1 Hibernate 实体关联关系映射----总结

收集整理：Volunteer 2009 年 11 月 16 日 星期一

1.1 引言

花了三天的业余时间，终于写完了 Hibernate 关联关系映射的所有实例，感觉还应该总结一下。

Hibernate 映射关系错综复杂，在实际中真的都能用到吗？不用行吗？

在我看来，Hibernate 提供这些映射关系，常用就是一对一和多对一，并且在能不用连接表的时候尽量不要用连接表。多对多会用到，如果用到了，应该首先考虑底层数据库设计是否合理。

在实际开发中，在 Hibernate 关联关系之外常常还有另外一种选择方案，表各自作为单表映射，业务逻辑控制外键关系（有时候就是一个相关联的列，但不一定要加外键约束），这样更加灵活，并且数据的完整性同样有保证。

当然，“单表映射，业务控制外键关系”并不是说 Hibernate 的实体关联功能是多余的，Hibernate 的实体关联的优点很多，随便拿本书都是讲优点，用好了会让开发人员感觉更方便，现在我也是两种方案结合使用。比如对于不很确定的两个实体，常常选用单表关联。

以前在初学 Hibernate 还没有完全搞清楚这些关联关系的时候，就是用单表映射，业务控制外键关系做的，发现没有任何问题，程序同样运行得很好。

看了这些是不是后悔浪费时间学习映射关系了？呵呵，Hibernate 的 OR Mapping 是 Hibernate 的灵魂，我相信 Hibernate 的创始人比我们一般人的理解更深刻。只有学会了这些才能体会 Hibernate 设计者的思想。学一个东西，不光自己写代码，还应该能看懂别人的代码才行。因此系统学习这些关联映射还是大有必要的。

以上都是我自己的观点。欢迎在此交流讨论。

Hibernate 在实际项目开发中，hbm.xml 包括数据库脚本都是通过 Xdoclet 生成的，在此不采用 Xdoclet 的目的是为了便于理解这些映射模型。实体-数据表-映射文件 三者对比看，太直观了。

瞌睡了，暂时先写到此，有新思路了再补上。。。。

2 Hibernate 关联关系映射实例速查

Hibernate 的映射关系很多，也比较复杂，也很容易忘记。这个基本上占据了 Hibernate 学习的七成时间。熟悉这些映射模型，需要大量的实践才能掌握。下面是我对 Hibernate 关联关系映射的一个总结，放到 blog 上一是自己查看方便，二来也可以和更多 Hibernate 开发人员交流分享。希望各位多多留言哦：）。

本文主要参考夏昕翻译的"Hibernate 参考文档 V3.12"，也在附件中给出了。

1. **本文的模块较多，映射关系部分分为一下模块：**

```
Hibernate 关联关系映射目录
 │
 ├──单向关联
 │    ├──  一对一外键单向关联
 │    ├──  一对一主键单向关联
 │    ├──  一对一连接表单向关联
 │    ├──  一对多外键单向关联
 │    ├──  一对多连接表单向关联
 │    ├──  多对一外键单向关联
 │    ├──  多对一连接表单向关联
 │    └──  多对多单向关联
 └──双向关联
      ├──  一对一外键双向关联
      ├──  一对一主键双向关联
      ├──  一对一连接表双向关联
      ├──  一对多外键双向关联
      ├──  一对多连接表双向关联
      └──  多对多双向关联
```

2. **本系列实例的开发环境：**

- ✓ Windows XP Professional 简体中文版
- ✓ MySQL 5.0.45
- ✓ Hibernate 3.12
- ✓ Java SDK 1.5.06
- ✓ IntelliJ IDEA 5.12

3. **系列实例中所用的 Hibernate 配置文件如下：**

```
<?xml version='1.0' encoding='gb2312'?>
    <!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "[url]http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd[/url]">
<hibernate-configuration>
    <session-factory>
      <!--指定连接数据库驱动-->
       <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
       <!--指定连接数据库的 url，hibernate 连接的数据库名-->
      <property name="connection.url">jdbc:mysql://localhost:3306/hbstudy</propert
    y>
         <!--指定连接数据库的用户名-->
         <property name="connection.username">root</property>
         <!--指定连接数据库的用户密码-->
         <property name="connection.password">leizhimin</property>
          <!--指定连接池的大小-->
        <property name="connection.pool_size">5</property>
        <!--指定数据库的方言-->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <!--根据需要自动创建数据库,测试环境用-->
        <property name="hbm2ddl.auto">create</property>
        <!--在控制台显示执行的 SQL 语句-->
        <property name="show_sql">true</property>
        <!-- Enable Hibernate's automatic session context management -->
        <property name="current_session_context_class">thread</property>
       <!--映射文件列表-->
        <!--单向关联-->
        <mapping resource="com/lavasoft/dx/_n_1_fk/Addressn1fk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_n_1_fk/Personn1fk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_n_1_tab/Addressn1tab.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_n_1_tab/Personn1tab.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_1_fk/Address11fk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_1_fk/Person11fk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_1_tab/Address11tab.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_1_tab/Person11tab.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_1_pk/Address11pk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_1_pk/Person11pk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_n_fk/Address1nfk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_n_fk/Person1nfk.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_n_tab/Address1ntab.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_1_n_tab/Person1ntab.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_n_n/Addressnn.hbm.xml"/>
        <mapping resource="com/lavasoft/dx/_n_n/Personnn.hbm.xml"/>
       <!--双向关联-->
        <mapping resource="com/lavasoft/sx/_1_n_fk/Address1nfk_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_n_fk/Person1nfk_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_n_tab/Address1ntab_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_n_tab/Person1ntab_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_n_n/Addressnn_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_n_n/Personnn_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_1_fk/Address11fk_sx.hbm.xml"/>
```

```
        <mapping resource="com/lavasoft/sx/_1_1_fk/Person11fk_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_1_pk/Address11pk_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_1_pk/Person11pk_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_1_tab/Address11tab_sx.hbm.xml"/>
        <mapping resource="com/lavasoft/sx/_1_1_tab/Person11tab_sx.hbm.xml"/>
    </session-factory>
    </hibernate-configuration>
```

4. 系列实例中所用到 **Session** 工厂是：

```
public class HibernateUtil {
    private static final SessionFactory sessionFactory;
        static {
          try {
            // Create the SessionFactory from hibernate.cfg.xml
            sessionFactory = new Configuration().configure().buildSessionFactory();
          } catch (Throwable ex) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("初始化 SessionFactory 失败！" + ex);
            throw new ExceptionInInitializerError(ex);
          }
    }
   public static final ThreadLocal session = new ThreadLocal();
   public static Session getCurrentSession() throws HibernateException {
          Session s = (Session) session.get();
          //当原 Session 为空或已关闭时，打开一个新的 Session
          if (s == null || !s.isOpen()) {
            s = sessionFactory.openSession();
            session.set(s);
          }
          return s;
       }
   public static void closeSession() throws HibernateException {
          Session s = (Session) session.get();
          session.set(null);
          if (s != null) {
            s.close();
          }
       }
    }
```

2.1 单向关联

2.1.1 Hibernate 一对一外键单向关联（见_1_1_fk.zip）

事实上，单向 1-1 与 N-1 的实质是相同的，1-1 是 N-1 的特例，单向 1-1 与 N-1 的映射配置也非常相似。只需要将原来的 many-to-one 元素增加 unique="true"属性，用于表示 N 的一端也必须是唯一的，在 N 的一端增加了唯一的约束，即成为单向 1-1。基于外键的单向 1-1 的配置将与无连接表 N-1 关联的 many-to-one 增加 unique="true"属性即可。

一、模型介绍

一个人（Person）对应一个地址（Address）。

二、实体（省略 **getter**、**setter** 方法）

```
public class Person11fk {
    private int personid;
    private String name;
    private int age;
    private Address11fk address11fk;
```

```
public class Address11fk {
    private int addressid;
    private String addressdetail;
```

