04

Liberando acesso aos endpoints públicos

Transcrição

[00:00] Nós já habilitamos o Spring security no projeto e verificamos que está funcionando. Tentamos acessar a url /tópicos e recebemos como resposta o código 401. Na aula de hoje, a ideia é configurar quais endereços quero liberar e quais quero proteger. Vou exigir que o cliente que está disparando a requisição esteja autenticado no sistema.

[00:27] Para fazer isso, todas as configurações de segurança vão ficar na classe que criamos na última aula. Lembra que tivemos que herdar a classe de WebSecurityConfigurerAdapter? Nessa classe que estamos herdando existem alguns métodos onde as configurações de autenticação e autorização têm que ficar. Temos que fazer então uma sobrescrita, porque por padrão ele tem um comportamento específico, mas quero sobrescrever com as regras da minha aplicação.

[01:04] Qual método tenho que sobrescrever? O método se chama configure. Mas tem um problema. Existem três métodos com esse nome. Nós vamos precisar usar os três. Vou escolher o primeiro, dar um enter. Pegar o segundo. E pegar o terceiro.

[01:54] Qual é a diferença entre eles? O primeiro, que recebe um authentication manager builder é um método que serve para configurar a parte de autenticação. A parte de controle de acesso, de login, fica nesse método.

[02:22] O segundo, que recebe um tal de http security, serve para fazer configurações de autorização. A parte de URLs, quem pode acessar cada url, perfil de acesso. E o terceiro, que recebe um tal de web security, serve para

fazermos configurações de recursos estáticos. São requisições para arquivo CSS, Javascript, imagens, etc. Não é nosso caso, já que estamos desenvolvendo só a parte do backend. O frontend fica na aplicação cliente. É separado. Mas se fosse uma aplicação em que o frontend está integrado, iríamos ensinar para o Spring que as requisições devem ser ignoradas, que não é para interceptar na parte de segurança.

[03:28] São esses três métodos configure que precisamos ter. Nesta aula, vamos utilizar o segundo. É nele que faço a configuração de autorização, das URLs do meu projeto, o que é público e o que preciso ter controle de acesso.

[03:48] Percebe que esse método recebe um tal de http (da classe HttpSecurity)? E esse parâmetro tem alguns métodos, por exemplo, http.authorizeRequests é o método que vamos precisar chamar para configurar quais requests vamos autorizar, e como vai ser essa autorização.

[04:15] Tem vários métodos, mas o que interessa para nós é o tal de antMatchers. Nós vamos falar para ele qual url quero filtrar e o que é para fazer, se é para emitir ou bloquear. Tem três versões dele. Na primeira, você pode passar só qual é a url, na sequência você diz se quer permitir ou não.

[05:00] No nosso projeto, como vai ficar essa parte? Na nossa API quero deixar público o endpoint que lista todos os tópicos, ou seja, o /tópicos, e o que detalha um tópico em específico, o /tópicos/id. Os outros três, para cadastrar, alterar e excluir, quero restringir. Não é para ser público.

[05:28] Do jeito que eu fiz, ele está liberando /tópicos independente do método. Eu quero liberar o /tópicos, mas não todos os métodos, só o método GET, que é para o método de listagem. Existe outra sobrecarga desse método antMatchers que antes de passar a url, posso passar que método quero filtrar.

[07:08] Estou só liberando acesso aos meus endpoints públicos. A princípio, está ok, já fiz minhas configurações. Vamos testar então no Postman. Eu tinh

tentando acessar o /tópicos e tinha dado 401. Agora, na teoria, é para liberar o acesso.

[07:50] Vamos testar o detalhar. Veio os dados da dúvida. Será que ele bloqueou as outras coisas que eu não configurei? Por exemplo, o DELETE. Vamos testar. Ele me deu o código de proibido. Como eu não liberei, o padrão é bloquear. Esse era o comportamento esperado.

[08:37] No próximo vídeo, vamos configurar os outros endereços. Como faço então para cadastrar e excluir? Se está tudo proibido, como faço para liberar? Vou ter que configurar a parte de autenticação.