Hibernate的关联映射之N-N关联映射

# 单向N-N的关联

单向的N-N关联和1-N关联的持久化类完全一样，控制关系的一端需要增加一个set集合属性，被关联的持久化实例以集合的形式存在。

N-N关联必须使用连接表，N-N关联与有连接表的1-N关联非常相似，只需要去掉<many-to-many.../>元素的unique="true"即可。其他的配置和1-N关联一样。

# 双向N-N的关联

对于双向的N-N关联，我们只需要转换为两个1-N关联模型即可。双向N-N关联两端都需要使用set集合属性，两端都增加对集合属性的访问。双向N-N同样必须使用连接表来建立两个实体之间的关联关系。

|  |
| --- |
| **--实体类**  **public** **class** Student {  **private** Integer id;  **private** String name;  **private** Set<Teacher> teachers;  //set,get，toString,无参构造器  **public** **class** Teacher {  **private** Integer id;  **private** String name;  **private** Set<Student> students;  //set,get，toString,无参构造器d |

双向N-N关联的映射文件需要使用<set.../>元素，用以映射集合属性。<set.../>属性还需要增加<key.../>子元素来映射外键列，同时还应该增加<many-to-many.../>子元素来映射关联实体类。两个映射文件如下：

|  |
| --- |
| <hibernate-mapping package="com.hibernate.domain">  <class name="Student" table="student">  <id name="id" column="student\_id">  <generator class="native" />  </id>    <property name="name" column="student\_name" />    <!-- 映射N-N关联实体，两边的table应该以样 -->  <set name="teachers" table="student\_teacher">  <!-- 映射关联的外键 列-->  <key column="student\_id" />  <!-- 映射关联类属性 -->  <many-to-many class="Teacher" column="teacher\_id" />  </set>  </class>  </hibernate-mapping> |
| <hibernate-mapping package="com.hibernate.domain">  <class name="Teacher" table="teacher">  <id name="id" column="teacher\_id">  <generator class="native" />  </id>    <property name="name" column="teacher\_name" />    <!-- 映射N-N关联实体，两边的table应该一样 -->  <set name="students" table="student\_teacher">  <!-- 映射关联的外键 列-->  <key column="teacher\_id" />  <!-- 映射关联类属性 -->  <many-to-many class="Student" column="student\_id"></many-to-many>  </set>  </class>  </hibernate-mapping>  双向N-N关联的双边都需要指定连接表的表名，外键列的列名，所以两个<set.../>元素的table属性的值必须指定且一样。<set.../>元素的两个子元素:<key../>、<many-to-many.../>都必须指定column属性。<key.../>和<many-to-many.../>分别是指定本持久化类、关联类在连接表中的外键列。因此两边的<key.../>和<many-to-many.../>的column属性的值应该是交叉相等。 |
| **public** **class** ManyToMany {  **private** SessionFactory sessionFactory;  **private** Session session;  **private** Transaction transaction;  @Before  **public** **void** init() {  //创建配置对象  Configuration config = **new** Configuration().configure();  //创建会话工厂对象  sessionFactory = config.buildSessionFactory();  //创建会话对象  session = sessionFactory.openSession();  //开启事务  transaction = session.beginTransaction();  }  @After  **public** **void** destory() {  //提交事务  transaction.commit();  //关闭会话  session.close();  //关闭会话工厂  sessionFactory.close();  }    @Test  **public** **void** many2Many(){  Teacher teacher1 = **new** Teacher();  teacher1.setName("teacher1");    Teacher teacher2 = **new** Teacher();  teacher2.setName("teacher2");    Student student1 = **new** Student();  student1.setName("student1");    Student student2 = **new** Student();  student2.setName("student2");    //建立两者之间的关系  Set<Student> ss = **new** HashSet<Student>();  ss.add(student1);  ss.add(student2);    teacher1.setStudents(ss);  teacher2.setStudents(ss);    session.save(teacher1);  session.save(teacher2);  session.save(student1);  session.save(student2);  }  } |

注意：这里只能由一边建立关联关系，就是说只能由Student对象建立与Teacher对象的关联或者由Teachert对象建立与Student对象的关联。否则将会出现主键重复错误