**SPRING REST USING SPRING BOOT 3**

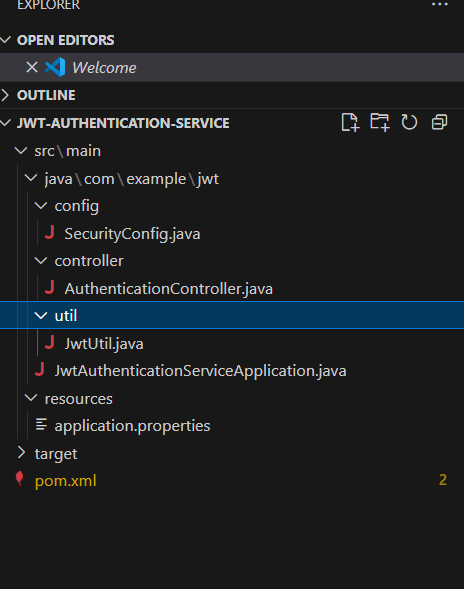
**Create authentication service that returns JWT**As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.  
  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.  
  
Request

curl -s -u user:pwd http://localhost:8090/authenticate

Response

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

**FOLDER STRUCTURE:**



**PROGRAM:**

**SecurityConfig.java**

package com.example.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

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

            .authorizeHttpRequests(auth -> auth.anyRequest().permitAll());

        return http.build();

    }

}

**AuthenticationController.java**

package com.example.jwt.controller;

import com.example.jwt.util.JwtUtil;

import org.springframework.http.ResponseEntity;

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;

@RestController

public class AuthenticationController {

    private final JwtUtil jwtUtil;

    public AuthenticationController(JwtUtil jwtUtil) {

        this.jwtUtil = jwtUtil;

    }

    @GetMapping("/authenticate")

    public ResponseEntity<?> authenticate(@RequestHeader(value = "Authorization", required = false) String authHeader) {

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

            System.out.println("Missing or invalid Authorization header.");

            return ResponseEntity.status(401).body("Missing Authorization header");

        }

        try {

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

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

            String credentials = new String(credDecoded);

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

            String username = values[0];

            String password = values[1];

            System.out.println("Username: " + username + ", Password: " + password);

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

                String token = jwtUtil.generateToken(username);

                System.out.println("Token generated successfully.");

                return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");

            } else {

                System.out.println("Invalid credentials.");

                return ResponseEntity.status(401).body("Invalid credentials");

            }

        } catch (Exception e) {

            System.out.println("Error decoding credentials: " + e.getMessage());

            return ResponseEntity.status(400).body("Invalid Authorization header format");

        }

    }

}

**JwtUtil.java**

package com.example.jwt.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

@Component

public class JwtUtil {

    private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

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

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

                .signWith(key)

                .compact();

    }

}

**JwtAuthenticationServiceApplication.java**

package com.example.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthenticationServiceApplication {

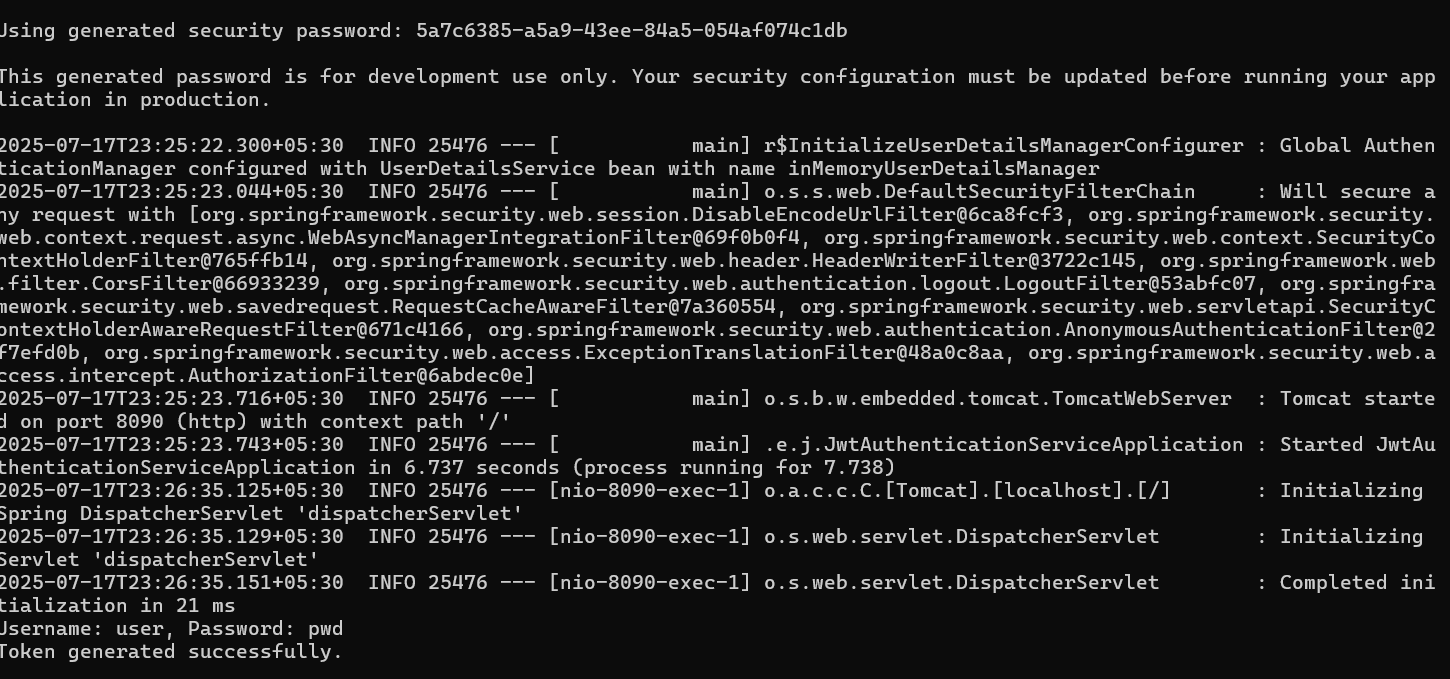
    public static void main(String[] args) {

        SpringApplication.run(JwtAuthenticationServiceApplication.class, args);

    }

}

**OUTPUT:**

****