02

Consultas com parâmetros dinâmicos

Transcrição

[00:00] Olá, pessoal! Estamos de volta ao treinamento de JPA. Continuando as discussões na parte de consultas, neste capítulo, nesta aula, ainda continuaremos discutindo mais um pouco sobre consultas. Nós já estudamos diversos assuntos sobre consultas, mas um deles ficou faltando. Eu estou com a classe "ProdutoDao".

[00:18] Temos algumas consultas usando a *named query*, usando função de agregação, enfim, vários tipos de consultas com filtros. Só que, por exemplo, essa consulta aqui - aliás essa aqui, que busca um produto pelo nome: `buscarPorNome`. Eu recebo uma *string*, que é o nome do produto, e eu faço o *select* filtrando pelo nome do produto.

[00:41] Só que, nessa consulta, o nome, ele é sempre obrigatório, então eu sempre tenho que passar o nome. Toda consulta que você faz um filtro dessa maneira, esse parâmetro, ele acaba ficando obrigatório. Se você passar nulo, se você não passar esse parâmetro, se você passar nulo, ele não vai fazer um *select* sem filtrar, um `SELECT p FROM Produto p`. Ele não vai ignorar o *where* só porque o parâmetro está nulo.

[01:03] Ele vai considerar que você quer carregar todos os produtos cujo nome seja nulo. Aqui está o problema, não é bem isso, porque no banco não terá nenhum produto com o nome nulo. Na verdade, eu queria ignorar esse parâmetro. Eventualmente, você vai querer esse tipo de *query*, onde os parâmetros são flexíveis, são opcionais.

[01:22] Imagine um sistema que tem uma tela de consulta, onde eu posso buscar produtos. Tem três campos nessa tela: eu posso buscar o produto pelo nome, pelo preço ou pela data de cadastro. Mas eu posso informar qualquer um desses parâmetros. Posso não informar nenhum parâmetro, ele vai trazer todos os produtos. Posso informar só o nome, ele tem que filtrar pelo nome, só a data de cadastro, tem que filtrar pela data de cadastro.

[01:44] Posso filtrar pelo preço e pela data de cadastro, ele tem que filtrar pelos dois. Então percebe? Tem que ser flexível, os parâmetros não são mais obrigatórios, são opcionais. Como montaríamos uma *query* dessa maneira? Vamos tentar fazer ela aqui. Vou quebrar a linha, vou criar um novo método na classe "ProdutoDao".

[02:04] Vou criar `public List<Produto>`, esse vai ser o retorno, `> buscarPorParametros()`, vou chamar assim. Imagine que aqui ele recebe os parâmetros `(String nome, )`, recebe o preço, que eu tinha comentado, `(String nome, BigDecimal preco, LocalDate dataCadastro)`. Esses são os três parâmetros possíveis, porém todos opcionais.

[02:35] Como ficaria o JPQL? Vamos criar aqui `String jpql = " ";`. Na teoria, ficaria assim: `"SELECT p FROM Produto p WHERE ";` e vem os filtros. Deixa eu quebrar a linha. `"SELECT p FROM Produto p WHERE p.nome = :nome";` e aí vai, `AND`, qual é o próximo parâmetro? `AND p.preco = :preco` e aí vai.

[03:12] Porém, esse é o jeito incorreto, que eu acabei de citar, porque desse jeito eu estou considerando que todos esses parâmetros aqui são sempre obrigatórios. Sempre são obrigatórios. Se você não informar um desses parâmetros, vem nulo e ele vai pensar que você quer filtrar pelo parâmetro nulo no banco de dados.

[03:29] Mas não é isso. Se um parâmetro, se eu não informar o preço, eu queria que a *query* fosse assim: `SELECT p FROM Produto p WHERE p.nome = :nome`, acabou. Não tem um *and*, não tem um *or* na sequência, ignora. Se eu inform

só o preço, eu quero que ele tire o `p.nome = :nome`, fique só `WHERE p.preco = :preco`.

[03:47] Porém, desse jeito não vai rolar. Nesse tipo de consulta, consulta dinâmica, consulta com parâmetros opcionais, temos que fazer uma gambiarra aqui. Já que não dá para dizer para a JPA: olha, JPA, esse parâmetro é opcional, se vier nulo, não é para filtrar que ele esteja nulo na coluna do banco de dados, é para ignorar ele da *query*. A JPA não faz isso automaticamente.

[04:09] Nós temos que fazer de um jeito aqui não tão bonito, mas é a solução. A consulta ficaria algo assim. Eu tenho que ter a consulta padrão, uma consulta que será sempre igual, `"SELECT p FROM Produto p";`. A princípio é isso. Agora eu tenho que verificar cada parâmetro desses se eles estão chegando, se não estão nulos.

[04:33] Eu vou ter que colocar *ifs* aqui. `if (nome != null)`, se o parâmetro nome não veio nulo é porque um nome foi digitado. Se você até quiser tomar cuidado, vai que não está nulo, mas está uma *string* vazia. Então se o nome não for nulo e não for vazio, `if (nome != null && !nome.trim().isEmpty())`. Se não é nulo e nem vazio é porque veio algo.

