**Week 4**

**JWT - handson**

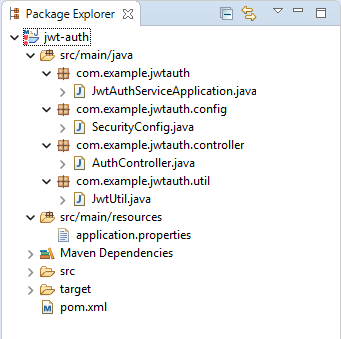
**Create authentication service that returns JWT**   
As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.  
  
**Request**

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

**Response**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

**FILE STRUCTURE**



**CODE**

*src\main\java\com\example\jwtauth\JwtAuthServiceApplication.java*

package com.example.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(JwtAuthServiceApplication.class, args);

    }

}

*src\main\java\com\example\jwtauth\config\SecurityConfig.java*

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.UserDetailsService;

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

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 SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.csrf().disable()

            .authorizeRequests()

                .antMatchers("/authenticate").authenticated()

                .anyRequest().permitAll()

            .and()

            .httpBasic();

        return http.build();

    }

    @Bean

    public PasswordEncoder passwordEncoder() {

        return NoOpPasswordEncoder.getInstance();

    }

    @Bean

    public UserDetailsService users() {

        InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

        manager.createUser(User.withUsername("user").password("pwd").roles("USER").build());

        return manager;

    }

}

*src\main\java\com\example\jwtauth\controller\AuthController.java*

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.http.HttpHeaders;

import org.springframework.http.ResponseEntity;

import org.springframework.util.Base64Utils;

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

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

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

import java.util.Collections;

import java.util.Map;

@RestController

public class AuthController {

    @Autowired

    private JwtUtil jwtUtil;

    @GetMapping("/authenticate")

    public ResponseEntity<Map<String, String>> authenticate(@RequestHeader(value = HttpHeaders.AUTHORIZATION, required = false) String authHeader) {

        try {

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

                return ResponseEntity.status(401).body(Collections.singletonMap("error", "Missing or invalid auth header"));

            }

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

            String credentials = new String(Base64Utils.decodeFromString(base64Credentials));

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

            if (parts.length != 2) {

                return ResponseEntity.status(401).body(Collections.singletonMap("error", "Invalid credential format"));

            }

            String token = jwtUtil.generateToken(parts[0]);

            return ResponseEntity.ok(Collections.singletonMap("token", token));

        } catch (Exception e) {

            e.printStackTrace();

            return ResponseEntity.status(500).body(Collections.singletonMap("error", "Server Error: " + e.getMessage()));

        }

    }

}

*src\main\java\com\example\jwtauth\util\JwtUtil.java*

package com.example.jwtauth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;

import java.util.Date;

@Component

public class JwtUtil {

    private final SecretKey secretKey = Keys.hmacShaKeyFor("my-super-secret-key-which-is-long-enough".getBytes());

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

                .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 minutes

                .signWith(secretKey, SignatureAlgorithm.HS256)

                .compact();

    }

}

*src\main\resources\application.properties*

server.port=8090

*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.example</groupId>

    <artifactId>jwt-auth</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <parent>

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

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

        <version>2.7.18</version>

    </parent>

    <properties>

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

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

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

    </properties>

    <dependencies>

        <!-- Spring Boot Web & Security -->

        <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>

        <!-- Updated JWT dependencies for Java 17+ / Java 20 -->

        <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>

**OUTPUT**

