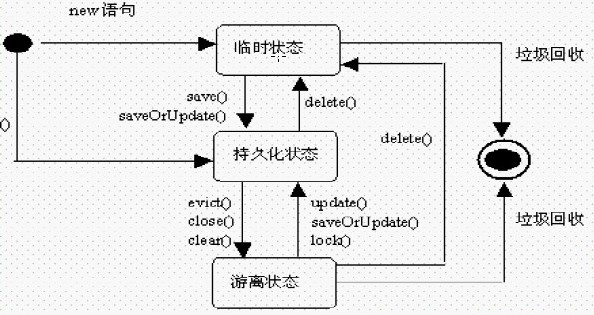
# 概述

Hibernate 中Java对象的状态   
1.临时状态 (transient)   
特征：   
   1】不处于Session 缓存中   
   2】数据库中没有对象记录   
Java如何进入临时状态   
   1】通过new语句刚创建一个对象时   
   2】当调用Session 的delete()方法，从Session 缓存中删除一个对象时。    
  
2.持久化状态(persisted)   
特征：   
   1】处于Session 缓存中   
   2】持久化对象数据库中设有对象记录   
   3】Session 在特定时刻会保持二者同步   
Java如何进入持久化状态   
   1】Session 的save()把临时－》持久化状态   
   2】Session 的load(),get()方法返回的对象   
   3】Session 的find()返回的list集合中存放的对象   
   4】Session 的update(),saveOrupdate()使游离－》持久化   
3.游离状态(detached)   
特征：   
   1】不再位于Session 缓存中   
   2】游离对象由持久化状态转变而来，数据库中可能还有对应记录。   
Java如何进入持久化状态－》游离状态   
   1】Session 的close()方法   
   2】Session 的evict()方法，从缓存中删除一个对象。提高性能。少用。

# 深入

学过hibernate的人都可能都知道hibernate有三种状态，transient(瞬时状态)，persistent(持久化状态)以及detached(离线状态)，大家伙也许也知道这三者之间的区别，比如瞬时状态就是刚new出来一个对象，还没有被保存到数据库中，持久化状态就是已经被保存到数据库中，离线状态就是数据库中有，但是session中不存在该对象。但是大家又是否对hibernate的session的那几个特殊方法一清二楚呢？或者说大家是否能够一眼就快速看出一个测试用例在反复的调用session的诸如save，update方法后会到底发出多少条SQL语句呢？本篇随笔将会给你答案，本篇随笔将会以大量的测试用例来掩饰hibernate的这三种状态的转变，相信看完本篇随笔的你会对hibernate的那三种状态有更深入的理解。

好了，废话不多说了，相信大家都知道hibernate的这三种状态的含义，那我们就通过一张图来开始我们的深入hibernate的三种状态之旅吧。



1.TestTransient

[IMG_257](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

User user = new User();

user.setUsername("aaa");

user.setPassword("aaa");

user.setBorn(new Date());

/\*

\* 以上user就是一个Transient(瞬时状态),此时user并没有被session进行托管，即在session的

\* 缓存中还不存在user这个对象，当执行完save方法后，此时user被session托管，并且数据库中存在了该对象

\* user就变成了一个Persistent(持久化对象)

\*/

session.save(user);

session.getTransaction().commit();

[IMG_258](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

此时我们知道hibernate会发出一条insert的语句，执行完save方法后，该user对象就变成了持久化的对象了

Hibernate: insert into t\_user (born, password, username) values (?, ?, ?)

