|  |
| --- |
| **Kitchen Story**  **(Project Source Code)** |

**Version History:**

|  |  |
| --- | --- |
| Author | Mayur Bharatkumar Solanki |
| Purpose | Source Code of the application |
| Date | 8th February 2022 |
| Version | 1.0 |

Table of Contents

[ **Project Link** 3](#_Toc90296037)

[ **Folder Structure** 4](#_Toc90296038)

[1. SecurityConfiguration.java 5](#_Toc90296039)

[1. ProductController.java 8](#_Toc90296040)

[2. RegistrationController.java 9](#_Toc90296041)

[1. JwtRequestFilter.java 11](#_Toc90296042)

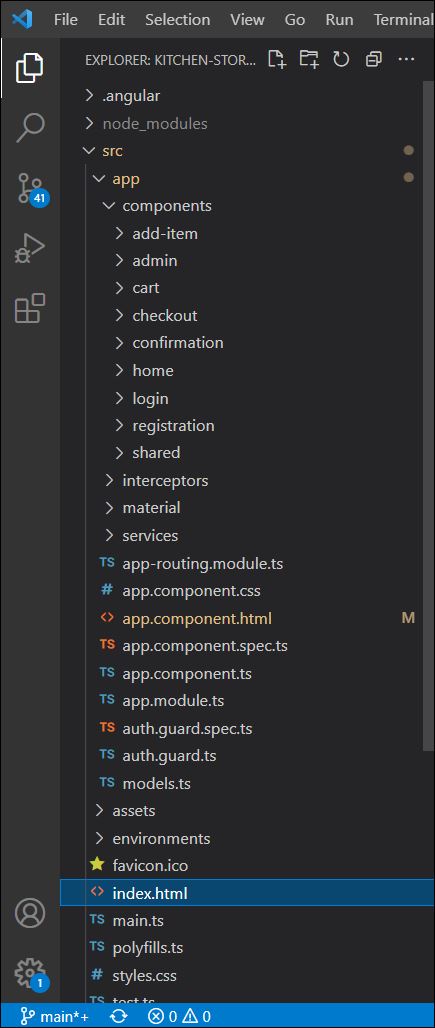
[1. Product.java 13](#_Toc90296043)

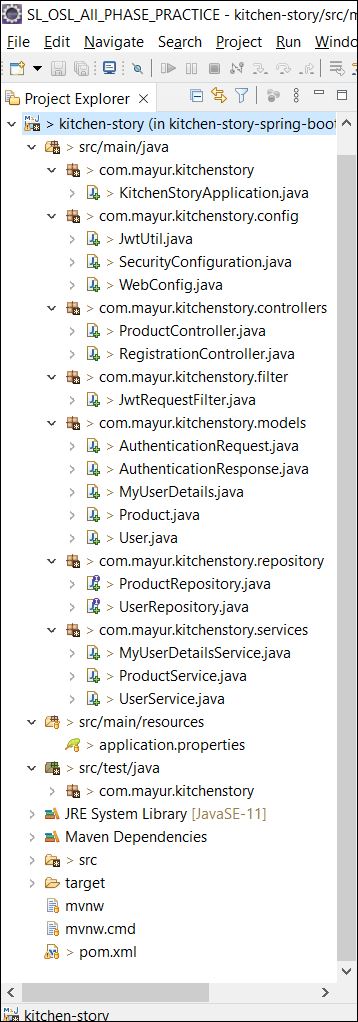
[2. User.java 15](#_Toc90296044)

* **Project Link**

|  |  |
| --- | --- |
| **Repository Name** | **Kitchen Story** |
| GitHub Link | https://github.com/mursky66/SL\_FSD\_PHASE4\_Project-.git |

* **Folder Structure**



****

* **Source Code**

**package com.mayur.kitchenstory**;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class KitchenStoryApplication {

public static void main(String[] args) {

SpringApplication.run(KitchenStoryApplication.class, args);

}

}

**package com.mayur.kitchenstory.config**;

import java.util.Arrays;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

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

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

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

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

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

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

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.CorsConfigurationSource;

import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import com.mayur.kitchenstory.filter.JwtRequestFilter;

@SuppressWarnings("deprecation")

@Configuration

@EnableWebSecurity

public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

@Autowired

UserDetailsService userDetailsService;

@Autowired

private JwtRequestFilter jwtRequestFilter;

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

@Bean

public DaoAuthenticationProvider authenticationProvider()

{

DaoAuthenticationProvider auth = new DaoAuthenticationProvider();

auth.setUserDetailsService(userDetailsService);

auth.setPasswordEncoder(passwordEncoder());

return auth;

}

@Override

@Bean

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

// authentication

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.authenticationProvider(authenticationProvider());

