Transcrição

Na aula de hoje discutiremos um pouco sobre o ciclo de vida das entidades da JPA. Sempre que trabalhamos com as entidades, quando instanciamos um objeto e chamamos os métodos do `@EntityManager`, elas ficam trafegando entre alguns estados no ciclo de vida e é interessante conhecer esses estados para entender melhor como a JPA trabalha "debaixo dos panos" (de forma oculta) e, principalmente, entender os erros, algumas *exceptions* que ela lança, a depender da situação e do que estivermos fazendo na aplicação.

Para deixar mais compreensível, vamos fazer uma alteração na classe `CadastroDeProduto`, temporariamente, para entendermos a questão do ciclo de vida. Vamos apagar a parte de `Produto celular`, as classes DAO, `ProdutoDao` e `CategoriaDao` e também o `categoria.Dao`, deixando apenas a nossa categoria.

```
public class CadastroDeProduto {

    public static void main(String[] args) {
        Categoria celulares = new Categoria("CELULARES");

        EntityManager em = JPAUtil.getEntityManager();

        em.getTransaction().begin();

        em.getTransaction().commit();
        em.close();
    }
```

```
}
```

COPIAR CÓDIGO

Então, estamos instanciando uma entidade, `celulares`, passando um parâmetro, `"CELULARES"`, que vai ser setado no atributo. Também criamos o `EntityManager`, demos um `begin()` na transação um `commit()` e um `close()`. Agora faremos manualmente a classe DAO para entendermos melhor e mais simples.

Então, tínhamos um `em.persist(celulares)` (EntityManager persist na categoria celulares). Como funciona a questão das entidades, do ciclo de vida. Toda vez que instanciamos uma entidade, por exemplo: `Categoria celulares = new Categoria("CELULARES")`, neste momento, ela ainda não está salva no banco de dados, o `EntityManager` não conhece essa categoria, então, como isso funciona do ponto de vista da JPA?

Dicutindo, portanto, sobre o **ciclo de vida**, quando damos "new", isto é, quando instanciamos uma entidade, para a JPA, a entidade está em um estado chamado de **TRANSIENT**. O estado transient é de uma entidade que nunca foi persistida, não está gravada no banco de dados, não tem um *id* e não está sendo gerenciada pela JPA.

A JPA desconhece essa entidade. É como se fosse um objeto Java puro, que não tem nada a ver com persistência. Esse é o primeiro estado de uma entidade. Nesse estado, se alteramos o atributo da entidade ou qualquer outra coisa que façamos com a entidade, o `EntityManager` não está gerenciando, nem verificando, e não vai sincronizar com o banco de dados se nós *commitarmos* e fizermos o *close* do `EntityManager`.

Por exemplo, se comentarmos a seguinte linha:

```
//em.persist(celulares);
```

COPIAR CÓDIGO

Nós instanciamos uma entidade, abrimos um `EntityManager`, demos um `begin()`, um `commit()` e um `close()`. Agora, se rodarmos o código, ele imprimirá no Console que só rodou os comandos para criar a tabela. Não deu *insert*, não fez nada. O motivo é que a entidade é transient, isto é, não está sendo gerenciada pela JPA. Assim, ela simplesmente ignora a entidade.

Porém, quando chamamos o método `persist()`, ela move do estado transient para o estado **MANAGED** ou gerenciado. Managed é o principal estado que uma entidade pode estar, portanto, tudo que acontece com uma entidade nesse estado, a JPA observará e poderá sincronizar com o banco de dados, a depender do que fizermos.

É o caso desta parte em `CadastroDeProduto`, quando chamamos o método `em.persist(celulares)` e, agora, a JPA está olhando para essa entidade. Desta maneira, se decidirmos alterá-la, a JPA observará. Ainda no `CadastroDeProduto`, havíamos criado o objeto de nome `"CELULARES"`.

