03

Evolutions

Transcrição

[00:00] O que temos até agora no nosso projeto? Temos um projeto com formulário de cadastro de produtos, banco de dados, três pontos de acesso para os produtos existentes com filtros pelos atributos do modelo, por exemplo aqui tipo ebook, e com já validação desses atributos para impedir que peçamos no banco de dados um atributo que não exista.

[00:24] Agora vamos começar a criar a parte do sistema referente ao uso do cliente, como cadastro, validação, confirmação, login e um painel de usuário para ele ver os próprios dados. Como com produto, vamos começar por formulário, já sabemos por experiência que é mais fácil fazer um formulário se tivermos os ajudantes e para usar os ajudantes utilizamos a fábrica de formulários que utiliza um modelo para criar um formulário personalizado para aquele usuário.

[00:55] Então vamos lá, vamos criar o modelo para eu poder criar um formulário a partir da fábrica, criamos aqui uma nova classe, chamada Usuario que estende do pacote, model aqui com.avaje.ebean.model. Vamos dizer que ela é uma entidade do banco de dados e que ela tem um id gerado automaticamente que é um private long id.

[01:33] O usuário precisa no mínimo de um nome também, então “private String nome” e os getters e os setters, “Ctrl + 3”, ggas, seleciona todos, cria os getters e setters. Mas e como isso se reflete no banco de dados? Já vimos rapidamente que o play utiliza aquele sistema de evoluções, para crias as tabelas automaticamente, atualizando a página aqui vemos que ele vai pedir para fazer uma evolução.

[02:07] Mas olha isso aqui, ele pede para dropar nossa tabela de produtos, antes de criar a tabela a tabela de produtos de novo e depois a de usuários, isso é meio estranho? Eu não quero perder todos os meus produtos ou meus usuários cadastrados toda vez que eu fizer uma mudança no banco de dados, não é um processo legal ficar reescrevendo todos nossos produtos.

[02:34] Então para um pouco, para tudo, vamos pausar o nosso cadastro de usuário e aprender um pouco como lidar com essas evolutions. Em desenvolvimento o play já era um arquivo conf/evolutions/default/1.sql, aqui que ele gera automaticamente os nossos creates tables, aqui nessa seção de ups, mas ele também cria se você quiser reverter o processo, aqui a seção de downs, que é o que contém os códigos, que é o que contem nessa tabela aqui.

[03:15] Mas eu não quero que ele fique dropando todas as nossas tabelas, toda vez que fizermos uma mudança, tem um jeito de contornar isso? Na real tem, se você for reparar aqui no topo, tem esse comentário dizendo que se você quiser que ele pare de gerar as nossas tabelas automaticamente, é só remover esse comentário, então vamos remover esse comentário, já parou de criar as nossas tabelas.

[03:43] Agora vamos ter que fazer isso por nós mesmos, e tem um jeito que vai deixar essa criação toda modularizada, que é fazer o que? Separar isso em arquivos diferentes, como as migrations que vemos utilizando Spring ou Rails. Como fazemos isso? Temos o arquivo 1.sql, será que não é só criar o arquivo 2.sql e criar o código ali? É exatamente isso, então vamos lá.

[04:13] Eu vou duplicar o arquivo 1, criar o arquivo dois e vamos separar a criação da tabela de produto da de usuário, do arquivo 1 removemos toda lógica da criação da tabela de usuário, removemos aqui o up dele e removemos aqui o down também. Então agora o arquivo 1 vai criar a tabela de produto e se ele for revertido vai remover a tabela de produto.

[04:38] No arquivo dois a mesma coisa, só que ao contrário, pegamos aqui a tabela de produto e removemos ela e o drop da tabela de produto, então o arquivo dois vai criar a tabela de usuário e caso desejemos reverter esse processo vamos

dar um drop na tabela de usuário. Vamos ver como fica isso agora na evolução que o play quer rodar.

[05:00] Atualizamos aqui, e olha só, ele vai sugerir aqui só para criar tabela de usuários, não vamos mais perder nossos produtos que aparecem quando acessamos aqui toda nossa api. Se dermos um apply aqui agora, vamos criar nossa tabela de produto, tem um detalhe importante, você não pode ter passado nenhum parâmetro, que nem aqui o tipo ebook e os preços, porque ao executar o script o play vai usar uma rota, ele vai adicionar uma rota aqui e se você tiver um parâmetro essa rota vai ser uma parte dos parâmetros.

[05:43] Então ele não vai conseguir executar a sua evolução. Aqui eu apliquei o script com uma rota sem parâmetros e parece que deu tudo certo, vamos dar uma olhada no nosso banco, vamos vir aqui mysql> show tables, e a nossa tabela de usuários está aqui.

[06:00] Se dermos um desc na tabela de usuário, vamos ver que ela tem que ter um ID e um nome certo? Aqui ID e nome. Mas e como fazemos se quisermos criar um novo campo dentro de uma tabela? Vamos criar um arquivo novo para isso, vamos fazer uma nova evolução e fazer tudo isso modularizado.

[06:21] Então agora ao em vez de 2.sql, vamos criar o 3.sql, mas o que vamos fazer aqui? Não sabemos ainda, então vamos criar novos campos para o usuário, usuário não basta só ter nome, vamos criar um email para ele e uma senha para ele poder fazer o login, então “private String email” e “private String senha”. Então agora podemos alterar nosso sql para representar isso, como fazemos isso? Não é mais um create table, é um alter table, “alter table usuário”.

[07:00] E o que vamos fazer? Adicionar a coluna email e a coluna email é um varchar(255) uma cadeia de caracteres até 255 caracteres, a mesma coisa com o campo senha, e o down? O down vamos apagar essa coluna, então “alter table usuario drop column email” e “alter table usuario drop column senha”.

[07:32] Então agora se rodarmos nosso evolution, ele vai adicionar esses dois campos e se precisarmos reverter ele vai apagar os dois campos, vamos ver lá, vou atualizar aqui e ele vai me pedir para fazer a evolução, eu dou um apply e podemos conferir no banco que ele criou os campos certinhos, conseguimos ver email e senha tudo bonitinho.

[08:00] Isso tem uma vantagem e uma desvantagem, vantagem que agora não perdemos mais o nosso trabalho, conseguimos manter nossos produtos cadastrados sem perder eles toda vez que fazemos uma alteração no banco, no entanto acabamos perdendo um pouco a nossa dinâmica de trabalho, porque precisamos gerar o código sql que antes era gerado automaticamente.

[08:33] E é isso, vimos como funciona os evolutions, vimos como usar as evolutions e como fazer evoluções manualmente, fizemos duas evoluções diferentes só nessa aula, agora podemos voltar a criar o nosso cadastro de cliente. Vamos partir para o formulário.