在本例子中 使用的多对多关系是 学生 与 课程之间的关系， 学生可以选择多门课程，一个课程也可以包含多个学生

1、多对多操作 映射配置

第一步： 创建Student 和 Course实体类

Student 和 Course 都是多对多的关系，所以，在创建的时候俩个实体类中都要加入 一个 Set 集合表示相对的实体集合

代码如下：

```
package daiwei.learning.hibernate.pojo;

import java.util.HashSet;
import java.util.Set;

/**
 * hibernate多对多练习
 * course <--> student
 * 一种课包含多个学生，一个学生可以选择多门课程
 *<p>Title:Course</p>
 *<p>description:</p>
 * @author DW
 * @date 2017年8月20日 下午3:07:35
 * @version 1.0
 */
public class Course {

    private Integer cid;
    private String cname;
    private String classroom;
    public Set<Student> students = new HashSet<>();



    public String getClassroom() {
        return classroom;
    }
    public void setClassroom(String classroom) {
        this.classroom = classroom;
    }

```

```
32
33      public Set<Student> getStudents() {
34          return students;
35      }
36      public void setStudents(Set<Student> students) {
37          this.students = students;
38      }
39      public Integer getCid() {
40          return cid;
41      }
42      public void setCid(Integer cid) {
43          this.cid = cid;
44      }
45      public String getCname() {
46          return cname;
47      }
48      public void setCname(String cname) {
49          this.cname = cname;
50      }
51
52
53  }
```

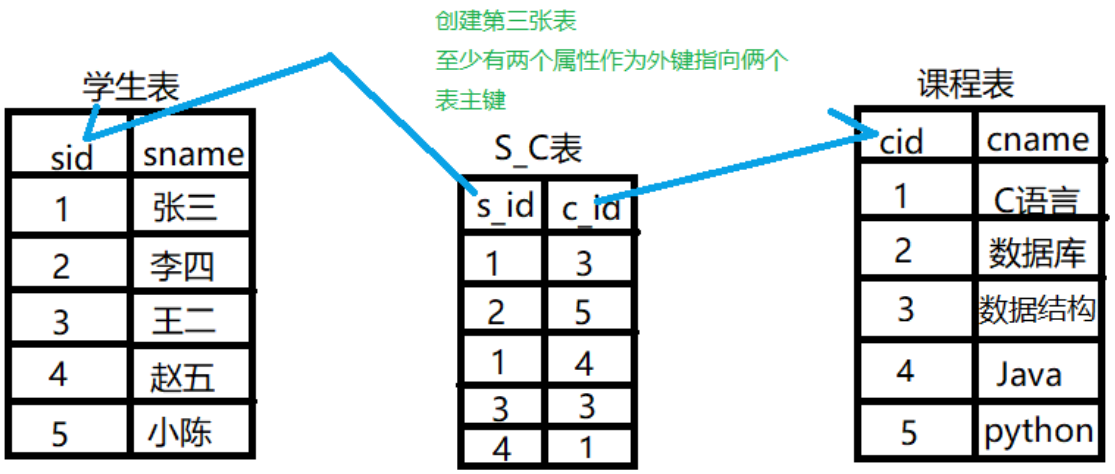
```
1   package daiwei.learning.hibernate.pojo;
2
3   import java.util.HashSet;
4   import java.util.Set;
5
6   /**
7    * hibernate多对多练习
8    * course <--> student
9    * 一种课包含多个学生，一个学生可以选择多门课程
10   *<p>Title:Course</p>
11   *<p>description:</p>
12   * @author DW
13   * @date 2017年8月20日 下午3:07:35
14   * @version 1.0
15   */
16  public class Student {
17
18      private Integer sid;
19      private String name;
20      private String gender;
```

```
    private Set<Course> courses = new HashSet<>();

    public Integer getSid() {
        return sid;
    }
    public void setSid(Integer sid) {
        this.sid = sid;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public Set<Course> getCourses() {
        return courses;
    }
    public void setCourses(Set<Course> courses) {
        this.courses = courses;
    }
}
```

