**sCreate 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"}

This can be incorporated as three major steps:

* Create authentication controller and configure it in SecurityConfig
* Read Authorization header and decode the username and password
* Generate token based on the user retrieved in the previous step

Let incorporate the above as separate hands on exercises.

**Solution:**

**Creating the authentication controller:**

package com.cognizant.spring\_learn.controller;

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

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

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

import com.cognizant.spring\_learn.service.AuthenticationService;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private AuthenticationService authenticationService;

@GetMapping("/authenticate")

public Map<String, String> authenticate(HttpServletRequest request) {

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

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

// Remove "Basic " and decode

String base64Credentials = authHeader.substring(6);

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

String credentials = new String(decodedBytes);

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

String username = userDetails[0];

String password = userDetails[1];

// Authenticate and return token

String token = authenticationService.generateToken(username, password);

return Map.of("token", token);

} else {

throw new RuntimeException("Missing or invalid Authorization header");

}

}

}

**Creating Authentication service:**

package com.cognizant.spring\_learn.service;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Service;

import java.security.Key;

import java.util.Date;

@Service

public class AuthenticationService {

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

private static final long EXPIRATION\_TIME = 600\_000; // 10 minutes

public String generateToken(String username, String password) {

// Basic check (you can replace this with DB validation or in-memory map)

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

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.signWith(SECRET\_KEY)

.compact();

} else {

throw new RuntimeException("Invalid credentials");

}

}

}

**Configuring Security:**

package com.cognizant.spring\_learn.config

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.web.SecurityFilterChain;

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

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults())

.csrf(csrf -> csrf.disable()); // Disable CSRF for simplicity

return http.build();

}

}

**Request:**

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

**Output:**

