OrerController.java**

**package com.touhid.onlineshop.controller;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

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

**import org.springframework.stereotype.Controller;**

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

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

**import org.springframework.web.servlet.HandlerExceptionResolver;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.touhid.onlineshop.exception.CustomError;**

**import com.touhid.onlineshop.model.Cart;**

**import com.touhid.onlineshop.model.Customer;**

**import com.touhid.onlineshop.model.CustomerOrder;**

**import com.touhid.onlineshop.service.CartService;**

**@Controller**

**public class OrderController implements HandlerExceptionResolver{**

**@Autowired**

**private CartService cartService;**

**@RequestMapping("/order/{cartId}")**

**public String createOrder(@PathVariable("cartId")long cartId){**

**CustomerOrder customerOrder=new CustomerOrder();**

**Cart cart=cartService.getCartById(cartId);**

**customerOrder.setCart(cart);**

**Customer customer=cart.getCustomer();**

**customerOrder.setCustomer(customer);**

**customerOrder.setBillingAddress(customer.getBillingAddress());**

**customerOrder.setShippingAddress(customer.getShippingAddress());**

**//customerOrderService.addCustomerOrder(customerOrder);**

**return "redirect:/checkout?cartId="+cartId;**

**}**

**public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler,**

**Exception ex) {**

**ModelAndView modelAndView=new ModelAndView();**

**CustomError error=new CustomError();**

**error.setMessage("Your request is not valid.Please Enter a valid request.");**

**modelAndView.addObject("customError", error);**

**modelAndView.setViewName("error\_page");**

**return modelAndView;**

**}**

**}**

**ProductController.java**

**package com.touhid.onlineshop.controller;**

**import java.util.ArrayList;**

**import java.util.List;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

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

**import org.springframework.data.domain.Page;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.ui.Model;**

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

**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.HandlerExceptionResolver;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.touhid.onlineshop.exception.CustomError;**

**import com.touhid.onlineshop.model.Product;**

**import com.touhid.onlineshop.service.ProductService;**

**@Controller**

**@RequestMapping("/product")**

**public class ProductController implements HandlerExceptionResolver{**

**@Autowired**

**private ProductService productService;**

**@RequestMapping("/productList/{productCategory}/{pageNumber}")**

**public String getProductByCategory(@PathVariable("pageNumber")int pageNumber,@PathVariable("productCategory")String productCategory,Model model){**

**Page<Product> page=productService.getAllProductByCategory(pageNumber, productCategory);**

**List<Product> products=new ArrayList<>();**

**for (Product product : page) {**

**products.add(product);**

**}**

**int currentPageNumber=page.getNumber()+1;**

**int beginIndex=Math.max(1, currentPageNumber-5);**

**int endIndex=Math.min(beginIndex+10, page.getTotalPages());**

**if (products.size()==0) {**

**model.addAttribute("msg","No product to show");**

**}**

**model.addAttribute("products",products);**

**model.addAttribute("productCategory",productCategory);**

**model.addAttribute("totalPages",page.getTotalPages());**

**model.addAttribute("currentPageNumber",currentPageNumber);**

**model.addAttribute("beginIndex",beginIndex);**

**model.addAttribute("endIndex",endIndex);**

**return "productList";**

**}**

**@RequestMapping(value="/productList/{productCategory}/{pageNumber}",method=RequestMethod.POST)**

**public String getProductBySearch(@RequestParam("searchTerm")String searchTerm,@PathVariable("pageNumber")int pageNumber,@PathVariable("productCategory")String productCategory,Model model){**