三、表模型

mysql> desc address_11fk;

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

mysql> desc person_11fk;

```
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| personid  | int(11)      | NO   | PRI | NULL    | auto_increment |
| name      | varchar(255) | YES  |     | NULL    |                |
| age       | int(11)      | YES  |     | NULL    |                |
| addressId | int(11)      | YES  | UNI | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
```

四、生成的 **SQL** 脚本

```
CREATE TABLE `address_11fk` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

```
CREATE TABLE `person_11fk` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  `addressId` int(11) default NULL,
  PRIMARY KEY  (`personid`),
  KEY `FK68A8818F3F45AA77` (`addressId`),
  CONSTRAINT `FK68A8818F3F45AA77` FOREIGN KEY (`addressId`) REFEREN
CES `address_11fk` (`addressid`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

五、映射方法：

```
    在 Person 中添加 Address 属性，映射配置为：
<!--用来映射关联 PO column 是 Address 在该表中的外键列名,增加 unique 变成"1-1"-->
    <many-to-one name="address11fk" column="addressId" unique="true"/>

<hibernate-mapping>
 <class name="com.lavasoft.dx._1_1_fk.Address11fk" table="ADDRESS_11fk">
      <id name="addressid"> <generator class="identity"/> </id>
      <property name="addressdetail"/>
  </class>
</hibernate-mapping>

<hibernate-mapping>
    <class name="com.lavasoft.dx._1_1_fk.Person11fk" table="PERSON_11fk">
        <id name="personid"> <generator class="identity"/> </id>
        <property name="name"/>
        <property name="age"/>
<!--用来映射关联 PO column 是 Address 在该表中的外键列名,增加 unique 变成"1-1"-->
      <many-to-one name="address11fk" column="addressId" unique="true"/>
    </class>
</hibernate-mapping>
```

六、测试方法：

```
public class Test_11fk {
    public static void main(String[] args){
        Person11fk p1=new Person11fk();
        p1.setAge(21);
        p1.setName("p1");
        Address11fk add1=new Address11fk();
        add1.setAddressdetail("郑州市经三路");

        p1.setAddress11fk(add1);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add1);
        session.save(p1);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果:

1) :正常保存. 推荐这么干!

```
        session.save(add1);
        session.save(p1);
```

```
Hibernate: insert into ADDRESS_11fk (addressdetail) values (?)
Hibernate: insert into PERSON_11fk (name, age, addressId) values (?, ?, ?)
```

2) :正常保存.

```
        session.save(p1);
        session.save(add1);
```

```
Hibernate: insert into PERSON_11fk (name, age, addressId) values (?, ?, ?)
Hibernate: insert into ADDRESS_11fk (addressdetail) values (?)
Hibernate: update PERSON_11fk set name=?, age=?, addressId=? where pers
onid=?
```

3) :正常保存.

```
//      session.save(p1);
        session.save(add1);
```

```
Hibernate: insert into ADDRESS_11fk (addressdetail) values (?)
```

4) : 发生异常,不能保存.

```
        session.save(p1);
//      session.save(add1);
```

```
Hibernate: insert into PERSON_11fk (name, age, addressId) values (?, ?, ?)
Exception in thread "main" org.hibernate.TransientObjectException: com.lavasof
t.dx._1_1_fk.Address11fk
```

2.1.2 Hibernate 一对一主键单向关联（见_1_1_pk.zip）

1-1 的关联可以基于主键关联，但基于主键关联的持久化类不能拥有自己的主键生成策略，它的主键由关联类负责生成。另外，另外，增加 one-to-one 元素来关联属性，必须为 one-to-one 元素增加 constrained="true"属性，表明该类主键由关联类生成。

一、模型介绍

一个人（Person）对应一个地址（Address）。

二、实体（省略 getter、setter 方法）

```
public class Person11pk {
    private int personid;
    private String name;
    private int age;
    private Address11pk address11pk;
```

```
public class Address11pk {
    private int addressid;
    private String addressdetail;
```

三、表模型

mysql> desc address_11pk;

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

mysql> desc person_11pk;

```
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| personid | int(11)      | NO   | PRI |         |       |
| name     | varchar(255) | YES  |     | NULL    |       |
| age      | int(11)      | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

四、生成的 SQL 脚本

```
/* Formatted on 2007/08/22 14:40 (QP5 v5.50) */
CREATE TABLE `address_11pk` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/22 14:41 (QP5 v5.50) */
CREATE TABLE `person_11pk` (
  `presonid` int(11) NOT NULL,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`presonid`),
  KEY `FK68A882C591BB393E` (`presonid`),
  CONSTRAINT `FK68A882C591BB393E` FOREIGN KEY (`presonid`) REFERENCES `address_11pk`
(`addressid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

五、映射方法：在 Person 中配置 id 生成策略为：

```
      <id name="personid">
            <!--基于主键关联时，主键生成策略是 foreign，表明根据关联类生成主键-->
            <generator class="foreign">
                <!--关联持久化类的属性名-->
                <param name="property">address11pk</param>
            </generator>
        </id>
        ......
        <!--用于映射 1-1 关联-->
      <one-to-one name="address11pk" constrained="true"/>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_1_pk.Person11pk" table="PERSON_11pk">
        <id name="personid" column="presonid">
            <!--基于主键关联时，主键生成策略是 foreign，表明根据关联类生成主键-->
            <generator class="foreign">
                <!--关联持久化类的属性名-->
                <param name="property">address11pk</param>
            </generator>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--用于映射 1-1 关联-->
        <one-to-one name="address11pk" constrained="true"/>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_1_pk.Address11pk" table="ADDRESS_11pk">
        <id name="addressid">
```

```
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
    </class>
</hibernate-mapping>
```

六、测试方法：

```
public class Test_11pk {
    public static void main(String[] args){
        Person11pk p1=new Person11pk();
        p1.setAge(21);
        p1.setName("p1");

        Address11pk add1=new Address11pk();
        add1.setAddressdetail("郑州市经三路");

        p1.setAddress11pk(add1);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add1);
        session.save(p1);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存. 推荐这么干!

```
        session.save(add1);
        session.save(p1);
```

```
Hibernate: insert into ADDRESS_11fk (addressdetail) values (?)
Hibernate: insert into PERSON_11fk (name, age, addressId) values (?, ?, ?)
```

2）:正常保存.

```
        session.save(p1);
        session.save(add1);
```

```
Hibernate: insert into PERSON_11fk (name, age, addressId) values (?, ?, ?)
Hibernate: insert into ADDRESS_11fk (addressdetail) values (?)
Hibernate: update PERSON_11fk set name=?, age=?, addressId=? where personid=?
```

3）:正常保存.

```
//      session.save(p1);
        session.save(add1);
```

```
Hibernate: insert into ADDRESS_11fk (addressdetail) values (?)
```

4) : 发生异常,不能保存.

```
        session.save(p1);
//      session.save(add1);
```

```
Hibernate: insert into PERSON_11fk (name, age, addressId) values (?, ?, ?)
Exception in thread "main" org.hibernate.TransientObjectException:
com.lavasoft.dx._1_1_fk.Address11fk
```

2.1.3 Hibernate 一对一连接表单向关联（见_1_1_tab.zip）

这种情况很少见，但 Hibernate 同样允许采用连接表关联 1-1.有连接表的 1-1 同样只需要将 N-1 的 many-to-one 元素增加 unique="true"属性即可。

一、模型介绍

一个人（Person）对应一个地址（Address)。

二、实体（省略 getter、setter 方法）

```
public class Person11tab {
    private int personid;
    private String name;
    private int age;
    private Address11tab address11tab;
```

```
public class Address11tab {
    private int addressid;
    private String addressdetail;
```

三、表模型

```
mysql> desc address_11tab;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

