**CREATE AUTHENTICATION SERVICE THAT RETURNS JWT**   
  
**SCEANRIO**

As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

**CODE:**

**Application.properties**

spring.application.name=jwt-auth-demo

server.port=8090

**JwtAuthDemoApplication.java**

package com.example.jwt\_auth\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthDemoApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthDemoApplication.class, args);

}

}

**SecurityConfig.java**

package com.example.jwt\_auth\_demo.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.web.SecurityFilterChain;

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

import static org.springframework.security.config.Customizer.withDefaults;

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

@SuppressWarnings("deprecation")

@Configuration

public class SecurityConfig{

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(withDefaults());

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

return new InMemoryUserDetailsManager(

User.withUsername("user")

.password("pwd")

.roles("USER")

.build()

);

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

}

**AuthenticationController.java**

package com.example.jwt\_auth\_demo.controller;

import java.util.Base64;

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

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

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

import com.example.jwt\_auth\_demo.model.AuthenticationResponse;

import com.example.jwt\_auth\_demo.util.JwtUtil;

import jakarta.servlet.http.HttpServletRequest;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public AuthenticationResponse generateToken(HttpServletRequest request) {

String header = request.getHeader("Authorization");

if (header == null || !header.startsWith("Basic ")) {

throw new RuntimeException("Missing or invalid Authorization header");

}

String base64Credentials = header.substring("Basic ".length()).trim();

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

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

String username = values[0];

String password = values[1];

if (!"user".equals(username) || !"pwd".equals(password)) {

throw new RuntimeException("Invalid Credentials");

}

String token = jwtUtil.generateToken(username);

return new AuthenticationResponse(token);

}

}

**AuthenticationResponse.java**

package com.example.jwt\_auth\_demo.model;

public class AuthenticationResponse {

private String token;

public AuthenticationResponse(String token){

this.token=token;

}

public String getToken(){

return token;

}

}

**JwtUtil.java**

package com.example.jwt\_auth\_demo.util;

import java.util.Date;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Base64;

@Component

public class JwtUtil {

private final String SECRET = "mysecretkey1234567890mysecretkey";

private Key getSigningKey() {

byte[] keyBytes = SECRET.getBytes();

return Keys.hmacShaKeyFor(keyBytes);

}

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(getSigningKey(), SignatureAlgorithm.HS256)

.compact();

}

}

OUTPUT

