hibernate延迟加载(get和load的区别) - xiaoluo501395377 - 博客园

在hibernate中我们知道如果要从数据库中得到一个对象，通常有两种方式，一种是通过session.get()方法，另一种就是通过session.load()方法，然后其实这两种方法在获得一个实体对象时是有区别的，在查询性能上两者是不同的。

一.load加载方式

当使用load方法来得到一个对象时，此时hibernate会使用延迟加载的机制来加载这个对象，即：当我们使用session.load()方法来加载一个对象时，此时并不会发出sql语句，当前得到的这个对象其实是一个代理对象，这个代理对象只保存了实体对象的id值，只有当我们要使用这个对象，得到其它属性时，这个时候才会发出sql语句，从数据库中去查询我们的对象。

```
session = HibernateUtil.openSession();  /*  * 通过load的方式加载对象时，会使用延迟加载机制，此时并不会发出sql语句，只有当我们需要使用的时候才会从数据库中去查询  */  User user = (User)session.load(User.class, 2);
```

我们看到，如果我们仅仅是通过load来加载我们的User对象，此时从控制台我们会发现并不会从数据库中查询出该对象，即并不会发出sql语句，但如果我们要使用该对象时：

```
session = HibernateUtil.openSession();  User user = (User)session.load(User.class, 2);
System.out.println(user);
```

此时我们看到控制台会发出了sql查询语句，会将该对象从数据库中查询出来：

```
Hibernate: select user0_.id as id0_0_, user0_.username as username0_0_, user0_.password as password0_0_, user0_.born as born0_0_ from user user0_ where user0_.id=?User [id=2, username=aaa, password=111, born=2013-10-16 00:14:24.0]
```

这个时候我们可能会想，那么既然调用load方法时，并不会发出sql语句去从数据库中查出该对象，那么这个User对象到底是个什么对象呢？

其实这个User对象是我们的一个代理对象，这个代理对象仅仅保存了id这个属性：

复制代码

```
session = HibernateUtil.openSession();  /*  * 通过load的方式加载对象时，会使用延迟加载机制，此时得到的User对象其实是一个  * 代理对象，该代理对象里面仅仅只有id这个属性  */
User user = (User)session.load(User.class, 2);  System.out.println(user.getId());

console:  2
```

复制代码

我们看到，如果我们只打印出这个user对象的id值时，此时控制台会打印出该id值，但是同样不会发出sql语句去从数据库中去查询。这就印证了我们的这个user对象仅仅是一个保存了id的代理对象，但如果我需要打印出user对象的其他属性值时，这个时候会不会发出sql语句呢？答案是肯定的：

复制代码

```
session = HibernateUtil.openSession();  /*  * 通过load的方式加载对象时，会使用延迟加载机制，此时得到的User对象其实是一个  * 代理对象，该代理对象里面仅仅只有id这个属性  */
User user = (User)session.load(User.class, 2);  System.out.println(user.getId());  // 如果此时要得到user其他属性，则会从数据库中查询  System.out.println(user.getUsername());
```

复制代码

此时我们看控制台的输出：

```
2Hibernate: select user0_.id as id0_0_, user0_.username as username0_0_, user0_.password as password0_0_, user0_.born as born0_0_ from user user0_ where user0_.id=?aaa
```

相信通过上述的几个例子，大家应该很好的了解了load的这种加载对象的方式了吧。

二、get加载方式

相对于load的延迟加载方式，get就直接的多，当我们使用session.get()方法来得到一个对象时，不管我们使不使用这个对象，此时都会发出sql语句去从数据库中查询出来：

```
session = HibernateUtil.openSession();  /*  * 通过get方法来加载对象时，不管使不使用该对象，都会发出sql语句，从数据库中查询  */  User user = (User)session.get(User.class, 2);
```

此时我们通过get方式来得到user对象，但是我们并没有使用它，但是我们发现控制台会输出sql的查询语句：

Hibernate: select user0_.id as id0_0_, user0_.username as username0_0_, user0_.password as password0_0_, user0_.born as born0_0_ from user user0_ where user0_.id=?

