[**Create authentication service that returns JWT**](https://github.com/Harishmoorthy2005/DN4.0-Deepskilling-6408031/tree/main/Week4_Spring%20REST%20using%20Spring%20Boot%203/5.%20JWT-handson/Create%20authentication%20service%20that%20returns%20JWT)

**pom.xml:**

<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

http://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>3.2.5</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>jwt-auth-service</artifactId>

<version>1.0</version>

<properties>

<java.version>20</java.version>

<maven.compiler.source>20</maven.compiler.source>

<maven.compiler.target>20</maven.compiler.target>

</properties>

<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>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>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**JwtAuthApplication.java:**

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**application.properties:**

server.port=8090

spring.security.user.name=user

spring.security.user.password=pwd

**JwtUtil.java:**

package com.example.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

public class JwtUtil {

private static final Key SECRET\_KEY = Keys.hmacShaKeyFor("mySuperSecretKey12345678901234567890".getBytes());

public static String generateToken(String username) {

long now = System.currentTimeMillis();

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(now))

.setExpiration(new Date(now + 1000 \* 60 \* 10))

.signWith(SECRET\_KEY, SignatureAlgorithm.HS256)

.compact();

}

}

**AuthController.java:**

package com.example.controller;

import com.example.util.JwtUtil;

import org.springframework.http.ResponseEntity;

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

import jakarta.servlet.http.HttpServletRequest;

import java.security.Principal;

import java.util.Map;

@RestController

public class AuthController {

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

Principal principal = request.getUserPrincipal();

if (principal == null) {

return ResponseEntity.status(401).build();

}

String username = principal.getName();

String token = JwtUtil.generateToken(username);

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

}

}

**SecurityConfig.java:**

package com.example.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;

@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();

return http.build();

}

}

