05

Mapeamento de chaves compostas

Transcrição

[00:00] Olá, pessoal! Vamos estudar mais um recurso da JPA. Até então, em todas as nossas entidades, fizemos um mapeamento da chave primária dessa maneira.

[00:10] Declaramos um atributo do tipo `private Long id;` e colocamos as anotações `@Id` , `@GeneratedValue` com a estratégia que depende do seu banco de dados. Esse é o tipo de mapeamento mais comum. Entretanto, às vezes, tem alguns sistemas, alguns bancos de dados, que precisamos mapear uma chave composta, a chave não é um único atributo, são duas ou mais informações. E a JPA, ela dá suporte para esse tipo de situação.

[00:32] Como fazemos para mapear uma chave composta? Vamos fazer isso na classe "Categoria". Vamos imaginar que a "Categoria", a chave da "Categoria", não é mais o ID, é o `nome` e uma outra informação aqui qualquer. Nós vamos tirar esse `@Id` , esse `@GeneratedValue` . Agora, o ID da "Categoria" vai ser a formação entre o atributo `nome` e um outro atributo aqui qualquer.

[00:59] Então, `private String` , a categoria, ela tem um nome e ela tem, sei lá, um tipo, alguma coisa do gênero, `private String tipo;` . E vamos imaginar que eu quero que esses dois atributos sejam a chave primária e não mais só um. Como eu faço para mapear a chave composta na JPA? A ideia seria extrairmos esses dois atributos para uma classe, que é a classe que representa a chave da categoria. Nós vamos extrair então criando uma nova classe.

[01:24] Continuando, "Ctrl + N", "class", vou chamar de "CategoriaId", geralmente o pessoal coloca esse nome. Aqui dentro dessa nova classe é que vamos jogar esses dois atributos, `private String nome;` e `private String tipo;`. Aquele mesmo esquema, vou gerar o construtor *default*, vou gerar o construtor que recebe esses dois parâmetros, "Source > Generate Constructor". Vou gerar os *getters* e *setters*, "Source > Generate Getters and Setters", OK.

[02:00] Nós vamos usar aquela mesma anotação `@Embeddable`, embutível, também podemos usar ela para chave composta, não só para atributos que queremos separar classes e mapear na mesma entidade, `@Embeddable`. Na "Categoria", agora o atributo que vai representar a chave será um atributo do tipo `private CategoriaId id;`, vou chamar de `id` o atributo.

[02:24] E temos que colocar uma anotação, que é o `@EmbeddedId`. Perceba que é parecido com aquele `@Embeddable` que nós vimos, dos dados pessoais do cliente, só que aqui é `@EmbeddedId`. Então a JPA sabe que aqui dentro estão os atributos que formam a chave primária. É dessa maneira que vai funcionar. Agora a "Categoria", quando for criada, vamos ter que fazer aquelas alterações.

[02:49] `this.categoriaId = new Categoria` - aliás, `this.id = new CategoriaId();`, eu chamei o atributo de `id`. Tenho que passar o `(nome, );`. O tipo, eu vou inventar um aqui fixo, só para não perdermos muito tempo com isso, `(nome, "cpto");`, mas você pegou a ideia, seria um outro atributo que passaríamos aqui como parâmetro. Mas só para não ter um impacto muito grande, vou fazer dessa maneira.

[03:22] O `getNome()`, vou fazer um *delegate*, `return this.id.getNome();`. Agora tudo está compilando certo. Está aqui, é assim que mapeamos uma chave composta, criando uma classe que segue aquele esquema do *embeddable*, só que para adicionar na classe não é o `@Embedded`, é o `@EmbeddedId`, que é o específico para ID, para a JPA saber que não são atributos comuns e sim a chave primária da classe.

[03:51] Porém, isso vai gerar um efeito, você já deve ter pensando, em todo o lugar que fazíamos um *find* - aqui, no nosso projeto, não tem, mas vamos fazer uma simulação aqui. Vou fazer aqui, no "CadastroDeProduto", só para agilizar.

[04:08] Seguindo, `em` e, para fazer um *find*, `em.find(Categoria.class, )` . Eu quero buscar a categoria. E agora, como eu passo a chave? Antes nós passávamos assim: `1l` - deixa só eu quebrar a linha. Passávamos aqui o número, o *long* que era o ID. E agora? Agora como a chave é uma classe, temos que instanciar um objeto do tipo `(Categoria.class, new CategoriaId());` .

[04:36] E nós passamos o nome da categoria, `("CELULARES", "xpto")` , `xpto` , que seria aquela de exemplo. Só importar aqui a classe. Então para fazer busca, para fazer consultas, temos que fazer essa mudança, toda consulta agora eu não passo um atributo, eu passo a classe que representa a chave primária e dentro da classe é que tem os valores que representam a chave composta.

[05:05] Dessa maneira que é feito o mapeamento de chave composta, bem tranquilo, nada de mais, a JPA suporta normalmente. `@Embeddable` . Geralmente, essas classes que têm `@Embeddable` , é comum colocarmos um `implements Serializable` .

[05:22] Agora eu não lembro se isso é do *hibernate* ou se é uma exigência da JPA, mas é comum colocarmos esse atributo aqui. Funciona, tal, mas às vezes pode dar algum tipo de problema, então sempre coloque o `Serializable` nas classes que têm o `@Embeddable` . Esse era o conteúdo de hoje, espero que vocês tenham gostado, vejo vocês no próximo vídeo.