**Cognizant Digital Nurture 4.0**

**Name: Gandu Lasya Sri**

**Email: gandulasyasri@gmail.com**

**Superset ID:6428164**

**Mandatory Hands-On Exercises**

**JWT-handson**

**Task: Create authentication service that returns JWT**

In this hands-on task, we are asked to create a simple authentication service using Spring Boot that returns a JWT (JSON Web Token) after successful login. Below is a complete explanation, like a student would describe it for a submission.

**Objective:**

To build a REST API endpoint (/authenticate) that returns a JWT token when a valid username and password are passed using basic authentication (like curl -u user:pwd http://localhost:8081/authenticate).

**Step 1: Create Authentication Controller**

**File:** AuthenticationController.java

1. First, I created a controller class with @RestController.
2. Inside it, I added a method mapped to /authenticate using @GetMapping.
3. This method takes the Authorization header from the HTTP request.
4. I added logging at the start and end of the method for debugging.

package com.example.jwtauth.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private Logger logger = LoggerFactory.getLogger(getClass());

@GetMapping("/authenticate")

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

logger.info("START - authenticate()");

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

String user = getUser(authHeader);

String token = generateJwt(user);

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

map.put("token", token);

logger.info("END - authenticate()");

return map;

}

private String getUser(String authHeader) {

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

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

String decoded = new String(decodedBytes);

String[] parts = decoded.split(":");

String username = parts[0];

logger.debug("Decoded username: {}", username);

return username;

}

private String generateJwt(String user) {

long currentTimeMillis = System.currentTimeMillis();

return io.jsonwebtoken.Jwts.builder()

.setSubject(user)

.setIssuedAt(new java.util.Date(currentTimeMillis))

.setExpiration(new java.util.Date(currentTimeMillis + 1200000))

.signWith(io.jsonwebtoken.SignatureAlgorithm.HS256, "mysupersecretkeymysupersecretkey")

.compact();

}

}

⚖️ **Step 2: Update Security Configuration**

**File:** SecurityConfig.java

http.authorizeHttpRequests()

.requestMatchers("/countries").hasRole("USER")

.requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.httpBasic();

**Explanation:**

This config allows both USER and ADMIN roles to access /authenticate.

**Step 3: Read and Decode Authorization Header**

private String getUser(String authHeader) {

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

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

String decoded = new String(decodedBytes);

String[] parts = decoded.split(":");

String username = parts[0];

logger.debug("Decoded username: {}", username);

return username;

}

**Explanation:**

The Authorization header contains a Base64 encoded username and password. We decode it and extract the username.

🔒 **Step 4: Generate JWT Token**

private String generateJwt(String user) {

long currentTimeMillis = System.currentTimeMillis();

return Jwts.builder()

.setSubject(user)

.setIssuedAt(new Date(currentTimeMillis))

.setExpiration(new Date(currentTimeMillis + 1200000)) // 20 minutes

.signWith(SignatureAlgorithm.HS256, "mysupersecretkeymysupersecretkey")

.compact();

}

**Explanation:**

We create a JWT using the username, issue time, and expiration. The key used is 256 bits as required by the JWT spec.

🌐 **Step 5: Testing via Browser and curl**

You can access the endpoint by opening your browser and going to:

http://localhost:8081/authenticate

The browser will prompt for username and password (e.g., lasya and 1234). After logging in, you should see a JSON response like:

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJsYXN5YSIsImlhdCI6MTY4NDY5MDYzMywiZXhwIjoxNjg0NjkxODMzfQ.GUlTljytEkGXFcMgJPJVLrcbyw9z1Qz5c7v2lPuNkLs"}

You can also test it with curl:

curl -u lasya:1234 http://localhost:8081/authenticate



