05

Listando no ProdutoDAO

Transcrição

[00:00] Fala, aluno. Tudo bom? Dando continuidade ao nosso curso de JDBC, vamos voltar à nossa classe `TestaInsercaoComProduto` e vamos revisar o que foi feito até agora, referente à nossa camada de persistência. Nós vimos que foi necessário criar uma DAO, porque essa DAO vai ficar responsável por toda a parte de comunicar com o nosso banco de dados.

[00:27] Então é ela quem vai conter as nossas queries, ela quem vai conter os nossos Prepared Statements, porque eles estavam repetidos nas nossas classes main. Nós vimos que código repetido nunca é uma boa prática no desenvolvimento de software. Então nós chegamos no `ProdutoDAO`, e esse `ProdutoDAO` agora, ele contém um método `.salvar();`.

[00:52] Todo mundo que precisar salvar um produto, vai apenas instanciar `ProdutoDAO`, passar uma connection para o seu construtor e vai passar o produto que deseja salvar. Só que nós vimos que agora nós podemos executar mais de um comando na nossa DAO, foi inclusive em um exemplo que nós estávamos usando para verificar, que agora eu quero salvar e depois eu quero listar.

[01:22] Então por isso que até extraímos a instância da nossa DAO, porque a partir desse momento precisamos só usar referências e ir chamando os seus métodos. Eu não preciso toda vez instanciar um `ProdutoDAO`, passando a connection, faço isso uma vez só, e chamo os seus métodos através da referência. Só que agora precisamos desenvolver o nosso método `.listar()`.

[01:48] Agora, como nós temos uma DAO e nós vimos que toda parte de persistência vai ficar nessa DAO, vamos implementar o nosso método `listar()` depois do método `.salvar()`. Hoje nós temos uma classe de modelo, que representa o nosso produto. Quando eu mando listar sem nenhum filtro, ou seja, sem nenhum where na nossa query, ele vai listar todo mundo.

[02:13] Então eu vou ter uma lista de produtos, porque se eu tiver um, ele vai ter uma lista com um produto; mas se tiver vários, vai ter uma lista com N produtos. Então o nosso método vai retornar uma lista de produtos e eu vou chamar ele de `listar()`, então fica `public List<Produto> listar()`. Eu já vou instanciar a nossa lista, que nós vamos precisar lá na frente.

[02:36] E eu vou chamar ela de `produtos()`, e vou instanciar aqui `List<Produto> produtos = new ArrayList<Produto>();`. Vou escrever o nosso SQL, que vai ser um *select* simples, ele não vai conter nenhum filtro, como já falado anteriormente, então vamos só fazer um `String sql = "SELECT ID, NOME, DESCRICAO FROM PRODUTO";`, onde temos o nome, o ID e a descrição.

[03:02] Como nós já estamos recebendo a nossa conexão pelo construtor, eu só preciso preparar, recuperar o meu Prepared Statement, e para isso eu vou usar `try(PreparedStatement pstm = connection.preparedStatement(sql));`, e passo o `(sql)`, que nós acabamos de escrever no ´String sql `. Lembrando sempre de adicionar o` throws SQLException`.

[03:29] Com o Prepared Statement em mãos, eu só preciso agora mandar executar com `pstm.execute();`. E esse código vai me trazer um resultado , isso nós já fizemos, vocês lembram bem. Esse resultado é um `try(ResultSet rst = pstm.getResultSet())`. Então enquanto eu tiver resultado, enquanto eu tiver um próximo resultado, traga para mim, com `while(rst.next())`.

[03:59] Só que agora estamos trabalhando com a nossa classe modelo. Então nada mais justo do que eu transformar esse `.getResultSet()`, que antes nós só guardávamos em strings, em integer e retornava, agora eu vou criar um

produto desse `.getResultSet()`. Então, se vocês lembram bem, nós criamos um construtor em `Produto` para que quando eu quiser inserir um produto, eu já passo um nome e uma descrição.

[04:27] E quando esse construtor for inserido, ele vai me retornar a sua chave gerada. Só que agora eu tenho um cenário diferente, esse `ProdutoDAO`, ele já tem o ID, já tem o nome, a descrição, então eu vou criar um novo construtor em `Produto`. Eu posso até fazer aqui, `public Produto`, só que agora ele vai conter todas as informações.

[04:54] Então eu quero `public Produto(Integer id, String nome, String descricao)`. Vocês já vão entender porque eu estou fazendo isso. Então `this.id = id;`, `this.nome = nome;` e `this.descricao = descricao;`. Eu vou fazer isso agora porque quando eu for instanciar o meu `Produto` em `ProdutoDAO`, o que eu vou precisar fazer é o seguinte, eu vou usar esse `new Produto(id, nome, descricao)`.

