**SPRING REST USING SPRING BOOT 3**

**5. JWT – handson**

**Create authentication service that returns JWT**

1. Spring initializr

* Project – maven
* Language – java
* Spring boot – 3.5.3
* Group – com.example
* Artifact and name – jwt-auth
* Package name – com.example.jwtauth
* Packaging – jar
* Java – 17
* Dependencies
* Spring web
* Spring security
* Spring boot dev tools
* Lombok
* Click on Generate

1. Importing

* Eclipse -> File -> Import -> Existing Maven project -> browse copy paste path

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.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</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 userDetailsService(PasswordEncoder encoder) {

UserDetails user = User.withUsername("user")

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

.roles("USER")

.build();

UserDetails admin = User.withUsername("admin")

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

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

@Bean

public SecurityFilterChain filterChain(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 passwordEncoder() {

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> authenticate(@RequestHeader("Authorization") String authHeader) {

String user = getUser(authHeader);

String token = generateToken(user);

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

map.put("token", token);

return map;

}

private String getUser(String authHeader) {

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

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

String credentials = new String(decoded);

return credentials.split(":")[0];

}

private String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(secretKey, SignatureAlgorithm.HS256)

.compact();

}

}

1. Make sure you set port number to 8090

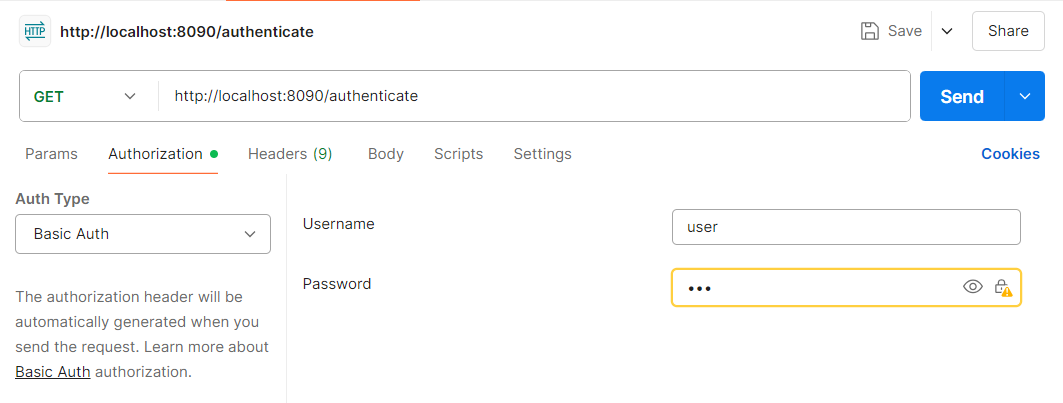
* server.port=8090

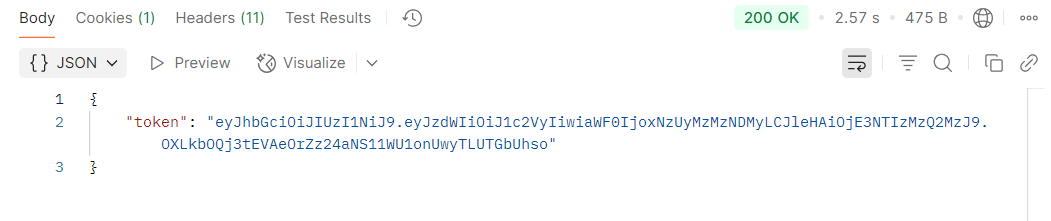
1. Start your application

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

1. In postman

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





1. Open terminal or cmd (optional)

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