}

// authorization

@Override

protected void configure(HttpSecurity http) throws Exception {

http.cors().and().csrf().disable()

.authorizeRequests().antMatchers("/authenticate", "/registration", "/products").permitAll().

anyRequest().authenticated()

.and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

}

@Bean

CorsConfigurationSource corsConfigurationSource()

{

CorsConfiguration configuration = new CorsConfiguration();

configuration.setAllowedOriginPatterns(Arrays.asList("\*"));

configuration.setAllowedMethods(Arrays.asList("HEAD", "GET", "POST", "PUT", "DELETE", "PATCH"));

configuration.setAllowCredentials(true);

configuration.setAllowedHeaders(Arrays.asList("Authorization", "Cache-Control", "Content-Type"));

final UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

source.registerCorsConfiguration("/\*\*", configuration);

return source;

}

}

**package com.mayur.kitchenstory.config;**

import org.springframework.context.annotation.Configuration;

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

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

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

@Configuration

@EnableWebMvc

public class WebConfig implements WebMvcConfigurer

{

@Override

public void addCorsMappings(CorsRegistry corsRegistry) {

corsRegistry.addMapping("/\*\*")

.allowedOrigins("http://localhost:4200")

.allowedMethods("PUT", "DELETE","GET", "POST")

.maxAge(3600L)

.allowedHeaders("\*");

}

}

**package com.mayur.kitchenstory.controllers**;

import java.util.List;

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

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

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

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

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

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

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

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

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

import com.mayur.kitchenstory.models.Product;

import com.mayur.kitchenstory.services.ProductService;

@RestController

@CrossOrigin(origins = "http://localhost:4200")

@RequestMapping("/products")

public class ProductController

{

@Autowired

private ProductService service;

@PostMapping

public String addProduct(@RequestBody Product product)

{

service.addProduct(product);

return "Product add success.";

}

@GetMapping

public List<Product> getProducts()

{

List<Product> productlist = service.getAllProducts();

return productlist;

}

@DeleteMapping("/delete/{prodId}")

public String deleteProduct(@PathVariable("prodId") String prodId) {

service.deleteProductById(prodId);

return "Product delete success.";

}

}

**package com.mayur.kitchenstory.controllers;**

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.BadCredentialsException;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

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

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

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

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

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

import com.mayur.kitchenstory.config.JwtUtil;

import com.mayur.kitchenstory.models.AuthenticationRequest;

import com.mayur.kitchenstory.models.AuthenticationResponse;

import com.mayur.kitchenstory.models.User;

import com.mayur.kitchenstory.services.MyUserDetailsService;

import com.mayur.kitchenstory.services.UserService;

@RestController

@CrossOrigin(origins = "http://localhost:4200")

public class RegistrationController {

@Autowired

private UserService service;

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtUtil jwtTokenUtil;

@Autowired

private MyUserDetailsService userDetailsService;

@PostMapping("/registration")

public void registerUserAccount(@RequestBody User user)

{

user.setRoles("USER");

service.registerUser(user);

}

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

public ResponseEntity<?> createAuthenticationToken(@RequestBody AuthenticationRequest authenticationRequest) throws Exception {

try {

authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(authenticationRequest.getEmail(), authenticationRequest.getPassword())

);

}

catch (BadCredentialsException e) {

throw new Exception("Incorrect username or password", e);

}

final UserDetails userDetails = userDetailsService

.loadUserByUsername(authenticationRequest.getEmail());

final String jwt = jwtTokenUtil.generateToken(userDetails);

return ResponseEntity.ok(new AuthenticationResponse(jwt));

}

}

**package com.mayur.kitchenstory.filter;**

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.mayur.kitchenstory.config.JwtUtil;

import com.mayur.kitchenstory.services.MyUserDetailsService;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@Component

public class JwtRequestFilter extends OncePerRequestFilter {

@Autowired

private MyUserDetailsService userDetailsService;

@Autowired

private JwtUtil jwtUtil;

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)

throws ServletException, IOException {

final String authorizationHeader = request.getHeader("Authorization");

String username = null;

String jwt = null;

if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {

jwt = authorizationHeader.substring(7);

username = jwtUtil.extractUsername(jwt);

}

if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

UserDetails userDetails = this.userDetailsService.loadUserByUsername(username);

if (jwtUtil.validateToken(jwt, userDetails)) {

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(

userDetails, null, userDetails.getAuthorities());

usernamePasswordAuthenticationToken

.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

}

}

chain.doFilter(request, response);

}

}

package com.mayur.kitchenstory.models;

