04

Usando JPQL

Transcrição

[00:00] Olá. Agora que nós já sabemos criar Querys através da *feature*, do Derived Query e do Spring Data, o pessoal da Recrutei solicitou que nós fizéssemos um novo relatório para eles que informado o nome, a data de contratação e um salário maior do que o informado no filtro, deve-se retornar todos os funcionários que se encaixarem nessa filtragem.

[00:24] Como nós já sabemos do Derived Query, já é possível que façamos essa Query lá para o pessoal do Recrutei. Então vamos lá. Entrando no nosso projeto, vamos até a nossa pasta principal, entrando na pasta do projeto, na pasta raiz, dentro da nossa pasta Repository, vamos dentro de pasta FuncionarioRepository. Então aqui eu já criei a Query com o Derived Query e eu queria mostrar como que ela ficaria.

[00:56] Então vamos lá, `findByNameAndSalarioGreaterThanAndDataContratacao` e aqui a data de contratação tem que passar a String de nome, o Double de salário e um LocalDate da data da contratação. Igual nós vimos na aula anterior com aquela tabela de como criar filtros, como fazer *selects* através de Derived Query. Porém, reparem que o método ficou com o nome muito extenso.

[01:33] Isso para criar, para a funcionalidade da aplicação não vai ter problema nenhum. Se caso você pegue esse método e coloque dentro da sua classe e chame ele lá dentro da RelatórioService, irá funcionar perfeitamente. Só que se você reparar bem, o nome do método acaba ficando um pouco extenso, um

pouco grande demais e isso pode acabar acarretando alguns problemas na hora de você chamar, de ficar grande e ficar difícil de você ler.

[02:04] Não tem problema nenhum caso você queira fazer assim. Pode fazer, não tem problema. Mas eu queria mostrar para vocês outra forma de criar Querys sem ser por Derived Query. E com isso você acaba não tendo de criar métodos muito grandes. Então isso que nós vamos fazer agora se chama JPQL. Baseia-se no quê? Em você criar o Querys com SQL só que utilizando o nome das entidades do Java. Então vamos lá.

[02:40] Deixa eu apagar esse método em Derived Query e vamos criar agora o mesmo método utilizando JPQL. Então ele tem de retornar uma lista de Funcionário e nós vamos colocar um nome. O nome do método não importa mais porque nós não estamos mais usando o Derived Query, mas ainda tem que ser algo que faça nós entendermos o que esse método irá fazer. Então vou colocar aqui, `findNomeDataContratacaoSalarioMaior`. Porque lendo isso nós vamos saber o que ele faz.

[03:32] Uma filtragem por Nome, por Data de contratação e um Salário Maior do que o informado. Na verdade eu vou deixar `findNomeSalarioMaiorDataContratacao`. E para que possamos fazer os filtros, fazer a consulta, o usuário tem que passar três atributos para nós.

[03:59] O primeiro é uma String de Nome, o segundo é um Double com o Salário que ele quer pesquisar, o Salário Maior do que o que ele quer pesquisar e por fim um LocalDate com a data. Vamos deixar só data, mas é a data de contratação. Agora se formos lá na nossa classe de relatório, a RelatórioService e chamar esse método, não vai acontecer nada porque o Spring Data vai olhar e falar, "Não entendo o que ele está querendo aqui" e não vai conseguir fazer porque ele vai achar que você está querendo utilizar Derived Query só que aqui não está no padrão Derived Query.

[04:43] Então nós temos que informar para o Spring Data que esse método te de fazer uma consulta, uma Query. E para informar isso para o Spring Data,

nós utilizamos a notação `@query`. Dentro do `@query` nós vamos escrever a Query que queremos executar aqui dentro. Então os mesmos comandos de SQL aqui, `select from`.

[05:12] Porém a entidade é conforme o nome escrito na nossa classe Java que no nosso caso é `funcionário`. E aqui eu quero colocar um *alias* para Funcionários para que eu não precise ficar utilizando esse nome extenso. Então vou colocar aqui como "f". Então faça um *select* em Funcionário e retorne todos os atributos de Funcionário. Aí nós vamos colocar o filtro, quando f igual, aqui também são os mesmos atributos da classe.

[05:49] No nosso caso aqui quando Nome for igual e aí tem que pegar o nome que o cliente está passando como parâmetro. Para pegarmos esse nome que o cliente está passando, nós utilizamos dois pontos e o mesmo nome aqui que nós utilizamos a variável que nosso caso foi nome. `f.nome = nome AND f.salario`. Salário é a mesma coisa, é o mesmo nome que nós demos a entidade. Então `f.nome = nome AND f.salario > = : salario AND f.dataContratacao = :data`.

[06:50] Deixa eu só quebrar aqui para que fique melhor para visualizarmos. Então selecione Funcionario e quando Funcionario for igual ao Nome, o Salario for maior que o Salario informado pelo cliente. Não pode ter espaço. É dois pontos e logo depois o Nome e a Data de Contratação for igual a Data de Contratação selecionada. Agora vou dar um “Close All”, clicando no canto superior esquerdo, e vamos voltar para as nossas Packages.