02

Derived Query

Transcrição

[00:00] Agora que nós já temos as nossas entidades criadas, os relacionamentos que o pessoal da Recrutei mandou feitos para nós e os nossos Cruds desenvolvidos, nós recebemos mais um desafio do pessoal da Recrutei. Eles querem visualizar um funcionário, mas eles querem consultar esse funcionário através do nome dele. Então o usuário da aplicação vai adicionar um nome e temos que retornar esse nome para o usuário que fez a consulta.

[00:34] Pois então, com esse desafio que o pessoal da Recrutei passou para nós, vamos pensar um pouco. Primeiro eu vou precisar criar um comando, algo no SQL para que ele possa fazer essa consulta para mim. E aí vem uma das *features*, uma das ferramentas mais interessantes do Spring Data porque se eu te falar que você consegue fazer essa consulta sem nenhuma linha de código SQL, você acredita?

[01:09] Parece coisa meio maluca, mas acredita em mim, é possível e conseguimos fazer isso com o Spring Data. Vamos lá. Se eu tenho que pesquisar um funcionário por nome e retornar uma lista de funcionários com aquele nome, então eu posso vir na minha pasta "FuncionarioRepository". Aqui dentro da nossa FuncionarioRepository é que nós vamos colocar essa consulta de Derived Query. Então o que queremos? Passado um nome, ele retorna uma lista de funcionários.

[01:40] Então o retorno aqui é `List`, o tipo do retorno é `funcionários`, uma lista de funcionário e agora escrevemos um método. Como eu disse, uma pesquisa começa pelo FindBy. Então `findBy`, e eu quero pesquisar pelo quê?

Vamos entrar aqui dentro da entidade de Funcionário e aqui temos os atributos. Eu tenho que falar o FindBy. Então eu tenho que fazer um `findBy + por algum atributo`.

[02:15] E o atributo que eu tenho aqui é Nome. Então eu quero fazer um FindBy pelo Nome. Vou colocar aqui `findByNome` e como parâmetro desse método que estamos criando dentro do Derived Query, eu vou passar uma String que contém o Nome que vai vir. Então `String nome`. Então procure um funcionário que tenha o Nome que vai vir aqui por parâmetro.

[02:47] Agora nós vamos vir aqui e aqui dentro das minhas classes de serviço eu vou criar uma nova classe. Então venho no meu pacote principal, no canto superior esquerdo, dentro da “Source Java > Service, dou um botão direito, New > Class". E essa Class eu vou chamar ela do quê? Eu vou chamar ela de RelatoriosService porque aqui dentro eu vou colocar todos os relatórios que o pessoal da Recrutei vai pedir para nós.

[03:25] E aqui, como nós já vimos em todos os outros Cruds, nós vamos dar as opções para que ele possa selecionar qual é o relatório que ele quer visualizar. Para não gastarmos mais tempo criando essa estrutura aqui, eu vou copiar a que já temos criado aqui e você também já viu como que faz isso na aula da criação do Cargo para fazer o Crud do Cargo. É o mesmo princípio. Aqui vamos disponibilizar um Switch de escolhas para o usuário.

[04:04] Então vou deixar só a opção 1 por enquanto porque aqui vamos colocando outras opções. E a primeira opção vai ser `Busca Funcionario nome` e vamos criar um método aqui para fazer a busca de Funcionário por Nome. Então `private void buscaFuncionarioNome` e vamos precisar do Scanner aqui porque o nosso usuário via precisar digitar o nome que é o que ele quer pesquisar. Então vamos adicionar aqui.

[05:11] Como nós vimos também, essa classe vai precisar do nosso FuncionarioRepository, pois é lá que está a lógica para fazer a consulta e nós vamos precisar dessa lógica para adicionar aqui dentro da nossa aplicação,

para fazer a consulta, então vamos criar via construtor. `Public RepositoryService` , criamos o construtor.