2.TestPersistent01

[IMG_259](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

User user = new User();

user.setUsername("aaa");

user.setPassword("aaa");

user.setBorn(new Date());

//以上u就是Transient（瞬时状态），表示没有被session管理并且数据库中没有

//执行save之后，被session所管理，而且，数据库中已经存在，此时就是Persistent状态 session.save(user);

//此时u是持久化状态，已经被session所管理，当在提交时，会把session中的对象和目前的对象进行比较

//如果两个对象中的值不一致就会继续发出相应的sql语句

user.setPassword("bbb");

//此时会发出2条sql，一条用户做插入，一条用来做更新

session.getTransaction().commit();

[IMG_260](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

在调用了save方法后，此时user已经是持久化对象了，被保存在了session缓存当中，这时user又重新修改了属性值，那么在提交事务时，此时hibernate对象就会拿当前这个user对象和保存在session缓存中的user对象进行比较，如果两个对象相同，则不会发送update语句，否则，如果两个对象不同，则会发出update语句。

Hibernate: insert into t\_user (born, password, username) values (?, ?, ?)

Hibernate: update t\_user set born=?, password=?, username=? where id=?

3.TestPersistent02

[IMG_261](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

session = HibernateUtil.openSession();

session.beginTransaction();

User u = new User();

u.setBorn(new Date());

u.setUsername("zhangsan");

u.setPassword("zhangsan");

session.save(u);

u.setPassword("222");

//该条语句没有意义 session.save(u);

u.setPassword("zhangsan111");

//没有意义 session.update(u);

u.setBorn(sdf.parse("1988-12-22"));

//没有意义 session.update(u);

session.getTransaction().commit();

[IMG_262](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

这个时候会发出多少sql语句呢？还是同样的道理，在调用save方法后，u此时已经是持久化对象了，记住一点：**如果一个对象以及是持久化状态了，那么此时对该对象进行各种修改，或者调用多次update、save方法时，hibernate都不会发送sql语句，只有当事物提交的时候，此时hibernate才会拿当前这个对象与之前保存在session中的持久化对象进行比较，如果不相同就发送一条update的sql语句，否则就不会发送update语句**

Hibernate: insert into t\_user (born, password, username) values (?, ?, ?)

Hibernate: update t\_user set born=?, password=?, username=? where id=?

4.TestPersistent03

[IMG_263](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

session = HibernateUtil.openSession();

session.beginTransaction();

User u = new User();

u.setBorn(sdf.parse("1976-2-3"));

u.setUsername("zhangsan2");

u.setPassword("zhangsan2");

session.save(u);

/\*

\* 以下三条语句没有任何意义

\*/

session.save(u);

session.update(u);

session.update(u);

u.setUsername("zhangsan3");

session.getTransaction().commit();

[IMG_264](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

相信这个测试用例，大家应该都知道结果了，没错，此时hibernate也会发出两条sql语句，原理一样的

Hibernate: insert into t\_user (born, password, username) values (?, ?, ?)

Hibernate: update t\_user set born=?, password=?, username=? where id=?

5.TestPersistent04

[IMG_265](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

//此时u是Persistent

User u = (User)session.load(User.class, 4);

//由于u这个对象和session中的对象不一致，所以会发出sql完成更新

u.setUsername("bbb");

session.getTransaction().commit();

[IMG_266](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

我们来看看此时会发出多少sql语句呢？同样记住一点：**当session调用load、get方法时，此时如果数据库中有该对象，则该对象也变成了一个持久化对象，被session所托管**。因此，这个时候如果对对象进行操作，在提交事务时同样会去与session中的持久化对象进行比较，因此这里会发送两条sql语句

Hibernate: select user0\_.id as id0\_0\_, user0\_.born as born0\_0\_, user0\_.password as password0\_0\_, user0\_.username as username0\_0\_ from t\_user user0\_ where user0\_.id=?

Hibernate: update t\_user set born=?, password=?, username=? where id=?

