**Spring REST using Spring Boot 3**

**EXERCISE 6: Create authentication service that returns JWT**

**Source Code**

**STEP 1: Project Setup**

**xml**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

**Package Structure**

arduino

com.cognizant.springsecurityjwt

controller

└──**>** AuthenticationController.java

config

└──**>** SecurityConfig.java

util

└──**>** JwtUtil.java

**1. AuthenticationController.java**

**java**

package com.cognizant.springsecurityjwt.controller;

import com.cognizant.springsecurityjwt.util.JwtUtil;

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

import org.springframework.security.core.Authentication;

import org.springframework.web.bind.annotation.\*;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public Map<String, String> generateToken(Authentication authentication) {

String username = authentication.getName();

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

return response;

}

}

**2. JwtUtil.java**

**java**

package com.cognizant.springsecurityjwt.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secretKey = "mysecretkey"; // Ideally, store this in properties or env

public String generateToken(String username) {

long currentTime = System.currentTimeMillis();

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(currentTime))

.setExpiration(new Date(currentTime + 1000 \* 60 \* 10)) // 10 minutes

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}

**3. SecurityConfig.java**

**java**

package com.cognizant.springsecurityjwt.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;

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

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

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/authenticate").authenticated()

.anyRequest().permitAll()

.and()

.httpBasic(); // Enables -u user:pwd basic auth

return http.build();

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance(); // Only for demo purposes

}

}

**4. Add User Credentials**

**application.properties (optional)**

Add in-memory user in SecurityConfig (alternative to application.properties)

**java**

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Bean

public InMemoryUserDetailsManager userDetailsService() {

var user = User.withUsername("user")

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

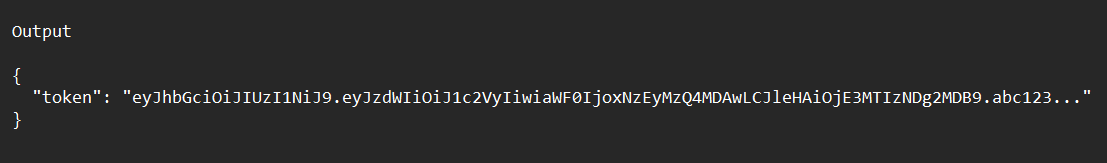
**5. Run & Test**

**Start your app on port 8090**

**Curl command**

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

**Console Output:**

****