[05:41] Agora aqui fora, `private final FuncionarioRepository funcionarioRepository` , vamos pegar essa entidade e colocar aqui e vamos falar que o nosso `funcionarioRepository = funcionarioRepository` que o Framework vai criar por injeção de dependência para nós no construtor. E agora eu vou fazer o seguinte, vamos voltar aqui no nosso Crud. Ele também vai pegar a mesma coisa. Vamos printar algo para o cliente e vamos pegar o nome que o cliente deseja. Vamos lá.

[06:33] `System.out ("Qual nome deseja pesquisar?")` , aí vamos criar uma String, chamar ela de Nome e vamos pegar o `scanner.next` para que possamos ter o nome que o usuário digitou no console. Então agora vamos criar uma lista de funcionários, vou dar o nome de List e vou falar para o `funcionarioRepository` para ele fazer o `findByNome` que o cliente digitou no console. E agora eu só preciso apresentar isso na tela.

[07:49] Então vou fazer aqui um ForEach. `list.forEach(System.out::println)` . Pronto, com isso eu já vou printar no console. E vamos colocar aqui o atributo que o método ou case 1 do Switch vai fazer a busca pelo nome. Ele vai precisar do Scanner, o argumento do método.

[08:30] E agora nós precisamos também lá na nossa classe principal, voltando no lado superior esquerdo da tela, pasta principal, na “Source Java > SpringDataApplication" que é a nossa classe Main, nos precisamos dar a opção ao cliente para que ele possa entrar também no Switch de relatórios, ele possa fazer consultas por relatórios. Então eu vou criar aqui também uma instância para que ele possa entrar lá na nossa classe de relatórios.

[09:05] Então `RelatorioService` e vamos passar um `RelatorioService` aqui pelo construtor também. Vamos adicionar aqui um `this.relatorioService = relatorioService` . Temos o nosso RelatorioService. Agora se ele digitar 4, ele

vai entrar dentro da opção de relatórios da aplicação. Então colocando aqui o case 4 e o case 4 vai para o `relatoriosService.inicial` passando o Scanner como argumento. Então eu vou lá no DBeaver. Na verdade eu não vou no DBeaver, eu vou executar a aplicação. Vamos executar a aplicação.

[10:19] Agora com a aplicação iniciada, eu vou vir na função de Funcionários e vou pedir para visualizar só para ver qual foi o nome do funcionário que eu cadastrei, que no caso foi Caio. Então vou pedir para eu sair da aplicação e vou na minha parte de relatórios. Vou pedir para buscar por um funcionário por nome e vou pesquisar, `Eu quero funcionário Caio` que é o que eu tenho registrado dentro da minha base de dados. Veja, a aplicação já trouxe o funcionário Caio. Eu vou acessar a documentação do Spring Data e vou entrar nessa parte de Query Creation.

[11:09] Isso que estamos vendo na tela se chama Derived Query. No que ela se baseia? Através de um comando, de um método escrito em Java, o próprio Spring Data consegue pegar método e criar uma Query para o seu banco de dados específico sem você ter que criar nenhum comando SQL. É muito legal isso. Então como fazemos isso?

[11:40] Primeiro esse comando, essa ferramenta que está dentro do Spring Data, nós chamamos ela através desse comando `findBy` . Quando colocamos FindBy significa que nós queremos fazer uma consulta por alguma coisa. E aí o nosso Spring Data já vai entender que é uma consulta por alguma coisa e vai consultar. Então aqui no exemplo que nós temos da documentação ele fala assim, `findByEmailAddressAndLastname` .

[12:10] Então ele quer que faça uma consulta pelo endereço de e-mail e pelo último nome. E ele passa como argumento nesse método os campos que de fato ele quer consultar e isso retorna para ele uma lista de usuários. Aqui no caso do exemplo da documentação. Mas não é só And que essa ferramenta sabe fazer e muito menos uma consulta simples por um único nome. Ele tem tudo isso que você pode utilizar: And, Or, Equals, Is, Between.

[12:47] Então é uma lista gigantesca que você pode utilizar. Caso você tenha a necessidade de utilizar algum outro que não vamos passar aqui no curso, você pode entrar na documentação e verificar qual é o modelo de consulta, qual é o tipo de consulta que você quer e verificar como que é o padrão, como que é um exemplo para fazer essa consulta no Derived Query.