04

Conhecendo o cache de primeiro nível

Transcrição

A forma mais comum de trabalhar com o *cache* é utilizar o que já vem por padrão nos `EntityManager`. O **cache de primeiro nível**, como é conhecido, impede que carreguemos a entidade duas vezes do banco. Por exemplo, ao executarmos o método `find` duas vezes para o mesmo `id`, somente a primeira chamada irá disparar uma *query*. Portanto, ao executarmos o `find` pela segunda vez em busca do mesmo objeto usando **o mesmo EntityManager** ele saberá que aquela entidade já foi carregada e irá reutilizar o resultado sem realizar uma nova busca ao banco de dados.

Vamos ver um exemplo, mas antes insira um produto no banco de dados e altere a estratégia do `Hibernate Mapping` para `update` na classe `JpaConfigurator`:

```
insert into Produto(nome) values('Livro de arquitetura');
```

```
public class JpaConfigurator {
    ...
    @Bean
    public LocalContainerEntityManagerFactoryBean getEntityManagerFactory(DataSource dataSource)

        ...
        props.setProperty("hibernate.hbm2ddl.auto", "update");
        ...
    }
}
```

Agora, vamos precisar de um `EntityManager` para testarmos o *cache*. No nosso projeto, gerenciar a JPA é responsabilidade do Spring IoC, dessa maneira, precisaremos fazer um `lookup` do `EntityManagerFactory` a partir do contexto do Spring:

```
public class TesteCache {
    public static void main(String[] args) {

        ApplicationContext ctx = new AnnotationConfigApplicationContext(JpaConfigurator.class);

        EntityManagerFactory emf = (EntityManagerFactory) ctx.getBean(EntityManagerFactory.class
        EntityManager em = emf.createEntityManager();
    }
}
```

Com o `EntityManager` pronto para usar, vamos realizar um `find` pelo *id* do produto que acabamos de cadastrar no banco:

```
public class TesteCache {
    public static void main(String[] args) {
```

```
        ApplicationContext ctx = new AnnotationConfigApplicationContext(JpaConfigurator.class);

        EntityManagerFactory emf = (EntityManagerFactory) ctx.getBean(EntityManagerFactory.clas
        EntityManager em = emf.createEntityManager();

        Produto produto = em.find(Produto.class, 1);
        System.out.println("Nome: " + produto.getNome());
    }
}
```

Podemos ver no console que ele realizou um `select` e logo após imprimiu o nome do produto:

```
Hibernate: select produto0_.id as id1_2_0_, produto0_.descricao as descrica2_2_0_, produto0_.li

Nome: Livro de arquitetura
```

Como esperávamos, quando ele realiza o primeiro `select` o `EntityManager` passa a conhecer os dados que foram retornados para aquele `id`. Mas o que acontecerá se fizermos um outro `select`? Vamos testar e descobrir!

```
public class TesteCache {
    public static void main(String[] args) {

        ApplicationContext ctx = new AnnotationConfigApplicationContext(JpaConfigurator.class);

        EntityManagerFactory emf = (EntityManagerFactory) ctx.getBean(EntityManagerFactory.clas
        EntityManager em = emf.createEntityManager();

        Produto produto = em.find(Produto.class, 1);
        System.out.println("Nome: " + produto.getNome());

        Produto outroProduto = em.find(Produto.class, 1);
        System.out.println("Nome: " + outroProduto.getNome());
    }
}
```

Agora, no console recebemos:

```
Hibernate: select produto0_.id as id1_2_0_, produto0_.descricao as descrica2_2_0_, produto0_.li

Nome: Livro de arquitetura
Nome: Livro de arquitetura
```

Analisando os resultados

Como afirmado anteriormente, sempre que buscarmos por alguma entidade, o cache de primeiro nível guarda os dados do objeto retornado do select. Dessa forma, caso haja uma nova busca pelo mesmo **id**, o cache já conhece seu resultado e reutiliza-os evitando que uma segunda query seja disparada, inutilmente, no banco.

Cache para vários EntityManagers?

Vamos imaginar o cenário que estamos trabalhando: uma aplicação web. Configuramos nosso projeto para que seja criado um novo `EntityManager` por requisição. Vamos criar mais um `EntityManager`, de exemplo, para tentar buscar um segundo produto:

```
public class TesteCache {
    public static void main(String[] args) {

        ApplicationContext ctx = new AnnotationConfigApplicationContext(JpaConfigurator.class);

        EntityManagerFactory emf = (EntityManagerFactory) ctx.getBean(EntityManagerFactory.clas

        EntityManager em = emf.createEntityManager();
        EntityManager em2 = emf.createEntityManager(); // criando o segundo EntityManager

        Produto produto = em.find(Produto.class, 1);
        System.out.println("Nome: " + produto.getNome());

        Produto outroProduto = em2.find(Produto.class, 1); // buscando a mesma entidade com o se
        System.out.println("Nome: " + outroProduto.getNome());

    }
}
```

Vimos que o cache de primeiro nível pode nos ajudar e muito a diminuir a quantidade de *queries* aumentando, assim, a performance da aplicação. Mas, o que acontecerá caso precisemos instanciar um outro `EntityManager` para realizar a segunda busca? Isso é algo muito comum quando trabalhamos com aplicações web onde queremos deixar um `EntityManager` viver **apenas** o tempo de uma requisição (como já fizemos anteriormente). Ou em casos onde trabalhamos com *Persistence Context Transactional* em que um *EntityManager* dura o tempo de uma transação. Nesses casos, como podemos fazer uso de um cache de entidade que seja acessível a partir de todos as instâncias de `EntityManager` que criarmos?