6.TestPersistent05

[IMG_267](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

//此时u是Persistent

User u = (User)session.load(User.class, 4);

u.setUsername("123");

//清空session session.clear();

session.getTransaction().commit();

[IMG_268](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

再看这个例子，当我们load出user对象时，此时user是持久化的对象，在session缓存中存在该对象，此时我们在对user进行修改后，**然后调用session.clear()方法，这个时候就会将session的缓存对象清空，那么session中就没有了user这个对象，这个时候在提交事务的时候，发现已经session中已经没有该对象了，所以就不会进行任何操作**，因此这里只会发送一条select语句

Hibernate: select user0\_.id as id0\_0\_, user0\_.born as born0\_0\_, user0\_.password as password0\_0\_, user0\_.username as username0\_0\_ from t\_user user0\_ where user0\_.id=?

7.TestDetached01

[IMG_269](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

//此时u是一个离线对象，没有被session托管

User u = new User();

u.setId(4);

u.setPassword("hahahaha");

//当执行save的时候总是会添加一条数据，此时id就会根据Hibernate所定义的规则来生成 session.save(u);

session.getTransaction().commit();

[IMG_270](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

我们看到，当调用了u.setId(4)时，此时u是一个离线的对象，因为数据库中存在id=4的这个对象，但是该对象又没有被session所托管，所以这个对象就是离线的对象，要使离线对象变成一个持久化的对象，应该调用什么方法呢？我们知道调用save方法，可以将一个对象变成一个持久化对象，但是，当save一执行的时候，此时hibernate会根据id的生成策略往数据库中再插入一条数据，所以如果调用save方法，此时数据库会发送一条插入的语句：

Hibernate: insert into t\_user (born, password, username) values (?, ?, ?)

所以对于离线对象，如果要使其变成持久化对象的话，我们不能使用save方法，而应该使用update方法

8.TestDetached02

[IMG_271](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

session = HibernateUtil.openSession();

session.beginTransaction();

User u = new User();

u.setId(5);

//完成update之后也会变成持久化状态 session.update(u);

u.setBorn(sdf.parse("1998-12-22"));

u.setPassword("world");

u.setUsername("world");

//会发出一条sql session.update(u);

session.getTransaction().commit();

[IMG_272](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

此时我们看到，当调用了update方法以后，此时u已经变成了一个持久化的对象，那么如果此时对u对象进行修改操作后，在事务提交的时候，则会拿该对象和session中刚保存的持久化对象进行比较，如果不同就发一条sql语句

Hibernate: update t\_user set born=?, password=?, username=? where id=?

9.TestDetached03

[IMG_273](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

session = HibernateUtil.openSession();

session.beginTransaction();

User u = new User();

u.setId(5);

//完成update之后也会变成持久化状态 session.update(u);

u.setBorn(sdf.parse("1998-12-22"));

u.setPassword("lisi");

u.setUsername("lisi");

//会抛出异常

u.setId(333);

session.getTransaction().commit();

[IMG_274](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

我们看这个例子，前面的操作一样，调用update方法后，user变成了一个持久化对象，在对user进行一些修改后，此时又通过 u.setId(333)方法设置了u的ID，那么这个时候，hibernate会报错，因为我们的u当前已经是一个持久化对象，**如果试图修改一个持久化对象的ID的值的话，就会抛出异常**，这点要特别注意

org.hibernate.HibernateException: identifier of an instance of com.xiaoluo.bean.User was altered from 5 to 333

10.TestDetached04

[IMG_275](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

User u = new User();

u.setId(5);

//现在u就是transient对象 session.delete(u);

//此时u已经是瞬时对象，不会被session和数据库所管理

u.setPassword("wangwu");

session.getTransaction().commit();

[IMG_276](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

接着我们来看这个例子，这里在调用了session.delete()方法以后，此时后u就会变成一个瞬时对象，因为此时数据库中已经不存在该对象了，既然u已经是一个瞬时对象了，那么对u再进行各种修改操作的话，hibernate也不会发送任何的修改语句，因此这里只会 有一条 delete的语句发生：

Hibernate: delete from t\_user where id=?

