**Create authentication service that returns JWT** 

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>demo</name>

<description>Demo project for Spring Boot JWT Authentication</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<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>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**DemoApplication.java:**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

**SecurityConfig.java:**

package com.example.demo;

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.config.Customizer;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

}

**AuthenticationController.java:**

package com.example.demo;

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.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.Date;

import java.util.Map;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@RestController

public class AuthenticationController {

private final String secretKey = "mySecretKey123456";

@GetMapping("/authenticate")

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

try {

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

return ResponseEntity.status(400).body("Missing or invalid Authorization header");

}

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

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

String credentials = new String(decodedBytes, StandardCharsets.UTF\_8);

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

if (values.length != 2) {

return ResponseEntity.status(400).body("Invalid basic authentication format");

}

String username = values[0];

String password = values[1];

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

String token = generateToken(username);

return ResponseEntity.ok(Map.of("token", token));

} else {

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

}

} catch (Exception e) {

return ResponseEntity.status(400).body("Error processing authentication");

}

}

private String generateToken(String username) {

long nowMillis = System.currentTimeMillis();

long expMillis = nowMillis + 3600000; // 1 hour expiration

Date now = new Date(nowMillis);

Date exp = new Date(expMillis);

return Jwts.builder()

.setSubject(username)

.setIssuedAt(now)

.setExpiration(exp)

.signWith(SignatureAlgorithm.HS256, secretKey.getBytes())

.compact();

}

}

**OUTPUT:**

curl -s -u user:pwd <http://localhost:8080/authenticate>

{"token":"<your.jwt.token.here>"}