Digital-Nurture-4.0-JavaFSE-main

Create authentication service that returns JWT

Name: Thilak Ragav M

Superset ID : 6409405

Week Task : Week - 4

# **5.JWT\_Handson**

**Create 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.

**SecurityConfig.java:**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public UserDetailsService userDetailsService() {

return new InMemoryUserDetailsManager(

User.withUsername("user")

.password("{noop}pwd")

.roles("USER")

.build()

);

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").authenticated()

.anyRequest().permitAll()

)

.httpBasic();

return http.build();

}

}

**AuthenticationController.java :**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.util.JwtUtil;

import org.springframework.security.core.Authentication;

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

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private final JwtUtil jwtUtil;

public AuthenticationController(JwtUtil jwtUtil) {

this.jwtUtil = jwtUtil;

}

@GetMapping("/authenticate")

public Map<String, String> createToken(Authentication authentication) {

String username = authentication.getName();

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

return response;

}

}

**JwUtil.java:**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private final String secret = "secret-key";

private final long expirationMs = 1000 \* 60 \* 10;

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + expirationMs))

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

}

**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);

}

}

**Output:**



