关联关系映射

关联关系概述

- ✓ 在RDBMS中，关系表是最常见的设计模式，表和表之间都有引用的主键和外键关系，而面向对象思想中类和类之间也有关联关系，可以用Hibernate来映射这种关系
- ✓ 例如员工和部门之间，新闻和新闻类型之间，都具有关联关系
- ✓ 关联关系大致可以分为两种：
  - ➢ 单项关联关系：例如员工信息中可以找到部门信息，新闻信息中可以找到新闻类型信息
  - ➢ 双向关联关系：例如员工信息中可以找到部门信息，但是部门信息中也包含了此部门的员工信息

关联关系分类

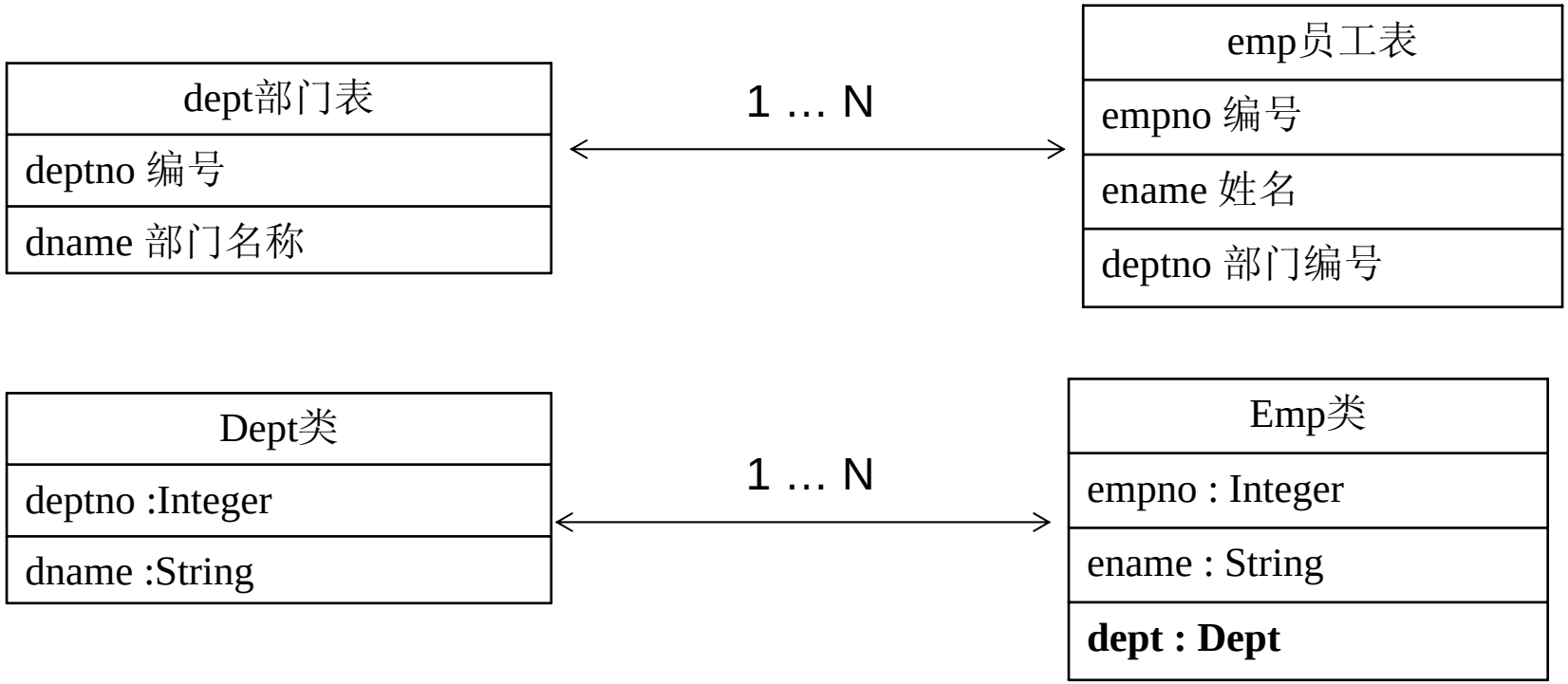
- ✓ 单向关联
  - ➢ 单向一对一
  - ➢ 单向一对多
  - ➢ 单项多对一
  - ➢ 单项多对多
- ✓ 双向关联
  - ➢ 双向一对一
  - ➢ 双向一对多
  - ➢ 双向多对多

