Best source for the Spring security:

<https://ivangfr.github.io/>

From Spring Security 5.7, the WebSecurityConfigurerAdapter is deprecated to encourage users

to move towards a component-based security configuration. It is recommended to create a bean of type SecurityFilterChain for security related configurations.

Before 5.7 we have the

Extends **WebSecurityConfigurerAdapter** that provide the below two methods:

protected void configure(HttpSecurity http) throws Exception = Which endpoint we need to provide the Security.

- IT IS CALLED AUTHORIZATION

protected void configure(AuthenticationManagerBuilder auth) = Provide the user details in that.

- IT IS CALLED AUTHENTICATION

In new we need to create two Beans

package com.security.config;

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.method.configuration.EnableMethodSecurity;

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

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

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

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

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

import org.springframework.security.web.SecurityFilterChain;

import com.security.service.UserInfoUserDetailSevice;

@Configuration

@EnableWebSecurity

@EnableMethodSecurity

public class SpringSecurity {

@Bean

public UserDetailsService userDetailsService() {

// public UserDetailsService userDetailsService(PasswordEncoder passwordEncoder) {

// UserDetails userDetails = User.withUsername("test").password(passwordEncoder.encode("test")).roles("Admin")

// .build();

// UserDetails userDetails1 = User.withUsername("test1").password(passwordEncoder.encode("test1")).roles("Simple")

// .build();

// return new InMemoryUserDetailsManager(userDetails, userDetails1);

return new UserInfoUserDetailSevice();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {

http.csrf().disable().authorizeHttpRequests().requestMatchers("/addUser").permitAll().and()

.authorizeHttpRequests()

.requestMatchers("/myLoans", "/myCards", "/contact", "/notices", "/myAccount", "/myBalance")

.authenticated().anyRequest().denyAll().and().formLogin();

return http.build();

}

@Bean

public AuthenticationProvider authenticationProvider() {

DaoAuthenticationProvider authenticationProvider = new DaoAuthenticationProvider();

authenticationProvider.setUserDetailsService(userDetailsService());

authenticationProvider.setPasswordEncoder(passwordEncoder());

return authenticationProvider;

}

}

package com.security.service;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

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

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

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

import org.springframework.stereotype.Component;

import com.security.config.UserInfoUserDetails;

import com.security.model.User;

import com.security.repository.UserRepository;

@Component

public class UserInfoUserDetailSevice implements UserDetailsService {

@Autowired

UserRepository repository;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

// TODO Auto-generated method stub

Optional<User> user = repository.findByName(username);

return user.map(UserInfoUserDetails::new).orElseThrow(() -> new UsernameNotFoundException("User Not Found"));

}

}

package com.security.config;

import java.util.Arrays;

import java.util.Collection;

import java.util.List;

import java.util.stream.Collectors;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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

import com.security.model.User;

public class UserInfoUserDetails implements UserDetails {

private String username;

private String password;

private List<GrantedAuthority> grantedAuthorities;

public UserInfoUserDetails(User user) {

this.username = user.getName();

this.password = user.getPwd();

this.grantedAuthorities = Arrays.stream(user.getRole().split(",")).map(SimpleGrantedAuthority::new)

.collect(Collectors.toList());

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

// TODO Auto-generated method stub

return grantedAuthorities;

}

@Override

public String getPassword() {

// TODO Auto-generated method stub

return password;

}

@Override

public String getUsername() {

// TODO Auto-generated method stub

return username;

}

@Override

public boolean isAccountNonExpired() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isAccountNonLocked() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isCredentialsNonExpired() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isEnabled() {

// TODO Auto-generated method stub

return true;

}

}

**JWT Token**

JWT token has three parts,

**eyJhbGciOiJIUzI1NiJ9**.eyJSb2xlIjoiQWRtaW4iLCJJc3N1ZXIiOiJJc3N1ZXIiLCJVc2VybmFtZSI6IkphdmFJblVzZSIsImV4cCI6MTY4MjQyNDY0MSwiaWF0IjoxNjgyNDI0NjQxfQ.**JRcgouYKLnJ6oChKcPhBv0RsBmYXGEz1JO8lupjdWe4**

**HEADER**

**PAYLOAD**

**VERIFY SIGNATURE**

**Above three components called as a claims in JWT.**

[**https://jwt.io/**](https://jwt.io/) **for decode the JWT token**