**WEEK – 4**

**Module 7 - Spring REST using Spring Boot 3**

**Superset ID:6407550**

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

**src\main\java**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("SpringLearnApplication started...");

}

}

**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.http.ResponseEntity;

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

import javax.crypto.SecretKey;

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.Date;

*@RestController*

public class AuthenticationController {

private static final SecretKey ***SECRET\_KEY*** = Keys.*secretKeyFor*(*SignatureAlgorithm*.***HS256***);

*@RequestMapping*("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

return ResponseEntity.*status*(401).body("Missing or invalid Authorization header");

}

// Decode Basic auth

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

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

String decodedCredentials = new String(decodedBytes);

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

if (parts.length != 2) {

return ResponseEntity.*status*(400).body("Invalid Authorization format");

}

String username = parts[0];

String password = parts[1];

// Validate credentials (hardcoded for demo)

if (!("user".equals(username) && "pwd".equals(password))) {

return ResponseEntity.*status*(401).body("Invalid credentials");

}

// Generate JWT

String token = Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 600000)) // 10 min

.signWith(***SECRET\_KEY***)

.compact();

return ResponseEntity.*ok*("{\"token\":\"" + token + "\"}");

}

}

**pom.xml**

package com.cognizant.spring\_learn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.http.ResponseEntity;

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

import javax.crypto.SecretKey;

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.Date;

*@RestController*

public class AuthenticationController {

private static final SecretKey ***SECRET\_KEY*** = Keys.*secretKeyFor*(*SignatureAlgorithm*.***HS256***);

*@RequestMapping*("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

return ResponseEntity.*status*(401).body("Missing or invalid Authorization header");

}

// Decode Basic auth

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

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

String decodedCredentials = new String(decodedBytes);

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

if (parts.length != 2) {

return ResponseEntity.*status*(400).body("Invalid Authorization format");

}

String username = parts[0];

String password = parts[1];

// Validate credentials (hardcoded for demo)

if (!("user".equals(username) && "pwd".equals(password))) {

return ResponseEntity.*status*(401).body("Invalid credentials");

}

// Generate JWT

String token = Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 600000)) // 10 min

.signWith(***SECRET\_KEY***)

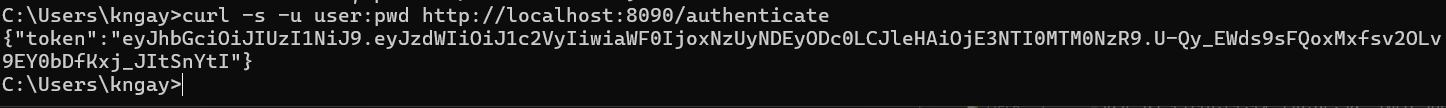
.compact();

return ResponseEntity.*ok*("{\"token\":\"" + token + "\"}");

}

}

**OUTPUT:**

****