Se na linha abaixo do `persist()` setarmos um nome para outro valor, por exemplo, `celulares.setNome("XPTO");` estamos mexendo na entidade, alterando um atributo dela. Como ela está no estado managed, a JPA está olhando isso.

```
        Categoria celulares = new Categoria("CELULARES");

        EntityManager em = JPAUtil.getEntityManager();
        em.getTransaction().begin();

        em.persist(celulares);
        celulares.setNome("XPTO");

        em.getTransaction().commit();
        em.close();
    }

}
```

COPIAR CÓDIGO

Quando commitarmos a transação, fizermos um `commit()`, ou, se não finalizarmos a transação, mas se tentarmos sincronizar o estado dessa entidade com o banco de dados, existe um método `flush()` no `EntityManager` que serve para este caso, em que não queremos *commitar* a transação, ainda faremos algumas operações, mas já desejamos sincronizar essa entidade com o banco de dados para ela gerar o id ou vinculá-la a outra.

No momento em que fazemos o `commit()` ou o `flush`, a JPA pega todas as entidades que estiverem no estado managed e sincroniza com o banco de dados. Então, tínhamos uma entidade que era transient, está gerenciada e, depois que commitamos, ele perceberá que a entidade não tem id, que ela era transient, ou seja, é necessário fazer um *insert* dela no banco de dados.

Vamos rodar ("Run As > 1 Java Application) e, no Console, ele deu um *insert* na categoria, "Hibernate: insert into categorias (id, nome) values (null, ?)", e, na sequência, deu um *update*, "Hibernate: update categorias set nome=? where id=?", significa que ele atualizou a categoria.

E o motivo de ter atulizado a catagoria é que fizemos o `persist()`, mas, na sequência, mudamos o nome dela. Como a categoria está gerenciada, se alteramos o atributo, ele saberá que é necessário fazer o *update*, pois atualizamos alguma informação da entidade. Em outras palavras, se a categoria está gerenciada, "managed", a JPA observará, e se alterarmos ela, a JPA sincronizará com o `commit()` ou com `flush()` ao banco de dados.

A partir do momento em que fechamos o `EntityManager`, isto é, `em.close()` ou `clear()` (para limpar as entidades gerenciadas do `EntityManager`), a categoria muda de estado. Se ela estava salva antes, passa para um estado chamado de **DETACHED**, que é um estado destacado.

O detached é um estado em que a entidade não é mais transient, porque tem id, já foi salva no banco de dados, porém, não está mais sendo gerenciada. Portanto, se mexermos nos atributos, a JPA não disparará *update* e nem fará mais nada. Vamos simular esse estado.

Em `CadastrDeProduto.java`, temos a entidade, o `persist()`, o atributo e o `commit()`, ele sincronizou com o banco de dados (fez o *insert* e o *update*) e fizemos o `em.close()` do `EntityManager`. Se, na sequência, pegarmos a entidade `celulares.setNome()` e mudarmos o nome de novo, por exemplo, para `"1234"` e rodarmos o código, ele não deveria fazer um segundo *update*.

```
      Categoria celulares = new Categoria("CELULARES");

        EntityManager em = JPAUtil.getEntityManager();
          em.getTransaction().begin();

          em.persist(celulares);
          celulares.setNome("XPTO");

          em.getTransaction().commit();
          em.close();

          celulares.setNome("1234");
      }

  }
```

COPIAR CÓDIGO

Nós mexemos no nome depois de fechar o `EntityManager`, então, ele não está mais "managed", está no estado "detached". Conferindo no Console, está correto, ele fez um *insert* e um único *update*, de quando ainda estava aberto o `EntityManager`, isto é, quando a entidade ainda estava gerenciada. Se alteramos qualquer coisa abaixo do `em.close()`, ele vai ignorar.

Esse é o ciclo de vida para quando estamos falando em persistência, em *insert*, criação de objetos. Estudamos o que acontece quando criamos uma entidade, instanciamos, chamamos o `persist()`, fechamos o `EntityManager` ou mexemos em um atributo da entidade. Nesta aula, nós simulamos todas essas etapas no código.

Agora, já entendemos como funciona a parte de criação de uma entidade. No próximo vídeo, discutiremos sobre outros cenários, outras possíveis transições de estados e como podem acontecer. Vejo vocês lá!!