**Page<Product> page=productService.getAllProductByBrandOrModel(pageNumber, searchTerm, productCategory);**

**List<Product> products=new ArrayList<>();**

**for (Product product : page) {**

**products.add(product);**

**}**

**if (products.size()==0) {**

**model.addAttribute("msg","No product to show");**

**}**

**int currentPageNumber=page.getNumber()+1;**

**int beginIndex=Math.max(1, currentPageNumber-5);**

**int endIndex=Math.min(beginIndex+10, page.getTotalPages());**

**model.addAttribute("products",products);**

**model.addAttribute("productCategory",productCategory);**

**model.addAttribute("totalPages",page.getTotalPages());**

**model.addAttribute("currentPageNumber",currentPageNumber);**

**model.addAttribute("beginIndex",beginIndex);**

**model.addAttribute("endIndex",endIndex);**

**return "productList";**

**}**

**@RequestMapping("/productList")**

**public String getAllProduct(Model model){**

**List<Product> products=productService.getAllProduct();**

**model.addAttribute("products",products);**

**return "productList";**

**}**

**@RequestMapping("/viewProduct/{productId}")**

**public String ProductDetailPage(@PathVariable("productId")long productId, Model model){**

**Product product=productService.getProductById(productId);**

**model.addAttribute("product",product);**

**model.addAttribute("search","search");**

**return "productDetail";**

**}**

**public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler,**

**Exception ex) {**

**ModelAndView modelAndView=new ModelAndView();**

**CustomError error=new CustomError();**

**error.setMessage("Your request is not valid.Please Enter a valid request.");**

**modelAndView.addObject("customError", error);**

**modelAndView.setViewName("error\_page");**

**return modelAndView;**

**}**

**}**

**RegisterController.java**

**package com.touhid.onlineshop.controller;**

**import java.util.List;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.validation.Valid;**

**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.crypto.password.PasswordEncoder;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.ui.Model;**

**import org.springframework.validation.BindingResult;**

**import org.springframework.web.bind.WebDataBinder;**

**import org.springframework.web.bind.annotation.InitBinder;**

**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.HandlerExceptionResolver;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.touhid.onlineshop.exception.CustomError;**

**import com.touhid.onlineshop.model.Authorities;**

**import com.touhid.onlineshop.model.BillingAddress;**

**import com.touhid.onlineshop.model.Cart;**

**import com.touhid.onlineshop.model.Customer;**

**import com.touhid.onlineshop.model.ShippingAddress;**

**import com.touhid.onlineshop.model.Users;**

**import com.touhid.onlineshop.service.AuthoritiesService;**

**import com.touhid.onlineshop.service.BillingAddressService;**

**import com.touhid.onlineshop.service.CartService;**

**import com.touhid.onlineshop.service.CustomerService;**

**import com.touhid.onlineshop.service.ShippingAddressService;**

**import com.touhid.onlineshop.service.UsersService;**

**import com.touhid.onlineshop.validator.PasswordValidator;**

**@Controller**

**public class RegisterController implements HandlerExceptionResolver{**

**@Autowired**

**private CustomerService customerService;**

**@Autowired**

**private BillingAddressService billingAddressService;**

**@Autowired**

**private ShippingAddressService shippingAddressService;**

**@Autowired**

**private UsersService usersService;**

**@Autowired**

**private AuthoritiesService authoritiesService;**

**@Autowired**

**private CartService cartService;**

**@Autowired**

**private PasswordValidator passwordValidator;**

**@RequestMapping("/register")**

**public String registerCustomer(Model model){**

**Customer customer=new Customer();**

**BillingAddress billingAddress=new BillingAddress();**

**ShippingAddress shippingAddress=new ShippingAddress();**

**customer.setBillingAddress(billingAddress);**

**customer.setShippingAddress(shippingAddress);**

**model.addAttribute("customer",customer);**

**return "registerCustomer";**

**}**

**@RequestMapping(value="/register",method=RequestMethod.POST)**

**public String registerCustomerPost(@Valid@ModelAttribute("customer")Customer customer,BindingResult result, @RequestParam(name="update",required=false) String update,@RequestParam(name="oldUserId",required=false)Long oldUserId ,Model model,HttpServletRequest request){**