一对多和多对一关联映射

- ✓ 关系型数据库中遍地都是一对多的关系，部门和员工，国家和省，客户和订单等等
- ✓ 在hibernate中数据库中的一对多关系被拆分成了两个关系
  - ➢ 多对一关系
    - ➢ 多的一端实体类包含一的一端的实体类类型的成员变量，换句话说就是在从表这边存储了匹配的主表数据
  - ➢ 一对多关系
    - ➢ 一的一端实体类包含一个集合类型的成员变量，集合中的元素类型正是多的一端的实体类类型，换句话说就是在主表这边存储了匹配的从表数据
- ✓ 以上两种关系只存在一种就是单向关联，两个都存在就是双向关联

多对一关系示例

- ✓ 员工表和部门表，在员工表中引用了部门表的部门编号

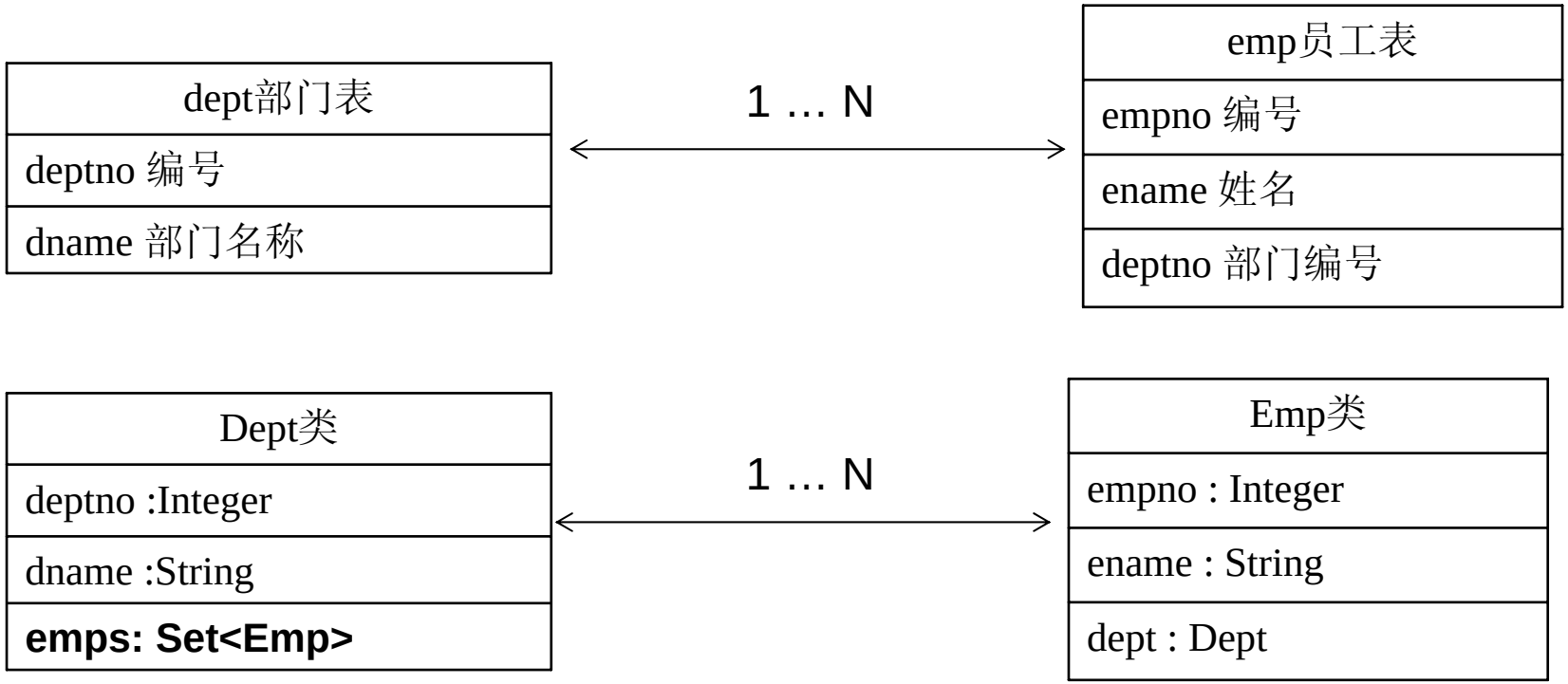


多的一端映射文件配置

```
<hibernate-mapping>
    <class name="com.Emp" table="EMP" schema="HB2">
        <id name="empno" type="java.lang.Integer">
            <generator class="increment" />
        </id>
        <property name="ename" type="java.lang.String"/>
        <!-- name属性设置类中的成员变量名，column是表中的外键列名 -->
        <many-to-one name="dept" column="deptno"/>
    </class>
</hibernate-mapping>
```

