**JWT-HANDSON**

**SecurityConfig.java (in package security)**

package com.cognizant.springlearn.security;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
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.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
  
@Configuration  
public class SecurityConfig {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(SecurityConfig.class);  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public InMemoryUserDetailsManager userDetailsService() {  
 UserDetails admin = User  
 .*withUsername*("admin")  
 .password(passwordEncoder().encode("pwd"))  
 .roles("ADMIN")  
 .build();  
  
 UserDetails user = User  
 .*withUsername*("user")  
 .password(passwordEncoder().encode("pwd"))  
 .roles("USER")  
 .build();  
  
 return new InMemoryUserDetailsManager(admin, user);  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf().disable()  
 .httpBasic()  
 .and()  
 .authorizeHttpRequests()  
 .requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")  
 .anyRequest().authenticated();  
  
 return http.build();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {  
 return config.getAuthenticationManager();  
 }  
}

**AuthenticationController.java (in the package controller)**

package com.cognizant.springlearn.controller;  
  
import io.jsonwebtoken.JwtBuilder;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestHeader;  
import org.springframework.web.bind.annotation.RestController;  
import io.jsonwebtoken.security.Keys;  
  
  
import java.util.Base64;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Map;  
  
import io.jsonwebtoken.security.Keys;  
import java.security.Key;  
  
@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);  
  
 String user = getUser(authHeader);  
 String token = generateJwt(user);  
  
 Map<String, String> map = new HashMap<>();  
 map.put("token", token);  
  
 *LOGGER*.info("END - authenticate()");  
 return map;  
 }  
  
 private String getUser(String authHeader) {  
 String encodedCredentials = authHeader.replace("Basic ", "");  
 byte[] decodedBytes = Base64.*getDecoder*().decode(encodedCredentials);  
 String decodedCredentials = new String(decodedBytes); // "user:pwd"  
 *LOGGER*.debug("Decoded credentials: {}", decodedCredentials);  
 return decodedCredentials.split(":")[0]; // return "user"  
 }  
  
 private String generateJwt(String user) {  
 String secret = "my-secret-key-that-is-long-enough-123456"; // At least 32 chars  
 Key key = Keys.*hmacShaKeyFor*(secret.getBytes());  
  
 return Jwts.*builder*()  
 .setSubject(user)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + 20 \* 60 \* 1000)) // 20 minutes  
 .signWith(key, SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
}

**Run the SpringLearnApplication.java and go to browser and type,**

[**http://localhost:8080/authenticate**](http://localhost:8080/authenticate)

**You will get a prompt to enter credentials**,

A screenshot of a computer

AI-generated content may be incorrect.

**I have entered “user” for username and “pwd” for password**

A screenshot of a login screen

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.