```
mysql> desc join_11tab;
+--------------+---------+------+-----+---------+-------+
| Field        | Type    | Null | Key | Default | Extra |
+--------------+---------+------+-----+---------+-------+
| personid     | int(11) | NO   | PRI |         |       |
| address11tab | int(11) | YES  | UNI | NULL    |       |
+--------------+---------+------+-----+---------+-------+
```

```
mysql> desc person_11tab;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
/* Formatted on 2007/08/20 16:52 (QP5 v5.50) */
CREATE TABLE `join_11tab` (
  `personid` int(11) NOT NULL,
  `address11tab` int(11) default NULL,
  PRIMARY KEY  (`personid`),
  UNIQUE KEY `address11tab` (`address11tab`),
  KEY `FK6B44BE20C4CC3D33` (`address11tab`),
  KEY `FK6B44BE209049BB1F` (`personid`),
  CONSTRAINT `FK6B44BE209049BB1F` FOREIGN KEY (`personid`) REFERENCES `person_11tab` (`personid`),
  CONSTRAINT `FK6B44BE20C4CC3D33` FOREIGN KEY (`address11tab`) REFERENCES `address_11tab` (`addressid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/20 16:53 (QP5 v5.50) */
CREATE TABLE `address_11tab` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/20 16:53 (QP5 v5.50) */
CREATE TABLE `person_11tab` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`personid`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

五、映射方法：在 Person 中添加 Address 属性，映射配置为：

```
        <!--使用 join 元素显式确定链接表-->
        <join table="join_11tab">
            <key column="personid"/>
            <!--映射 1-1 关联属性，其中 unique= “true” 属性确定为 “1-1” -->
            <many-to-one name="address11tab" unique="true"/>
        </join>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_1_tab.Person11tab" table="PERSON_11tab">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--使用 join 元素显式确定链接表-->
        <join table="join_11tab">
            <key column="personid"/>
            <!--映射 1-1 关联属性，其中 unique= “true” 属性确定为 “1-1” -->
            <many-to-one name="address11tab" unique="true"/>
        </join>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_1_tab.Address11tab" table="ADDRESS_11tab">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_11tab {
    public static void main(String[] args){
        Person11tab p1=new Person11tab();

        p1.setAge(21);
        p1.setName("p1");

        Address11tab add1=new Address11tab();
        add1.setAddressdetail("郑州市经三路");

        p1.setAddress11tab(add1);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add1);
        session.save(p1);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存. 推荐这么干!

```
        session.save(add1);
        session.save(p1);
```

```
Hibernate: insert into ADDRESS_11tab (addressdetail) values (?)
Hibernate: insert into PERSON_11tab (name, age) values (?, ?)
Hibernate: insert into join_11tab (address11tab, personid) values (?, ?)
```

2）:正常保存.

```
        session.save(p1);
        session.save(add1);
```

```
Hibernate: insert into PERSON_11tab (name, age) values (?, ?)
Hibernate: insert into join_11tab (address11tab, personid) values (?, ?)
Hibernate: insert into ADDRESS_11tab (addressdetail) values (?)
Hibernate: update join_11tab set address11tab=? where personid=?
```

3）:正常保存.

```
//      session.save(p1);
        session.save(add1);
```

```
Hibernate: insert into ADDRESS_11tab (addressdetail) values (?)
```

4）: 发生异常,不能保存.

```
        session.save(p1);
//      session.save(add1);
```

```
Hibernate: insert into PERSON_11tab (name, age) values (?, ?)
Hibernate: insert into join_11tab (address11tab, personid) values (?, ?)
Exception in thread "main" org.hibernate.TransientObjectException: com.lavasoft.dx._1_1_tab.Address11tab
```

2.1.4 Hibernate 一对多外键单向关联 （见_1_n_fk.zip）

这种情况很少见，但 Hibernate 同样允许采用连接表关联 1-1.有连接表的 1-1 同样只需要将 N-1 的 many-to-one 元素增加 unique="true"属性即可。

一、模型介绍

一个人（Person）对应多个地址（Address），比如家庭地址、公司地址。

二、实体（省略 getter、setter 方法）

```
public class Person1nfk implements Serializable {
    private int personid;
    private String name;
    private int age;
    private Set addresses=new HashSet();
```

```
public class Address1nfk implements Serializable {
    private int addressid;
private String addressdetail;
```

三、表模型

mysql> desc address_1nfk;

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
| personid      | int(11)      | YES  | MUL | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

mysql> desc person_1nfk;

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
  /* Formatted on 2007/08/21 10:06 (QP5 v5.50) */
  CREATE TABLE `address_1nfk` (
    `addressid` int(11) NOT NULL auto_increment,
    `addressdetail` varchar(255) default NULL,
    `addresses` int(11) default NULL,
    PRIMARY KEY  (`addressid`),
    KEY `FK9B93456DC08D1667` (`addresses`),
    CONSTRAINT `FK9B93456DC08D1667` FOREIGN KEY (`addresses`) REFERENCES `person_1nfk` 
(`personid`)
  ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=gbk;
```

```
  /* Formatted on 2007/08/21 10:07 (QP5 v5.50) */
  CREATE TABLE `person_1nfk` (
    `personid` int(11) NOT NULL auto_increment,
    `name` varchar(255) default NULL,
    `age` int(11) default NULL,
    PRIMARY KEY  (`personid`)
  ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

五、映射方法：在实体类 Person 里面添加 Address 的集合，即可形成一对多关系。

```
        <!--映射集合属性，关联到持久化类,inverse="false"表示主控端在 Person1nfk 端，
lazy="false"表示不采用延迟加载-->
        <set name="addresses"
             table="ADDRESS_1nfk"
             cascade="all"
        >
            <!--确定关联的外键列-->
            <key column="personid"/>
            <!--用以映射到关联类属性-->
            <one-to-many class="com.lavasoft.dx._1_n_fk.Address1nfk"/>
        </set>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_n_fk.Person1nfk" table="PERSON_1nfk">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--映射集合属性，关联到持久化类,inverse="false"表示主控端在 Person1nfk 端，
lazy="false"表示不采用延迟加载-->
        <set name="addresses"
             table="ADDRESS_1nfk"
             cascade="all"
        >
            <!--确定关联的外键列-->
            <key column="personid"/>
            <!--用以映射到关联类属性-->
            <one-to-many class="com.lavasoft.dx._1_n_fk.Address1nfk"/>
        </set>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_n_fk.Address1nfk" table="ADDRESS_1nfk">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_1nfk {
    public static void main(String[] args){
        Address1nfk add1=new Address1nfk();
        Address1nfk add2=new Address1nfk();
        Person1nfk p=new Person1nfk();

        add1.setAddressdetail("郑州市经三路");
        add2.setAddressdetail("合肥市宿州路");
        p.setName("wang");
        p.setAge(30);
        p.getAddresses().add(add1);
        p.getAddresses().add(add2);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add1);
        session.save(add2);
        session.save(p);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）：正常保存.

```
//      session.save(add1);
//      session.save(add2);
        session.save(p);
```

```
Hibernate: insert into PERSON_1nfk (name, age) values (?, ?)
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
```

```
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
Hibernate: update ADDRESS_1nfk set personid=? where addressid=?
Hibernate: update ADDRESS_1nfk set personid=? where addressid=?
```

2）:正常保存.

```
        session.save(add1);
        session.save(add2);
        session.save(p);
```

```
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
Hibernate: insert into PERSON_1nfk (name, age) values (?, ?)
Hibernate: update ADDRESS_1nfk set personid=? where addressid=?
Hibernate: update ADDRESS_1nfk set personid=? where addressid=?
```

3）:正常保存.

```
        session.save(add1);
        session.save(add2);
//      session.save(p);
```

```
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
```