一对多关系示例

- ✓ 员工表和部门表，在员工表中引用了部门表的部门编号



一的一端映射文件配置

```
<hibernate-mapping>
    <class name="com.Dept" table="DEPT" schema="HB2">
        <id name="deptno" type="java.lang.Integer">
            <generator class="assigned" />
        </id>
        <property name="dname" type="java.lang.String"/>

        <!-- 设置集合属性的名称为emps-->
        <set name="emps">
            <!-- 设置外键列名 -->
            <key column="deptno"/>
            <!-- 设置关联类的类型 -->
            <one-to-many class="com.Emp"/>
        </set>

    </class>
</hibernate-mapping>
```

cascade关键属性

- ✓ cascade是<set>标签的一个属性，设置在“<one-to-many>”的一端，表示级联设置，所谓的级联就是在“一”端的变动会对“多”端造成影响
- ✓ 属性值
  - cascade=“none”,（默认）
  - cascade="save-update",
  - cascade="delete",
  - cascade="all"

级联的属性

- ✓ none就是不使用级联操作，默认级联是none。
- ✓ save-update也就是只有对象保存操作（持久化操作）或者是持久化对象的更新操作，才会级联操作关联对象（子对象）。
- ✓ delete对持久化对象的删除操作时会进行级联操作关联对象（子对象）。
- ✓ all对持久化对象的所有操作都会级联操作关联对象（子对象）。

<set>标签的其他属性

- ✓ inverse属性，默认是false，表示，表示自己是主控方，而一般我们都在“一”的一端设置<set>标签的inverse属性为true，表示反转，即对方是主控方，维护关系
  - <set name="emps" cascade="all" inverse="true">
- ✓ order-by属性，可以把集合中的数据做排序，只需要指定排序的列名就可以了，可以设置升序或者降序排列
  - <set name="emps" cascade="all" inverse="true" order-by="empno">

一对一关联映射

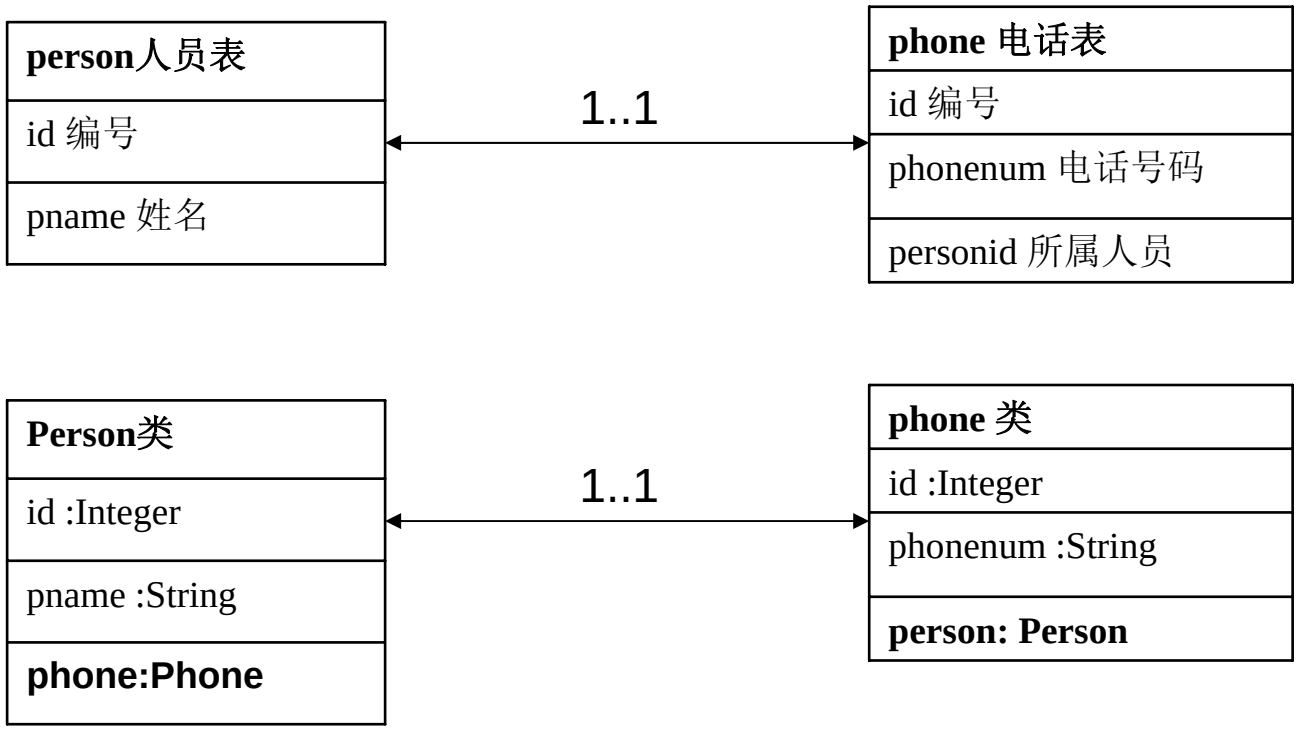
- ✓ 在关系型数据库中，一个表的一条记录仅匹配另一个表的一条数据（如果匹配多条数据那就是一对多关系了）
- ✓ 一对一关系分成两种情况
  - ➢ 基于外键的一对一关联
  - ➢ 基于主键的一对一关联

