06

Restringindo o acesso aos endpoints privados

Transcrição

[00:00] Nós já liberamos acesso à nossa API, ao endpoint de listagem, de detalhes de tópicos. A ideia dessa aula é restringir o acesso aos outros endereços e exigir a parte de autenticação.

[00:15] No método configure liberamos acesso para requisições do tipo GET, para /tópicos e /tópicos/alguma coisa. Na sequência, o ideal é colocar isso também: anyRequest().authenticated(). Qualquer outra requisição tem que estar autenticada. O Spring vai devolver 403, e só vamos conseguir acessar, disparar requisições para esse endereço se o cliente que estiver disparando as requisições estiver autenticado.

[00:48] Só que não temos implementada essa parte de autenticação. Vamos ter que começar a implementação a lógica. Essa parte é um pouco chata, porque são vários códigos, várias configurações. Para começar, vamos precisar fazer o seguinte: para o cliente disparar requisições, ele vai precisar estar autenticado, mas aí precisamos ter a ideia de autenticação no projeto, vamos ter que ter, por exemplo, uma classe que representa o usuário, a senha dele, e o perfil de acesso.

[01:24] No nosso projeto, não sei se vocês chegaram a olhar, mas no pacote modelo, já temos uma classe chamada usuário. A ideia é que essa classe seja utilizada para o usuário fazer autenticação do sistema. Essa classe é uma classe de domínio da nossa aplicação, e transformamos em uma entidade de JPA. Então, existe já no banco uma tabela para guardar os usuários.

[01:45] Temos nessa classe o id, o nome, o e-mail e a senha. O e-mail e a senha são as informações que ele vai utilizar para se autenticar no sistema. Porém, o Spring security precisa saber que essa é a classe que representa o usuário. Precisamos dizer para ele. Como ensinamos isso?

[02:08] Vamos precisar implementar uma interface. Existe uma interface no Spring security que precisamos implementar na classe usuário. Preços implementar uma interface chamada UserDetails, que é a interface para dizer que essa é a classe que tem detalhes de um usuário. Dá erro de compilação porque no Java se você implementou uma interface, você é obrigado a sobrescrever, implementar os métodos que estão definidos nela.

[02:51] São vários métodos que ele gerou, que precisamos sobrescrever. Por exemplo, o método Getpassword. Para saber a senha do seu usuário, o Spring vai chamar o método Getpassword, então precisamos devolver qual atributo representa a senha. No nosso caso, é this.senha.

[03:26] Tem alguns métodos que desenvolvem boolean que é caso você faça controle na sua aplicação, da conta do usuário, se a conta está bloqueada, se tem data de expiração ou coisas do gênero, você devolveria os atributos que representam essas informações. No nosso caso não vamos ter esse controle mais fino, mais detalhado. Vamos devolver true em todos os métodos.

[04:10] Faltou o método que tem que devolver uma collection de grantedAuthority. Para o Spring security, além de ter uma classe usuário, precisa ter uma classe também que representa o perfil do usuário. Qual o perfil relacionado com as permissões de acesso dele. Por isso ele tem mais esse método, que é para devolver qual atributo contém a coleção com os perfis desse usuário. Só que, é claro, não existe esse atributo.

[04:45] Na classe usuário, além do id, nome, e-mail e senha vamos ter que ter mais um atributo private. Vou dar um new só para inicializar a coleção, para não ficar nula, vou importar o ArrayList.

[05:15] Nós não temos no nosso projeto uma classe perfil. Vamos criar essa classe, que vai ser uma entidade da JPA. Vou ter que ter uma tabela para guardar os perfis de acesso do nosso projeto. Vou criar a classe perfil no pacote modelo mesmo. Essa classe tem que ser uma entidade, isso já vimos como fazer no outro treinamento. E precisamos ter um id. Essa classe, além do id, só vai ter um atributo, que é uma string com o nome do perfil. Vou gerar os getters e setters do id e do atributo nome.

[06:11] Tudo ok. Só que se olharmos no método, ele ainda dá erro, porque a coleção tem que herdar de grantedAuthority, que é uma interface. A classe que representa o perfil também precisamos implementar uma interface do Spring. E a interface que vamos implementar é a grantedAuthority. Nela só tem um único método, que é o getAuthority, para devolvermos qual atributo tem o nome do authority, o nome do perfil.

[06:52] Ele fica dando o warning porque tem que ter o atributo do serialVersionUID. Vou mandar o Eclipse gerar esse atributo, só para parar de dar erro. Não é obrigatório porque o Java gera o atributo.

[07:13] Pronto, já ensinei para o Spring que a nossa classe usuário representa, e a classe perfil é a classe que tem perfil de acesso ao usuário. Só faltou um detalhe. Como a classe perfil é uma entidade, preciso dizer qual a cardinalidade do relacionamento, de usuário para perfil. Como um usuário pode ter vários perfis e um perfil pode estar atrelado a mais de um usuário ao mesmo tempo, vai ser um @ManyToMany.

[07:47] Por padrão, na JPA, todo relacionamento que é ManyToMany, se eu carregar do banco de dados, ele não carrega a lista, porque é lazy, só que eu vou colocar o fetch para ser Eager, porque quando eu carregar o usuário já carrego a lista de perfis, porque vou precisar dos perfis de acesso do usuário.

[08:09] Já mapeei a classe usuário, a classe perfil, seguindo as interfaces do Spring. Fiz o primeiro passo. Agora, na classe de configuração, depois que configurei as URLs e disse que qualquer outra url tem que estar autenticada,

vou dar um enter e continuar, usando and().formLogin(). Existe esse método que é para falar para o Spring gerar um formulário de autenticação. O Spring já tem um formulário de autenticação e um controller que recebe as requisições desse formulário. Então vou chamar esse método porque quero utilizar esse formulário padrão do Spring.

[08:48] Já está tudo computando. Vou rodar o projeto. Ele vai inicializar, vamos ver se vai dar algum erro. Perfeito. Vou testar isso diretamente no browser só para vermos a página de login. Se eu tentar entrar no localhost:8080/. Se eu der um enter aqui, ele me joga para uma tela de login. O Spring detectou que o / não estava liberado, que eu configurei o endform login, ele redirecionou para a página de login. Nós não criamos essa página no projeto, ela é do próprio Spring. Está até em inglês, e é legal, porque ele colocou o Bootstrap.

[10:00] Porém, eu não ensinei para ele a lógica de autenticação. Qualquer coisa que eu digitar vai dar um erro. Não tem um provider de autenticação, não ensinamos isso. É o assunto do próximo vídeo.