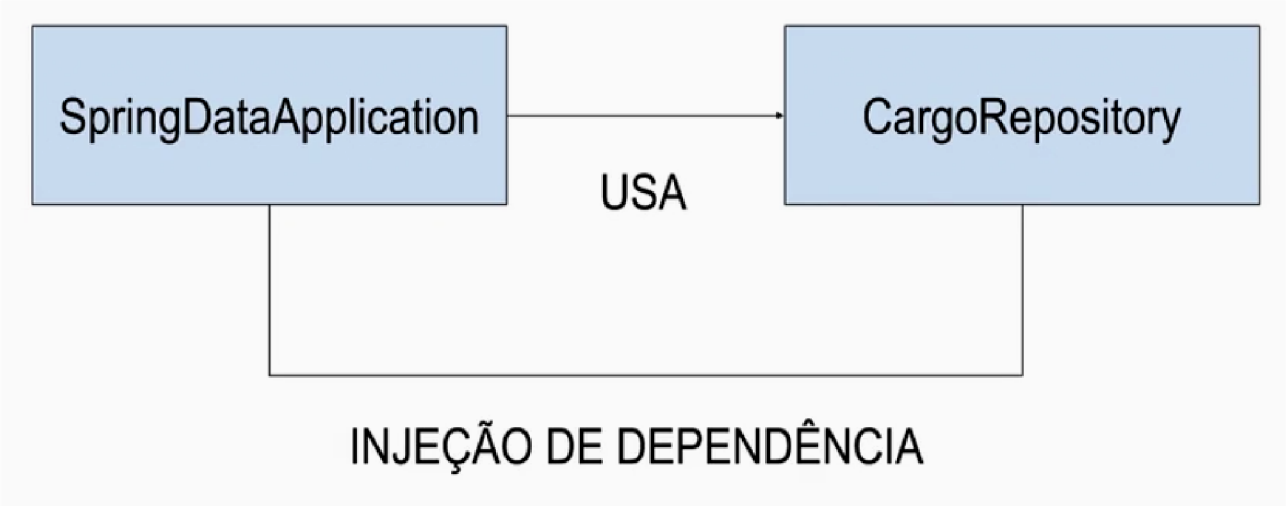
09

Inserindo valores

Transcrição

[00:00] Olá. Bem-vindo a mais uma aula. Agora que já criamos um Repository e já vimos a diferença da agilidade em utilizar o Repository do Spring Data e utilizar o JPA da maneira convencional, precisamos pegar esse Repository que nós criamos e utilizar ele no SpringDataApplication. Porém existe um pequeno problema. Sabemos que em Java nós não podemos dar *new* em interfaces.



[00:29] Então como podemos pegar uma instância de CargoRepository e mandar ela lá para o SpringDataApplication para que possamos salvar um determinado registro em outra classe. Aí entra a injeção de dependência. O Framework do Spring é muito conhecido por injeção de dependência, ele é muito bom isso.

[00:53] Tem várias outras coisas que ele é bom, mas é essa é uma das grandes *features* que diferencia ele no mercado. E aqui vamos entender como que podemos utilizar essa injeção de dependência dentro da nossa aplicação. Então a ideia é mais ou menos essa que nós estamos vendo na tela. Nós temos a

SpringDataApplication que ela precisa utilizar uma classe externa do seu escopo, que é a CargoRepository.

[01:20] E para isso nós precisamos de uma instância de CargoRepository dentro da SpringDataApplication. Então nós vamos enviar essa instância através de injeção de dependência. Vou entrar no Eclipse. Antes de eu ir para SpringDataApplication, deixa eu dar um "Close All" para mostrar para vocês o caminho.

[01:44] Vou pedir para vocês entrarem na pasta inicial, na Source do Java, dentro do pacote do projeto, na pasta Repository, entra na CargoRepository porque a CrudRepository ou todos os Repositorys do Spring, eles precisam que sejam passados para eles dois parâmetros para que eles funcionem. Então vamos colocar o diamante. O primeiro parâmetro é qual é o tipo do objeto que você quer que seja criado um repositor dele, que você quer manipular ele a base de dados.

[02:21] No nosso caso aqui é a tabela de Cargo que queremos manipular. Então aqui é Cargo e o segundo parâmetro é qual é o tipo do `Id`. Se viermos em "Cargo" nós vamos ver que o `Id` é tipo Integer. Então eu venho aqui e coloco `Integer`. Pronto. Agora eu já tenho aqui o CrudRepository configurado para trabalhar com Cargo e o IdInteger.