第二步：创建俩个实体类相应的映射文件 Student.hbm.xml文件和 Course.hbm.xml 文件

在两个映射文件中，由于这是多对多的关系，所以俩个实体中都要包含一个 Set 标签，并且多对多的关系，是通过第三张表来维护俩个类之间的映射关系，所以在 set 标签中要添加 table 属性，即创建第三张表，来维护俩个表的关系（如图）。



映射文件参考代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<!--
    private Integer sid;
    private String name;
    private String gender;
    private Set<Course> courses = new HashSet<>(); -->
    <class name="daiwei.learning.hibernate.pojo.Student" table="tb_student">
        <id name="sid" column="sid">
            <generator class="native"></generator>
        </id>
        <property name="name" column="sname"></property>
        <property name="gender" column="gender"></property>
    <!--
        由于是多对多关系，所以在 Set 属性中要添加 table 属性，通过第三表来维护多对多的关系
            key标签： 我方在这个维护关系的第三张表的外键。
            manyToMany class ： set集合包含的元素类型，该类型在第三张表的外键。
     -->
        <set name="courses" table="course_student">
            <key column="student_id"></key>
            <many-to-many class="daiwei.learning.hibernate.pojo.Course" column="course_id"></many-to-many>
        </set>
    </class>
</hibernate-mapping>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

<!-- private Integer cid;
    private String cname;
    private String classroom;
    public Set<Student> students = new HashSet<>();
    -->
    <class name="daiwei.learning.hibernate.pojo.Course" table="tb_course">
        <id name="cid" column="cid">
            <generator class="native"></generator>
        </id>
        <property name="cname" column="cname"></property>
        <property name="classroom" column="classromm"></property>

        <set name="students" table="course_student">
            <key column="course_id"></key>
            <many-to-many class="daiwei.learning.hibernate.pojo.Student"
column="student_id"></many-to-many>
        </set>

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

Set 标签中

table属性：要创建的维护两张表关系的第三张表名

key 标签：

column 属性：第三张表 course_student 表中指定该类（正在配置的外键名）

many-to-many 标签：

class 属性：set集合中包含的类名（配置这一方对应的多方的类名）

column 属性：第三张表中 指向 Set中的类的外键名

两个配置完后，第三张表名和column要相互对应

注：在创建这几个表之前，要保证数据库中没有名字相同的表

2、多对多操作 级联保存操作

根据学生保存课程

第一步：在学生（课程）的配置文件中的 set 标签进行配置，cascade 值为 save-update
第二步：写代码实现

1、创建学生和课程对象， 把课程（学生）放到学生（课程）里， 然后保存学生（课程）就可以了。

Student.hbm.xml参考配置代码

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<!--
    private Integer sid;
    private String name;
    private String gender;
    private Set<Course> courses = new HashSet<>(); -->
    <class name="daiwei.learning.hibernate.pojo.Student" table="tb_student">
        <id name="sid" column="sid">
            <generator class="native"></generator>
        </id>
        <property name="name" column="sname"></property>
        <property name="gender" column="gender"></property>
    <!--
        由于是多对多关系，所以在 Set 属性中要添加 table 属性，通过第三表来维护多对多的关系
            key标签： 我方在这个维护关系的第三张表的外键。
            manyToMany class : set集合包含的元素类型，该类型在第三张表的外键。
     -->
        <set name="courses" table="course_student" cascade="save-update">
            <key column="student_id"></key>
            <many-to-many class="daiwei.learning.hibernate.pojo.Course" column="course_id"></many-to-many>
        </set>
    </class>
</hibernate-mapping>
```

Course.hbm.xml 参考配置代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

<!-- private Integer cid;
    private String cname;
    private String classroom;
    public Set<Student> students = new HashSet<>();
    -->
    <class name="daiwei.learning.hibernate.pojo.Course" table="tb_course">
        <id name="cid" column="cid">
            <generator class="native"></generator>
        </id>
        <property name="cname" column="cname"></property>
        <property name="classroom" column="classromm"></property>

        <set name="students" table="course_student" >
            <key column="course_id"></key>
            <many-to-many class="daiwei.learning.hibernate.pojo.Student"
column="student_id"></many-to-many>
        </set>

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

测试参考代码：

