**Ex:06 Create authentication service that returns JWT**

**PROGRAM:**

JwtAuthServiceApplication:

package com.example.jwt\_auth\_service;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class JwtAuthServiceApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(JwtAuthServiceApplication.class, args);  
 }  
}

SecurityConfig.java:

package com.example.jwt\_auth\_service.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 );  
 return http.build();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {  
 return config.getAuthenticationManager();  
 }  
}

UserConfig.java:

package com.example.jwt\_auth\_service.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
  
@Configuration  
public class UserConfig {  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 return new InMemoryUserDetailsManager(  
 User.*withUsername*("user")  
 .password("{noop}pwd") // {noop} means plain text password  
 .roles("USER")  
 .build()  
 );  
 }  
}

AuthController.java:  
package com.example.jwt\_auth\_service.controller;  
  
import com.example.jwt\_auth\_service.util.JwtUtil;  
import jakarta.servlet.http.HttpServletRequest;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Base64;  
import java.util.HashMap;  
import java.util.Map;  
  
@RestController  
public class AuthController {  
  
 @Autowired  
 private AuthenticationManager authenticationManager;  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @RequestMapping(value = "/authenticate", method = RequestMethod.*POST*)  
 public Map<String, String> authenticate(HttpServletRequest request) {  
 String authHeader = request.getHeader("Authorization");  
  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 throw new RuntimeException("Missing or invalid Authorization header.");  
 }  
  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);  
 String credentials = new String(credDecoded);  
 final String[] values = credentials.split(":", 2);  
  
 try {  
 Authentication auth = authenticationManager.authenticate(  
 new UsernamePasswordAuthenticationToken(values[0], values[1])  
 );  
  
 String token = jwtUtil.generateToken(values[0]);  
  
 Map<String, String> response = new HashMap<>();  
 response.put("token", token);  
 return response;  
 } catch (AuthenticationException e) {  
 throw new RuntimeException("Invalid credentials");  
 }  
 }

}  
**OUTPUT:**