**import** java.io.Serializable;

**public** **class** AuthenticationRequest **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 4662341638943225147L;

**private** String email;

**private** String password;

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

//need default constructor for JSON Parsing

**public** AuthenticationRequest() { }

**public** AuthenticationRequest(String email, String password) {

**super**();

**this**.email = email;

**this**.password = password;

}

}

package com.mayur.kitchenstory.models;

**import** java.io.Serializable;

**public** **class** AuthenticationResponse **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 7646727820952460100L;

**private** **final** String jwt;

**public** AuthenticationResponse(String jwt) {

**this**.jwt = jwt;

}

**public** String getJwt() {

**return** jwt;

}

}

**package com.mayur.kitchenstory.models;**

import java.util.Arrays;

import java.util.Collection;

import java.util.List;

import java.util.stream.Collectors;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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

public class MyUserDetails implements UserDetails {

private static final long serialVersionUID = 1L;

private String email;

private String password;

private List<GrantedAuthority> authorities;

public MyUserDetails() {};

public MyUserDetails(User user) {

this.email = user.getEmail();

this.password = user.getPassword();

this.authorities = Arrays.stream(user.getRoles().split(","))

.map(SimpleGrantedAuthority::new)

.collect(Collectors.toList());

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

return authorities;

}

@Override

public String getPassword() {

return password;

}

@Override

public String getUsername() {

return 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;

}

}

**package com.mayur.kitchenstory.models;**

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "products")

public class Product {

@Id

private String id;

private String productName;

private double price;

private String description;

private String image;

public Product() {}

public Product(String productName, double price, String description, String image) {

this.productName = productName;

this.price = price;

this.description = description;

this.image = image;

}

public String getImage() {

return image;

}

public void setImage(String image) {

this.image = image;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

}

**package com.mayur.kitchenstory.models;**

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.index.Indexed;

import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "users")

public class User {

@Id

private String id;

private String firstName;

private String lastName;

@Indexed(unique = true)

private String email;

private String password;

private String roles;

public User() {};

public User(String firstName, String lastName, String email, String password) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.password = password;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

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 String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getRoles() {

return roles;

}

public void setRoles(String roles) {

this.roles = roles;

}

}

**package com.mayur.kitchenstory.repository;**

import org.springframework.data.mongodb.repository.MongoRepository;

import org.springframework.stereotype.Repository;

import com.mayur.kitchenstory.models.Product;

@Repository

public interface ProductRepository extends MongoRepository<Product, String> {

}

**package com.mayur.kitchenstory.repository;**

import java.util.Optional;

import org.springframework.data.mongodb.repository.MongoRepository;

import org.springframework.stereotype.Repository;

import com.mayur.kitchenstory.models.User;

@Repository

public interface UserRepository extends MongoRepository<User, String> {

Optional<User> findByEmail(String email);

}

**package com.mayur.kitchenstory.services**;

import java.util.Optional;

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

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.mayur.kitchenstory.models.MyUserDetails;

import com.mayur.kitchenstory.models.User;

import com.mayur.kitchenstory.repository.UserRepository;

@Service

public class MyUserDetailsService implements UserDetailsService

{

@Autowired

private UserRepository repository;

@Override

public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

Optional<User> user = repository.findByEmail(email);

user.orElseThrow(() -> new UsernameNotFoundException("Not found: "+email));

UserDetails details = user.map(MyUserDetails::new).get();

return details;

}

public String getLoggedInUsername()

{

String username;

Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {

username = ((UserDetails)principal).getUsername();

} else {

username = principal.toString();

}

return username;

}

}

**package com.mayur.kitchenstory.services;**

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.mayur.kitchenstory.models.Product;

import com.mayur.kitchenstory.repository.ProductRepository;

@Service

public class ProductService {

@Autowired

private ProductRepository repository;

public void addProduct(Product product)

{

repository.save(product);

}

public List<Product> getAllProducts()

{

return repository.findAll();

}

public Optional<Product> getProductById(String id)

{

return repository.findById(id);

}

public void deleteProductById(String id)

{

repository.deleteById(id);

}

}

**package com.mayur.kitchenstory.services;**

import java.util.List;

import java.util.Optional;

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

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

import org.springframework.stereotype.Service;

import com.mayur.kitchenstory.models.User;

import com.mayur.kitchenstory.repository.UserRepository;

@Service

public class UserService {

@Autowired

private UserRepository repository;

@Autowired

PasswordEncoder passwordEncoder;

public void registerUser(User user)

{

passwordEncoder.encode(user.getPassword());

repository.save(user);

}

public Optional<User> findUserByEmail(String email)

{

return repository.findByEmail(email);

}

public List<User> getAllRegisteredUser()

{

return repository.findAll();

}

}

**package com.mayur.kitchenstory**;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class KitchenStoryApplicationTests {

@Test

void contextLoads() {

}

}