2.1.5 **Hibernate 一对多连接表单向关联** （见_1_n_tab.zip）

一、模型介绍

一个人（Person）对应多个地址（Address），比如家庭地址、公司地址。

二、实体（省略 getter、setter 方法）

```
public class Person1ntab {
    private int personid;
    private String name;
    private int age;
    private Set addresses=new HashSet();
```

```
public class Address1nfk implements Serializable {
    private int addressid;
    private String addressdetail;
```

三、表模型

```
mysql> desc join_1ntab;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| personid   | int(11) | NO   | PRI |         |       |
| addressid  | int(11) | NO   | PRI |         |       |
+------------+---------+------+-----+---------+-------+
```

mysql> desc person_1ntab;

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

mysql> desc address_1ntab;

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
/* Formatted on 2007/08/21 10:58 (QP5 v5.50) */
CREATE TABLE `address_1ntab` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/21 10:58 (QP5 v5.50) */
CREATE TABLE `join_1ntab` (
  `personid` int(11) NOT NULL,
  `addressid` int(11) NOT NULL,
  PRIMARY KEY  (`personid`,`addressid`),
  UNIQUE KEY `addressid` (`addressid`),
  KEY `FK6B6078C3C8DF5BFF` (`personid`),
  KEY `FK6B6078C3C2B11347` (`addressid`),
  CONSTRAINT `FK6B6078C3C2B11347` FOREIGN KEY (`addressid`) REFERENCES `address_1ntab`
(`addressid`),
  CONSTRAINT `FK6B6078C3C8DF5BFF` FOREIGN KEY (`personid`) REFERENCES `person_1ntab`
(`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/21 10:58 (QP5 v5.50) */
```

```
CREATE TABLE `person_1ntab` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`personid`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_n_tab.Person1ntab" table="PERSON_1ntab">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--映射集合属性，join_1ntab 是连接表表名-->
        <set name="addresses"
             table="join_1ntab"
                >
            <!-- “column="personid"” 确定 PERSON_1ntab 表关联到连接表的外键列名-->
            <key column="personid"/>
            <!-- “column="addressid"” 关联 PERSON_1ntab 表的 Address1ntab 对象的 id 在
连接表中的列名-->
            <!-- “unique="true"表示 1-N，Person1ntab 是 1，Address1ntab 是多” -->
            <many-to-many
                    column="addressid"
                    unique="true"
                    class="com.lavasoft.dx._1_n_tab.Address1ntab"/>
        </set>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._1_n_tab.Address1ntab" table="ADDRESS_1ntab">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_1ntab {
    public static void main(String[] args){
        Address1ntab add1=new Address1ntab();
        Address1ntab add2=new Address1ntab();
        Address1ntab add3=new Address1ntab();
        Person1ntab p1=new Person1ntab();
        Person1ntab p2=new Person1ntab();

        add1.setAddressdetail("郑州市经三路");
        add2.setAddressdetail("合肥市宿州路");
        add3.setAddressdetail("北京市长安路");
        p1.setName("wang");
        p1.setAge(30);
        p2.setName("lee");
        p2.setAge(50);

        p1.getAddresses().add(add1);
        p1.getAddresses().add(add2);
        //p2.getAddresses().add(add2);
        p2.getAddresses().add(add3);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add1);
        session.save(add2);
        session.save(add3);
        session.save(p1);
        session.save(p2);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存.

```
        session.save(add1);
        session.save(add2);
        session.save(add3);
        session.save(p1);
        session.save(p2);
```

```
Hibernate: insert into PERSON_1nfk (name, age) values (?, ?)
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
Hibernate: insert into ADDRESS_1nfk (addressdetail) values (?)
Hibernate: update ADDRESS_1nfk set personid=? where addressid=?
Hibernate: update ADDRESS_1nfk set personid=? where addressid=?
```

2.1.6 **Hibernate 多对一外键单向关联** （见_n_1_fk.zip）

一、模型介绍

多个人（Person）对应一个地址（Address）。

二、实体（省略 getter、setter 方法）

```
public class Personn1fk {
    private int personid;
    private String name;
    private int age;
    private Addressn1fk addressn1fk;
```

```
public class Addressn1fk {
    private int addressid;
    private String addressdetail;
```

三、表模型

mysql> desc address_n1kf;

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

mysql> desc person_n1kf;

```
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| personid  | int(11)      | NO   | PRI | NULL    | auto_increment |
| name      | varchar(255) | YES  |     | NULL    |                |
| age       | int(11)      | YES  |     | NULL    |                |
| addressId | int(11)      | YES  | MUL | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
CREATE TABLE `address_n1kf` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;

DROP TABLE IF EXISTS `person_n1kf`;
```

```
CREATE TABLE `person_n1kf` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  `addressId` int(11) default NULL,
  PRIMARY KEY  (`personid`),
  KEY `FK4571AF54A2A3EE48` (`addressId`),
  CONSTRAINT `FK4571AF54A2A3EE48` FOREIGN KEY (`addressId`) REFERENCES
`address_n1kf` (`addressid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._n_1_fk.Personn1fk" table="PERSON_n1fk">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--用来映射关联 PO column 是 Address 在该表中的外键列名-->
        <many-to-one name="addressn1fk" column="addressId"/>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._n_1_fk.Addressn1fk" table="ADDRESS_n1fk">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_n1fk {
    public static void main(String[] args){
        Personn1fk p1=new Personn1fk();
        Personn1fk p2=new Personn1fk();

        p1.setAge(21);
        p1.setName("p1");

        p2.setAge(23);
        p2.setName("p2");
        Addressn1fk add=new Addressn1fk();
        add.setAddressdetail("郑州市经三路");

        p1.setAddressn1fk(add);
        p2.setAddressn1fk(add);

        Session session=HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add);
        session.save(p1);
        session.save(p2);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存. 推荐这么干!

```
        session.save(p1);
        session.save(p2);
```

```
Hibernate: insert into ADDRESS_n1kf (addressdetail) values (?)
Hibernate: insert into PERSON_n1kf (name, age, addressId) values (?, ?, ?)
Hibernate: insert into PERSON_n1kf (name, age, addressId) values (?, ?, ?)
```

2）:正常保存.

```
        session.save(p1);
        session.save(p2);
        session.save(add);
```

```
Hibernate: insert into PERSON_n1kf (name, age, addressId) values (?, ?, ?)
Hibernate: insert into PERSON_n1kf (name, age, addressId) values (?, ?, ?)
Hibernate: insert into ADDRESS_n1kf (addressdetail) values (?)
Hibernate: update PERSON_n1kf set name=?, age=?, addressId=? where personid=?
Hibernate: update PERSON_n1kf set name=?, age=?, addressId=? where personid=?
```

3）:正常保存.

```
        session.save(add);
//      session.save(p1);
//      session.save(p2);
```

```
Hibernate: insert into ADDRESS_n1kf (addressdetail) values (?)
```

4）: 发生异常,不能保存.

