单向多对一或一对多关联关系的保存和更新方法

1、save()方法同时保存主从表

需求：

* 保存操作

* 要求：

* 创建一个客户，创建一个联系人

* 建立联系人和客户的单向多对一关联关系。

* 先保存联系人，再保存客户

* 问题：

* 当我们执行保存操作时，会产生3条sql语句。其中两条insert和1条update。

* update语句是多余的。

* 解决办法：

* 保存原则：先保存主表，再保存从表

```
public void test1(){
    //1.建立客户实体
    Customer c = new Customer();
    c.setCustName("中粮商务公园");
    c.setCustLevel("普通客户");

    //建立联系人实体
    LinkMan l = new LinkMan();
    l.setLkmName("老马");
    l.setLkmGender("male");
    l.setLkmPhone("010-34384983");

    //2.建立他们的单向多对一关联关系
    l.setCustomer(c);

    //获得Session并开启事务
    Session s = HibernateUtil.getCurrentSession();
    Transaction tx = s.beginTransaction();
    //3.保存：先保存客户，再保存联系人
    s.save(c);
    s.save(l);
    //提交事务
    tx.commit();
}
```

2、update()方法更新从表或主表

需求：（创建主表，查从表，更新从表，配从表级联操作）

（创建从表，查主表，更新主表，配主表级联操作）

* 更新从表信息

* 要求：

* 查询id为1的从表。

* 创建一个新的客户。

* 给id为1的联系人分配创建的这个客户

* 更新联系人。

* 问题：

* 当我们更新一个持久态对象，该对象中关联了一个临时态对象时，更新不成功，会报错

* object references an unsaved transient instance - save the transient instance before flushing: com.itheima.domain.Customer

* 解决办法：

* 配置级联保存更新：级联操作

* 要操作谁，就找谁的配置文件。要级联操作谁，就找谁的配置。

* 配置属性：　cascade　　配置的值：　save-update

* <many-to-one name="customer" class="Customer" column="lkm_cust_id" **cascade="save-update"**></many-to-one>

```java
public void test2(){
    //1.建立主表
    Customer c = new Customer();//临时态
    c.setCustName("华强北");
    c.setCustLevel("普通客户");
    c.setCustSource("电话");
    c.setCustIndustry("商务办公");
    c.setCustAddress("宝安区留仙二路");
    c.setCustPhone("0755-32983248");

    Session s = HibernateUtil.getCurrentSession();
    Transaction tx = s.beginTransaction();
    //2.查询id为1的从表
    LinkMan l1 = s.get(LinkMan.class,1L);//持久态

    //3.建立他们的单向关联关系
    l1.setCustomer(c);// 联系人是持久态，客户是临时态

    //4.更新从表
    s.update(l1);//先保存临时态客户，再更新联系人。

    tx.commit();
}
```

双向多对一或一对多关联关系的保存和更新方法

1、save()方法保存主从表信息

需求：

* 保存操作

* 要求：

* 创建一个主表，创建一个从表

* 建立联系人和客户的双向一对多或多对一关系。

* 先保存客户，再保存联系人

* 问题：

* 我们先保存客户，再保存联系人，在双向关联关系时，仍然会出现2条Insert和1条update语句。

* 其中update语句是多余的。

* 解决办法：

* 让 主表方 放弃维护关联关系的权利。

* 使用配置文件配置：

* 配置的属性：**inverse**，是否放弃维护的权利

* 配置的值：**true**放弃，**false**不放弃（默认值）

* 注意：

* **inverse**属性只应该出现在**set**标签上。

```
public void saves(){
    //1.建立两个实体
            Customer c = new Customer();
            c.setCustName("双向CCCC");
            c.setCustLevel("普通客户");

            Linkman l = new Linkman();
            l.setLkmName("CCCC双向C");
            l.setLkmGender("male");

            //2.建立他们的双向一对多关联关系
            l.setCustomer(c);
            c.getLinkmans().add(l);

            Session s = HibernateUtil.getCurrentSession();
            Transaction tx = s.beginTransaction();
            //3.保存：先保存客户，再保存联系人
            s.save(c);
            s.save(l);

            tx.commit();
}
```

