02

# Salvando o registro

## Transcrição

[00:00] Olá. Agora que nós já temos uma aplicação Spring Data que consegue se conectar ao banco de dados e inserir valores no banco de dados, nós precisamos entregar ao cliente, ao usuário dessa aplicação uma maneira de ele inserir esses valores dinâmicos. Então vamos lá. Primeiro vamos alinhar as expectativas. Nós poderíamos fazer isso através de um site ou através de uma API.

[00:24] Só que se adicionarmos esse contexto a aplicação, ele vai ficar muito grande e você vai ter muita coisa para aprender e não vai focar no objetivo que é aprender o Spring Data. Então para que possamos reduzir aqui as coisas que nós devemos aprender e focar no Spring Data, eu vou utilizar o Console da aplicação Java para pegar os valores que o cliente insere e salvar no banco de dados, pois assim vai ficar bem simples.

[00:53] Então para conseguirmos pegar os valores que o cliente insere dentro do Console do Java, nós utilizamos a classe Scanner passando `scanner = new Scanner (System.in)` . Agora nós já temos o Scanner para que possamos pegar as informações que o cliente vai inserir no Console. Só que eu preciso também disponibilizar para o cliente uma função para que ele possa sair dessa aplicação caso ele não queira mais executar ação nenhuma de inserir, não queira mais fazer nada dentro da aplicação.

[01:48] Para isso vou criar aqui uma variável Boolean. Eu vou chamar ela de `system` e vou atribuir o valor `true` a ela. E o que eu vou falar? Enquanto essa variável estiver `true` , eu quero que as coisas aqui dentro da nossa aplicação

funcione. E aqui também eu vou apresentar para o cliente um cabeçalho para que ele possa saber as opções que ele tem. Então vamos lá.

[02:26] `System.out.println ("Qual acao voce quer executar?")` . E a primeira ação que ele vai ter disponível é exatamente a de sair da aplicação. Então caso ele digite 0, ele vai sair. E também precisamos disponibilizar para ele uma função para que ele possa salvar o registro dinâmico que ele quer criar.

[03;00] Mas para isso eu não vou adicionar essa lógica dentro da nossa classe principal porque senão a classe vai ficar muito grande e vai ficar difícil para leitura. Então para isso vem aqui no nosso projeto ao canto superior esquerdo da tela na pasta Java, pasta principal, botão direito, "New > Package" e crie uma Package chamada "service". Dentro dessa Package nós vamos adicionar essas classes que vão fazer a inserção.

[03:27] Aqui você cria a classe CrudCargoService. Dentro dessa classe nós vamos criar a lógica para inserir os valores no banco de dados. Como eu havia dito na criação do Repository nós precisamos colocar algumas anotações para que o Spring possa identificar que nós vamos utilizar essa classe em outra. Outra classe vai depender dessa daqui que estamos criando.

[03:58] Para essa aqui não utilizamos o `@repository` . Como a definimos como sendo uma Service nós vamos colocar um `@service` aqui em cima. E como tínhamos visto aqui, para fazer a inserção nós precisamos de uma dependência, de uma instância de CargoRepository. Como essa lógica não vai ficar mais na nossa classe principal, nós precisamos criar também, precisamos passar essa instância para a nossa Classe de Crud. Então `private final CargoRepository` .

[04:40] Vou criar aqui o construtor e vou passar o CargoRepository através do construtor e vou falar que o meu `cargoRepository = cargoRepository` que está vindo injetado pelo construtor.

[05:02] Pois bem, agora eu também vou criar um método que eu vou chamar de Inicial porque o cliente vai selecionar a função que ele deseja executar na aplicação e a aplicação vai informar aqui para nós, vai jogar ele aqui para que ele possa escolher quais as funções que ele deseja executar dentro da nossa classe de Cargo. Então `inicial`.

[05:28] Dentro da classe Inicial eu vou criar aqui a primeira função que a nossa classe Cargo vai ter que é a de salvar. Então `private void salvar` e aqui dentro vou criar a lógica para salvar um registro. A primeira coisa que eu vou precisar é que como dissemos que o valor que o cliente insere no console está dentro do Scanner, eu vou precisar de um Scanner dentro do meu Salvar para pegar o valor que o cliente digitou no Console.

[06:05] Agora eu vou colocar aqui uma informação, um cabeçalho para o cliente saber o que ele tem que inserir. Então a primeira coisa que eu quero que ele insira é a descrição, `Descricao do cargo`. Então passando a descrição do cargo eu vou criar uma variável do tipo String, vou chamar de `Descricao` e vou pegar a descrição que o cliente escreveu do Console. E como que eu faço isso?

[06:45] Com o `scanner.next`. Com isso eu pego o valor de texto, o valor String que o cliente inseriu no console. Como a descrição em mãos que o cliente inseriu, vou criar um `Cargo cargo = new Cargo`. E agora eu vou fazer o quê? O meu cargo eu vou atribuir a ele a descrição que o cliente acabou de escrever. Feito isso vou pegar o Repository.

[07:28] É o CargoRepository e vou salvar esse cargo com a descrição dinâmica que o cliente acabou de passar para nós. E feito isso eu vou dar outra informação para o cliente falando que foi salvo. Pronto. Nós já temos aqui a lógica. Só adicionando aqui ao Inicial para que ele chame o Salvar. Aqui nós já temos a lógica para salvar o registro na base de dados.

[08:04] Agora nós precisamos chamar essa classe que faz essa lógica aqui dentro. Como eu disse, a função de salvar não está mais dentro da nossa classe

`main`, a nossa `application`. Então eu não preciso mais de uma variável de cargo. Eu preciso de uma variável para fazer a injeção com a Service.

[08:28] Então eu não vou mais em CargoRepository, eu vou na ServiceRepository porque é aqui na Service que temos a lógica para salvar. Então vou pegar aqui a classe de Service e vou passar aqui. Vou chamar de `cargoService`. Então vou ter uma instância de CargoService. Deixa eu fazer um *import* de `cargoService`. Perfeito.

[09:01] Já tenho aqui um CargoService para que eu possa trabalhar. Posso apagar esses comandos e vou dar uma nova opção para o cliente. Então se o cliente digitar 1, ele vai ir para as funções de cargo. E aqui eu vou criar uma variável `int`, vou chamar `action` que é a ação que o cliente quer dizer executar, vou pegar o Scanner e para pegar um valor inteiro que o cliente digita no console, eu tenho o `nextInt`.

[09:37] E agora eu vou fazer, se a ação que o cliente digitou for igual a 1, eu quero que ele chame o `cargoService`, o método `inicial` passando o `scanner` que nós já temos. Se ele digitar qualquer coisa além de 1, eu quero que a nossa variável `System` passe a ser *false* para que assim a aplicação desligue. Vamos executar a aplicação para ver se ela está funcionando. Então vamos lá, vamos esperar ele iniciar. Pronto, está iniciando.

[10:23] Deixa eu maximizar para que possamos ver melhor aqui. Então vamos esperar. Está iniciando. Agora ele dá as opções para nós, "Qual é a ação que você deseja executar?". No caso eu quero inserir um cargo. Então coloquei um aqui e aí ele pede a descrição. A descrição aqui eu vou colocar `gerente`. Então estou criando aqui a descrição de gerente. Foi salvo.

[10:58] Agora vou lá na nossa base, no DBeaver e vou dar, dentro da tabela, um "Renovar". Pronto, GERENTE está salvo aqui na tabela de Cargo. Com isso já conseguimos salvar. Agora vou deixar como desafio para você, criar a função para atualizar esse registro. Então até a próxima.