基于外键的一对一关联

- ✓ 这种映射非常简单，只要把两个实体之间建立一个many-to-one的关系，然后在many端增加一个属性unique=”true”，设置多的一端值不允许重复即可
- ✓ 如果是双向一对一关联，那么就需要在one端实体类中增加一个many端类的成员变量，在one的一方增加一个<one-to-one>标签

基于外键一对一关联示例

- ✓ 人员表(主表)和电话表（从表），一个人员对应一个电话



映射文件配置

Phone.hbm.xml

```
<hibernate-mapping>
    <class name="com.Phone" table="PHONE">
        <id name="id" type="java.lang.Integer">
            <generator class="assigned" />
        </id>
        <property name="phonenum" type="java.lang.String"/>
        <many-to-one name="person" column="personid" class="com.Person" unique="true"/>
    </class>
</hibernate-mapping>
```

Person.hbm.xml

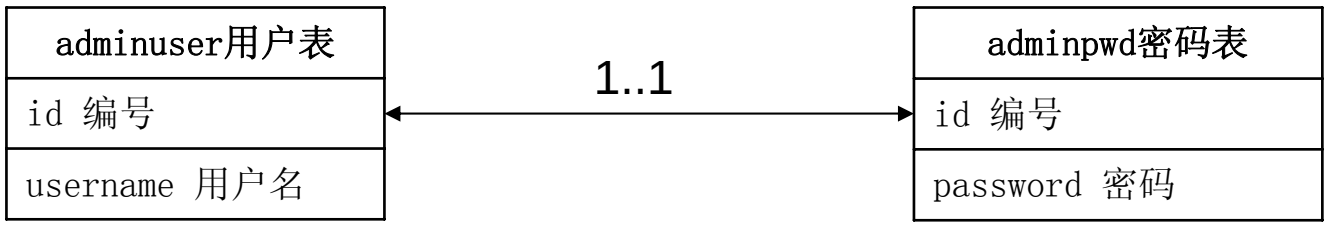
```
<hibernate-mapping>
    <class name="com.Person" table="PERSON" schema="HB2">
        <id name="id" type="java.lang.Integer">
            <generator class="assigned" />
        </id>
        <property name="pname" type="java.lang.String"/>
        <!-- 这里的name是Person类的属性名称，property-ref是Phone类的属性名称 -->
        <one-to-one name="phone" class="com.Phone" property-ref="person"/>
    </class>
</hibernate-mapping>
```