```
//      session.save(add);
        session.save(p1);
        session.save(p2);
```

```
Hibernate: insert into PERSON_n1kf (name, age, addressId) values (?, ?, ?)
Hibernate: insert into PERSON_n1kf (name, age, addressId) values (?, ?, ?)
Exception in thread "main" org.hibernate.TransientObjectException: com.lavasoft.dx._n_1_fk.Addressn1kf
```

2.1.7 Hibernate 多对一连接表单向关联 （见_n_1_tab.zip）

一、模型介绍

多个人（Person）对应一个地址（Address）。

二、实体（省略 getter、setter 方法）

```
public class Personn1tab {
    private int personid;
    private String name;
    private int age;
    private Addressn1tab addressn1tab;
```

```
public class Addressn1tab {
    private int addressid;
    private String addressdetail;
```

三、表模型

```
mysql> desc address_n1tab;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

```
mysql> desc join_n1tab;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| personid    | int(11) | NO   | PRI |         |       |
| addressn1tab | int(11) | YES  | MUL | NULL    |       |
+-------------+---------+------+-----+---------+-------+
```

```
mysql> desc person_n1tab;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
CREATE TABLE `address_n1tab` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=gbk;
```

```
CREATE TABLE `join_n1tab` (
  `personid` int(11) NOT NULL,
  `address11fk` int(11) default NULL,
  PRIMARY KEY  (`personid`),
  KEY `FKAC780AAADAE3A82C` (`personid`),
  KEY `FKAC780AAAC6242A64` (`address11fk`),
  CONSTRAINT  `FKAC780AAAC6242A64`  FOREIGN  KEY  (`address11fk`)  REFERENCES
`address_n1tab` (`addressid`),
  CONSTRAINT  `FKAC780AAADAE3A82C`  FOREIGN  KEY  (`personid`)  REFERENCES
`person_n1tab` (`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
CREATE TABLE `person_n1tab` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`personid`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._n_1_tab.Personn1tab" table="PERSON_n1tab">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--使用 join 元素显式确定链接表-->
        <join table="join_n1tab">
            <!—映射关联所用的外键-->
```

```xml
            <key column="personid"/>
            <many-to-one name="addressn1tab"/>
        </join>
    </class>
</hibernate-mapping>
```

```xml
<hibernate-mapping>
    <class name="com.lavasoft.dx._n_1_tab.Addressn1tab" table="ADDRESS_n1tab">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
    </class>
</hibernate-mapping>
```

六、测试方法

```java
public class Test_n1tab {
    public static void main(String[] args){
        Personn1tab p1=new Personn1tab();
        Personn1tab p2=new Personn1tab();

        p1.setAge(21);
        p1.setName("p1");

        p2.setAge(23);
        p2.setName("p2");

        Addressn1tab add=new Addressn1tab();
        add.setAddressdetail("郑州市经三路");

        p1.setAddressn1tab(add);
        p2.setAddressn1tab(add);

        Session session=HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add);
        session.save(p1);
        session.save(p2);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）：正常保存. 推荐这么干!

```
        session.save(p1);
        session.save(p2);
```

```
Hibernate: insert into ADDRESS_n1tab (addressdetail) values (?)
Hibernate: insert into PERSON_n1tab (name, age) values (?, ?)
Hibernate: insert into join_n1tab (address11fk, personid) values (?, ?)
Hibernate: insert into PERSON_n1tab (name, age) values (?, ?)
Hibernate: insert into join_n1tab (address11fk, personid) values (?, ?)
```

2）：正常保存.

```
        session.save(p1);
        session.save(p2);
        session.save(add);
```

```
 Hibernate: insert into PERSON_n1tab (name, age) values (?, ?)
 Hibernate: insert into join_n1tab (address11fk, personid) values (?, ?)
 Hibernate: insert into PERSON_n1tab (name, age) values (?, ?)
 Hibernate: insert into join_n1tab (address11fk, personid) values (?, ?)
```

2.1.8 Hibernate 多对多单向关联 （见_n_n.zip）

一、模型介绍

~~多个人（Person）对应多个地址（Address）。~~
一个人可对应多个地址，一个地址也可以对应多个人。

二、实体（省略 **getter**、**setter** 方法）

```
public class Personnn {
    private int personid;
    private String name;
    private int age;
    private Set addresses=new HashSet();
```

```
public class Addressnn {
    private int addressid;
    private String addressdetail;
```

三、表模型

mysql> desc person_nn;

```
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

mysql> desc join_nn;

```
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| personid  | int(11) | NO   | PRI |         |       |
| addressid | int(11) | NO   | PRI |         |       |
+-----------+---------+------+-----+---------+-------+
```

mysql> desc person_nn;

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
/* Formatted on 2007/08/21 11:13 (QP5 v5.50) */
CREATE TABLE `address_nn` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/21 11:14 (QP5 v5.50) */
CREATE TABLE `join_nn` (
  `personid` int(11) NOT NULL,
  `addressid` int(11) NOT NULL,
  PRIMARY KEY  (`personid`,`addressid`),
  KEY `FKAAB98CF5E008E752` (`personid`),
  KEY `FKAAB98CF5239F6A16` (`addressid`),
  CONSTRAINT `FKAAB98CF5239F6A16` FOREIGN KEY (`addressid`) REFERENCES `address_nn`
(`addressid`),
  CONSTRAINT `FKAAB98CF5E008E752` FOREIGN KEY (`personid`) REFERENCES `person_nn`
(`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/21 11:14 (QP5 v5.50) */
CREATE TABLE `person_nn` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`personid`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.dx._n_n.Personnn" table="PERSON_nn">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--映射集合属性，join_1ntab 是连接表表名-->
        <set name="addresses"
             table="join_nn"
                >
            <!-- “column="personid"” 确定 PERSON_1ntab 表关联到连接表的外键列名-->
            <key column="personid"/>
            <!-- “column="addressid"” 关联 PERSON_1ntab 表的 Address1ntab 对象的 id 在
连接表中的列名-->
            <many-to-many
                    column="addressid"
                    class="com.lavasoft.dx._n_n.Addressnn"/>
        </set>
    </class>
</hibernate-mapping>

<hibernate-mapping>
    <class name="com.lavasoft.dx._n_n.Addressnn" table="ADDRESS_nn">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_nn {
    public static void main(String[] args){
        Addressnn add1=new Addressnn();
        Addressnn add2=new Addressnn();
        Addressnn add3=new Addressnn();
        Personnn p1=new Personnn();
        Personnn p2=new Personnn();

        add1.setAddressdetail("郑州市经三路");
        add2.setAddressdetail("合肥市宿州路");
        add3.setAddressdetail("北京市长安路");
        p1.setName("wang");
        p1.setAge(30);
        p2.setName("lee");
        p2.setAge(50);

        p1.getAddresses().add(add1);
        p1.getAddresses().add(add2);
        p2.getAddresses().add(add2);
        p2.getAddresses().add(add3);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(add1);
        session.save(add2);
        session.save(add3);
        session.save(p1);
        session.save(p2);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存.

```
        session.save(add1);
        session.save(add2);
        session.save(add3);
        session.save(p1);
        session.save(p2);
```

```
Hibernate: insert into ADDRESS_nn (addressdetail) values (?)
Hibernate: insert into ADDRESS_nn (addressdetail) values (?)
Hibernate: insert into ADDRESS_nn (addressdetail) values (?)
Hibernate: insert into PERSON_nn (name, age) values (?, ?)
Hibernate: insert into PERSON_nn (name, age) values (?, ?)
Hibernate: insert into join_nn (personid, addressid) values (?, ?)
Hibernate: insert into join_nn (personid, addressid) values (?, ?)
Hibernate: insert into join_nn (personid, addressid) values (?, ?)
Hibernate: insert into join_nn (personid, addressid) values (?, ?)
```

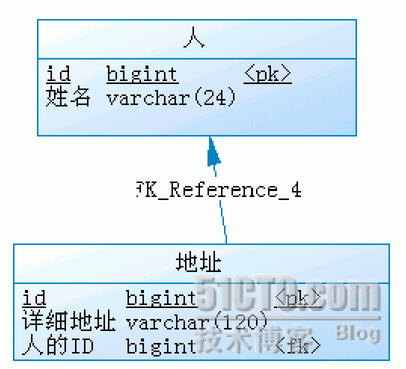
2.2 双向关联

2.2.1 Hibernate 一对一外键双向关联（见_1_1_fk_bidirection.zip）

一对一外键关联是一对多外键关联的特例，只是在多的一方加了个唯一性约束。

一、模型

一个人对应一个地址。



```
/*==============================================================*/
/* DBMS name:      MySQL 5.0                                    */
/* Created on:     2008-12-9 0:12:54                            */
/*==============================================================*/

drop table if exists address;
drop table if exists person;
/*==============================================================*/
/* Table: address                                               */
/*==============================================================*/
create table address
(
   id              bigint not null auto_increment comment 'ID',
   detail          varchar(120) not null comment '详细地址',
   personid        bigint comment '人的ID',
   primary key (id)
)type = InnoDB;
alter table address comment '地址';
/*==============================================================*/
/* Table: person                                                */
/*==============================================================*/
create table person
(
   id              bigint not null auto_increment comment 'ID',
   name            varchar(24) not null comment '姓名',
  primary key (id)
)type = InnoDB;
alter table person comment '人';
alter table address add constraint FK_Reference_4 foreign key (personid)
     references person (id) on delete restrict on update restrict;
```

