pom.xml:

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

AuthenticationController.java:

package com.cognizant.springlearn.controller;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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;

@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);

// Decode and extract username

String user = getUser(authHeader);

LOGGER.debug("Extracted User: {}", user);

// Generate JWT token

String token = generateJwt(user);

LOGGER.debug("Generated Token: {}", token);

// Return token in response map

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

map.put("token", token);

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

return map;

}

private String getUser(String authHeader) {

LOGGER.debug("START getUser()");

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

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

String decodedCredentials = new String(decodedBytes); // e.g., "user:pwd"

LOGGER.debug("Decoded Credentials: {}", decodedCredentials);

String username = decodedCredentials.split(":", 2)[0];

LOGGER.debug("Extracted Username: {}", username);

LOGGER.debug("END getUser()");

return username;

}

private String generateJwt(String user) {

LOGGER.debug("START generateJwt()");

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date((new Date()).getTime() + 1200000)); // 20 mins

builder.signWith(SignatureAlgorithm.HS256, "secretkey");

String token = builder.compact();

LOGGER.debug("END generateJwt()");

return token;

}

}