07

Native Query

Transcrição

[00:00] Vamos lá. Agora o pessoal da Recrutei pediu mais um relatório para nós que é informado a data de contratação do funcionário. Nós temos que retornar uma lista de funcionário ou todos os funcionários que tenham a data de contratação superior a data informada. Então vamos lá. Nós já aprendemos a fazer aqui Derived Querys e JPQL.

[00:31] Porém vamos pensar, já trabalhamos com duas *features* do Spring Data. Mas será que o Spring Data não suporta Querys nativas ou a Query como ela é executada lá no MariaDB. Ele suporta sim e é isso que vamos ver agora. Então nós vamos fazer essa consulta que o pessoal da Recrutei solicitou através da Query nativa.

[00:55] Você vai ver que a Query nativa é muito parecida com a JPQL, mudando o fato de que em vez de nós utilizarmos os valores de atributos das nossas entidades, os nomes dos atributos das nossas entidades, nós temos que usar os nomes dos atributos que está nas tabelas do nosso MariaDB. Então vamos lá. Entrando no nosso projeto inicial, na pasta raiz do nosso projeto, lado superior esquerdo, na Package “Repository > FuncionarioRepository".

[01:25] E aqui nós vamos criar o nosso método para que possamos fazer essa consulta de retornar uma Data de Contratação superior a Data de Contratação que nós desejamos e vamos utilizar a Query nativa do MariaBD. Então vai retornar uma lista de funcionário. Eu vou chamar isso aqui de `findDataContratacaoMaior`.

[01:58] E para fazer essa data de contratação ele tem que informar para nós um LocalDate com a Data de Contratação. Da mesma forma de quando nós utilizamos o JPQL, se nós formos rodar isso e chamar, o Spring Data também não vai saber por que ele vai tentar executar isso aqui como se fosse uma Derived Query, mas esse método aqui também não está no padrão de uma Derived Query. Então para fazer uma Query nativa, nós também temos de utilizar a notação `@Query`.

[02:34] E aqui nós vamos escrever a nossa Query, `SELECT FROM`. Porém, como eu disse, aqui não utilizamos mais o nome dos atributos nosso e sim do banco de dados. Então vamos no DBeaver e vamos ver que a nossa tabela se chama "Funcionários" no plural e não mais "Funcionário". Então aqui a pesquisa é por `funcionarios`. E aqui eu também vou colocar um *alias* porque eu não quero ficar chamando esse nome gigantesco. Vou chamar de "f".

[03:21] Aqui eu posso colocar um asterisco e aqui eu vou falar `WHERE f.data`. Vamos voltar lá no DBeaver para vermos que na nossa entidade o nome é "dataContratacao" com o C maiúsculo depois do data, conforme nós vimos aqui. Porém no banco de dados ele utiliza a forma de escrita do SQL. Então ele coloca o *underline* entre as duas palavras. E nós temos que chamar dessa forma. Então é `data_contratacao >= :data`, aí para pegar o parâmetro é da mesma forma.

[04:12] Porém se nós executamos assim, o Spring Data vai achar que você está tentando fazer um JPQL e vai dar erro porque o nome que nós estamos utilizando para tabelas são diferentes entre o nome para o JPQL e para utilizar a NativeQuery. Então para nós conseguimos falar para o Spring Data executar isso aqui como uma NativeQuery, nós temos que falar que isso que estamos passando, essa String é o valor da nossa Query.

[04:46] E depois nós temos que colocar outro parâmetro que nós vamos chamar de NativeQuery e esse `nativeQuery = true` para informar para o Spring Data que esse comando que nós estamos utilizando é uma NativeQuery. Então ago

que nós já fizemos isso vamos até o nosso RelatórioService para chamar essa consulta também.

[05:14] Voltando no nosso projeto, na pasta raiz, agora no Package “Service > RelatórioService" e vamos um método, `private void` , eu vou chamar `buscaFuncionarioDataContratacao` . Esse método vai precisar também de um Scanner para pegar a data de contratação que o usuário vai digitar no console. E aqui podemos pegar também essa data que nós já criamos na outra consulta porque nós também vamos precisar fazer a mesma formatação. Então vamos lá.

[06:02] O nosso `funcionarioRepository.findDataContratacaoMaior` vai receber o nosso `LocalDate` convertido. E isso vai retornar para nós, essa consulta, um `list<Funcionario` . E vamos chamar isso de "list" e por fim vamos printar esse "list" no console para mostrar ao nosso usuário. Por fim só vamos dar essa opção também de consulta para o usuário dentro da aplicação.

[06:45] Então vamos colocar aqui. Se ele digitar 3, a busca vai ser por Busca Funcionario Data Contratacao. E vamos colocar essa opção também no nosso Switch de seleção de relatório. Então se ele seleciona 3 ele vai para a busca de funcionário por data de contratação. Então executando a aplicação.

[07:28] Vamos lá, Relatório, opção 3 de busca por data de contratação e eu quero todo funcionário que tenha sido contratado depois do dia 01/01/2009. Está lá, o nosso Caio foi contratado nessa data. Então com isso nós também já vimos mais uma nova função que é executar com as Querys nativas do seu SQL.