[05:30] E nesse `new Produto`, eu vou fazer assim, eu vou pegar o `srt.getInt(1),`, que vai estar no primeiro Index, que é o ID, vou pegar uma `rst.getString(2)`, que é o nosso nome, que está no segundo Index, e vou pegar outra `rst.getString(3)`, que é a nossa descrição, que está no terceiro Index. Pronto, esse `new Produto(rst.getInt(1), rst.getString(2), rst.getString(3));` eu agora estou transformando ele em um produto.

[05:58] Como eu tenho que retornar a lista de produtos, o que eu vou fazer? Eu vou pegar a minha lista e vou `produtos.add(produto);`. Então o que eu estou fazendo? Eu recuperei o primeiro produto - na verdade eu recuperei a primeira linha do nosso banco de dados, transformei ela em produto e adicionei na minha lista de produtos, porque agora eu tenho que adicionar o `return produtos;`.

[06:26] Eu vou retornar essa lista de produtos, que era intenção desde o começo. Então agora eu tenho o meu método já pronto, retornando um

produto, agora já utilizando a nossa classe de modelo. No nosso método `TestaInsercaoComProduto`, agora eu vou mudar o nome dele, eu vou dar um "Refactor > Rename" e vou botar "TestaInsercaoEListagemComProduto", para fazer mais sentido.

[06:57] Vou pedir para ele mudar em todo mundo que eu usei como referência. E agora eu vou fazer o seguinte, eu vou recuperar uma lista de produtos, que eu vou chamar `List<Produto> listaDeProdutos = produtoDAO.listar();`, vou chamar a nossa DAO e o método `listar();`. Para vermos se deu certo, eu vou fazer o seguinte, eu vou pegar a lista de produtos, vou usar o `stream()` para usar um `.foreach`.

[07:29] Vou botar `lp`, de lista de produtos, vou usar a *lambda* e fazer o seguinte, como nós sobrescrevemos o `toString`, eu posso já dar um `println()` no objeto, então fica `listaDeProdutos.stream().foreach(lp -> System.out.println(lp));`. Quando sobrescrevemos o `Produto`, eu coloquei `("O produto criado foi:`, o que não vai fazer muito sentido na listagem.

[08:00] Então eu vou botar `return String.format("O produto é: %d, %s, %s", this.id, this.nome, this.descricao)`. Na ´TestaInsercaoEListagemComProduto`, qual vai ser o objetivo agora? Nós vamos inserir a cômoda e depois vamos já mostrar os produtos que já existem na tabela, que são aqueles dois que viemos trabalhando desde o começo, e o novo produto, que é a cômoda.

[08:24] para verificarmos se não tem nenhum lixo, vamos usar a nossa classe `TestaRemocao`. Não tinha. Agora, se eu executar o `TestaInsecaoEListagemComProduto`, ele tem que me trazer três produtos, exatamente isso: os dois produtos que viemos utilizando desde o começo, como eu informei, e o novo, que é a nossa cômoda vertical.

[08:48] Agora nós vemos que o nosso código já ficou melhor trabalhado, digamos assim, porque eu tenho uma camada de persistência, onde eu tenho

tudo o que é referente à persistência de produto, à listagem, à inserção, se eu tiver uma futura, eu posso colocar o remoção agora em `ProdutoDAO`, o remover. Enfim, tudo o que vai ser referente a produto e for trabalhar com as queries SQL, remover, o CRUD, digamos assim, vai ficar dentro de `ProdutoDAO`.

[09:23] E as nossas classes passam a ficar mais enxutas, elas ficam apenas para testar essas nossas operações com banco de dados. Eu não tenho mais queries espalhadas em todo canto, eu só preciso mesmo instanciar a nossa DAO, passar uma conexão para ela e eu posso usar quantos métodos eu quiser, desde que faça sentido, é claro.

[09:50] Então agora a nossa aplicação, ela já fica com uma cara cada vez mais de uma aplicação que você vai encontrar no mundo corporativo, nas empresas, nas grandes empresas. A preocupação é sempre essa, é melhorar o código, refatorar, não repetir código, usar padrões que já são utilizados no mundo todo, enfim é isso que viemos fazendo aqui.

[10:15] A ideia é que saiamos com esse conhecimento, com essa questão bem aprimorada, para ser automático, já pensarmos em desenvolver códigos dessa maneira, de maneiras que qualquer desenvolvedor vai ver o seu código e falar: isso eu conheço. Enfim, porque a ideia é essa: você escrever códigos de fácil manutenção. Então é isso, aluno. Espero que vocês tenham gostado e até o próximo vídeo.