[05:00] Agora, sim, eu tenho que filtrar pelo nome. Eu pego o meu JPQL e vou complementando. Eu já tenho um `SELECT p FROM Produto p`, eu coloco aqui dentro do `if`, `jpql = "WHERE p.nome = :nome ";`, coloca isso na *query*. E aqui vai, continua lá - digitei errado, jpql. Agora, se o preço veio. O preço não é *string*, não tem o vazio.

[05:33] `if (preco != null)` eu continuo, `jpql = "AND p.preco = :preco ";`. Eu tenho que fazer esses *ifs* para verificar se o parâmetro está vindo, já que ele é opcional. Porém, vamos ter um problema, o `WHERE`, eu coloquei o `WHERE` nesse primeiro `if`. Se ele digitou o nome, a *query* vai ter o *where*. E se ele não digitou o nome, só digitou o preço?

[05:59] A *query* vai ficar `SELECT p FROM Produto p AND p.preco` ? *And*? E onde está o *where*? É mesmo. Eu vou ter que fazer outro *if* para ver se eu já adicionei o *where*. Dentro do segundo *if*, eu faço outro *if*. *If* e *where* já foi adicionado, aí eu tenho que ter uma variável booleana para controlar se o *where* já foi adicionado. Meu Deus do céu, já começou a complicar.

[06:19] Qual é a gambiarra que o pessoal faz? Como o *where*, eu preciso ter o *where*, mas eu só preciso ter uma única vez, eu vou colocar todos os filtros como *and* e vou colocar um *where* aqui, nessa parte, que é a parte fixa, `"SELECT p FROM Produto p WHERE"` . Mas e aí? O que eu coloco no *where*?

[06:34] O pessoal faz essa gambiarra aqui, `WHERE 1 =1 ";` . Como assim `1=1` ? 1 é igual a 1? Verdadeiro. Sempre vai dar verdadeiro, então não importa, sempre vai dar verdadeiro, é só para forçar que eu sempre terei o *where* na *query*. Fica essa gambiarra, um negócio meio esquisito, mas funciona assim.

[06:52] `WHERE 1=1 ";` tem um parâmetro, eu coloco `AND` , tem um parâmetro, eu coloco `AND` , tem um parâmetro, eu coloco `AND` . Eu faço um *if* para cada parâmetro. Tenho o preço, agora é `if (dataCadastro != null)` , `jpql = "AND p.dataCadastro = : dataCadastro "` .

[07:19] Terminei a minha *query* aqui. Porém, aí é que vem o problema, eu vou precisar desses três *ifs* de novo, porque eu preciso *setar* agora o parâmetro. Se entrou no primeiro *if*, tem esse parâmetro `:nome:` . Se entrou nesse segundo *if*, tem esse parâmetro `:preco` . Agora eu preciso criar a *query*.

[07:38] Eu preciso fazer isso aqui: `em.createQuery(jpql, )` , passo o meu `jpql` , é objeto do tipo Produto que essa *query* devolve, `(jpql, Produto.class);` . Só que agora eu não posso fazer isso, igual estávamos fazendo antes, `.setParameter("nome", nome)` , e passa o parâmetro `nome` que veio no método.

[08:03] Porque é opcional, pode ser que não tenha esse parâmetro na *query*. Então aqui, o que temos que fazer? Temos que criar a *query*, atribuí-la a uma

variável do tipo *typedQuery*, que é a *query* da JPA, tem essa classe da JPA, `TypedQuery<Produto> query = em.createQuery()`.

[08:19] Aqui faz mais três *if elses* para ver quais parâmetros têm que ser substituídos. Aqui, de novo, se o nome foi preenchido, só que agora não vou mais mexer no JPQL. Eu pego o meu objeto `query.setParameter("nome", nome)`, tenho o parâmetro `"nome"`, então eu vou *seta-lo*. Se entrou no *if* é porque tem o parâmetro. Vou substituir aqui o parâmetro nome.

[08:43] E a mesma coisa para o preço e a mesma coisa para data de cadastro. No segundo seria o parâmetro `"preco"`, no terceiro seria o parâmetro `"dataCadastro"`. Aqui seria o parâmetro do método `dataCadastro`, aqui o parâmetro `preco`.

[08:59] Aqui sim, no final, no final de tudo isso, `return query.getResultList();`. O nosso método fica dessa maneira. Quando você tem uma busca por parâmetros opcionais, dinâmicos, você acaba ficando com o código desse jeito. Você monta uma *query* padrão e, como você não sabe onde o *where* vai ser adicionado, você sempre adiciona ele fazendo essa gambiarra, `WHERE 1=1`.

[09:26] E aí vai. Tem o primeiro parâmetro? Bota o *and*, concatena na *query*. Tem o segundo parâmetro? Concatena na *query*. Tem o terceiro parâmetro? Vai concatenando. Cria o `TypedQuery`, tem que fazer os *ifs* de novo, para *setar* o parâmetro no objeto *query*. E fica com esse código meio poluído. Essa é a solução tradicional, usando o JPQL, você tem que fazer os *ifs* e *elses* duplicados, para cada parâmetro, dois *ifs*.

[09:54] Porém, para evitar esse tipo de situação, a JPA tem mais um recurso para você fazer consulta. Nós já vimos bastante do JPQL, só que existe um outro recurso para fazer consulta, que você não precisa fazer duas vezes esses *ifs*. No próximo vídeo conheceremos ele. Vejo vocês lá.