04

Compondo Specifications

Transcrição

[00:00] Agora que nós já sabemos como criar uma estrutura de uma Specification, nós vamos criar as demais Specifications que a Recrutei solicitou. Então vamos voltar na pasta raiz do nosso projeto, entrar na pasta Specification e vamos criar as demais Specifications. É muito simples criar Specifications utilizando o Spring Data. A facilidade é que ele já traz todos os objetos criados.

[00:31] Então você pode basicamente utilizar a mesma estrutura e só alterar a operação que você quer executar. Vou mostrar para vocês como é fácil. Aqui já criamos a Specification de Nome para fazer o filtro por nome. Eu vou copiar essa mesma Specification e vou transformar ela nas demais para vocês verem como que a estrutura é basicamente a mesma. Copiei CPF, Salário e Data de Contratação. Então vamos lá, `cpf` .

[01:03] Eu estou alterando o nome do método e aqui vou fazer a consulta e ele vai me passar um CPF. O pessoal da Recrutei quer o Builder do Criteria por Equals, ele quer fazer um comparativo de igualdade. E como é um comparativo de igualdade, eu posso pegar o valor que o cliente informou e fazer a consulta direto. Entretanto no Root eu tenho que falar para ele pegar o atributo de Nome CPF.

[01:33] Então ele vai fazer o comparativo do atributo de nome CPF com o valor de CPF que o nosso cliente passou por parâmetro. A mesma coisa para Salario. Só que o Salario tem um diferencial. Ele vai precisar passar um Double de

salário porque é o tipo que temos dentro da nossa entidade. E aqui ele quer um comparativo por GreaterThan.

[01:59] Então vamos utilizar o `greaterThan` e falar para o nosso Root fazer a busca por salário e não preciso de nenhum comando especial do SQL, posso passar direto o meu salário. E para finalizar, a `dataContratacao`. A data de contratação do funcionário é um LocalDate. Então fazemos `localDate` e aqui a `dataContratacao` e também vamos utilizar o método `greaterThan`.

[02:45] Vamos pedir para ele pesquisar por data de contratação e passar a nossa data de contratação. Então vejam como que o Spring Data agiliza o nosso desenvolvimento. A estrutura é basicamente a mesma, nós só mudamos qual é a operação que nós queremos, o nome dos nossos atributos e é basicamente a mesma coisa. Já conseguimos criar as nossas Specifications muito rápido.

[03:10] Agora que nós já temos a nossa Specification, vamos fechar aqui e vamos voltar para a PackageService, para a classe RelatorioFuncionarioDinamico. Eu havia dito que se o usuário não quiser fazer uma pesquisa por um determinado atributo, ele não precisa. Então eu preciso fazer uma verificação por isso.

[03:35] Entretanto como eu quero apresentar esse valor em dar a possibilidade para o usuário acessar essa classe, acessar os recursos que nós estamos criando aqui, eu vou anotar com a notação de `@service` para que possamos depois fazer a injeção de dependência dentro dessa classe. Vamos lá. Como nós estamos utilizando Console aqui, o Console não permite que o usuário não digite um valor.

[04:02] Então vou tomar como se o usuário digitar algum valor específico, eu vou entender que ele não quer executar a consulta daquele valor. Por exemplo, If se o usuário digitar no `if(nome.equalsIgnoreCase ("NULL")`, eu vou atribuir ao Nome o valor *null*, `nome = null`. Porque se o valor estiver nulo, ele não vai fazer a consulta por usuário. Vou copiar aqui porque também é a mesma estrutura para CPF.

[04:48] Se caso o usuário, digite um CPF *null* é porque ele também não quer fazer uma consulta por CPF. Para Salario é a mesma coisa, `("Digite o salario")` e aqui Salario. Mas o Salario é um Double e o usuário tem que passar um Double, então `nextDouble` . E eu não posso fazer uma verificação de Double por String. Então o que eu vou fazer? Se o usuário digitar em Salario o valor 0, eu vou entender que ele não quer também fazer a consulta por salário.

[05:35] E por fim nós temos a consulta por data de contratação. Entretanto na data de contratação, o usuário da aplicação vai escrever uma data de contratação no Console, uma String e eu tenho que converter essa String para LocalDate. Então para isso vou precisar de um *formatter* para formatar o texto digitado para um LocalDate.

[06:01] Vou criar aqui um `private final` , vou pegar aqui um `DateTimeFormatter` e chamar ele de `formatter` e falar que ele é igual a um `DateTimeFormatter.ofPattern` e o padrão que ele vai passar é de `("dd/MM/yyyy")` . Deixa eu minimizar para ficar melhor para visualizar, esse é o padrão que o usuário tem que escrever no Console.

[06:43] Então vamos lá, vou copiar aqui também porque é a mesma estrutura, `("Digite a data de contratacao")` e aqui vou pegar uma String, vou pedir para ele pegar a String e aqui vou colocar só a data porque ainda não é a data de contratação convertido. Vou criar aqui um LocalDate e aqui sim é a `dataContratacao` que é a data de contratação de fato que nós queremos pesquisar na nossa base.

[07:23] Então vamos falar, se a data que o usuário digitou no console for igual a *null* também, se ele digitar `NULL` no Console, significa que ele não quer fazer consulta pela data de contratação. Entretanto se digitar um valor que não seja nulo, nós vamos pegar a nossa data de contratação e vamos atribuir à data de contratação um `LocalDate.parse` . Então ele vai *parsiar* o texto que o cliente digitou para um LocalDate.

[08:04] Vamos atribuir aqui a data e vamos pedir para o *parse* fazer a formatação conforme o *formatter* que criamos. Agora nós só temos que criar aqui as nossas classes Where, as nossas condições também. Como eu quero dar a opção para o cliente de ele fazer vários filtros, eu vou utilizar a operação OR. Então ele pode fazer por Nome, por CPF, por Salário ou por Data. E se ele quiser, por exemplo, o Nome ou Salário, vai trazer Nome e Salário conforme a condição que ele solicitou.

[08:41] Entretanto você também pode fazer a consulta por Entity, o cliente fazer, "Eu quero salário e nome", então eu faço uma junção desses dois filtros. No caso eu vou utilizar `.or` e vou colocar aqui mais uma consulta, `.or(SpecificationFuncionario.cpf)`, vamos passar o CPF digitado pelo usuário no Console, `.or(SpecificationFuncionario.salario)` e o Salário que o usuário também digitou no Console.

[09:34] Por fim `or(SpecificationFuncionario.dataContratacao)` que ele digitou no console. Mas no caso já é a data de contratação convertida. Por fim vamos printar isso no Console. Então `funcionarios.forEach(System.out::println)` para que ele possa quebrar a linha. Já criamos as estruturas, já criamos as Querys e na próxima aula vamos mostrar como apresentar isso para o cliente. Então até a próxima.