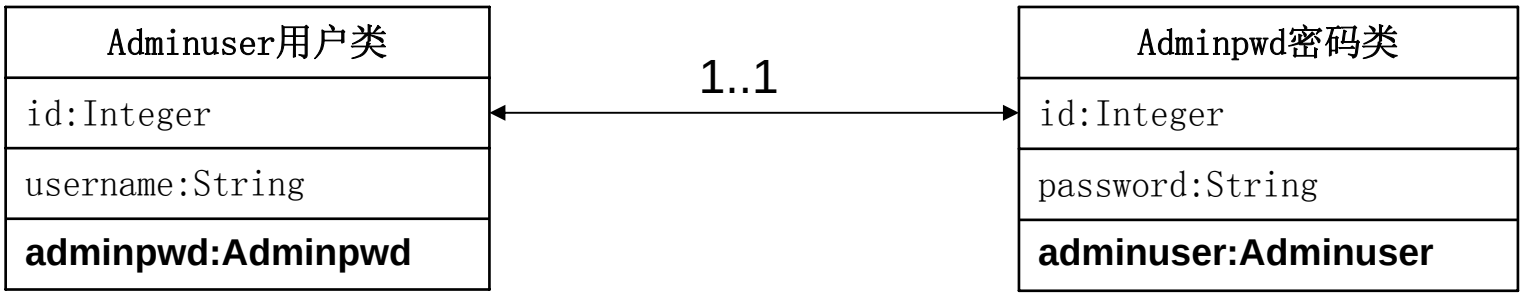
基于主键的一对一关联

- ✓ 这种映射关系对应的两个表都是只有主键没有外键，但是其中一个表的主键值是来源于另外一个表的主键值，也就是说一个表的主键值是主动生成的，但是另外一个表的主键值是引用了别的表的主键值，其实还是有主从之分的，不妨把主动生成主键的表看成主表，被动生成主键值的看成从表
- ✓ 在主表映射中正常加入<one-to-one>标签，正常选择生成主键的方式，但是从表中映射增加的<one-to-one>标签需要多出constrainted=”true”属性，并且主键生成方式只能是“foreign”，外键引用方式

基于主键的一对一关联示例

- ✓ 管理用户表和管理密码表，一个用户对应一个密码，密码表的主键值来源于用户表的主键值





用户实体映射文件配置

映射文件**Adminuser.hbm.xml**

```
<hibernate-mapping>
    <class name="com.Adminuser" table="ADMINUSER" schema="HB2">
        <id name="id" type="java.lang.Integer">
            <column name="ID" precision="6" scale="0" />
            <generator class="increment" />
        </id>
        <property name="username" type="java.lang.String">
            <column name="USERNAME" length="50" />
        </property>
            <!—加入一对一标签，在这里name表示引用密码对象的成员变量名 -->
        <one-to-one name="adminpwd"/>
    </class>
</hibernate-mapping>
```

密码实体映射文件配置

映射文件**Adminpwd.hbm.xml**

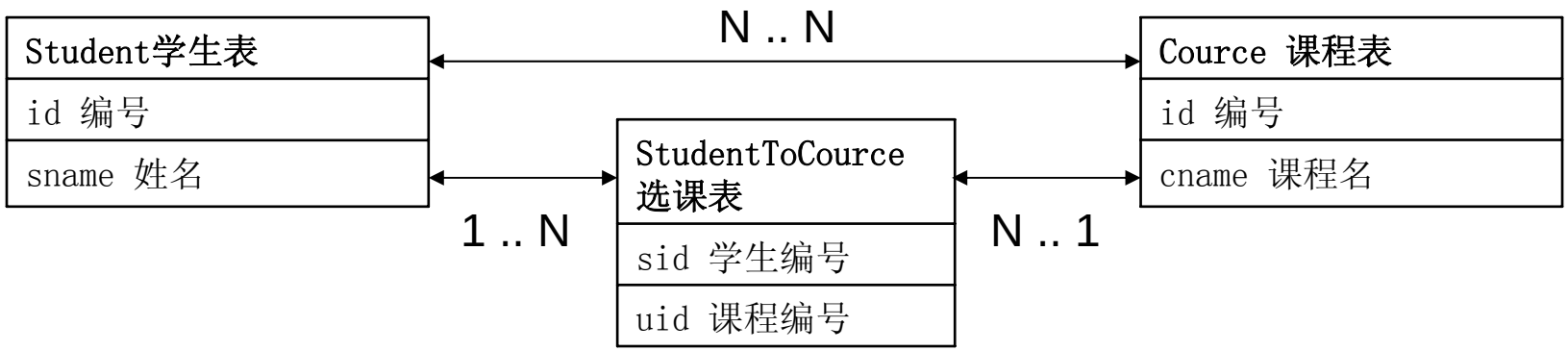
```
<hibernate-mapping>
   <class name="com.Adminpwd" table="ADMINPWD" schema="HB2">
      <id name="id" type="java.lang.Integer">
         <!—主键生成策略必须是foreign外键，表示主键值来源于引用-->
         <generator class="foreign">
            <!-- <param>标签属性名固定是property，值是类中引用变量名 -->
            <param name="property">adminuser</param>
         </generator>
      </id>
      <property name="password" type="java.lang.String"/>

     <!-- 加入一对一标签，constrained属性必须为true，表示约束主键值必须是用户表中存在的值-->
      <one-to-one name="adminuser" constrained="true"/>
   </class>
</hibernate-mapping>
```