二、对象模型

```
public class Person implements java.io.Serializable {
  private Long id;
  private String name;
  private Address address;
```

```
public class Address implements java.io.Serializable {
  private Long id;
  private Person person;
  private String detail;
```

三、映射文件

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.
0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="entity.Person" table="person">
    <id name="id" type="java.lang.Long">
      <column name="id" />
      <generator class="identity" />
    </id>
    <property name="name" type="java.lang.String">
      <column name="name" length="24" not-null="true">
        <comment>姓名</comment>
      </column>
    </property>
    <one-to-one name="address" cascade="all" />
  </class>
</hibernate-mapping>
```

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.
0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="entity.Address" table="address" catalog="testdb">
    <id name="id" type="java.lang.Long">
```

```xml
      <column name="id" />
      <generator class="identity" />
    </id>
    <property name="detail" type="java.lang.String">
      <column name="detail" length="120" not-null="true">
        <comment>详细地址</comment>
      </column>
    </property>
    <many-to-one name="person" class="entity.Person"
     fetch="select" unique="true">
      <column name="personid">
        <comment>人的 ID</comment>
      </column>
    </many-to-one>
  </class>
</hibernate-mapping>
```

```xml
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>

  <session-factory>
    <property name="connection.username">root</property>
    <property name="connection.url">
     jdbc:mysql://localhost:3306/testdb
    </property>
    <property name="dialect">
     org.hibernate.dialect.MySQLDialect
    </property>
    <property name="connection.password">xiaohui</property>
    <property name="connection.driver_class">
     com.mysql.jdbc.Driver
    </property>
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <mapping resource="entity/Person.hbm.xml" />
    <mapping resource="entity/Address.hbm.xml" />

  </session-factory>
```

```
</hibernate-configuration>
```

四、测试

```
import org.hibernate.Transaction;
import entity.Address;
import entity.Person;
import utils.HibernateSessionFactory;

public class Test {
  public static void main(String[] args) {
    savePerson();
  }

  public static void savePerson() {
    Person person = new Person("张三");
    Address address = new Address("XX 街 X 号");
    person.setAddress(address);
    address.setPerson(person);

    Session session = HibernateSessionFactory.getSession();
    Transaction tx = session.beginTransaction();
    session.save(person);
    tx.commit();
  }
}
```

运行日志：

```
Hibernate:
    insert
    into
        person
        (name)
    values
        (?)
Hibernate:
    insert
    into
        testdb.address
        (detail, personid)
    values
        (?, ?)
```

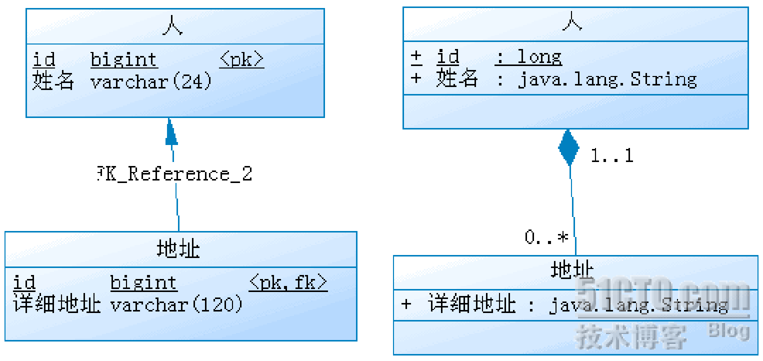
2.2.2 Hibernate 一对一主键双向关联

一对一主键映射在一对一映射中还算是最为常用的。

一、模型

一个人 Person 对应一个地址 Address。

二、数据模型和对象模型图



导出建表 **SQL** 如下：

```
/*==============================================================*/
/* DBMS name:      MySQL 5.0                                    */
/* Created on:     2008-12-8 23:05:32                           */
/*==============================================================*/


drop table if exists address;
drop table if exists person;

/*==============================================================*/
/* Table: address                                               */
/*==============================================================*/
create table address
(
   id            bigint not null comment 'ID',
   detail        varchar(120) not null comment '详细地址',
   primary key (id)
) type = InnoDB;

alter table address comment '地址';

/*==============================================================*/
```

```
/* Table: person                                                */
/*==============================================================*/
create table person
(
    id              bigint not null auto_increment comment 'ID',
    name            varchar(24) not null comment '姓名',
    primary key (id)
) type = InnoDB;

alter table person comment '人';

alter table address add constraint FK_Reference_2 foreign key (id)
        references person (id) on delete restrict on update restrict;
```

三、对象模型代码

```
public class Person implements java.io.Serializable {
  private Long id;
  private String name;
  private Address address;
```

```
public class Address implements java.io.Serializable {
  private Long id;
  private Person person;
  private String detail;
```

四、映射代码

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.
0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="entity.Person" table="person">
    <id name="id" type="java.lang.Long">
      <column name="id" />
      <generator class="identity" />
    </id>
    <property name="name" type="java.lang.String">
      <column name="name" length="24" not-null="true">
        <comment>姓名</comment>
      </column>
    </property>
    <!-- cascade="all"：在保存 person 对象的时候，级联保存 person 对象关联的 addres
s 对象    -->
    <one-to-one name="address" cascade="all" />
  </class>
</hibernate-mapping>
```

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.
0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="entity.Address" table="address" catalog="mydb">
    <id name="id" type="java.lang.Long">
      <column name="id" />
      <!-- class="foreign": 一对一主键映射中，使用另外一个相关联的对象的标识符 -->
      <generator class="foreign">
        <param name="property">person</param>
      </generator>
    </id>
    <property name="detail" type="java.lang.String">
      <column name="detail" length="120" not-null="true">
        <comment>详细地址</comment>
      </column>
    </property>
    <!-- 表示在 address 表存在一个外键约束，外键参考相关联的表 person -->
    <one-to-one name="person" constrained="true" />
  </class>
</hibernate-mapping>
```

五、**Hibernate** 配置

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>

<session-factory>
  <property name="connection.username">root</property>
  <property name="connection.url">
    jdbc:mysql://localhost:3306/mydb
  </property>
  <property name="dialect">
    org.hibernate.dialect.MySQLDialect
  </property>
```

```xml
  <property name="connection.password">xiaohui</property>
  <property name="connection.driver_class">
    com.mysql.jdbc.Driver
  </property>
  <property name="show_sql">true</property>
  <property name="format_sql">true</property>
  <mapping resource="entity/Person.hbm.xml" />
  <mapping resource="entity/Address.hbm.xml" />

</session-factory>

</hibernate-configuration>
```

测试很简单就不写了。

2.2.3 Hibernate 一对一连接表双向关联（见_1_1_tab_bidirection.zip）

一、模型介绍

一个人（Person）对应一个地址（Address）。

二、实体（省略 getter、setter 方法）

```java
public class Person11tab_sx {
    private int personid;
    private String name;
    private int age;
    private Address11tab_sx address11tab_sx;
```

