**Spring REST using Spring Boot 3**

Create authentication service that returns JWT

**SpringSecurityJwtApplication.java**

package com.cognizant.springsecurity;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringSecurityJwtApplication {

public static void main(String[] args) {

SpringApplication.run(SpringSecurityJwtApplication.class, args);

}

}

**2. SecurityConfig.java**

package com.cognizant.springsecurity.config;

import com.cognizant.springsecurity.filter.JwtFilter;

import com.cognizant.springsecurity.service.CustomUserDetailsService;

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.config.annotation.authentication.configuration.AuthenticationConfiguration;

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

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

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

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

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

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

import org.springframework.security.web.SecurityFilterChain;

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

@Configuration

@EnableMethodSecurity

public class SecurityConfig {

@Autowired

private JwtFilter jwtFilter;

@Autowired

private CustomUserDetailsService userDetailsService;

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

return http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/", "/public", "/authenticate").permitAll()

.anyRequest().authenticated()

)

.sessionManagement(sess -> sess.sessionCreationPolicy(SessionCreationPolicy.STATELESS))

.authenticationProvider(authProvider())

.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class)

.build();

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config)

throws Exception {

return config.getAuthenticationManager();

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

@Bean

public DaoAuthenticationProvider authProvider() {

DaoAuthenticationProvider provider = new DaoAuthenticationProvider();

provider.setUserDetailsService(userDetailsService);

provider.setPasswordEncoder(passwordEncoder());

return provider;

}

}

**3. CustomUserDetailsService.java**

package com.cognizant.springsecurity.service;

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

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 java.util.ArrayList;

@Service

public class CustomUserDetailsService implements UserDetailsService {

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

if ("admin".equals(username)) {

return new User("admin", "admin123", new ArrayList<>());

} else {

throw new UsernameNotFoundException("User not found with username: " + username);

}

}

}

**4. TestController.java**

package com.cognizant.springsecurity.controller;

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

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

@RestController

public class TestController {

@GetMapping("/")

public String welcome() {

return "Welcome to JWT secured app!";

}

@GetMapping("/public")

public String publicApi() {

return "This is a public endpoint";

}

@GetMapping("/hello")

public String hello() {

return "Hello, this is a secured endpoint!";

}

}

**5. JwtFilter.java (Dummy for now)**

package com.cognizant.springsecurity.filter;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component

public class JwtFilter extends OncePerRequestFilter {

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain filterChain)

