**5. JWT-handson**

**Create authentication service that returns JWT  
code:**

JwtUtil.java :

package com.cognizant.spring\_learn2.util;

import java.util.Date;

import org.springframework.stereotype.Component;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@Component

public class JwtUtil {

private final String secretKey = "mysecretkey";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

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

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

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}

AuthenticationController.java :

package com.cognizant.spring\_learn2.Controller;

import java.util.Base64;

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 com.cognizant.spring\_learn2.util.JwtUtil;

@RestController

public class AuthenticationController {

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

private final JwtUtil jwtUtil;

public AuthenticationController(JwtUtil jwtUtil) {

this.jwtUtil = jwtUtil;

}

@GetMapping("/authenticate")

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

LOGGER.info("START - authenticate()");

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

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

}

String[] credentials = new String(Base64.getDecoder()

.decode(authHeader.substring("Basic ".length())))

.split(":");

String username = credentials[0];

String password = credentials[1];

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

throw new RuntimeException("Invalid credentials");

}

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

LOGGER.info("END - authenticate()");

return response;

}

}

**SecurityConfig.java :**

package com.cognizant.spring\_learn2.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;

@EnableWebSecurity

@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()

)

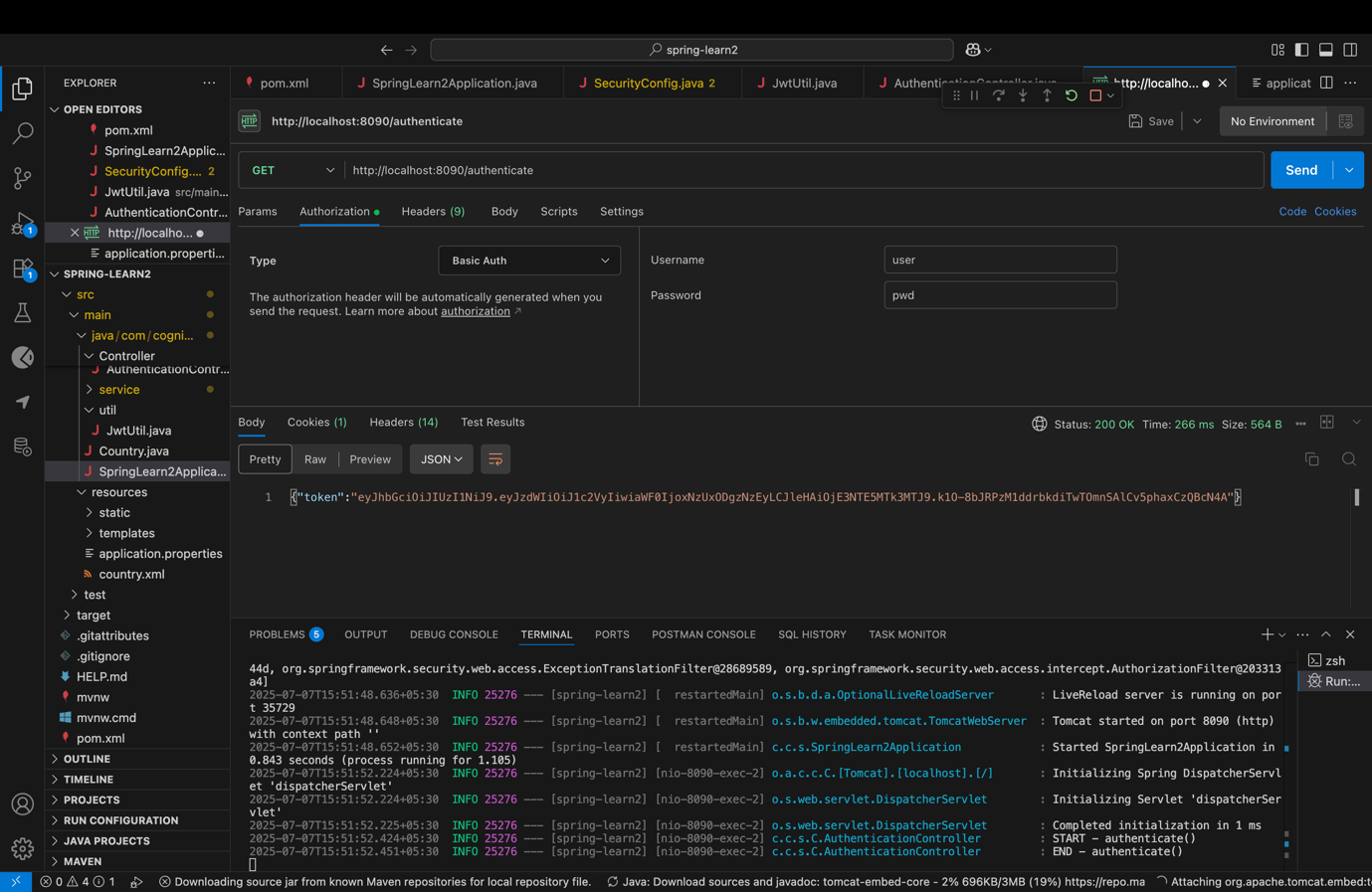
.httpBasic(); // Enable basic auth

return http.build();

}

}

**Output:**

****