06

Registro de acesso com Actions

Transcrição

[00:00] Na aula anterior vimos como pegar o conteúdo do cabeçalho da requisição para assim autenticar o nosso usuário, autenticar o acesso a nossa API, na aula de hoje eu tinha falado que íamos armazenar esses acessos do usuário, para que isso possa ser usado para gerar relatórios ou para que o cliente possa ter uma noção de quanto que ele está usando o serviço, para isso vamos ver uma funcionalidade do play que não vimos ainda que são as ações.

[00:27] As ações ou actions servem como um tipo de interceptador da requisição, a action ocorre antes da lógica do controller, mas caso a requisição seja autenticada, a autenticação acontece primeiro, então se o usuário não estiver autenticado corretamente, a ação não ocorre. E para que serve essa ação? Ela serve para você executar um trecho de código antes da requisição passar pela lógica do controlador.

[00:57] Sendo que podemos atribuir essa ação a vários métodos, evitando assim uma repetição de código, e como que se faz uma action? Vamos criar aqui uma ação, eu vou dar um “Ctrl + N” e criar uma nova classe do pacote ações, vamos criar uma ação de registro de acesso, então AcaoDeRegistroDeAcesso e eu vou estender a classe que ela estende aqui, ela estende uma ação simples, uma Action.Simple e para que consigamos fazer a ação, podemos ver que tem um erro aqui ela precisa implementar esse método chamado call.

[01:46] Que recebe um contexto, que é o contexto da nossa requisição, eu vou renomear aqui para contexto e vemos o que faz, em primeiro lugar como vamos executar uma ação sem alterar a requisição, eu vou vir aqui e retornar delegate.call e passar o contexto de volta, não vamos alterar o nosso contexto. O que vamos fazer aqui nessa ação? A princípio vamos só mandar alguma coisa no registro de texto, no log?

[02:20] Vamos lá, então Logger.debug e vamos mandar uma mensagem aqui, ação de registro, e como fazemos para essa ação ser executada? Vou vir aqui no API controller e eu vou anotar um controle inteiro com a ação, @With(AcaoDeRegistroDeAcesso.class), então agora todos os métodos do nosso controller passam pela ação de registro de acesso antes de serem executados.

[03:01] Eu poderia fazer essa anotação em cada um dos métodos, mas como eu quero que toda ação do API controller seja registrada eu vou anotar direto no controller. Vamos testar? Eu vou vir aqui no nosso console e usar aquele código de acessar a url passando nosso header aqui com o token. Eu vou fazer um acesso e recebemos aqui o nosso produto, porque o meu token está correto, eu já tinha pego na aula anterior.

[03:33] E aqui se viermos no log, conseguimos ver o registro que escrevemos ali na nossa ação, mas se eu errar o meu token, ou passar um token inválido? Eu vou alterar aqui e receber o meu erro, deixa eu limpar aqui, vamos lá, podemos ver que não aconteceu nenhum outro registro aqui no log da aplicação, quando fazemos uma requisição, ele primeiro autentica, se a autenticação passar, ele vai fazer a ação e então ele executa a lógica do controller.

[04:10] Claro, podemos criar um modelo para fazer o registro desses acessos, eu vou vir aqui e criar uma nova classe chamada registro de acesso, estendendo a classe model do pacote avaja Ebean, essa classe tem que ser uma entidade, aquela coisa que já vimos, entity, tem que ter um ID, GeneratedValue, private Long id, ela vai precisar ter o que mais também? Ela vai ter um usuário só que vai ser um pouco diferente porque um usuário pode ter vários acessos.

[04:50] Então vamos fazer em vez da anotação @OneToOne, vamos usar a anotação @ManyToOne, vamos precisar também da URI que foi acessada, então private string uri, e vamos precisar da data de acesso, para caso queiramos usar isso para alguma coisa, então private Date data.

[05:22] E como todos esses campos são imutáveis, vamos instanciar tudo no nosso construtor, então vamos lá, public registro de acesso e vamos receber o Usuario usuario, vamos receber a uri também que é algo externo, então String uri, e vamos gravar a nossa data no momento atual, então vamos lá, eu vou atribuir esse usuário ao campo usuário, a uri ao campo uri e vou criar aqui this.data = new Date().

