Segurança IV

Validando o Token

Uma vez que **entregamos o Token pro cliente-side,** temos que saber **como validar esse Token quando a requisição chegar**. A validação do **Token tem que ocorrer antes da resposta da API**, pensando nisso o **Spring oferece uma classe** pra esse cenário que é a **"OncePerRequestFilter"**.

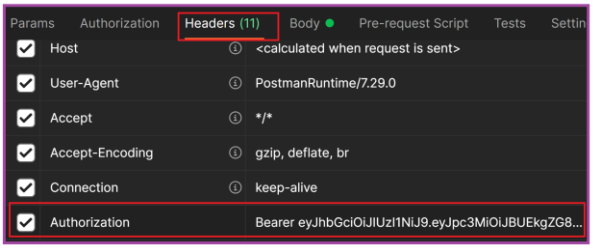
A **classe que contém a lógica de validação do Token** portanto **deve herdar** dessa **classe do Sring,** assim sendo **ela** sempre **vai ser executada antes da resposta,** precisamos **também implementar o método que faz esse filtro**, ele contém a **REQUISIÇÃO**, a **RESPOSTA** e o **FILTRO**:

```
public class AutenticacaoViaTokenFilter extends OncePerRequestFilter {

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {
```

Para **habilitar o Filtro temos que programar a configuração do Spring**, fazemos isso na **Classe de segurança que tem o "Configure" de autorização de request**, temos que dizer que nosso Filtro vai ser chamado antes **do filtro interno do Spring:**

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
            .antMatchers(HttpMethod.GET, "/topicos").permitAll()
            .antMatchers(HttpMethod.GET, "/topicos/*").permitAll()
            .antMatchers(HttpMethod.POST, "/auth").permitAll()
            .anyRequest().authenticated()
            .and().csrf().disable()
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and().addFilterBefore(new AutenticacaoViaTokenFilter(), UsernamePasswordAuthenticationFilter.class);
}
```

A **ideia é recuperar o Token enviado pelo Client-Side** na **requisição***(ele envia via Header no parâmetro de Authorization):*

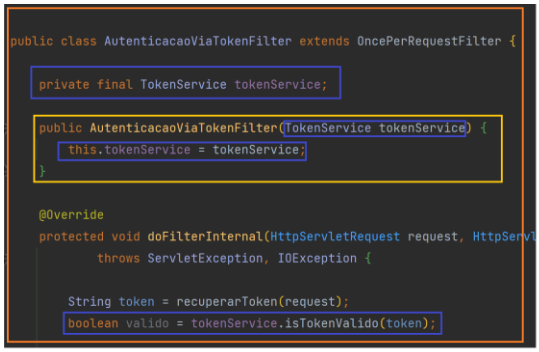



```
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletRespor
        throws ServletException, IOException {

    String token = recuperarToken(request);

private String recuperarToken(HttpServletRequest request) {
    String token = request.getHeader( s: "Authorization");

    if(token == null || !token.startsWith("Bearer ")){
        return null;
    }

    return token.substring(7);
}
```

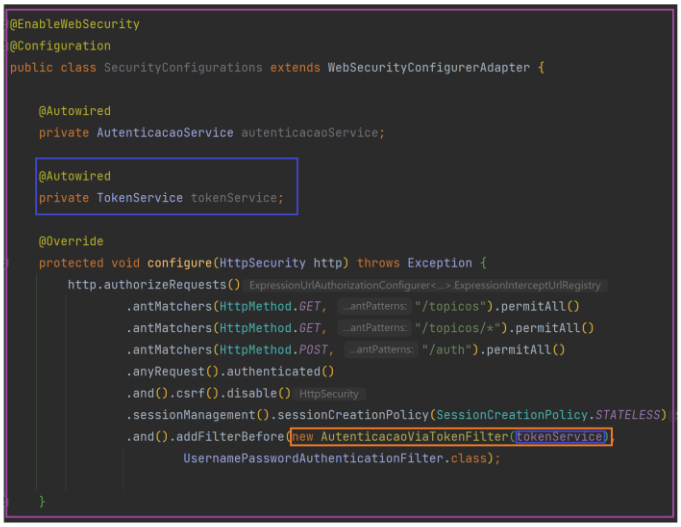
O substring é para pegar o Token sem o seu tipo que é identificado no início do Authorization*

Depois de recuperado o Token, é necessário fazer a validação dele. Podemos deixar a validação na **classe de Service do Token**, mas aqui entra um ponto importante, não é possível fazer injeção de dependência nessa classe de **AutenticaçãoViaTokenFilter.**

Isso porque **ela não foi anotada como uma componente do Spring**, então temos que receber **a Service do Token** pelo **construtor dela**, e como essa **Classe é instanciada** na **classe de Configuração de Segurança**, podemos passar o **ServiceToken** por **ela**:



