Hibernate中Se ion的缓存及对象的状态 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022_Hibernate_E4_c104_252522.htm 对于session这个接口的学习可以说是最痛苦也是最复杂的，因为它所涉及的方面太多了，一些隐藏的机制也很多，谁让它是Central API呢。 对于它的几个最基本的方法如save()、0delete()、flush()等的学习都花了我一定的时间。在深入了解这些这些方法前，了解session的缓存机制以及Hibernate中Java对象的状态对我们是很有帮助的。 一．Session的缓存 Java是纯面向对象的语言，因此不可能像C语言那样直接操纵内存，例如声明一段可用的内存空间。在Java里面，缓存通常是指Java对象的属性占用的内存空间，通常是一些集合类型的属性。在session接口的实现类SessionImpl中定义了一系列的Java集合，这些Java集合就构成了Session的缓存。 使用缓存的一个很明显的好处就是可以减少数据库访问的频率，提高应用程序的性能，因为从内存中读取数据显然要比从数据库中查询快多了。根据我个人的理解，Session的缓存实际上起到了一个“过渡仓库”作用。就像魔兽中的英雄一样，身上都会背有一个包，用来存放常用的物品如补血药水、补魔药水、回城卷等等。如果想用回城卷而身上没有回程卷的话就要跑到商店去shopping了，这样就会浪费大量的时间了，除非你此刻就在商店旁边；如果想用的回城卷的时候身上就有的话，英雄就可以直接用而不必大老远的跑到商店去了。我们的Session的缓存可以说就相当于英雄身上的背包，我的应用程序就是英雄，而数据库就是商店咯，如下图所示。 当然这个比喻不是很准确了，比方说在Hibernate应用中

我们可以向数据库插入一条新的记录，而在魔兽中你是不可能给商店增加存货量的，只是为了便于理解，才作了这么一个对比。 二．Hibernate中Java对象的状态 在一个Hibernate应用中，Java对象可以处于以下三个状态之一： 1．临时状态(Transient)。处于这个状态的对象还被没有纳入Hibernate的缓存管理体系，跟任何session都不关联，在数据库中也没有对应的记录。 2．持久化状态(Persistent)。处于这个状态的对象位于Session的缓存中，并且和数据库中的一条数据记录相对应。 3．游离状态(Detached)。处于这个状态的对象不再位于Session的缓存中，它与临时对象的最大区别在于，游离对象在数据库中还可能存在一条与它对应的记录。 上述3个状态之间是可以相互转化的，而且我们所说的状态都是针对某一个session实例而言的，比方说，对象A对于session1而言是处于持久化状态的，因为它处于session1的缓存中，但是对于session2而言对象A并不在它的缓存中，因此它是处于游离状态的。 对于这几个状态的理解花费了我一定的时间，因为总是有一些稀奇古怪的念头在我脑海中产生。比如说，对于临时状态的定义，如果我新建一个对象，然后人为的让它属性的值和数据库中的一条记录对应，包括id的取值都一样。此时它能否说是处于游离状态呢？因为它和一条记录想对应呀。实际上这些情况都是由于一些不和规范的操作而产生的。在Hibernate应用中，无论Java对象处于临时状态、持久化状态还是游离状态，应用程序都不应该修改它的OID。OID的值应该由Hibernate来维护和负责，实际上Hibernate在同步缓存中的对象与数据库中的记录时，都是通过OID来进行关联和映射的，如果应用程序人为的修改了对象的OID，就会导

致一些莫名其妙的错误，而且这样也不利于数据的同步。