02

Por que autenticar via token

Transcrição

[00:00] Na última aula, nós aprendemos sobre o Spring security, como proteger nossa API REST liberando endpoints e em outros exigindo autenticação. Porém, a autenticação que fizemos foi a tradicional, com usuário e senha, e o servidor, sempre que o usuário efetua login, cria uma sessão para guardar essas informações. Mas isso não é uma boa prática no modelo REST. O ideal é que a nossa autenticação seja stateless. Se vocês lembram, no curso anterior tivemos uma discussão sobre o modelo REST, e uma das características desse modelo é que toda a comunicação seja de modelo stateless. Ou seja, o cliente dispara uma requisição, leva todas as informações necessárias, o servidor processa, executa o que tem que executar, devolve a resposta e acabou.

[00:57] Eu tenho um slide mostrando essa ideia da autenticação via session. Em uma aplicação web tradicional, esse é o modelo utilizado. Toda vez que o usuário vai se autenticar no sistema, ele entra no formulário de login, digita o e-mail, a senha, quando faz o login o sistema cria uma sessão e nessa session ele armazena as informações do usuário. Para o servidor conseguir diferenciar um do outro, cada sessão tem id único. Em uma aplicação Java, essa sessão é chamada de jsessionid.

[01:53] Esse id é devolvido como resposta para o navegador, no formato de um cookie. O navegador armazena isso em um cookie, armazena o id da sessão. Nas próximas requisições que esse usuário disparar nesse navegador, o browser automaticamente envia esse parâmetro como um cookie. Quando chega uma requisição para o servidor, ele verifica se está vindo um cookie Jsessionid. Se estiver, ele recupera o id, com os dados.

[02:37] Dentro da sessão tem as informações. Isso vai contra um dos princípios do REST, que é de ser stateless. Nesse modelo, para cada usuário que estiver logado na aplicação vou ter um espaço na memória armazenando as informações. Isso consome espaço de memória, e se o servidor cair vou perder todas as sessões. Se eu quiser ter escalabilidade, se eu quiser ter um balanceamento de carga com múltiplos servidores, eu teria problema de compartilhamento.

[03:07] No modelo REST o ideal é trabalhar com a autenticação de maneira stateless. Com o Spring security é possível fazer isso. Conseguimos explicar para o Spring que não é para ele criar a sessão, que toda vez que o usuário se logar vou fazer a lógica de autenticação, mas não é para criar uma session. Só que aí, nas próximas requisições que os clientes dispararem, o servidor não sabe se está logado ou não, porque não tem nada armazenado. O cliente vai ter que mandar alguma informação dizendo quem é ele, se ele está logado, se tem permissão para acessar. Isso geralmente é feito via tokens. O pessoal costuma usar a especificação JSON web token para fazer esse tipo de autenticação. Ou seja, a cada requisição, o cliente vai mandar um token identificando quem é o usuário que está disparando essa requisição, se ele tem permissão para disparar.

[04:00] Nos próximos vídeos vamos aprender como utilizar o JSON web token, uma aplicação com Java utilizando Spring Boot.