**Create authentication controller and configure it in SecurityConfig**

package com.example.jwtDemo.controller;

import java.sql.Date;

import java.util.Base64;

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

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

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

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@RestController

public class AuthenticationController {

private final String secretKey = "A123vasanthne3982ui982hud92udu2h9ubeuh39uh89fh3f8y3by3bd8uwy7b8273gd83bd7823hdy8bd873be3g8y3ybd83g3yd83db8ye3h8hd7behhjgf";

@GetMapping("/authenticate")

public String authenticate(@RequestHeader("Authorization") String authHeader){

if (authHeader != null && authHeader.startsWith("Basic ")) {

String base64Credentials = authHeader.substring("Basic ".length());

byte[] decoded = Base64.getDecoder().decode(base64Credentials);

String[] credentials = new String(decoded).split(":", 2);

String username = credentials[0];

String password = credentials[1];

if ("user".equals(username) && "pwd".equals(password)) {

String token = Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 10 \* 60 \* 1000)) // 10 minutes

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

return token;

}

}

return "unauthorized";

}

}

package com.example.jwtDemo.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Bean;

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

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

import org.springframework.security.web.SecurityFilterChain;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

.and()

.httpBasic();

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

return new InMemoryUserDetailsManager(

User.withUsername("user")

.password("pwd")

.roles("USER")

.build()

);

}

@Bean

public static NoOpPasswordEncoder passwordEncoder() {

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

}

}

