**Create authentication service that returns JWT**

FILE:SecurityConfig.java

package com.cognizant.spring\_learn.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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.configuration.EnableWebSecurity;

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

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

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

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

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(SecurityConfig.class);

*@Override*

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")

.and()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");

}

*@Bean*

public PasswordEncoder passwordEncoder() {

***LOGGER***.info("Start");

return new BCryptPasswordEncoder();

}

*@Override*

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable()

.httpBasic()

.and()

.authorizeRequests()

.antMatchers("/countries").hasRole("USER")

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated();

}

}

FILE:AuthenticationController.java

package com.cognizant.spring\_learn.controller;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

*@RestController*

public class AuthenticationController {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(AuthenticationController.class);

private String getUser(String authHeader) {

***LOGGER***.debug("Start: getUser()");

***LOGGER***.debug("Raw Authorization header: {}", authHeader);

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

***LOGGER***.debug("Encoded credentials: {}", encodedCredentials);

byte[] decodedBytes = Base64.*getDecoder*().decode(encodedCredentials);

String decodedCredentials = new String(decodedBytes);

***LOGGER***.debug("Decoded credentials: {}", decodedCredentials);

String username = decodedCredentials.split(":")[0];

***LOGGER***.debug("Extracted username: {}", username);

***LOGGER***.debug("End: getUser()");

return username;

}

*@GetMapping*("/authenticate")

public Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {

***LOGGER***.info("START: /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

Map<String, String> map = new HashMap<>();

String token = generateJwt(user);

map.put("token", token);

***LOGGER***.info("Authenticated user: {}", user);

***LOGGER***.info("END: /authenticate");

return map;

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.*builder*();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date((new Date()).getTime() + 1200000));

builder.signWith(*SignatureAlgorithm*.***HS256***, "secretkey");

String token = builder.compact();

return token;

}

}

Output:



