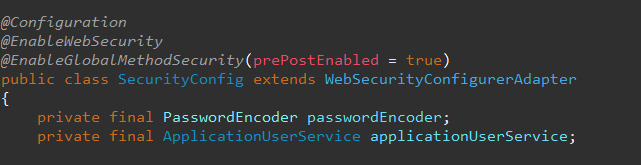
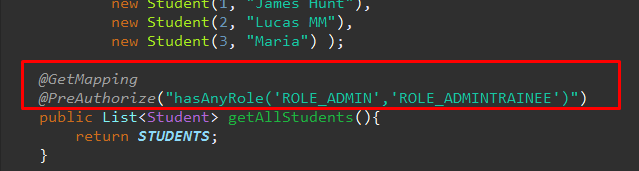
Spring Boot Security

O spring boot security tem o objetivo de cuidar da parte de autenticação e autorização do aplicativo.

Inicialmente é necessário uma classe de configuração exemplo



A anotação **@EnableGlobalMethodSecirity( prePostEnabled = true )** serve para permitir que os métodos dos controllers tenham o direito de guardarem suas próprias rotas.



**No Spring existe vários elementos que permite você torna-lo super customizavel**

WebSecurityConfigAdapter: é a classe que te dá acesso á métodos que modificam o spring boot security, permitindo a customização da autenticação e autorização.

## **Interfaces de de autenticação:**

* *UserDetailsService*: interface, representa uma interfaçe que permite o desenvolvedor customizar o serviço que realiza a busca de dados através de uma implementação DAO ou JPA, também lança uma exceção caso não encontre o usuário.
* *UserDetails*: interface, representa os dados do usuário como nome, senha, authorities que serão implementados pelo desenvolvedor, uma objeto dessa implementação é gerado quando o UserDetailsService encontra o usuário.
* *AuthenticationProvider*: interface, é quem coordena o processo, ele recebe o UserDetailsService, e o encoder.
* *PasswordEncoder*: interface, quem define como a senha será encriptografada

**Classes:**

* SimpleGrantedAuthorities: classe, representa as “roles” e “authorities” do usuário.
* DaoInterface: interface, é uma classe que recomendo você criar, assim na hora de implementar um DAO que busque o usuario a implementação do UserDetailsService não irá depender de uma implementação de busca do banco de dados, garantindo o baixo acoplamento, e a fácil troca de uma classe de busca de dados por outra.
* @Bean DaoAuthenticationProvider: uma classe que é um provedor de autenticação simples, pode ser injetada. Que permite usar uma implementação do UserService e um Encoder a nossa escolha.

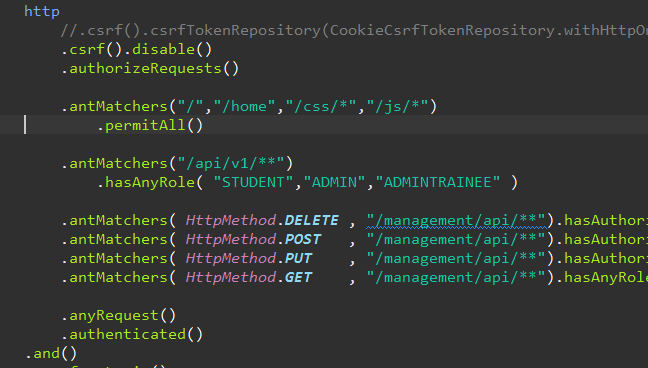
## **Recursos de Autorização**

Para usar os recursos de autorização é necessário sobre escrever o método abaixo,

*"@Override protected void configure(HttpSecurity http) throws Exception {}"*

O objeto http é uma api fluente:

Tirando o csrf tudo comessa com o **authorizeRequests()**



## **Permitir todos**

.anyRequest().permitAll();

## **Bloquear todos**

.anyRequest.authenticated()

## **Bloquear um recurso a todos menos quem tem autoridade**

antMatchers( “/minha-rota” )

.hasAuthority( “USER:ADD\_USER” ) ou .hasRole(“ADMIN”)

## **Um ROLE é ADMN se o valor da grantedAuthority registrado for:** ROLE\_ADMIN

## .hasRole(“ADMIN”) é igual .hasAuthority(“ROLE\_ADMIN”)

## **Filtros de url e role por antMatchers**

antMatchers(‘/minha-rota’)

antMatchers( HttpMethod.[METODO HTTP]) , “/minha-rota” )

.permitAll() ou .hasAuthority( “MyAuthority” ) ou .hasRole(“MyRole”)

Permite que uma rota escolhida no antMatchers seja protejida por uma authority ou role

anyRequest() -> qualquer rota

.authenticated()

**.and() -> recomeça o loop**

[METODO HTTP: GET|POST|PUT|DELETE]

Exemplo antMatchers(método opcional,”/localSeguro”)

.hasAuthority(“ADMIN”) ou .hasAnyAuthority(“ADMIN”,”SUPER\_ADMIN”)

.antMatchers(método opcional, “/localPublico”)

.permitAll()

**.anyRequest().authenticated() -> proteger todas as rotas que não tenham permit all.**

## **Exemplo de uma rota para o antMatcher**

“/minha\_rota” -> administrar minha\_rota

“/minha\_rota/\*\*” -> administrar

## **Configurar Login**

.formLogin()

.loginPage(“/loginPage”)

.permitAll()

.defaultSuccessUrl(“/rota de sucesso”,true)

As vezes o usuario já acessou um recurso e caio na rota de login , com o true o usuario então é não é enviado de volta para esse recurso mas sim para a rota de sucesso

.passwordParameter(“campo-password-no-html”)

.usernameParameter(“campo-username-no-html”)

.and()

## **Configurar Remember Me**

Referencia:

.rememberMe()

.tokenValiditySeconds( (int) TimeUnit.DAYS.toSeconds(21) )

.remembeMeParameter(“campo-remember-me-no-htm”)

.key(“minha\_chave\_secreta\_para\_gerar\_o\_cookie\_de\_remember\_me”)

## **Configurar Logout**

Referencia: <https://www.baeldung.com/spring-security-logout>

.logout()

.logoutUrl(“/url-to-logout”)

//.logoutRequestMatcher( new AntPathRequestMatcher("/logout","GET"))

Ele fica no modo get quando o csrf é desativado, o modo get basicamente significa que se eu mandar um get na url de logout ele vai sair, mas se o csrf está ativado então é necessário mandar um post nessa rota.

.clearAuthentication(true) // manda o securityContext limpar a autenticação // geralmente já é automático esse item

.invalidateHqttpSession(true)// manda o securityContext eliminar a autenticação //

Também é automatico

.logoutSuccessUrl(“/path-to-other-page”)

.deleteCookies(“JSESSIONID”)

.logoutSuccessHandler( LogoutSuccessHandler seuBeamSuccessHandler(){})