```
public class AutenticacaoViaTokenFilter extends OncePerRequestFilter {

    private final TokenService tokenService;

    public AutenticacaoViaTokenFilter(TokenService tokenService) {
        this.tokenService = tokenService;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServl
            throws ServletException, IOException {

        String token = recuperarToken(request);
        boolean valido = tokenService.isTokenValido(token);
```



```
@EnableWebSecurity
@Configuration
public class SecurityConfigurations extends WebSecurityConfigurerAdapter {

    @Autowired
    private AutenticacaoService autenticacaoService;

    @Autowired
    private TokenService tokenService;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
                .antMatchers(HttpMethod.GET, ...antPatterns: "/topicos").permitAll()
                .antMatchers(HttpMethod.GET, ...antPatterns: "/topicos/*").permitAll()
                .antMatchers(HttpMethod.POST, ...antPatterns: "/auth").permitAll()
                .anyRequest().authenticated()
                .and().csrf().disable() HttpSecurity
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and().addFilterBefore(new AutenticacaoViaTokenFilter(tokenService),
                        UsernamePasswordAuthenticationFilter.class);

    }
```

O **método de validação do Token é simples**, basta fazer o **parser com o JWTS** passando a **chave da criptografia assinada do token**, esse **parser** retorna um **objeto caso a validação tenha dado certo** e **se não tiver, lança uma exceção**:

```
public boolean isTokenValido(String token) {
    try{
        Jwts.parser().setSigningKey(this.secret).parseClaimsJws(token);
        return true;

    }catch (Exception e){
        return false;
    }
}
```

Agora que o **Token foi recuperado e validado eu preciso dizer para o Spring** que eu **não preciso mais que ele faça a autenticação de Login e senha**, pois esse processo já foi feito no primeiro login do usuário, agora o **Spring deve reconhece-lo como autenticado pelo Token**.

O **Spring** nos permite ***"força-lo"*** a **reconhecer um usuário autenticado pelo contexto de segurança**, fazemos ***isso com uma chamada Estática*** da classe de **SecurityContextHolder** onde p**egando o contexto eu passo o usuário que deve ser autenticado de maneira "forçada".**

Para recuperar esse usuário podemos usar o próprio Token, já que ***ele contém o ID do usuário que foi setado na sua criação***, recuperando **esse ID**, basta **fazer uma consulta do usuário** e instanciar um **UserNamePasswordAuthenticationToken** passando o **usuário**, ***(a senha pode ser null)*** e as Authorities que **ele** possuiu (perfis de autoridade) :

***Lembrando que essa classe não permite injeção automática com @AutoWired por motivos ja comentados, então to recebendo o repository de usuário pelo construtor e passando pela classe de configuração de segurança já que ela instância a classe de autenticação do token ***

Então agora **todo o processo de Autenticação por Token** vai ser feito a **cada request do usuário**, **ele** vai **passar o Token devolvido no primeiro acesso,** e a **API vai recupera-lo**, **valida-lo** e então **autenticar o usuário**, no fim **devolve a resposta** da **requisição**: