**WEEK-4**

**SPRING REST USING SPRING BOOT 3**

* **Create authentication service that returns JWT**

1. Pom.xml

* Add below dependencies

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>

* Right click on project -> Maven -> Update project.

1. SecurityConfig.java

* Src/main/java -> Package (com.example.jwtauth.config) -> class

(SecurityConfig)

package com.example.jwtauth.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.core.userdetails.User;

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

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public UserDetailsService configureUsers(PasswordEncoder encoder) {

UserDetails normalUser = User.withUsername("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

UserDetails superAdmin = User.withUsername("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(normalUser, superAdmin);

}

@Bean

public SecurityFilterChain setupSecurity(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

@Bean

public PasswordEncoder encoder() {

return new BCryptPasswordEncoder();

}

}

1. AuthenticationController.java

* Src/main/java -> package (com.example.jwtauth.controller) -> class (AuthenticationController)

package com.example.jwtauth.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

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

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

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

import java.security.Key;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Key secretKey = Keys.hmacShaKeyFor(

"my-secret-key-which-is-very-secure".getBytes()

);

@GetMapping("/authenticate")

public Map<String, String> generateJwt(@RequestHeader("Authorization") String authorizationHeader) {

String username = extractUsername(authorizationHeader);

String jwtToken = createJwtToken(username);

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

responseMap.put("token", jwtToken);

return responseMap;

}

private String extractUsername(String header) {

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

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

String[] credentials = new String(decoded).split(":");

return credentials[0];

}

private String createJwtToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 20 \* 60 \* 1000)) // 20 minutes

.signWith(secretKey, SignatureAlgorithm.HS256)

.compact();

}

}

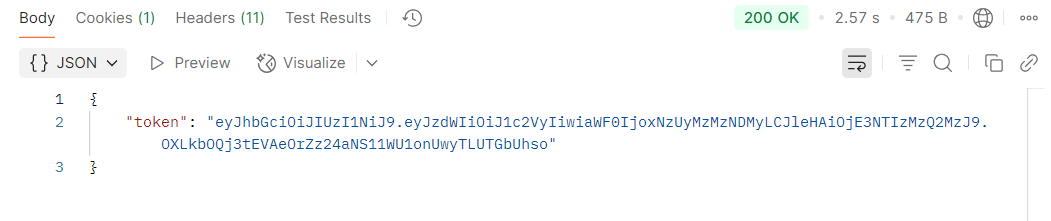
1. Start your application

* JwtAuthApplication.java -> Run As -> Java Application

1. In postman

* <http://localhost:8090/authenticate>
* Authorization tab -> Auth type -> Basic Auth, then give username and password

**Output:**



**Explanation:**

* To manage authentication and token generation, I established a Spring Boot project and included dependencies for the JSON Web Token (jjwt), Spring Web, and Spring Security libraries.
* I created a security filter that limits access to the /authenticate endpoint using basic authentication and constructed a SecurityConfig class to define in-memory users (admin and user) with encrypted passwords.
* After that, I created an AuthenticationController class with a /authenticate endpoint that uses a secret key to create a signed JWT after first extracting the username from the request header.
* The service replies with a JWT token that contains the subject, issue date, expiration, and signature after receiving a legitimate login and password through Postman's Basic Auth.