**Create authentication service that returns JWT** 

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

package com.cognizant.springlearn.controller;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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 java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

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

@GetMapping("/authenticate")

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

LOGGER.info("START");

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

String user = getUser(authHeader);

String token = generateJwt(user);

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

map.put("token", token);

LOGGER.info("END");

return map;

}

private String getUser(String authHeader) {

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

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

String decodedCredentials = new String(decodedBytes);

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

LOGGER.debug("Decoded User: {}", username);

return username;

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date((new Date()).getTime() + 1200000)); // 20 minutes

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

String token = builder.compact();

LOGGER.debug("Generated Token: {}", token);

return token;

}

}

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable().httpBasic().and()

.authorizeRequests()

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

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

}

@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() {

return new BCryptPasswordEncoder();

}

Output:

START

Authorization Header: Basic dXNlcjpwd2Q=

Decoded User: user

Generated Token: eyJhbGciOiJIUzI1NiJ9...

END