**WEEK-4**

**JWT Hands-On**

**Exercise-1: Create authentication service that returns JWT**

**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>

<groupId>com.cognizant.spring\_learn</groupId>

<artifactId>jwt-auth</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>jwt-auth</name>

<description>JWT Authentication Project</description>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

</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>

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.5.3</version>

</plugin>

</plugins>

</build>

</project>

**src/main/resource/properties.xml:**

spring.application.name=jwt-auth

server.port=8090

**src/main/java/JwtAuthApplication:**

package com.cognizant.spring\_learn.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

System.out.println("START - JwtAuthApplication main()");

SpringApplication.run(JwtAuthApplication.**class**, args);

System.out.println("END - JwtAuthApplication main()");

}

}

**SecurityConfig.java:**

**package** com.cognizant.spring\_learn.jwtauth.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

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

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

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

**import** org.springframework.security.authentication.AuthenticationManager;

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

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

@Configuration

**public** **class** SecurityConfig {

@Bean

**public** UserDetailsService userDetailsService() {

**var** user = User.*withUsername*("user")

.password("pwd")

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(user);

}

@Bean

**public** AuthenticationManager authenticationManager(AuthenticationConfiguration config) **throws** Exception {

**return** config.getAuthenticationManager();

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** NoOpPasswordEncoder.getInstance(); }

}

**AuthController.java:**

**package** com.cognizant.spring\_learn.jwtauth.controller;

**import** com.cognizant.spring\_learn.jwtauth.util.JwtUtil;

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

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.Authentication;

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

**import** java.util.Base64;

**import** jakarta.servlet.http.HttpServletRequest;

@RestController

**public** **class** AuthController {

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

**private** JwtUtil jwtUtil;

@GetMapping("/authenticate")

**public** String authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

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

**throw** **new** RuntimeException("Missing or invalid Authorization header.");

}

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

String credentials = **new** String(Base64.getDecoder().decode(base64Credentials));

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

String username = values[0];

String password = values[1];

Authentication authentication = authenticationManager.authenticate(

**new** UsernamePasswordAuthenticationToken(username, password)

);

**if** (authentication.isAuthenticated()) {

String token = jwtUtil.generateToken(username);

**return** "{\"token\":\"" + token + "\"}";

} **else** {

**throw** **new** RuntimeException("Invalid credentials");

}

}

}

**JwtUtil.java:**

**package** com.cognizant.spring\_learn.jwtauth.util;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** io.jsonwebtoken.security.Keys;

**import** org.springframework.stereotype.Service;

**import** java.security.Key;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map;

@Service

**public** **class** JwtUtil {

**private** **final** Key key = Keys.secretKeyFor(SignatureAlgorithm.***HS256***);

**public** String generateToken(String username) {

Map<String, Object> claims = **new** HashMap<>();

**return** Jwts.builder()

.setClaims(claims)

.setSubject(username)

.setIssuedAt(**new** Date(System.*currentTimeMillis*()))

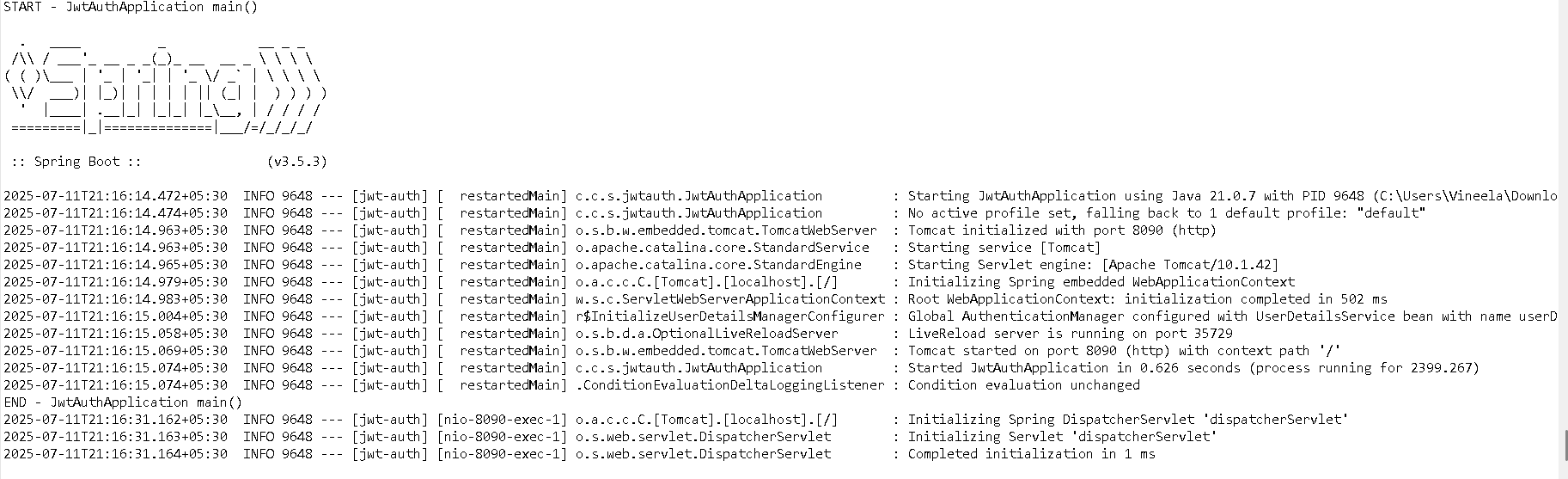
.setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60 \* 10)) .signWith(key)

.compact();

}

}

**OUTPUT:**

****