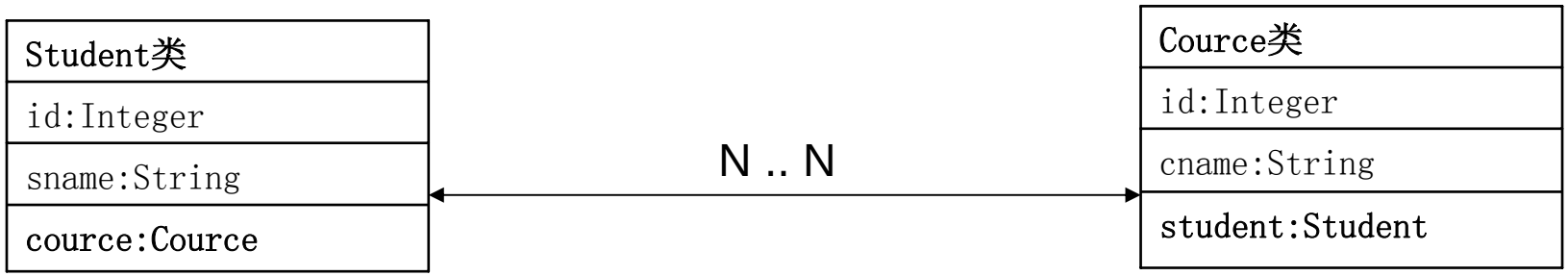
多对多关联映射

- ✓ 多对多关系在数据库中很少是两个表的直接关系，一般都是由两个一对多关系组成，例如学生表和课程表，一个学生可以选多门不同的课，但是一门课也可以被多个不同学生选择，所以学生表和课程表之间就形成了多对多的关系，多个学生对应多个课程，中间需要有一个学生选课表来衔接
- ✓ 如果是双向N-N关联，两端都需要增加一个集合对象，并且两边映射文件都需要增加一个<set>标签以及<many-to-many>标签
- ✓ 不需要创建中间表的实体类和映射文件

多对多关联映射示例

- ✓ 创建学生表，课程表，和学生选课表，学生选课表为中间表





学生实体映射文件

```
<hibernate-mapping>
    <class name="com.Student" table="student">
        <id name="id" type="java.lang.Integer">
            <generator class="increment"/>
        </id>
        <property name="sname" type="java.lang.String"/>
        <!-- 设置集合属性，table指定连接表名 -->
        <set name="cource" table="StudentToCource">
            <!-- 设置本实体在连接表中的外键列名 -->
            <key column="sid"/>
            <!-- 设置对方实体在连接表中的外键列名和对方实体类名-->
            <many-to-many column="cid" class="com.Cource"/>
        </set>
    </class>
</hibernate-mapping>
```

课程实体映射文件

```
<hibernate-mapping>
    <class name="com.Cource" table="cource">
        <id name="id" type="java.lang.Integer">
            <generator class="increment"/>
        </id>
        <property name="cname" type="java.lang.String"/>
        <!-- 设置集合属性，table指定连接表名 -->
        <set name="student" table="StudentToCource">
            <!-- 设置本实体在连接表中的外键列名 -->
            <key column="cid"/>
            <!-- 设置对方实体在连接表中的外键列名和对方实体类名-->
            <many-to-many column="sid" class="com.Student"/>
        </set>
    </class>
</hibernate-mapping>
```