throws ServletException, IOException {

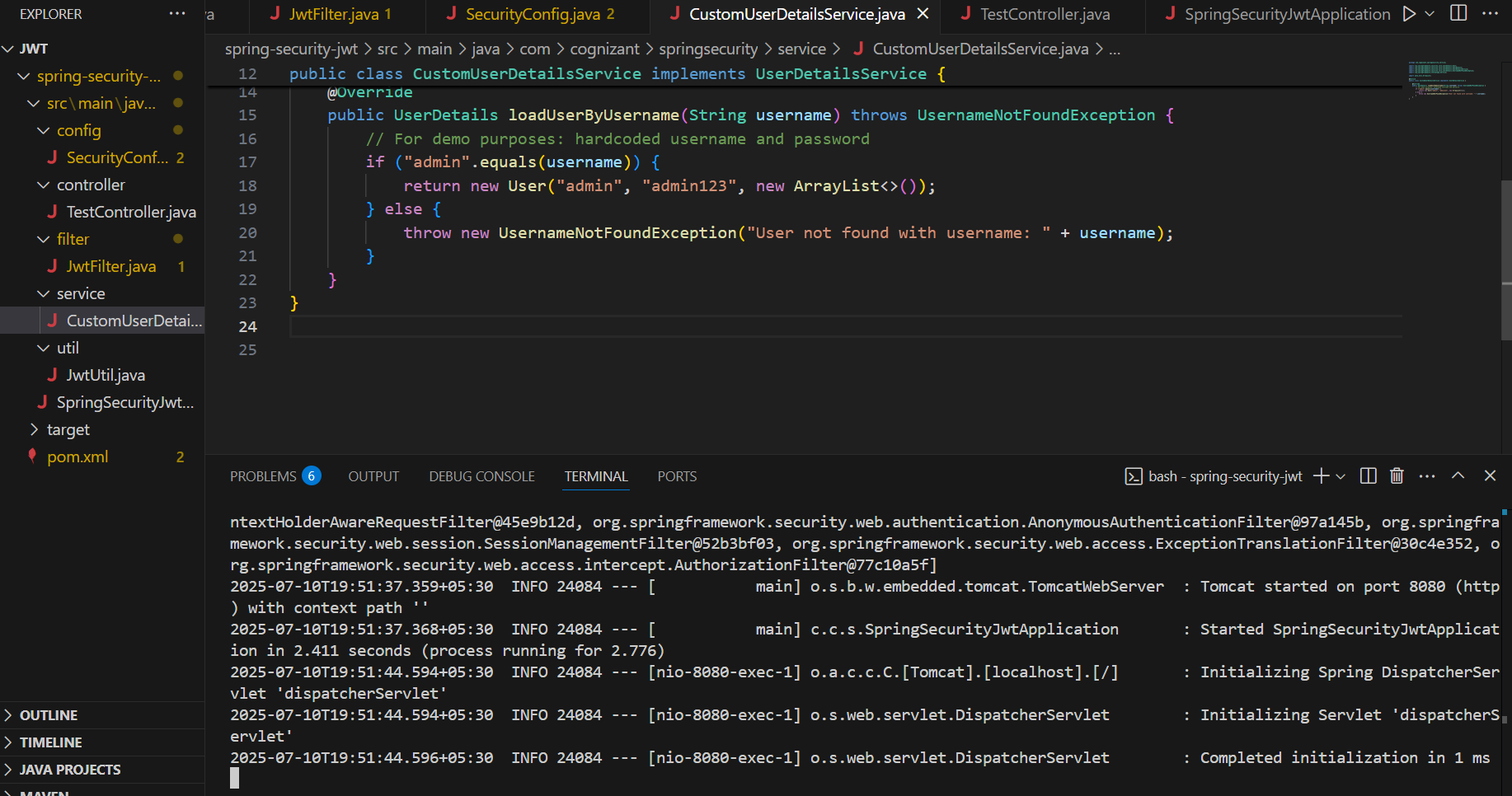
// Dummy filter: just forward all requests

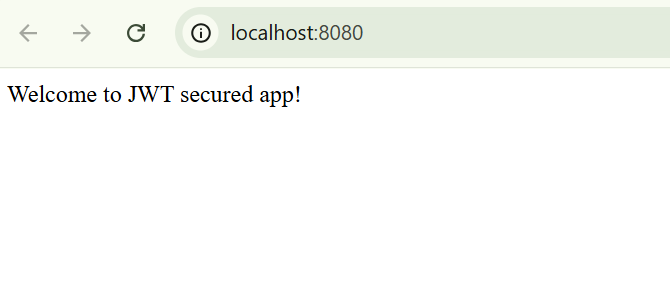
filterChain.doFilter(request, response);

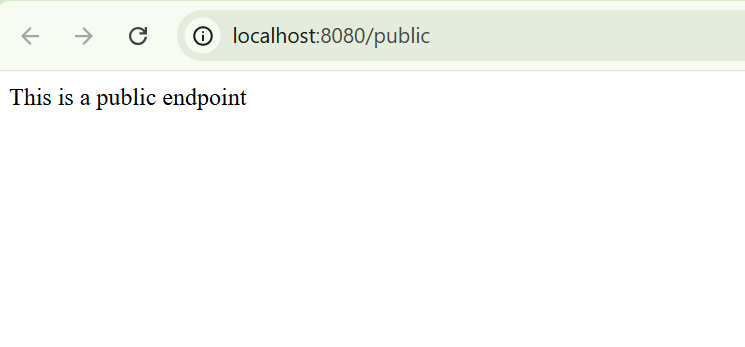
}

}

**Output:**

****

****

****