```java
public class Address11tab_sx {
    private int addressid;
    private String addressdetail;
    private Person11tab_sx person11tab_sx;
```

三、表模型

mysql> desc person_11tab_sx;

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
mysql> desc join_11tab_sx;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| addressid | int(11) | NO   | UNI |         |       |
| personid  | int(11) | NO   | PRI |         |       |
+-----------+---------+------+-----+---------+-------+
```

```
mysql> desc address_11tab_sx;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
/* Formatted on 2007/08/22 17:35 (QP5 v5.50) */
CREATE TABLE `person_11tab_sx` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/22 17:34 (QP5 v5.50) */
CREATE TABLE `address_11tab_sx` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/22 18:35 (QP5 v5.50) */
CREATE TABLE `join_11tab_sx` (
  `addressid` int(11) NOT NULL,
  `personid` int(11) NOT NULL,
  PRIMARY KEY  (`personid`),
  UNIQUE KEY `addressid` (`addressid`),
  UNIQUE KEY `personid` (`personid`),
  KEY `FKF4AA80E44327AAB6` (`personid`),
  KEY `FKF4AA80E460C0C9F0` (`addressid`),
  CONSTRAINT `FKF4AA80E460C0C9F0` FOREIGN KEY (`addressid`) REFERENCES
`address_11tab_sx` (`addressid`),
  CONSTRAINT `FKF4AA80E44327AAB6` FOREIGN KEY (`personid`) REFERENCES
`person_11tab_sx` (`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.sx._1_1_tab.Person11tab_sx" table="PERSON_11tab_sx">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <join table="join_11tab_sx"
              optional="true">
            <key column="personid"
                 unique="true"/>
            <many-to-one name="address11tab_sx"
                         column="addressid"
                         not-null="true"
                         unique="true"/>
        </join>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.sx._1_1_tab.Address11tab_sx" table="ADDRESS_11tab_sx">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
        <join table="join_11tab_sx"
              optional="true"
              inverse="true">
            <key column="addressid"
                 unique="true"/>
            <many-to-one name="person11tab_sx" column="personid"
                         not-null="true" unique="true"/>
        </join>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_11tab_sx {
    public static void main(String[] args){
        Address11tab_sx add = new Address11tab_sx();
        Person11tab_sx p = new Person11tab_sx();
        add.setAddressdetail("郑州市经三路");
        p.setAge(12);
        p.setName("wudalang");

        add.setPerson11tab_sx(p);
        p.setAddress11tab_sx(add);

        Session session = HibernateUtil.getCurrentSession();
        Transaction tx = session.beginTransaction();
        session.saveOrUpdate(p);
        session.saveOrUpdate(add);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存.

```
        session.saveOrUpdate(p);
        session.saveOrUpdate(add);
```

```
Hibernate: insert into PERSON_11tab_sx (name, age) values (?, ?)
Hibernate: insert into ADDRESS_11tab_sx (addressdetail) values (?)
Hibernate: insert into join_11tab_sx (addressid, personid) values (?, ?)
```

2.2.4 Hibernate 一对多外键双向关联（见_1_n_fk_bidirection.zip）

一、模型介绍

一个人（Person）对应多个地址（Address）。

二、实体（省略 getter、setter 方法）

```
public class Person1nfk_sx implements Serializable {
    private int personid;
    private String name;
    private int age;
    private Set addresses=new HashSet();
```

```
public class Address1nfk_sx implements Serializable {
    private int addressid;
    private String addressdetail;
    private Person1nfk_sx person1nfkSx;
```

三、表模型

mysql> desc person_1nfk_sx;

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

mysql> desc address_1nfk_sx;

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
| personid      | int(11)      | NO   | MUL |         |                |
+---------------+--------------+------+-----+---------+----------------+
```

四、生成的 SQL 脚本

```
  /* Formatted on 2007/08/22 17:42 (QP5 v5.50) */
  CREATE TABLE `address_1nfk` (
    `addressid` int(11) NOT NULL auto_increment,
    `addressdetail` varchar(255) default NULL,
    `personid` int(11) default NULL,
    PRIMARY KEY  (`addressid`),
    KEY `FK9B93456DA6D6C1F5` (`personid`),
    CONSTRAINT `FK9B93456DA6D6C1F5` FOREIGN KEY (`personid`) REFERENCES `person_1nfk`
(`personid`)
  ) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
  /* Formatted on 2007/08/22 17:42 (QP5 v5.50) */
  CREATE TABLE `person_1nfk` (
    `personid` int(11) NOT NULL auto_increment,
    `name` varchar(255) default NULL,
    `age` int(11) default NULL,
    PRIMARY KEY  (`personid`)
  ) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.sx._1_n_fk.Person1nfk_sx" table="PERSON_1nfk_sx">
```

```
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--映射集合属性，关联到持久化类-->
        <set name="addresses" inverse="true" cascade="all">
            <!--column 用于指定外键列名-->
            <key column="personid" not-null="true"/>
            <!--映射关联类-->
            <one-to-many class="com.lavasoft.sx._1_n_fk.Address1nfk_sx"/>
        </set>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.sx._1_n_fk.Address1nfk_sx" table="ADDRESS_1nfk_sx">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
        <!--映射关联属性，column 属性指定外键列名-->
        <many-to-one name="person1nfk"
                     class="com.lavasoft.sx._1_n_fk.Person1nfk_sx"
                     fetch="select"
                     cascade="save-update">
            <column name="personid" not-null="true"/>
        </many-to-one>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_1nfk_sx {
    public static void main(String[] args){
        Address1nfk_sx add1=new Address1nfk_sx();
        Address1nfk_sx add2=new Address1nfk_sx();
        Person1nfk_sx p=new Person1nfk_sx();

        add1.setAddressdetail("郑州市经三路");
        add2.setAddressdetail("合肥市宿州路");
        p.setName("wang");
        p.setAge(30);

        p.getAddresses().add(add1);
        p.getAddresses().add(add2);
        add1.setPerson1nfk(p);
        add2.setPerson1nfk(p);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(p);
        session.saveOrUpdate(add1);
        session.saveOrUpdate(add2);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1):正常保存.

```
        session.save(p);
        session.saveOrUpdate(add1);
        session.saveOrUpdate(add2);
```

```
Hibernate: insert into PERSON_1nfk_sx (name, age) values (?, ?)
Hibernate: insert into ADDRESS_1nfk_sx (addressdetail, personid) values (?, ?)
Hibernate: insert into ADDRESS_1nfk_sx (addressdetail, personid) values (?, ?)
```

2.2.5 Hibernate 一对多连接表双向关联（见_1_n_tab_bidirection.zip）

一、模型介绍

一个人（Person）对应多个地址（Address）。

二、实体（省略 getter、setter 方法）

```
public class Person1ntab_sx {
    private int personid;
    private String name;
    private int age;
    private Set addresses=new HashSet();
```

```
public class Address1ntab_sx {
    private int addressid;
    private String addressdetail;
    private Person1ntab_sx person1ntab_sx;
```

三、表模型

mysql> desc person_1ntab_sx;

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

mysql> desc address_1ntab_sx;

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

mysql> desc join_1ntab_sx;

```
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| addressid | int(11) | NO   | PRI |         |       |
| personid  | int(11) | NO   | PRI |         |       |
+-----------+---------+------+-----+---------+-------+
```

四、生成的 SQL 脚本

```
/* Formatted on 2007/08/22 17:52 (QP5 v5.50) */
CREATE TABLE `address_1ntab_sx` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/22 17:52 (QP5 v5.50) */
CREATE TABLE `person_1ntab_sx` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/22 17:52 (QP5 v5.50) */
CREATE TABLE `join_1ntab_sx` (
  `addressid` int(11) NOT NULL,
  `personid` int(11) NOT NULL,
  PRIMARY KEY  (`personid`,`addressid`),
  KEY `FK8F869F61F93DDD6` (`personid`),
  KEY `FK8F869F61FC0F682A` (`addressid`),
  CONSTRAINT  `FK8F869F61FC0F682A`  FOREIGN  KEY  (`addressid`)  REFERENCES
`address_1ntab_sx` (`addressid`),
  CONSTRAINT `FK8F869F61F93DDD6` FOREIGN KEY (`personid`) REFERENCES `person_1ntab_sx`
(`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.sx._1_n_tab.Person1ntab_sx" table="PERSON_1ntab_sx">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--映射集合属性，关联到持久化类-->
        <!--table="join_1ntab_sx"指定了连接表的名字-->
        <set name="addresses"
             table="join_1ntab_sx"
             cascade="all">
            <!--column="personid"指定连接表中关联当前实体类的列名—>
            <key column="personid" not-null="true"/>
```

