**Hands-on: 6**

**Create authentication service that returns JWT**   
  
User credentials needs to be sent to authentication service request that generates and returns the JWT.

**Objective:**

To create a secure /authenticate endpoint that accepts credentials via HTTP Basic Auth, decodes them, generates a JWT, and returns it as a JSON response. The system will:

* Authenticate users using Spring Security
* Generate a JWT on successful login
* Protect API endpoints by requiring a valid JWT

**Authentication Flow**

1. **User sends credentials** (username and password) to /authenticate
2. **Spring Security** validates the credentials using CustomUserDetailsService
3. If valid:
   * A JWT is generated and returned using jjwt library
4. If invalid:
   * Returns HTTP 401 Unauthorized

**Dependencies in 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>2.7.17</version> <!-- ✅ Use a stable version -->

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>JWT-Token</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>JWT-Token</name>

<description>JWT Token Authentication Service</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Security -->

<dependency>

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

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

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<!-- JSON Web Token -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

<!-- Spring Boot Logging (optional as it's included by default) -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Plugin -->

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**SecurityConfig.java:**

package com.cognizant.jwt.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.\*;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig extends ~~WebSecurityConfigurerAdapter~~ {

*@Override*

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER")

.and()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN");

}

*@Override*

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.httpBasic()

.and()

.authorizeRequests()

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN");

}

*@Bean*

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

**AuthenticationController.java:**

package com.cognizant.jwt.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

*@RestController*

public class AuthenticationController {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(AuthenticationController.class);

*@GetMapping*("/authenticate")

public Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {

***LOGGER***.info("START /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader); // if this fails, 500 error

String token = generateJwt(user); // if this fails, 500 error

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

map.put("token", token);

***LOGGER***.info("END /authenticate");

return map;

}

private String getUser(String authHeader) {

***LOGGER***.debug("Decoding Authorization header...");

// Sample header: Basic dXNlcjpwd2Q=

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

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

String decodedString = new String(decodedBytes); // user:pwd

***LOGGER***.debug("Decoded: {}", decodedString);

return decodedString.split(":")[0]; // extract user

}

private String generateJwt(String user) {

return Jwts.*builder*()

.setSubject(user)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 1200000)) // 20 mins

.signWith(*SignatureAlgorithm*.***HS256***, "secretkey")

.compact();

}

}

**JwtTokenApplcation.java**:

package com.cognizant.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class JwtTokenApplication {

public static void main(String[] args) {

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

}

}

**Test in Postman:**

