一，hibernate一级缓存的管理

如果持久态对象不再一级缓存中，可以更新数据库吗？

不能。

把对象移除一级缓存的方法：

session.evict(obkect)：把一个对象移除一级缓存

session.clear():把一级缓存的所有对象移除

```
//演示一级缓存的管理
                Session session = hibernateUtils.getSession();
                Transaction ts = session.beginTransaction();
                //p1是一个持久态对象，存储在一级缓存中
                person p1 = session.get(person.class, 6);
                p1.setName("老王");
                //把p1对象移除一级缓存
                session.evict(p1);
                ts.commit();
                session.close();
```

二，hibernate的延迟加载策略

延迟加载时为了减少程序和数据库访问的次数，提高数据库执行的性能。

延迟加载的执行机制：

1）在查询一个对象的时候，不会查询对象的属性或者其关联的数据。

2）在需要使用到对象的属性或关联数据的时候才会去查询数据库

结论：按需加载

1、类（属性）级别延迟加载

load（）：只有load方法才支持类级别的延迟加载。

get（）：方法不支持类级别的延迟加载。

具体事例代码：

```
//演示类级别的延迟加载
```

```
Session session = hibernateUtils.getSession();
Transaction ts = session.beginTransaction();
//p1是一个持久态对象，存储在一级缓存中
//load（）只有load方法才支持类级别的延迟加载
person p1 = session.load(person.class, 6);
System.out.println(p1.getName());
ts.commit();
session.close();
```

如果不想要操作是延迟加载的话，需要在实体对应的*.hbm.xml文件中的class标签中将lazy属性的值设置为false，因为其默认情况下是lazy=“true”，即支持延迟加载的，将其更改为lazy=“false”时，就表示取消延迟加载。

2、关联级别延迟加载

以一对多为例

一方：<set></se

修改一对多的延迟加载配置：

<set name="多端在一端实体中的名字" inverse=“true” lazy=“false></set>

多方：<many-to-one/>

修改多对一的延迟加载配置：

<many-to-one name="一端在多端实体中的属性名" inverse=“true” lazy=“false”/>

注意：在实际开发过程中，我们并不需要对延迟加载的方式进行修改，一般就直接保持默认配置。