 理解Hibernate的三种状态，更利于理解Hibernate的运行机制，这些可以让你在开发中对疑点问题的定位产生关键性的帮助。

三种状态

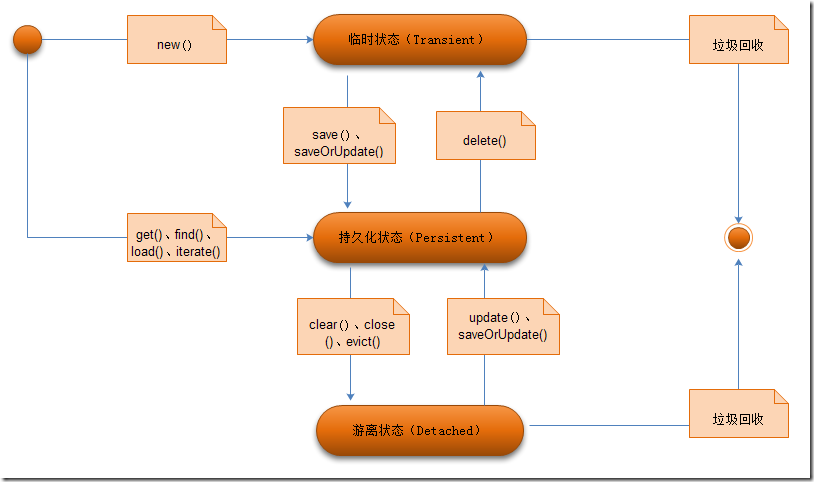
临时状态（Transient）：在通过new关键字，实例化一个对象开始，该对象就进入了临时状态，但它还没有被持久化，没有保存在Session当中。

持久化状态（Persistent）：对象被加入到Session缓存当中，如通过session.save(entity)，Hibernate把实体保存到seesion当中，entity就处在持久化状态中。

游离状态（Detached）：对象脱离了session缓存，如通过session清理，将对象保存到数据库中，原来在session中的对象仍然与内存中，该对象就处于游离状态。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 临时状态  （Transient） | 持久化状态  （Persistent） | 游离状态  （Detached） |
| 是否存于Session缓存中 | × | √ | × |
| 数据库中是否有对应记录 | × | √ | √ |

对象生命周期



下面以一个对象从生成到保存到数据库的流程做一下解析：

    当一个对象被实例化出来以后，该对象是临时状态，当调用方法session.save(entity)，后该对象被加入到session缓存中，进入持久化状态，这时数据库中还不存在相关的记录，当session提交数据库事务时，这里隐含做了两件事，一件事是隐式调用session.flush()，其作用先是清理缓存（相当于调用了session.clear()），再生成一条对应的insert语句，但该语句还没有提交，第二件事是对刚才生成的语句进行提交，从而在数据库中生成了对应的记录。至此原entity对象就在数据库中生成了一条对应的记录，而它本身也脱离了session缓存，处于游离状态，该对象经过垃圾回收机制处理被回收。一个hibernate的保存对象过程就此结束。

实例代码如下：

1: Session session = null;

2: try {

3: session = HibernateUtil.openSession();

4: //开始事务

5: session.beginTransaction();

6: //member对象进入临时状态

7: Member member = new Member();

8: member.setMobile("18208195215");

9: member.setUsername("liping");

10: //member对象进入持久化状态

11: session.save(member);

12: //提交事务，隐式包含了session.flush()的动作

13: session.getTransaction().commit();

14: //提交完成后，member处于游离状态

15: } catch (HibernateException e) {

16: e.printStackTrace();

17: if (session != null)

18: session.getTransaction().rollback();

19: } finally {

20: if (session != null)

21: session.close();

22: }