[02:50] Vou salvar, fechar a classe e agora eu vou na SpringDataApplication. Entrando na pasta "SpringDataApplication" eu vou querer que seja executado um comando após o *Start* da aplicação. Então a nossa aplicação vai iniciar, vai iniciar o Framework Spring e depois disso eu quero que seja executado alguma outra coisa.

[03:14] Para isso nós temos uma interface dentro do Java chamado CommandLineRunner. Ela me obriga a implementação um método chamado Run. Ele é executado logo após a finalização do método Main. Então depois de iniciar a nossa aplicação e iniciar o SpringFramework, nós vamos cair nesse método Run e vai ser executado tudo o que está aqui dentro. E como eu havia

falado no momento da injeção de dependência, nós vamos precisar de uma instância da CargoRepository aqui.

[03:49] Então se eu pegar aqui e fizer `repositor = new Car` , eu não tenho nenhuma classe, eu só tenho uma interface, não tenho nenhuma classe que eu posso estar fazendo o `new` aqui. Então vamos fazer a injeção de dependência desse Repository. Pegar uma instância dele.

[04:23] Já tenho a CargoRepository, vou dar `private` e um `final` . E aí ele vai dar erro para mim falando “você tem uma instância, mas essa é uma instância nula, não foi inicializada. Você não pode utilizar ela”, porque se eu vier aqui embaixo e dar um 'Repository.save' ele simplesmente vai dar *null pointer* porque eu não preenchi essa instância com nada.

[04:48] Então agora eu vou criar a `public SpringDataApplication` que é o construtor da minha classe Spring Data e vou falar o quê? Eu quero que quando você crie a SpringDataApplication você crie com uma instância do CargoRepository. Vou vir aqui e fazer `this.repository = repositor;` o meu Repository que está vazio que é igual a esse Repository que o Spring Data vai criar para mim. Então o que o Spring Data vai fazer?

[05:22] Preciso criar aqui o SpringDataApplication. Mas ele está querendo um CargoRepository. E o que ele vai fazer? Eu anotei aqui com o Repository. Então o SpringBootApplication vai varrer e falar, "Isso é um tipo de entidade que eu entendo, então crie uma instância desse cara aqui". Por isso que ele é muito bom com injeção de dependência, o Framework do Spring.

[05:48] Agora que eu já tenho um objeto, um Repository, um CargoRepository que eu já posso salvar, eu preciso criar um cargo. Como eu disse, deixa eu criar o cargo, `cargo = new Cargo();` . Como nós vimos na aula que criamos a nossa tabela, nós colocamos o `@GeneratedValue` e eu não preciso mais me preocupar em *setar* um cargo, o próprio Framework já vai fazer isso para mim.

[06:22] A única coisaa que eu preciso me preocupar é em *setar* uma descrição para o cargo porque esse cargo já vai ter um ID gerado pelo próprio Framework. Então `setDescription` e aqui eu vou colocar `DESENVOLVEDOR DE SOFTWARE`. Então beleza, já tenho aqui o primeiro "cargo" criado. E como eu faço agora para salvar esse cargo que eu acabei de criar lá na nossa base de dados? `Repository.save`.

[07:03] Se caso fosse no JPA tradicional, teríamos que fazer todo aquele esquema de abrir transação, pegar o EndityManager e depois fechar a persistência. Aqui essa única linha já vai salvar na base de dados. Vou salvar aqui. Voltando ao nosso pacote inicial do lado esquerdo, dentro da pasta "SpringDataApplication", botão direito, "Run As > Java Application".

[07:35] Ele vai iniciar a aplicação e como eu disse, vai vir, vai rodar o comando `main`, vai iniciar o Framework e logo após iniciado o Framework ele já vai executar a classe `run` para salvar o nosso cargo. Vamos dar uma conferida aqui. O Console está *startando*. Ok, *startado*. Agora que ele *statou*, eu vou no DBeaver, vou dar um "Renovar", "Refresh" e agora vou pedir para visualizar a tabela. Vou vir em "Ver dados".

[08:37] Vejam só, já está salvo aqui. "ID1 - DESENVOLVEDOR DE SOFTWARE". Então viram como o Framework é muito rápido para fazer essas funções para nós? Então espero que vocês tenham gostado dessa aula. Nos vemos na próxima.