05

Estados no delete da entidade

Transcrição

Nós estávamos discutindo sobre os estados da entidade da JPA e ainda não estudamos a parte de consultas, que será tema da próxima aula. Mas, apenas para complementar, existe mais uma situação no ciclo de vida, que é quando uma entidade está no banco de dados.

Isto é, já fechamos o `EntityManager` e abrimos um novo, não temos mais uma referência para a entidade no estado detached, então, como faremos para trazê-la para o estado managed. Basicamente, queremos trazê-la do banco para o estado managed, para isso, precisaremos dos métodos `find()` / `createQuery()`. Nós veremos estes métodos de consulta na próxima aula.

Portanto, existe mais essa transição de estados. Além do managed para o banco de dados, quando estamos commitando ou fazendo um `flush()` no `EntityManager`, existe a transição do banco de dados para o managed, quando fazemos uma consulta, uma *query*. Agora, voltando para o código, em `CadastroDeProduto.java`, faltou apenas um último estado, que é quando excluímos uma entidade.

Para excluir, temos a seguinte situação: do estado managed, podemos chamar o método `remove()` do `EntityManager` e ela passa para o estado **REMOVED**. Quando o `commit()` ou o `flush()` for chamado, ele vai sincronizar o `remove()` com o banco de dados disparando um *delete*. Vamos simular essa situação.

Em `CadastroDeProduto.java`, temos a entidade, persistimos, atualizamos o nome, fizemos um `flush()`, ele disparou o *update*, demos um `clear()`,

voltamos a entidade para o estado managed chamando o `merge()`, atualizamos o nome e fizemos um `flush()`.

Depois disso, ela ainda está managed, porque não fizemos um `clear()` nem `close()`, então podemos excluí-la do banco de dados com `em.remove()`, passando a entidade `celulares`, e quando fizermos um `flush()`, ela deve disparar um *delete* no banco de dados.

```
celulares = em.merge(celulares);
celulares.setNome("1234");
em.flush();
em.remove(celulares);
em.flush();
```

COPIAR CÓDIGO

Vamos rodar o código e ver a saída no Console. Ele fez o *insert*, o *update*, o *select* devido ao `merge()`, mais um *update* e um *delete*. Então, a JPA deleta baseada no *id*, no atributo da chave primária. Com isso, fechamos todos os cenários possíveis, todas as transições de estados de uma entidade JPA.

Quando uma entidade nasce, ela está no estado transient, para salvá-la no banco de dados, temos que movê-la para o estado managed e, então, entra o método `persist()`. Commitamos a transação ou fizemos um `flush()` no `EntintyManager`, ele sincroniza com o banco de dados. Se fecharmos esse `EntityManager` (ou se dermos um `clear()`), a entidade vai para o estado detached.

Se temos a entidade detached, podemos chamar o método `merge()` para trazê-la novamente ao método managed, ou, se ela já está no banco de dados, é possível fazer um `find()` ou uma `createQuery()` e, por fim, se quisermos excluí-la do banco de dados, do managed, chamamos o `remove()` e ela passa para o estado de removed.

Esses são os estados possíveis e as transições que acontecem na JPA. Conforme vamos utilizando o `EntityManager` e esses métodos, ocorrem transições de estados. Se quisermos complementar a nossa classe DAO, `CategoriaDao.java`, nós já temos o método `cadastrar()` e o `atualizar()`, para fazer um método para excluir, será parecido.

```
public void remover(Categoria categoria) {
    this.em.remove(categoria);
```

COPIAR CÓDIGO

A única ressalva é que essa categoria precisa estar no estado managed. Pode acontecer de não termos essa garantia, de não sabermos se ela está managed. Se ela estiver detached e chamarmos para o `remove()`, será que teremos problemas? Vamos simular o nosso código do método `main()`. Antes de fazer o `remove()`, vamos fazer o `clear()`. Agora ela está detached, e, na sequência, chamaremos o `remove()` para testar se ele deletará do banco de dados.

```
celulares = em.merge(celulares);
celulares.setNome("1234");
em.flush();
em.clear();
em.remove(celulares);
em.flush();
```

COPIAR CÓDIGO

Vamos rodar e ao analisar o Console, receberemos uma *exception*, "IlegalArgumentException: Removing a detached instance", ou seja, não é permitido remover uma entidade que está detached, ela precisa estar managed. Em `CategoriaDao.java`, nós não sabíamos se, no método `atualizar()`, a classe estava managed e chamamos o `merge()` para garantir.

Podemos fazer o mesmo no método `remover()`, chamar o método `merge()` para forçá-la a ficar managed, e. na sequência, fazer o `remove()`. Então,

pegaremos o `merge()` reatribuindo o objeto, isto é, `categoria = em.merge(categoria)` , e depois fazemos o `remove(categoria)` em cima da categoria.

```
public void remover(Categoria categoria) {
    em.merge(categoria);
        this.em.remove(categoria);
    }
```

COPIAR CÓDIGO

É importantíssimo lembrar de reatribuir. Estamos fazendo `merge()` , mas não guardamos a entidade *mergeada*, a entidade que está no estado managed, então, estamos mexendo na categoria que ainda está detached, por isso, precisamos reatribuir. Podemos fazer desta maneira, `categoria = em.merge(categoria);` , só para garantir que a entidade está managed.

Assim fica uma classe DAO com a JPA, temos o `cadastrar()` , o `atualizar()` e o `remover()` , faltam apenas os métodos de consulta, que discutiremos na próxima aula. Espero que tenham aprendido um pouco sobre transições de estados de uma entidade, como elas funcionam, as possíveis transições e as excessões. Vejo vocês na próxima aula!! Abraços!!