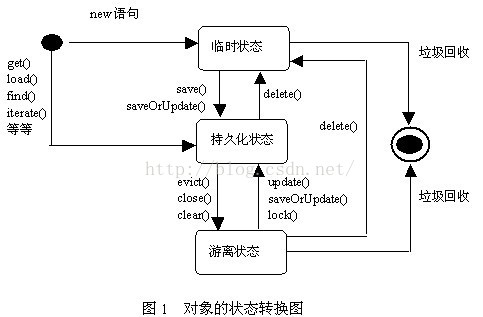
Hibernate持久化对象

hibernate采用低侵入式的设计，它对持久化类几乎不做任何要求。也就是说hibernate操作的持久化类基本都是普通Java对象。对于持久化类的要求这里不做说明。只就持久化对象的状态和各个状态之间的转换。 Hibernate持久化对象有如下几种状态：

1. **瞬态**：对象有new操作符创建，但是并没与Hibernate Session关联。处于瞬态的对象是不会被持久化到数据库中的。如果程序中失去了瞬态对象的引用，瞬态对象就会被垃圾回收机制销毁。
2. **持久化**：持久化实力在数据库中有对应的记录，并且拥有一个持久化标识。对于持久化对象，它必须要与指定的Hibernate Session关联起来。
3. **托管**：该对象曾经处于持久化装填，但随着与之关联的Session被关闭了，那么该对象也就变为了托管状态。

下图是hibernate持久化实例的状态演化图：



## 瞬态转变为持久化状态

当我们通过new新建一个实例时，这个实例就处于瞬态。瞬态可以通过以下几个方法转换为持久化状态。

|  |
| --- |
| save(Object obj)：将对象变为持久化状态，该对象的属性将被保存到数据库中。  persist(Object obj)：将对象变为持久化状态，该对象的属性将被保存到数据库中  save(Object obj,Object pk)：将对象保存到数据库，保存到数据库时，指定主键值  persist(Object obj,Object pk)：将对象保存到数据库，保存到数据库时，指定主键值 |

如果对象的标识属性是generated类型的，那么hibernate将会在执行save()方法时自动生成标识属性值，并且将该标识属性值分配给该对象，并且标识属性值会在sava()被调用时自动产生并分配给该对象。

如果对象的标识属性是assigned类型的，或者是复合主键，那么该标识属性值应该在调用save()方法之前手动赋予给该对象。

在使用save()和persist()方法的时候，有一个区别：使用save()方法保存持久化对象时，该方法返回持久化对象的标识属性值。但是persist不会返回任何值。

## 加载持久化实例

我们可以使用load()方法来加载一个持久化实例，这种加载时根据持久化类的标识属性值加载持久化实例的，其实质就是根据主键从数据表中加载一条新的纪录。News new = seesion.load(News.class,new integer(PK));

同时也可以使用get()方法加载一个持久化实例。它和load方法的相同点在于两者都是根据主键装载持久化实例的。不同就在于get()会立即访问数据库，而laod()会延迟加载，不会立即访问数据库。

一旦加载了该持久化实例后，该实例就会处于持久化状态，这是如果对该持久化实例所做的修改将会保持到数据库中。如：new.setTitle("11111111");

这段代码会在session.flush之前自动保持到数据库中。也就是说，修改对象最简单的方法就是在Session处于打开状态时加载它，然后只见修改即可。

|  |
| --- |
| New n = seesion.load(News.class,new Integer(PK));  n.setTitlte("新标题");  Session.flush(); |

## 托管状态

当一个对象处于脱管的状态后，程序应该使用新的session来保存这些修改。hibernate提供了update()、merge()和updateOrSace()等方法来保存这些修改。

|  |
| --- |
| News news = firstSession.load(News.class,new Integer(pk));  //第一个session关闭  firstSession.close();  //修改脱管状态下的持久化对象  n.setTitle("新标题");  //打开第二个session  Session secondeSession = sf.openSession();  //保存脱管对象所做的修改  secondeSession.update(n); |

当我们用另一个session来保存这种修改后，该脱管对象会再次回到持久化状态。当需要使用update来保存程序对持久化对象所在的修改时，如果不清楚该对象是否曾经持久化过，那么可以选择updateOrSave()方法，该方法会自动判断该对象是否曾经持久化过，如果持久化过，则使用update()来操作，否是使用save()方法。