04

Validando o token

Transcrição

[00:00] Nós já conseguimos recuperar o token do cabeça authorization e imprimimos para ver se estava chegando certinho. O próximo passo é ir ao nosso filter. Preciso verificar se o token está válido, se é um token gerado pela aplicação ou se algum cliente mandou um token, mas digitou qualquer coisa aleatoriamente.

[00:33] Para fazer isso, preciso usar aquela API naquela biblioteca que estamos usando, do JSON web token. Tínhamos isolado o código dela naquela classe token service. Nessa a classe vamos criar um método que recebe um token e me devolve um boolean me dizendo se está válido ou não.

[00:52] Eu vou simplesmente fazer a chamada para essa classe e guardar o boolean que foi devolvido. Vou criar uma variável boolean, chamar de valido = tokenservice.istokenvalido(token).

[01:21] Dá um erro porque não existe esse atributo. Vou precisar criar um atributo do tipo TokenService. A ideia seria injetar esse TokenService. Porém tem um problema. Nesse tipo de classe não conseguimos fazer injeção de dependências. Não dá para colocar um @AutoWired, até porque na classe de security Configuration nós que instanciamos manualmente a classe.

[01:58] Eu posso receber via construtor. Já que não posso injetar pelo atributo, vou injetar da mesma maneira, só que via construtor. Quem for dar new no nosso filter vai ter que passar o token service como parâmetro. Vou salvar. Só que na verdade quem está dando new na classe token filter somos nós mesm pela classe security Configuration.

[02:32] Mas aqui estou na classe security Configuration. Nessa a classe eu consigo fazer injeção de dependências. Resolvemos o problema. Não consigo injetar no filter, mas na classe Configuration eu consigo. Na hora de dar new no filter, eu passo o token service que foi injetado. Problema resolvido. Ele está só reclamando porque o método isTokenValido não existe. Temos que criar.

[03:10] Aqui temos aquele método para gerar o token. Preciso ter esse método para fazer a validação, para validar se o token que está chegando está ok ou não. Para fazer isso, vamos usar de novo o tal de jwts. Só que não vou chamar o builder, porque não quero criar um novo token. Vou chamar o método parser, que é o método que tem a lógica para fazer o passe de um token. Você passa para ele um token, ele vai descriptografar e verificar se está ok.

[03:43] Na sequência, temos que chamar primeiro setSigningKey. Tenho que passar aquele secret da nossa aplicação, que é a chave que ele usa para criptografar e descriptografar. Tem um método chamado parseClaimsJws. Esse é o método que vamos chamar passando como parâmetro o token.

[04:18] Esse método devolve o Jws claims, que é um objeto onde consigo recuperar o token e as informações que setei dentro do token. Mas quando eu fizer essa chamada, se o token estiver válido, ele devolve o objeto. Se estiver inválido ou nulo, ele joga uma exception. Eu vou fazer um try catch, vou colocar o código dentro do try. Se ele rodou tudo ok, o token está válido. Se chegou na linha de baixo é porque o token está válido, retorna true, porque não quero recuperar nenhuma informação do token nesse método. Se deu alguma exception, ele vai entrar no false.

[05:27] Por enquanto, a única coisa que preciso fazer é devolver se está válido ou se não está válido. Só isso. Salvei. Vou voltar para o meu filter. Nesse exercício, só vamos validar se está ok ou não. Vamos fazer aquele teste via Postman. Vou mandar um token correto, ver se está válido e se vai imprimir true. Depois mando um token aleatório, ver se vai imprimir falso. E vou mandar vazio.

[06:15] Preciso do cabeçalho authorization, bearer, crio a requisição para fazer o DELETE. No Postman, inclusive, tem um cabeçalho authorization. Posso ir direto por ali. Ele já seta automaticamente o cabeçalho, não preciso digitar. Vou disparar a requisição. Deu proibido, porque ainda não fiz a lógica de autenticar no Spring, mas para mim o que interessa é que imprimiu true. Vou disparar de novo com o token errado. Imprimiu false. Vou disparar vazio. Imprimiu false.

[08:05] Funcionou. Nosso código já está validando se o token que foi recuperado é válido ou não. Esse era o conteúdo de hoje. No próximo vídeo, já que eu já recuperei o token do cabeçalho, já vi que está válido, tenho que falar para o Spring autenticar o usuário. Se não estiver válido, simplesmente não vai autorizar.