**Handson Exercise 6 - Read Authorization header and decode the username and password**

**CODE:**

**SecurityConfig Class: -**

package com.cognizant.ex4handson2;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationProvider;  
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
 public UserDetailsService userDetailsService(PasswordEncoder encoder) {  
 UserDetails user = User.*withUsername*("user")  
 .password(encoder.encode("pwd"))  
 .roles("USER")  
 .build();  
  
 return new InMemoryUserDetailsManager(user);  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public AuthenticationProvider authProvider(UserDetailsService userDetailsService, PasswordEncoder encoder) {  
 DaoAuthenticationProvider provider = new DaoAuthenticationProvider();  
 provider.setUserDetailsService(userDetailsService);  
 provider.setPasswordEncoder(encoder);  
 return provider;  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/countries").hasRole("USER")  
 .requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")  
 .anyRequest().authenticated()  
 )  
 .httpBasic();  
  
 return http.build();  
 }

}

**AuthController Class: -**

package com.cognizant.ex4handson2;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.\*;  
  
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);  
  
 Map<String, String> map = new HashMap<>();  
 map.put("token", "");  
  
 *LOGGER*.info("End /authenticate");  
 return map;  
 }

private String getUser(String authHeader) {  
 *LOGGER*.info("Start getUser");  
 String encodedCredentials = authHeader.substring("Basic ".length());  
 *LOGGER*.debug("Encoded credentials: {}", encodedCredentials);

byte[] decodedBytes = Base64.*getDecoder*().decode(encodedCredentials);  
 String decodedString = new String(decodedBytes); // "user:pwd"  
 *LOGGER*.debug("Decoded string: {}", decodedString);  
  
 String username = decodedString.split(":")[0];  
 *LOGGER*.info("End getUser");  
  
 return username;  
}

}

**JwtUtil Class: -**

package com.cognizant.ex4handson2;  
import io.jsonwebtoken.\*;  
import io.jsonwebtoken.security.Keys;  
import org.springframework.stereotype.Component;  
import java.util.Date;  
import java.security.Key;  
  
@Component  
public class JwtUtil {  
 private final String SECRET\_KEY = "mysecretkeymysecretkeymysecretkey123";  
  
 public String generateToken(String username) {  
 long expirationTime = 1000 \* 60 \* 60;  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + expirationTime))  
 .signWith(getSignKey(), SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
  
 private Key getSignKey() {  
 return Keys.*hmacShaKeyFor*(SECRET\_KEY.getBytes());  
 }  
}

**Main Class: -**

package com.cognizant.jwtHandson;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class JwtHandsonApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(JwtHandsonApplication.class, args);  
 }  
  
}

**OUTPUT: -**