```xml
            <!--unique="true"表示当前实体类是"1",不是"n"—>
            <many-to-many column="addressid"
                          unique="true"
                          class="com.lavasoft.sx._1_n_tab.Address1ntab_sx"/>
        </set>
    </class>
</hibernate-mapping>

<hibernate-mapping>
    <class name="com.lavasoft.sx._1_n_tab.Address1ntab_sx"
           table="ADDRESS_1ntab_sx">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
        <!--映射关联属性，column 属性指定外键列名-->
        <join   table="join_1ntab_sx"
                inverse="true"
              optional="true">
            <key column="addressid"/>
            <many-to-one name="person1ntab_sx"
                         column="personid"
                         cascade="all"
                         not-null="true"/>
        </join>
    </class>
</hibernate-mapping>
```

六、测试方法

```java
public class Test_1ntab_sx {
    public static void main(String[] args){
        Address1ntab_sx add1=new Address1ntab_sx();
        Address1ntab_sx add2=new Address1ntab_sx();
        Person1ntab_sx p=new Person1ntab_sx();

        add1.setAddressdetail("郑州市经三路");
        add2.setAddressdetail("合肥市宿州路");
        p.setName("wang");
        p.setAge(30);

        p.getAddresses().add(add1);
        p.getAddresses().add(add2);
        add1.setPerson1ntab_sx(p);
        add2.setPerson1ntab_sx(p);
        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
//      session.save(p);
        session.saveOrUpdate(add1);
        session.saveOrUpdate(add2);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存.

```
//      session.save(p);
        session.saveOrUpdate(add1);
        session.saveOrUpdate(add2);
```

```
Hibernate: insert into PERSON_1ntab_sx (name, age) values (?, ?)
Hibernate: insert into ADDRESS_1ntab_sx (addressdetail) values (?)
Hibernate: insert into ADDRESS_1ntab_sx (addressdetail) values (?)
Hibernate: insert into join_1ntab_sx (personid, addressid) values (?, ?)
Hibernate: insert into join_1ntab_sx (personid, addressid) values (?, ?)
```

2.2.6 Hibernate 多对多双向关联（见_n_n_bidirection.zip）

一、模型介绍

~~多个人（Person）对应多个地址（Address）。~~
一个人可对应多个地址，一个地址也可以对应多个人。

二、实体（省略 getter、setter 方法）

```
public class Personnn_sx {
    private int personid;
    private String name;
    private int age;
    private Set addresses=new HashSet();
```

```
public class Addressnn_sx {
    private int addressid;
    private String addressdetail;
    private Set persons = new HashSet();
```

三、表模型

```
mysql> desc person_nn_sx;
```

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| personid | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255) | YES  |     | NULL    |                |
| age      | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

```
mysql> desc address_nn_sx;
```

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| addressid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| addressdetail | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

```
mysql> desc join_nn_sx;
```

```
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| addressid | int(11) | NO   | PRI |         |       |
| personid  | int(11) | NO   | PRI |         |       |
+-----------+---------+------+-----+---------+-------+
```

四、生成的 SQL 脚本

```
/* Formatted on 2007/08/22 17:59 (QP5 v5.50) */
CREATE TABLE `address_nn_sx` (
  `addressid` int(11) NOT NULL auto_increment,
  `addressdetail` varchar(255) default NULL,
  PRIMARY KEY  (`addressid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/22 17:59 (QP5 v5.50) */
CREATE TABLE `person_nn_sx` (
  `personid` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `age` int(11) default NULL,
  PRIMARY KEY  (`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

```
/* Formatted on 2007/08/22 17:59 (QP5 v5.50) */
CREATE TABLE `join_nn_sx` (
  `addressid` int(11) NOT NULL,
  `personid` int(11) NOT NULL,
  PRIMARY KEY  (`personid`,`addressid`),
  KEY `FK6EBBC5EF6C600921` (`personid`),
  KEY `FK6EBBC5EF2A92FF3D` (`addressid`),
  CONSTRAINT `FK6EBBC5EF2A92FF3D` FOREIGN KEY (`addressid`) REFERENCES `address_nn_sx`
(`addressid`),
  CONSTRAINT `FK6EBBC5EF6C600921` FOREIGN KEY (`personid`) REFERENCES `person_nn_sx`
(`personid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

五、映射方法

```
<hibernate-mapping>
    <class name="com.lavasoft.sx._n_n.Personnn_sx" table="PERSON_nn_sx">
        <id name="personid">
            <generator class="identity"/>
        </id>
        <property name="name"/>
        <property name="age"/>
        <!--映射集合属性，关联到持久化类-->
        <!--table="join_1ntab_sx"指定了连接表的名字-->
        <set name="addresses"
             table="join_nn_sx"
             cascade="all">
            <!--column="personid"指定连接表中关联当前实体类的列名—>
            <key column="personid" not-null="true"/>
            <!--column="addressid"是连接表中关联本实体的外键-->
            <many-to-many column="addressid"
                          class="com.lavasoft.sx._n_n.Addressnn_sx"/>
        </set>
    </class>
</hibernate-mapping>
```

```
<hibernate-mapping>
    <class name="com.lavasoft.sx._n_n.Addressnn_sx"
           table="ADDRESS_nn_sx">
        <id name="addressid">
            <generator class="identity"/>
        </id>
        <property name="addressdetail"/>
        <!--table="join_nn_sx"是双向多对多的连接表-->
        <set name="persons"
             inverse="true"
             table="join_nn_sx">
            <!--column="addressid"是连接表中关联本实体的外键-->
            <key column="addressid"/>
            <many-to-many column="personid"
```

```
                            class="com.lavasoft.sx._n_n.Personnn_sx"/>
        </set>
    </class>
</hibernate-mapping>
```

六、测试方法

```
public class Test_nn_sx {
    public static void main(String[] args){
        Addressnn_sx add1=new Addressnn_sx();
        Addressnn_sx add2=new Addressnn_sx();
        Personnn_sx p1=new Personnn_sx();
        Personnn_sx p2=new Personnn_sx();

        add1.setAddressdetail("郑州市经三路");
        add2.setAddressdetail("合肥市宿州路");
        p1.setName("wang");
        p1.setAge(30);
        p2.setName("zhang");
        p2.setAge(22);

        p1.getAddresses().add(add1);
        p1.getAddresses().add(add2);
        p2.getAddresses().add(add2);
        add1.getPersons().add(p1);
        add2.getPersons().add(p1);
        add2.getPersons().add(p2);

        Session session= HibernateUtil.getCurrentSession();
        Transaction tx=session.beginTransaction();
        session.save(p1);
        session.save(p2);
//      session.saveOrUpdate(add1);
//      session.saveOrUpdate(add2);
        tx.commit();
        HibernateUtil.closeSession();
    }
}
```

七、测试结果

1）:正常保存.

```
        session.save(p1);
        session.save(p2);
//      session.saveOrUpdate(add1);
//      session.saveOrUpdate(add2);
```

```
Hibernate: insert into PERSON_nn_sx (name, age) values (?, ?)
Hibernate: insert into ADDRESS_nn_sx (addressdetail) values (?)
Hibernate: insert into ADDRESS_nn_sx (addressdetail) values (?)
Hibernate: insert into PERSON_nn_sx (name, age) values (?, ?)
Hibernate: insert into join_nn_sx (personid, addressid) values (?, ?)
Hibernate: insert into join_nn_sx (personid, addressid) values (?, ?)
Hibernate: insert into join_nn_sx (personid, addressid) values (?, ?)
```