11.TestDetached05

[IMG_277](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

User u = new User();

u.setId(4);

u.setPassword("zhaoliu");

//如果u是离线状态就执行update操作，如果是瞬时状态就执行Save操作

//但是注意：该方法并不常用 session.saveOrUpdate(u);

session.getTransaction().commit();

[IMG_278](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

这里我们来看看 saveOrUpdate这个方法，这个方法其实是一个"偷懒"的方法，如果对象是一个离线对象，那么在执行这个方法后，其实是调用了update方法，如果对象是一个瞬时对象，则会调用save方法，记住：**如果对象设置了ID值，例如u.setId(4)，那么该对象会被假设当作一个离线对象，此时就会执行update操作**。

Hibernate: update t\_user set born=?, password=?, username=? where id=?

如果此时我将u.setId(4)这句话注释掉，那么此时u就是一个瞬时的对象，那么此时就会执行save操作，就会发送一条insert语句

Hibernate: insert into t\_user (born, password, username) values (?, ?, ?)

12.TestDetached06

[IMG_279](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

//u1已经是持久化状态

User u1 = (User)session.load(User.class, 3);

System.out.println(u1.getUsername());

//u2是离线状态

User u2 = new User();

u2.setId(3);

u2.setPassword("123456789");

//此时u2将会变成持久化状态，在session的缓存中就存在了两份同样的对象,在session中不能存在两份拷贝，否则会抛出异常

session.saveOrUpdate(u2);

[IMG_280](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

我们再来看一下这个例子，此时我们的u1已经是持久化的对象了，保存在session缓存中，u2通过调用saveOrUpdate方法后也变成了一个持久化的对象，此时也会保存在session缓存中，这个时候session缓存中就存在了一个持久化对象有两个引用拷贝了，这个时候hibernate就会报错

org.hibernate.NonUniqueObjectException: **a different object with the same identifier value was already associated with the session**: [com.xiaoluo.bean.User#3]

一个session中不能存在对一个持久化对象的双重copy的，要解决这个方法，我们这里又要介绍session的另一个方法  merge方法，这个方法的作用就是解决一个持久化对象两分拷贝的问题，这个方法会将两个对象合并在一起成为一个对象。

[IMG_281](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

　　　　　　　session = HibernateUtil.openSession();

session.beginTransaction();

//u1已经是持久化状态

User u1 = (User)session.load(User.class, 3);

System.out.println(u1.getUsername());

//u2是离线状态

User u2 = new User();

u2.setId(3);

u2.setPassword("123456789");

//此时u2将会变成持久化状态，在session的缓存中就存在了两份同样的对象,在session中不能存在两份拷贝，否则会抛出异常// session.saveOrUpdate(u2);

//merge方法会判断session中是否已经存在同一个对象，如果存在就将两个对象合并 session.merge(u2);

//最佳实践：merge一般不用

session.getTransaction().commit();

[IMG_282](http://www.cnblogs.com/xiaoluo501395377/p/javascript:void(0);)

我们看到通过调用了merge方法以后，此时会将session中的两个持久化对象合并为一个对象，但是merge方法不建议被使用

Hibernate: select user0\_.id as id0\_0\_, user0\_.born as born0\_0\_, user0\_.password as password0\_0\_, user0\_.username as username0\_0\_ from t\_user user0\_ where user0\_.id=?

zhangsan

Hibernate: update t\_user set born=?, password=?, username=? where id=?

终于写完了本篇随笔，本篇随笔可能概念性的内容比较少，基本都是通过测试用例来分析hibernate的三种状态可能会出现的各种情况。

最后总结一下：

**①.对于刚创建的一个对象，如果session中和数据库中都不存在该对象，那么该对象就是瞬时对象(Transient)**

**②.瞬时对象调用save方法，或者离线对象调用update方法可以使该对象变成持久化对象，如果对象是持久化对象时，那么对该对象的任何修改，都会在提交事务时才会与之进行比较，如果不同，则发送一条update语句，否则就不会发送语句**

**③.离线对象就是，数据库存在该对象，但是该对象又没有被session所托管**