**if (result.hasErrors()) {**

**return "registerCustomer";**

**}**

**List<Customer> customerList=customerService.getAllCustomers();**

**for (Customer customer2 : customerList) {**

**if (customer2.getCustomerEmailAddress().equals(customer.getCustomerEmailAddress())) {**

**model.addAttribute("emailMsg","Email already exists");**

**return "registerCustomer";**

**}**

**if (customer2.getUsername().equals(customer.getUsername())) {**

**model.addAttribute("usernameMsg","Username already exists");**

**return "registerCustomer";**

**}**

**}**

**//System.out.println("--------------------------update:"+update+" ------------"+oldUserId);**

**if (update!=null && update.equalsIgnoreCase("update")) {**

**Customer oldCustomer=customerService.getCustomerBycustomerId(oldUserId);**

**customer.getBillingAddress().setBillingAddressId(oldCustomer.getBillingAddress().getBillingAddressId());**

**customer.getShippingAddress().setShippingAddressId(oldCustomer.getShippingAddress().getShippingAddressId());**

**billingAddressService.addBillingAddress(customer.getBillingAddress());**

**shippingAddressService.addShippingAddress(customer.getShippingAddress());**

**String oldUsername=oldCustomer.getUsername();**

**oldCustomer.setBillingAddress(customer.getBillingAddress());**

**oldCustomer.setShippingAddress(customer.getShippingAddress());**

**oldCustomer.setEnabled(true);**

**oldCustomer.setBillingAddress(customer.getBillingAddress());**

**oldCustomer.setShippingAddress(customer.getShippingAddress());**

**oldCustomer.setUsername(customer.getUsername());**

**oldCustomer.setPassword(customer.getPassword());**

**oldCustomer.setCustomerEmailAddress(customer.getCustomerEmailAddress());**

**oldCustomer.setCustomerName(customer.getCustomerName());**

**oldCustomer.setCustometPhoneNumber(customer.getCustometPhoneNumber());**

**customerService.addCustomer(oldCustomer);**

**Users users=usersService.findUserByusername(oldUsername);**

**users.setCustomerId(oldUserId);**

**users.setEnabled(true);**

**users.setPassword(oldCustomer.getPassword());**

**users.setUsername(oldCustomer.getUsername());**

**usersService.addUsers(users);**

**Authorities authorities=authoritiesService.findAuthoritiesByusername(oldUsername);**

**authorities.setAuthorityType("ROLE\_USER");**

**authorities.setUsername(oldCustomer.getUsername());**

**authoritiesService.addAuthorities(authorities);**

**Cart cart=oldCustomer.getCart();**

**cart.setCustomer(oldCustomer);**

**cart.setGrandTotal(0);**

**cartService.addCart(cart);**

**oldCustomer.setCart(cart);**

**customerService.addCustomer(oldCustomer);**

**autoLogin(oldCustomer,request);**

**return "updateCustomerSuccess";**

**}**

**else {**

**billingAddressService.addBillingAddress(customer.getBillingAddress());**

**shippingAddressService.addShippingAddress(customer.getShippingAddress());**

**customer.setEnabled(true);**

**customerService.addCustomer(customer);**

**Users users=new Users();**

**users.setCustomerId(customer.getCustomerId());**

**users.setEnabled(true);**

**users.setPassword(customer.getPassword());**

**users.setUsername(customer.getUsername());**

**usersService.addUsers(users);**

**Authorities authorities=new Authorities();**

**authorities.setAuthorityType("ROLE\_USER");**

**authorities.setUsername(customer.getUsername());**

**authoritiesService.addAuthorities(authorities);**

**Cart cart=new Cart();**

**cart.setCustomer(customer);**

**cart.setGrandTotal(0);**

**cartService.addCart(cart);**

**customer.setCart(cart);**

**customerService.addCustomer(customer);**

**autoLogin(customer,request);**

**}**

**return "registerCustomerSuccess";**

**}**

**private void autoLogin(Customer customer,HttpServletRequest request) {**

**String username=customer.getUsername();**

**String password=customer.getPassword();**

**// generate session if one doesn't exist**

**request.getSession();**

**Authentication authentication=new UsernamePasswordAuthenticationToken(username, password);**

**SecurityContextHolder.getContext().setAuthentication(authentication);**

**}**

**@RequestMapping("/customer/update")**

**public String customerUpdate(){**

**return "updateCustomer";**

**}**

**@RequestMapping(value="/customer/update",method=RequestMethod.POST)**

**public String customerUpdatePost(@RequestParam("username")String username,@RequestParam("password")String password,Model model){**

**Customer customer=customerService.findCustomerByusernameAndpassword(username, password);**

**if (customer==null) {**

**return "updateCustomer";**

**}**

**model.addAttribute("customer",customer);**

**model.addAttribute("update","update");**

**return "registerCustomer";**

**}**

**public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler,**

**Exception ex) {**

**ModelAndView modelAndView=new ModelAndView();**

**CustomError error=new CustomError();**

**error.setMessage("Your request is not valid.Please Enter a valid request.");**

**modelAndView.addObject("customError", error);**

**modelAndView.setViewName("error\_page");**

**return modelAndView;**

**}**

**@InitBinder**

**public void initBinder(WebDataBinder binder) {**

**binder.addValidators(passwordValidator);**

**}**

**}**

**AuthoritiesDao.java**

**package com.touhid.onlineshop.dao;**

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

**import org.springframework.stereotype.Repository;**

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

**import com.touhid.onlineshop.model.Authorities;**

**@Repository**

**public interface AuthoritiesDao extends CrudRepository<Authorities, Long>{**

**Authorities findAuthoritiesByusername(String username);**

**}**

**BillingAddressDao.java**

package com.touhid.onlineshop.dao;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.touhid.onlineshop.model.BillingAddress;

@Repository

public interface BillingAddressDao extends CrudRepository<BillingAddress, Long>{

}

**CartDao.java**

package com.touhid.onlineshop.dao;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.touhid.onlineshop.model.Cart;

@Repository

public interface CartDao extends CrudRepository<Cart, Long> {

Cart findUserBycartId(long cartId);

}

package com.touhid.onlineshop.dao;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.touhid.onlineshop.model.Cart;

@Repository

public interface CartDao extends CrudRepository<Cart, Long> {

Cart findUserBycartId(long cartId);

}