05

Visualizar e deletar

Transcrição

[00:00] Olá. Agora que já temos aqui os nossos métodos para salvar e atualizar, nós vamos precisar criar um método para que possamos visualizar e para que possamos também deletar registros da nossa tabela. Assim finalizamos o Crud. Então agora eu vou criar um método para que possamos visualizar os registros que estão inseridos na nossa tabela. Então vamos lá, `private` seguindo o padrão aqui, `void visualizar` .

[00:35] Para visualizar eu não preciso de parâmetro porque como nós vamos ver, o método do Spring Data para que possamos retornar todos os valores inseridos em uma tabela não vai parâmetro. Então vamos aí, `cargoRepository.findAll` . O `findAll` ele retorna para nós o `iterable` que é uma coleção e ele retorna todos os registros que nós temos dentro da nossa tabela. E aí passamos qual é o tipo do atributo que no nosso caso é Cargo.

[01:16] Então `findAll` vai retornar uma `iterable<Cargo>` e eu vou chamar isso de Cargos no plural. Agora eu quero fazer o que? Eu vou pedir para essa Iterable fazer para mim um `forEach(cargo -->)` e aí ele vai pegar item por item, cargo por cargo que está nessa Iterable e vai fazer um `system.out.println` . Ele vai printar na tela para nós essas Iterables.

[02:05] Só que o que acontece, se eu *printar* esse cara assim cru, ele vai printar o endereço de memória da variável e não é o que eu quero. Eu quero que ele printe os valores. Para isso eu preciso sobrescrever aqui na minha classe o `ToString` . Com o `ToString` sobrescrito eu posso definir como que eu quero

que ele printe no Console, que aqui eu deixei no modo *default*. Deixa eu voltar só para mostrar um negócio para vocês.

[02:37] Caso você não saiba sobrescrever o `ToString`, você vem aqui em cima na opção "Source" e seleciona o "Generate toString". Seleciona as "Fields" que você quer no seu "toString" e dá um "Generate" e aí ele gera o "toString" para você. Então agora aqui eu só preciso dar a opção para o cliente, para o usuário da aplicação para que ele possa visualizar. Então vamos colocar aqui. Ele vai ter a opção 3, que no caso aqui eu deixei como `visualizar`.

[03:19] E se ele digitar 3, ele vai visualizar os registros da tabela. Não tem parâmetro. Vamos aproveitar também e criar o método para deletar um registro. Então tem um registro na minha tabela, preciso que ele não exista mais. Aqui já é deletar mesmo o registro da tabela, não é deleção lógica. Então vamos lá. Vamos criar o método `private void deletar`.

[04:03] Para deletar, aí sim eu vou precisar do Id do registro na tabela para que ele possa deletar aquele registro em específico. Então vamos no CargoRepository e nós vamos perceber que tem aqui o deletar. O que é o deletar? Você cria uma entidade com todos os registros para que você possa deletar exatamente essa entidade na base. Você tem o Deletar todos ou você pode deletar por um Id, que é o que vamos utilizar.

[04:43] Deletando por um Id você só vai deletar aquele registro e aí você só vai precisar de um atributo da sua classe para deletar. Não precisa de um objeto completo para deletar. Então feito isso, agora eu só vou printar para o cliente que ele foi deletado. Agora vamos criar aqui também a opção para deletar, que vai ser a opção 4 e vamos colocar também aqui a opção para ele deletar. Se ele selecionar 4, ele vai poder deletar um registro.

[05:31] Nesse caso nós temos de passar um Scanner como parâmetro. Vamos executar a nossa aplicação para ver como ela vai se comportar. Vamos aguardar ela iniciar. Aguardando. Classe e o projeto iniciado, cargo. Vamos primeiro visualizar os cargos que nós temos hoje na nossa tabela. Visualizando

nós temos aqui, 1 é o programador, 3 é recursos humanos e 4 é gerente. Agora que nós já temos aqui os Ids eu posso atualizar.

[06:21] Então vou vir para atualizar e vou atualizar novamente o registro 3. Vou voltar ele para RH. Agora eu posso visualizar e está lá como RH. Então vendo? Eu não preciso mais ir no meu banco de dados para visualizar as informações. E agora vamos testar aqui também o "deletar" que eu vou deletar aqui o registro 4. Agora ele foi deletado e eu posso visualizar e nós não temos mais o registro 4.

[06:54] Antes de finalizarmos aqui, eu gostaria de mostrar algo para vocês porque vocês podem estar pensando, "Bom, ok, se ele mostrou isso, mas poderia se fazer também com JPA". Então vamos fazer um comparativo de como ficou aqui a nossa classe, de como estamos utilizando o nosso repositório para salvar registros e como ficaria ela no JPA. Então essa parte aqui você não precisa escrever. É só para fazermos um comparativo.

[07:26] Vou criar na nossa pasta Service uma nova classe. Vou chamar ela de JpaCrudCargoService. Eu já tenho aqui o código JPA, vou copiar ele e colar para que possamos dar uma analisada e fazer o comparativo entre um e outro. Então deixa eu pegar o código e vou adicionar ele aqui. Vamos ver.

[08:07] Se tivéssemos utilizando o JPA precisaríamos criar um Entity Manager Factory, atribuir esse Entity Manager Factory a um Entity Manager e aí poderíamos fazer `getTransaction`, iniciar uma transação, criar uma persistências, fazer *commits* e fechar transações. Então veja, vamos nos focar primeiro no método para salvar um registro. Veja a quantidade de itens que eu preciso para salvar um registro. Agora vamos fazer um comparativo.

[08:39] E aqui perceba que não estamos criando a opção dinâmica para o cliente digitar dinamicamente. Não estamos utilizando o Scanner aqui. Essas funções aqui nós utilizamos para salvar. Então vamos ver na nossa classe como fazemos para salvar. Apenas isso.

[09:01] Veja, o Framework abstrai essas coisas que antes precisávamos escrever na mão para coisas mais simples, para coisas mais rápidas, com menos preocupação e com menos linhas de códigos nós conseguimos dar mais agilidade ao desenvolvimento. Então o comparativo entre como nós faríamos com o JPA e como está hoje com o Spring Data é um bom exemplo do porquê é bom utilizar o Framework. Então até a próxima aula.