**Create authentication service that returns JWT**

UserCredentials.java:

package com.cognizant.springlearn.security;

public class UserCredentials {

private String username;

private String password;

public String getUsername() { return username; }

public void setUsername(String username) { this.username = username; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

}

SecurityConfig.java:

package com.cognizant.springlearn.security;

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.builders.AuthenticationManagerBuilder;

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

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("user")

.password(passwordEncoder().encode("secret"))

.roles("USER");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

@Override

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

}

JwtUtil.java:

package com.cognizant.springlearn.security;

import io.jsonwebtoken.Claims;

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\_KEY = "secretkey123";

private final long EXPIRATION\_MS = 1000 \* 60 \* 60; // 1 hour

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

public Boolean validateToken(String token, String username) {

final String user = extractUsername(token);

return (user.equals(username) && !isTokenExpired(token));

}

public String extractUsername(String token) {

return extractAllClaims(token).getSubject();

}

private boolean isTokenExpired(String token) {

return extractAllClaims(token).getExpiration().before(new Date());

}

private Claims extractAllClaims(String token) {

return Jwts.parser().setSigningKey(SECRET\_KEY)

.parseClaimsJws(token).getBody();

}

}

AuthenticationController.java:

package com.cognizant.springlearn.security;

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.PostMapping;

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

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

@RestController

public class AuthenticationController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtUtil jwtUtil;

@PostMapping("/authenticate")

public String createAuthenticationToken(@RequestBody UserCredentials creds) throws Exception {

try {

Authentication auth = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(

creds.getUsername(), creds.getPassword()));

} catch (AuthenticationException e) {

throw new Exception("Incorrect username or password", e);

}

return jwtUtil.generateToken(creds.getUsername());

}

}