05

Limitando dados de uma consulta

Transcrição

Agora nós faremos mais uma consulta, mas ela será um pouco diferente. Primeiro, vamos copiar o seguinte trecho de `ProdutoDao.java` que está filtrando pelo nome do produto.

```
public List<Produto> buscarPorNomeDaCategoria(String nome) {
        String jpql = "SELECT p FROM Produto p WHERE
p.categoria.nome = :nome";
        return em.createQuery(jpql, Produto.class)
                .setParameter("nome", nome)
                .getResultList();
    }
```

COPIAR CÓDIGO

Até então, nas consultas que estávamos fazendo, nós carregávamos a entidade inteira com `SELECT p`, portanto, ele traz a entidade com todos os atributos. Mas, se quisermos apenas o preço do produto, isto é, buscar um produto por um determinado nome, não todas as informações dele. Então, qual o sentido de carregar uma entidade inteira, se queremos só um atributo? É possível filtrar um atributo devolvido ao invés da entidade inteira? Sim, e aprenderemos nesta aula.

O método deste retorno não será um `List<Produto>` e nem um `Produto`, porque queremos trazer apenas um atributo que, no caso, é o preço. Se entrarmos na classe `Produto.java`, o preço é do tipo `BigDecimal`, então, na nossa classe DAO, o retorno do método é `BigDecimal`.

```
public BigDecimal buscarPorNome(String nome) {
    String jpql = "SELECT p FROM Produto p WHERE
p.categoria.nome = :nome";
        return em.createQuery(jpql, Produto.class)
                    .setParameter("nome", nome)
                    .getResultList();
    }
```

COPIAR CÓDIGO

Queremos devolver apenas o `BigDecimal`, o preço do produto. Vamos também renomear o nome do método, `buscarPrecoDoProdutoComNome()` e então passamos o nome como parâmetro para buscarmos apenas o preço. Na `Query`, para filtrar, precisamos pensar também como se fosse orientação a objetos. No lugar de `SELECT p`, faremos, `SELECT p.preco`, isto é, o nome do atributo.

Com isso a JPA sabe que o nosso `SELECT` não é para trazer a entidade inteira, e, sim, um único atributo dessa entidade.

```
public BigDecimal buscarPorNome(String nome) {
    String jpql = "SELECT p.preco FROM Produto p WHERE
p.categoria.nome = :nome";
        return em.createQuery(jpql, Produto.class)
                    .setParameter("nome", nome)
                    .getResultList();
    }
```

COPIAR CÓDIGO

O resto continua igual, `FROM Produto p WHERE p.nome = :nome`, depois fazemos `em.createQuery(jpql, Produto.class)`, nela, apenas substituiremos o `Produto.class` por `BigDecimal.class`, porque agora o retorno será esse. Em seguida, temos um pequeno problema: nós setamos tudo, mas o método que estamos chamando é o `getResultList()`, que não devolve um `BigDecimal` e, sim, um `List`.

Se queremos apenas um único registro em uma *query*, não podemos chamar o `getResultList()`, nós chamaremos o método `getSingleResult()`, que serve para carregar uma única entidade ou um único registro que virá na consulta, ou seja, não é uma lista, mas, sim, um único resultado. Feito isso, a princípio está pronto o nosso método e podemos testar.

```
public BigDecimal buscarPorNome(String nome) {
    String jpql = "SELECT p.preco FROM Produto p WHERE
p.categoria.nome = :nome";
        return em.createQuery(jpql, BigDecimal.class)
                    .setParameter("nome", nome)
                    .getSingleResult();
    }
```

COPIAR CÓDIGO

Vamos abrir a classe `CadastroDeProduto.java` e criar a seguinte linha.

```
BigDecimal precoDoProduto =
produtoDao.buscarPrecoDoProdutoComNome()
```

COPIAR CÓDIGO

E passaremos o nome `"Xiaomi Redmi"`. Na sequência, daremos um `System.out.println()` para conferir se ele está carregando corretamente o `precoDoProduto`, que, no caso, deveria ser 800. Vamos rodar ("Run As > 1 Java Application")

```
BigDecimal precoDoProduto =
produtoDao.buscarPrecoDoProdutoComNome("Xiaomi Redmi")
System.out.println(precoDoProduto)
```

COPIAR CÓDIGO

Analisando o Console, notaremos que ele carregou "800.00". Agora, só para garantir que esse foi o `System.out` correto, vamos concatenar `"Preco do Produto: "` com `+precoDoProduto`.

```
System.out.println("Preco do Produto: " +precoDoProduto)
```

COPIAR CÓDIGO

Vamos rodar e observar o Console, e bem ao final, encontraremos "Preco do Produto: 800.00". Ao fazer uma consulta, não precisamos, necessariamente, carregar a entidade inteira. Podemos limitar qual é o atributo para trazer a informação que queremos, evitando um *select* desnecessário com uma grande quantidade de informações.

Inclusive, o SQL que ele gerou é bem enxuto: "Select", "produto0_.preco as col_0_0_". Estes nomes o Hibernate gera na hora de montar a *query*. Provavelmente existe um algoritmo que, para cada atributo, aumenta esses números só para simplificar. Enfim, ele montou a *query* corretamente e trouxe apenas um atributo.

Então, eventualmente, se precisarmos carregar um único atributo de uma entidade, podemos montar a *query* dessa maneira e funcionará corretamente. Espero que tenham gostado do vídeo!! Abraços!!