package com.simplilearn.capstone2.configuration;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Lazy;

import org.springframework.http.HttpHeaders;

import org.springframework.security.authentication.AuthenticationManager;

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

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

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

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

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

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

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

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

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

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

import org.springframework.stereotype.Component;

import com.simplilearn.capstone2.service.JwtService;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

@Component

public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {

@Autowired

private JwtAuthenticationEntryPoint jwtauthenticationentrypoint;

//@Lazy

@Autowired

private JwtRequestFilter jwtrequestfilter;

//@Lazy

@Autowired

private UserDetailsService jwtservice;

@Bean

@Override

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Override

protected void configure(HttpSecurity httpsecurity) throws Exception {

httpsecurity.cors();

httpsecurity.csrf().disable().authorizeRequests()

.antMatchers("/authenticate","/registerNewUser","/movieticket/showall","getTicketById/{ticketId}","/getTicketDetails/{isSingleTicketCheckout},{ticketId}")

.permitAll().antMatchers(HttpHeaders.ALLOW)

.permitAll().anyRequest().authenticated()

.and()

.exceptionHandling()

.authenticationEntryPoint(jwtauthenticationentrypoint).and().sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS);

httpsecurity.addFilterBefore(jwtrequestfilter, UsernamePasswordAuthenticationFilter.class);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder authenticatnManagerBuilder) throws Exception {

authenticatnManagerBuilder.userDetailsService(jwtservice).passwordEncoder(passwordEncoder());

}

}

/\*

\* All these annotations will take care of role based authentication like

\* specifically for admin n for user dont authenticate end point specified

\* inside antmatchers here-- .authorizeRequests().antMatchers("").permitAll()

\*

\* .sessionCreationPolicy(SessionCreationPolicy.STATELESS)-->for user security purpose this

\* will not store the user password in seesion

\*\*/