因此我们可以看到，使用load的加载方式比get的加载方式性能要好一些，因为load加载时，得到的只是一个代理对象，当真正需要使用这个对象时再去从数据库中查询。

三、使用get和load时的一些小问题

当了解了load和get的加载机制以后，我们此时来看看这两种方式会出现的一些小问题：

①如果使用get方式来加载对象，当我们试图得到一个id不存在的对象时，此时会报NullPointException的异常

```
session = HibernateUtil.openSession();            /*             * 当通过get方式试图得到一个id不存在的user对象时，此时会报NullPointException异常             */            User user = (User)session.get(User.class, 20);
System.out.println(user.getUsername());
```

此时我们看控制台的输出信息，会报空指针的异常：

Hibernate: select user0_.id as id0_0_, user0_.username as username0_0_, user0_.password as password0_0_, user0_.born as born0_0_ from user user0_ where user0_.id=?java.lang.NullPointerException   ....

这是因为通过get方式我们会去数据库中查询出该对象，但是这个id值不存在，所以此时user对象是null，所以就会报NullPointException的异常了。

②如果使用load方式来加载对象，当我们试图得到一个id不存在的对象时，此时会报ObjectNotFoundException异常：

复制代码

```
session = HibernateUtil.openSession();            /*             * 当通过get方式试图得到一个id不存在的user对象时，此时会报ObjectNotFoundException异常             */            User user = (User)session.load(User.class, 20);
System.out.println(user.getId());            System.out.println(user.getUsername());
```

复制代码

我们看看控制台的输出：

20Hibernate: select user0_.id as id0_0_, user0_.username as username0_0_, user0_.password as password0_0_, user0_.born as born0_0_ from user user0_ where user0_.id=?org.hibernate.ObjectNotFoundException: No row with the given identifier exists: [com.xiaoluo.bean.User#20].....

为什么使用load的方式和get的方式来得到一个不存在的对象报的异常不同呢？？其原因还是因为load的延迟加载机制，使用load时，此时的user对象是一个代理对象，仅仅保存了当前的这个id值，当我们试图得到该对象的username属性时，这个属性其实是不存在的，所以就会报出ObjectNotFoundException这个异常了。

③org.hibernate.LazyInitializationException异常

接下来我们再来看一个例子：

复制代码

```
publicclass UserDAO{    public User loadUser(int id)    {        Session session = null;        Transaction tx = null;
User user =  null;        try        {            session = HibernateUtil.openSession();            tx =
session.beginTransaction();            user = (User)session.load(User.class, 1);            tx.commit();        }
catch (Exception e)        {            e.printStackTrace();            tx.rollback();        }        finally        {
HibernateUtil.close(session);        }        return user;    }}
```

复制代码

复制代码

```
    @Test    publicvoid testLazy06()    {        UserDAO userDAO = new UserDAO();        User user =
userDAO.loadUser(2);        System.out.println(user);    }
```

复制代码

模拟了一个UserDAO这样的对象，然后我们在测试用例里面来通过load加载一个对象，此时我们发现控制台会报LazyInitializationException异常

org.hibernate.LazyInitializationException: could not initialize proxy - no Session   ....

这个异常是什么原因呢？？还是因为load的延迟加载机制，当我们通过load()方法来加载一个对象时，此时并没有发出sql语句去从数据库中查询出该对象，当前这个对象仅仅是一个只有id的代理对象，我们还并没有使用该对象，但是此时我们的session已经关闭了，所以当我们在测试用例中使用该对象时就会报LazyInitializationException这个异常了。

所以以后我们只要看到控制台报LazyInitializationException这种异常，就知道是使用了load的方式延迟加载一个对象

了，解决这个的方法有两种，一种是将load改成get的方式来得到该对象，另一种是在表示层来开启我们的session和关闭session。

至此，hibernate的两种加载方式get和load已经分析完毕！！！