[06:05] Está tudo ok, agora falta criar os getters, eu não vou criar os setters porque todos esses dados são imutáveis, então select getters. No modelo de usuário também queremos uma referência a esses acessos, então vamos vir aqui no modelo de usuário e vamos adicionar esses acessos, mas com a anotação @OneToMany, que é a anotação contraria do @ManyToOne que indica que o usuário tem vários acessos.

[06:37] Então @OneToMany e vamos mapear isso para variável usuário do outro modelo, private, vai ser uma lista de registros de acesso e eu vou chamar isso de acessos, preciso importar aqui a classe lista e eu vou gerar o getter e setter desse modelo aqui. Agora temos toda nossa modelagem pronta, podemos gerar o sql, no caso para não ficar escrevendo o sql todo eu vou fazer a mesma coisa que nas aulas anteriores, copiar aqui uma evolution e pegar da minha colinha todo sql já pronto de registro de acesso.

[07:25] Com usuário, uri, data e os índices necessários, tanto ups quanto down, peguei isso aqui tudo vou copiar aqui nessa evolution que eu acabei de criar, tendo nosso código sql pronto, podemos atualizar nosso banco fazendo um acesso qualquer aqui, temos agora uma evolução para o registro de acesso, eu vou atualizar o nosso banco e podemos efetivamente fazer o registro desse acesso, salvar ele no banco.

[08:00] Então vamos lá na classe registro de acesso, deixa eu fechar essas coisas que não vamos mais usar, vamos aqui na ação, para ter nosso usuário eu preciso pegar o código de acesso e para pegar o código de acesso não temos o usuário logado aqui, temos que pegar do header, então vamos pegar do contexto.request().getHeader() e vamos pegar o header API-Token, que era aquele que estávamos usando para guardar o código do token do usuário, guarda isso em uma variável codigo.

[08:35] O que mais precisamos da nossa requisição, precisamos da uri, então context.request().uri() assim já temos qual a uri que o nosso usuário acessou, então agora precisamos do usuário para poder gerar o nosso registro de acesso, então eu vou injetar aqui o usuarioDAO para podermos pegar ele do banco, UsuarioDAO usuarioDAO; e podemos pegar aqui nosso usuário, então usuarioDAO.comToken passamos o código, só que do mesmo jeito que na aula anterior já estamos autenticado quando chegamos aqui.

[09:25] Então eu vou dar um get direto aqui e pegar o nosso usuário sem precisar conferir se ele existe ou não, e a partir daí podemos fazer um novo registro de acesso, passando o usuário e a uri, guardando isso em uma variável e acesso.save(). Salvamos nosso acesso no nosso banco, agora podemos fazer alguns testes, vamos primeiro ver aqui no nosso banco como que ficou, eu vou dar um select * from registro_de_acesso.

[10:08] E vamos ver que ele está vazio, agora eu vou fazer uma requisição e vamos ver esse registro ser salvo no banco, se eu fizer aqui com um token inválido ele não tem que adicionar nada, mas se eu alterar aqui e fizer meu token ser válido, vamos ver que ele recebe os produtos e que ele vai salvar um registro no banco com ID do usuário 18, que por acaso é o me usuário.

[10:40] Agora vamos fazer com que o usuário tenha conhecimento do número de acessos que ele já fez, vamos lá no painel e fazemos uma entrada para mostrar para ele quantos acessos ele já fez, então painel.scala.html e adicionamos aqui no fim da classe um outro parágrafo que tenha o nosso registro de acessos. Então vai ser outro panel-body.

[11:25] Ele vai ter uma frase “você já fez” o número, vamos colocar aqui um Strong, @usuario.getAcessos().size(), “você já fez n acessos a nossa API”, parece bom, vamos ver como fica isso na tela? Vou fazer o login aqui, no caso eu acho que já estou logado, então vou para o painel, usuário/painel, não estou logado, então vamos lá, marco@caelum.com.br,

senha é marco, vemos aqui, “você já fez um acesso a nossa API”, agora vou fazer mais um para garantir que isso está direito.

[12:12] Fiz mais um acesso, vou dar um F5 aqui, dois acessos. O que vimos nesse vídeo? Vimos como criar ações que ocorrem por toda requisição e se a requisição for autenticada essa ação só ocorre caso a autenticação tenha ocorrido corretamente, vimos também como recuperar a uri da requisição e usar isso para registrar um acesso a API para o nosso usuário, na próxima aula vamos finalmente fazer a autenticação de administradores, fazendo com que só eles possam cadastrar produtos. E vamos fazer as telas de gerenciamento dos administradores, acabando o nosso projeto.