```java
@Test
    public void cascadeAdd() {

        Transaction tx = null;

        try {
            Session session = HibernateUtils.getSessionObj();
            tx = session.beginTransaction();
            tx.begin();

            Course c1 = new Course();
            c1.setCname("C语言");
            c1.setClassroom("w2301");
```

```
14
15              Course c2 = new Course();
16              c2.setCname("数据结构");
17              c2.setClassroom("w2303");
18
19              Course c3 = new Course();
20              c3.setCname("Java程序设计");
21              c3.setClassroom("w2303");
22
23
24              Student s1 = new Student();
25              s1.setName("dw");
26              s1.setGender("man");
27
28              Student s2 = new Student();
29              s2.setName("wanger");
30              s2.setGender("man");
31
32              Student s3 = new Student();
33              s3.setName("zhangsan");
34              s3.setGender("man");
35
36              s1.getCourses().add(c1);
37              s1.getCourses().add(c3);
38  //          c1.getStudents().add(s1);
39  //          c2.getStudents().add(s1);
40
41              s2.getCourses().add(c2);
42              s2.getCourses().add(c3);
43  //          c2.getStudents().add(s2);
44  //          c3.getStudents().add(s2);
45
46              s3.getCourses().add(c1);
47              s3.getCourses().add(c3);
48  //          c1.getStudents().add(s3);
49  //          c3.getStudents().add(s3);
50
51              session.save(s1);
52              session.save(s2);
53              session.save(s3);
54  //          session.save(c1);
55  //          session.save(c2);
56  //          session.save(c3);
57
58              tx.commit();
59          } catch (Exception e) {
```

```
            e.printStackTrace();
            tx.rollback();
        }
    }
}
```

3、多对多操作 级联删除操作

删除学生级联删除课程

步骤一：保留级联保存中配置，即学生中的 cascade 设置为 "save-update，delete"。

步骤二：通过session.get() 方法获得要删除的对象。

步骤三：通过调用 session.delete() 方法，删除刚查出来的对象，这样和该学生有关的 course_student 记录也会被删除。

参考测试代码：

```
    @Test
    public void casacadeDelete() {
        Transaction tx = null;
        try {
            Session session = HibernateUtils.getSessionObj();
            tx = session.beginTransaction();
            tx.begin();
            Student student = session.get(Student.class, 1);
            session.delete(student);

            tx.commit();
        } catch (Exception e) {
            tx.rollback();
            e.printStackTrace();
        }
    }
```

4、多对多操作 维护第三张表

1、让某个学生拥有某个课程（增选）

第一步：通过session.get() 方法获得有关的学生和课程对象。

第二步：通过调用student.getCourse().add(course)， 将 课程加入到学生的课程集合中即可(持久态会自动更新数据库)。

参考代码：

```
@Test
public void remianRealationADD() {
    Transaction tx = null;
    try {
        Session session = HibernateUtils.getSessionObj();
        tx = session.beginTransaction();
        tx.begin();
        Student student = session.get(Student.class, 2);
        Course course = session.get(Course.class, 2);
        student.getCourses().add(course);

        session.saveOrUpdate(student);

        tx.commit();
    } catch (Exception e) {
        tx.rollback();
    }
}
```

2、让某个学生没有某个课程（退选）

第一步：通过session.get() 方法获得有关的学生和课程对象。

第二步：调用student.getCourses().remove(course), 将某个课程移除该学生的集合中即可（持久态会自动更新数据库）

参考代码：

```
@Test
public void remianRealationREMOVE() {
    Transaction tx = null;
    try {
        Session session = HibernateUtils.getSessionObj();
        tx = session.beginTransaction();
        tx.begin();
```

```
            Student student = session.get(Student.class, 2);
            Course course = session.get(Course.class, 2);
            student.getCourses().remove(course);

            session.saveOrUpdate(student);

            tx.commit();
        } catch (Exception e) {
            tx.rollback();
        }
    }
```