非常重要

先主后从

Customer.hbm.xml

```
        key:作用：用于映射外键。
            属性：column：指定外键的名称。
        one-to-many:作用：指定当前实体和集合元素之间的关系是一对多。
            属性：class：指定集合元素对应的实体类
        inverse="true"   主动放弃 对关联关系的维护权利
            -->
        <set name="linkmans"
             table="cst_linkman" inverse="true">
            <key column="lkm_cust_id"></key>
            <one-to-many class="Linkman" />
        </set>
    </class>
</hibernate-mapping>
```

2、update()方法更新从表或主表

需求：

双向多对一或一对多更新 inverse="true" 不能有这个配置（不维护会导致更新失败）

1、建立从表 更新主表 s.update(c) 建立从表 查主表 更新主表 并设置 主表的级联保存更新

2、建立主表 更新从表 s.update(lk); 建立主表 查从表 更新从表 并设置 从表的级联保存更新

3、查出主表同时查出从表 更新：更新主表或者从表都可以

* 更新操作

* 要求：

* 创建一个联系人，查询id为1的客户

* 建立客户和联系人的双向一对多关联关系

* 更新客户

* 问题：当我们更新用一个实体它关联了一个临时态对象，会报错。

* 解决办法：配置级联保存更新

* 配置位置：要操作谁找谁的配置文件，要级联操作谁找谁的配置。

* 配置的属性：cascade

* 配置的值：save-update

```
* <set name="linkmans" table="cst_linkman" cascade="save-update" >
  <key column="lkm_cust_id"></key>
  <one-to-many class="LinkMan"/>
 </set>
*/
public void test2(){
    //1.创建一个从表
    LinkMan l = new LinkMan();//临时态
    l.setLkmName("联系人1");
    l.setLkmGender("male");
    l.setLkmPhone("010-34384983");
    l.setLkmMobile("13811111111");
    l.setLkmPosition("teacher");
    l.setLkmEmail("laoma@itcast.com");
    l.setLkmMemo("给力");

    Session s = HibernateUtil.getCurrentSession();
    Transaction tx = s.beginTransaction();
    //2.查询id为1的主表
    Customer c1 = s.get(Customer.class, 1);//持久态
    //3.建立双向关联关系
    l.setCustomer(c1);
    c1.getLinkmans().add(l);
    //4.更新主表
    s.update(c1);
    tx.commit();
}
```

```java
public void saTest2(){
    Session s = HibernateUtil.getCurrentSession();
    Transaction t = s.beginTransaction();
    //1、查出联系人
    Linkman lk = s.get(Linkman.class, 2L);
    //2、查出客户为5的
    Customer customer = s.get(Customer.class, 7);
    //建立双向关联关系
    lk.setCustomer(customer);
    customer.getLinkmans().add(lk);
    //更新
    s.update(customer);
    //2、联系人更新到客户里去
    t.commit();
}
```

查询更新处理

查询和删除就直接操作

Session s = HibernateUtil.getCurrentSession();

s.get(xxxx)　查询方法

s.delete(xxxx)　删除方法

s.update(xxxx)　更新（改的方法）

s.save(xxxx)　增加（插入方法）

双向一对多删除操作

* 问题：

* 1、删除联系人能不能删除？

* 可以。删除从表不用管其他的。

* 2、删除客户能不能删除？

* 看有没有联系人引用该客户：

* 有引用：

* 看是否配置了inverse属性：

* 配置了inverse属性=true：

* 默认情况下不能删除，因为主表不维护和从表的关联关系。

* 就想删除：

```
 * 先删除从表数据，再删除主表数据
 * 级联删除。
 * 配置的方式：cascade属性
 * 配置的值：delete
 * 配置了inverse属性=false：
 * 可以删除，会把外键置为null。当数据库中表的外键字段有非空约束时，不能删除了。
 * 没有引用： 可以删除
 * 实际开发中的：
 * 级联删除慎用。
 * 删除主表，就删除主表，删除不成功，就直接报错。
 */
public void test3(){
 Session s = HibernateUtil.getCurrentSession();
 Transaction tx = s.beginTransaction();
 //1.查询id为1的客户
 Customer c1 = s.get(Customer.class, 2);//持久态
 //2删除客户
 s.delete(c1);
 tx.commit();

}
```