01

# Token da API e segurança do usuário

### Transcrição

[00:00] Nas aulas anteriores fizemos o que? Fizemos o sistema de login e log out do usuário e vimos como funciona o sistema de autenticação do play que é baseado em anotações com uma classe customizada que altera as regras de negócio de autenticação do seu sistema. Vamos fazer então a autenticação da própria API de produtos, em geral fazemos isso enviando uma chave de acesso junto com cada uma das requisições, seja por um cabeçalho ou seja por um parâmetro extra chamado chave ou token, ou qualquer coisa.

[00:40] Então vamos criar essa chave, esse token, só que do usuário e vamos mostrar também essa informação no painel do usuário deixando o painel mais bonitinho, vamos criar o modelo do nosso token? Eu vou vir aqui na nossa classe de usuário dar um “Ctrl + N” class, criar o token da API que estende model do Ebean. Agora configuramos ele para ser reconhecida no banco e adicionamos os atributos necessários que ela vai precisar para funcionar.

[01:20] Primeiro deles é um ID, Long id, o segundo deles, como é um token único de usuário, vai precisar de um usuário vai precisar de um usuário, então uma relação @OneToOne com um private de Usuario usuario, e vamos precisar também de pelo menos um código que nem fizemos no token de cadastro, então private String codigo.

[01:46] Uma outra coisa que é interessante termos, é uma data de expiração, não vamos efetivamente usar a data de expiração para invalidar o token do usuário, mas é uma coisa que você pode fazer depois que terminamos o nosso projeto que é bem útil e aumenta a segurança do projeto, no caso aqui vamos usar para aumentar a segurança do próprio código, utilizando ele para gerar um hash, então vamos criar uma data aqui, date do java.util chamado expiracao.

[02:22] Vamos criar um construtor que já gere o nosso código e já sete a data de expiração e isso a partir de um usuário, então public TokenDaAPI, ele vai receber um usuário e guardar ele no campo que temos aqui do nosso modelo, e setamos a data de expiração, eu vou setar com a data atual, mas você pode adicionar um ano nela, para que o token inspire em um ano e tenha que ser revalidado.

[03:00] Vou criar aqui uma data atual, new Date e eu vou criar o código, this.codigo vai ser do mesmo jeito que foi no token de cadastro, vai ser um código encriptado para aumentar a segurança, Crypt.sha1, vamos gerar primeiro uma string aleatória, Crypt.generateSecureCookie, a partir daí podemos também adicionar a data de expiração como string para garantir que fique mais difícil de alguém adivinha, então expiracao.ToString e podemos também adicionar o hash do objeto do usuário, então usuario.ToString.

[03:50] E temos um código bem difícil de ser adivinhando, então bem seguro. Geramos uns getters e setters, seleciona tudo ok. E nosso modelo está pronto, agora precisamos inserir ele no banco, criar evolução para que a tabela dele seja gerado no banco, então eu vou copiar aqui o 4.sql só que eu não vou gerar ele na mão, eu vou fazer que nem da última vez e pegar uma colinha aqui para gerar a tabela.

[04:20] Então vamos ver aqui, ele está gerando a tabela com ID, usuário, código e expiração, com uma referência para o nosso modelo de usuário, e aqui os drops tanto da referência da chave do usuário, quanto da tabela em si. Então já temos o nosso sql, podemos pegar o nosso token e quando o usuário terminar de se cadastrar geramos um token para ele.

[04:50] Mas antes aqui no nosso modelo eu vou adicionar uma referência para esse token, para podermos acessar tanto o usuário a partir do token quanto o token a partir do usuário, então é uma referência de tabela, então vai ser @OneToOne private tokenDaAPI token, vamos gerar os getters e setters, só que aqui eu não quero que tenha uma

referência na tabela do usuário, quero que tenha só no modelo, então vamos falar que isso aqui é mapeável pela tabela do token, pelo atributo usuário.

[05:30] Agora sim podemos começar a usar o nosso token, aqui no controller, sempre que confirmamos o cadastro de um usuário, então confirma usuário, podemos gerar um token para ele, então depois dele ser verificado eu vou gerar um token, new tokenDaAPI passando um usuário como parâmetro aqui para ele já gerar os nossos modelos lá dentro, vou salvar esse token, tokenDaAPI.save e vou atualizar o objeto do usuário, usuario.setToken com o nosso token da API. Aqui ele já atualiza o nosso usuário e já temos tudo salvo no banco.

[06:11] Agora podemos resolver um problema de segurança que estávamos tendo aqui que eu não cheguei nem a comentar, aqui na nossa sessão estamos inserindo o email do usuário, só que isso é muito ruim porque isso fica exposto para quem estiver fuçando o cookies do seu browser, então seria mais legal passar um código, alguma coisa assim, e agora que já temos um token com código super seguro, podemos ao em vez de inserir o email, inserimos o próprio código.

[06:45] Como essa linha de código aqui é repetida duas vezes, então já vamos ter que alterar ela duas vezes, eu vou extrair para um método que faz isso para gente, então, insereUsuarioDaSessao, passando o usuário, ao em vez de mudar em todos lugares que esse método é chamado, mudamos só aqui. E em vez de usuario.getEmail vamos pegar o token e vamos pegar o getCodigo do token.

[07:15] Já estamos bem mais seguro aqui, já não guardamos mais nosso email na sessão, guardamos um código aleatório que não saberíamos reproduzir direito e precisamos fazer uma mudança aqui, olha só, quando fazemos a autenticação, estamos pegando o email da sessão, precisamos pegar agora o código, então vamos mudar isso aqui para código e vamos vir e no DAO ao em vez de pegar um usuário com email, vamos pegar um usuário a partir do token.

[07:57] Vamos criar esse método e fazer a lógica aqui que é igual a todos que fizermos anteriormente, usuários onde o código do token, então equal token.codigo, seja igual a esse código que recebemos do parâmetro, findUnique porque supostamente só vai ter um, guarda isso em uma variável de usuário e podemos passar aqui Optional.ofNullable como temos feito para seguir boas práticas.

[08:45] Agora a lógica do usuário autenticado volta a funcionar bonitinha, então vamos ver no nosso browser como que ficou? Ele já está pedindo para fazer evolução, para gerar a nossa tabela no banco, eu vou dar um apply e criamos um novo usuário e tentamos fazer login com ele para dar um ok. Usuario/novo eu vou criar aqui marco com email marco@caelum.com.br, senha marco, confirmação marco.

[09:15] Eu já limpei o nosso banco para garantir que não teríamos problemas em criar um novo usuário, então vamos lá, eu vou pegar aqui o nosso email, copiar o link de confirmação, vou dar um “Ctrl + C” aqui e vou acessar esse link para confirmarmos nosso usuário, supostamente já somos redirecionados para o painel caso tudo dê certo.

[09:40] Então eu vou dar um ok aqui, estamos no painel do usuário “Bem Vindo”, só que não sabemos qual é o nosso token, como que eu vou fazer um acesso a API se eu não tenho o token para mandar como parâmetro, então vamos alterar esse painel para deixar ele um pouco mais bonito, com umas mensagem, uma explicação de como usar a API e também mostrar o token do usuário quando ele precisar?

[10:05] Eu tenho uma cola aqui que tem toda a estilização, mas não tem a lógica de com vamos fazer isso, então eu vou pegar o nosso painel e vou substituir aqui esse código Html, se eu der um F5 aqui, você vai ver que não tem nada sendo gerado dinamicamente, temos o painel do usuário, bem vindo usuário genérico, esse é o seu painel, aqui você tem a chave de acesso pessoal, se você clicar aqui você consegue ver a chave.

[10:43] Só que para isso falta uma coisa, isso aqui é um código JavaScript, não importamos nenhum JavaScript no nosso projeto, então vamos lá no main e vamos importar o Bootstrap e o jQuery no fim da nossa main, é o lugar mais ideal

para você importar JavaScript, então eu vou importar um JavaScript aqui, para importar JavaScript usamos o source, temos que importar os arquivos que estão na pasta public.

[11:10] Eu vou subir aqui em cima para passar por uma coisa que não vimos exatamente como funciona, só nos exercícios anteriores, como você importa um asset da pasta public, você usa essa rota aqui routes.assests.versioned, então vamos vir aqui e vamos usar essa rota para importar o JavaScript. Eu vou copiar aqui nos dois lugares e vamos importar tanto o Bootstrap quanto o jQuery.

[11:45] Os dois estão na mesma pasta, estão em bootstrap/js/o nome do arquivo, no caso bootstrap.mim.js e aqui jquery.mi.js, com esses dois JavaScripts já conseguimos fazer o botão funcionar, só que tem um detalhe aqui, o Bootstrap requer o jQuery e estamos importando o jQuery antes, então subimos aqui e agora o jQuery está antes do Bootstrap e assim ele vai funcionar.

[12:20] Eu vou fechar aqui o main, e vamos atualizar aqui o painel para ver o botão funcionar, ele deve mostrar a palavra token e olha só, ele esconde, para não ficar exposto, caso alguém passe atrás de você e fique lendo e descubra o seu token. Vamos ver como funciona esse painel, vamos precisar de um usuário para ele, então eu vou ter que importar aqui e receber por um parâmetro um usuário da classe usuário.

[13:00] Vamos usar ele aqui e vamos pegar o token dele para escrever aqui no lugar do token, mas antes vamos resolver o erro de compilação que estamos tendo agora e passar esse usuário para nossa tela, aqui no faz login não, no painel, vamos ter que pegar o usuário da sessão e carregar ele do banco de dados, então vamos fazer isso. Session(AUTH) que é um código do nosso usuário.

[13:40] E agora temos que pegar ele a partir do usuarioDAO, então usuarioDAO.comToken(codigo), só que assim, aqui ele já está autenticado, ele já passou por toda aquela lógica de pegar o usuário com token, conferir se está presente ou se não está, e assim ele chega aqui no nosso método, então eu não preciso conferir se o usuário está existente aqui de novo, então eu vou só dar um get mesmo e jogar na variável usuário e passar para view.

[14:06] Não precisa se preocupar com a confirmação de que esse modelo existe, porque ele realmente já existe, vamos ver, compilado com sucesso, eu vou dar um F5 aqui para remover os erros de compilação do eclipse e podemos finalmente alterar a nossa página para usar os dados do usuário, vamos lá, então @usuario.getNome() e aqui em baixo vamos pegar o código do token, então @usuario.geToken().getCodigo().

[14:45] Agora atualizando ali, temos os dados bonitinho do nosso usuário, “Bem vindo, marco, sua chave de acesso é essa”. Então o que fizemos nessa aula? Fizemos o token do usuário, atrelamos os dois modelos de modo circular e fizemos um painel todo bonitinho aqui, para explicar como fazer a requisição e mostrar o token de um modo relativamente seguro, com um botãozinho que esconde ele.

[15:15] Na próxima aula, vamos fazer a autenticação em si usando do mesmo sistema de autenticação do usuário, para que toda chamada a nossa API, seja autenticada com as credencias do usuário.