**只为成功找方法，不为失败找借口！**

[**MyBatis学习总结(五)——实现关联表查询**](http://www.cnblogs.com/xdp-gacl/p/4264440.html)

**一、一对一关联**

**1.1、提出需求**

　　根据班级id查询班级信息(带老师的信息)

**1.2、创建表和数据**

　　创建一张教师表和班级表，这里我们假设一个老师只负责教一个班，那么老师和班级之间的关系就是一种一对一的关系。

[复制代码](javascript:void(0);)

1 CREATE TABLE teacher(

2 t\_id INT PRIMARY KEY AUTO\_INCREMENT,

3 t\_name VARCHAR(**20**)

4 );

5 CREATE TABLE class(

6 c\_id INT PRIMARY KEY AUTO\_INCREMENT,

7 c\_name VARCHAR(**20**),

8 teacher\_id INT

9 );

10 ALTER TABLE class ADD CONSTRAINT fk\_teacher\_id FOREIGN KEY (teacher\_id) REFERENCES teacher(t\_id);

11

12 INSERT INTO teacher(t\_name) VALUES('teacher1');

13 INSERT INTO teacher(t\_name) VALUES('teacher2');

14

15 INSERT INTO class(c\_name, teacher\_id) VALUES('class\_a', **1**);

16 INSERT INTO class(c\_name, teacher\_id) VALUES('class\_b', **2**);

[复制代码](javascript:void(0);)

　　表之间的关系如下：

**1.3、定义实体类**

　　1、Teacher类，Teacher类是teacher表对应的实体类。

[复制代码](javascript:void(0);)

1 package me.gacl.domain;

2

3 /\*\*

4 \* @author gacl

5 \* 定义teacher表对应的实体类

6 \*/

7 public class Teacher {

8

9 //定义实体类的属性，与teacher表中的字段对应

10 private int id; //id===>t\_id

11 private String name; //name===>t\_name

12

13 public int getId() {

14 return id;

15 }

16

17 public void setId(int id) {

18 this.id = id;

19 }

20

21 public String getName() {

22 return name;

23 }

24

25 public void setName(String name) {

26 this.name = name;

27 }

28

29 @Override

30 public String toString() {

31 return "Teacher [id=" + id + ", name=" + name + "]";

32 }

33 }

[复制代码](javascript:void(0);)

　　2、Classes类，Classes类是class表对应的实体类

[复制代码](javascript:void(0);)

1 package me.gacl.domain;

2

3 /\*\*

4 \* @author gacl

5 \* 定义class表对应的实体类

6 \*/

7 public class Classes {

8

9 //定义实体类的属性，与class表中的字段对应

10 private int id; //id===>c\_id

11 private String name; //name===>c\_name

12

13 /\*\*

14 \* class表中有一个teacher\_id字段，所以在Classes类中定义一个teacher属性，

15 \* 用于维护teacher和class之间的一对一关系，通过这个teacher属性就可以知道这个班级是由哪个老师负责的

16 \*/

17 private Teacher teacher;

18

19 public int getId() {

20 return id;

21 }

22

23 public void setId(int id) {

24 this.id = id;

25 }

26

27 public String getName() {

28 return name;

29 }

30

31 public void setName(String name) {

32 this.name = name;

33 }

34

35 public Teacher getTeacher() {

36 return teacher;

37 }

38

39 public void setTeacher(Teacher teacher) {

40 this.teacher = teacher;

41 }

42

43 @Override

44 public String toString() {

45 return "Classes [id=" + id + ", name=" + name + ", teacher=" + teacher+ "]";

46 }

47 }

[复制代码](javascript:void(0);)

**1.4、定义sql映射文件classMapper.xml**

[复制代码](javascript:void(0);)

1 <?xml version="1.0" encoding="UTF-8" ?>

2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

3 <!-- 为这个mapper指定一个唯一的namespace，namespace的值习惯上设置成包名+sql映射文件名，这样就能够保证namespace的值是唯一的

4 例如namespace="me.gacl.mapping.classMapper"就是me.gacl.mapping(包名)+classMapper(classMapper.xml文件去除后缀)

5 -->

6 <mapper namespace="me.gacl.mapping.classMapper">

7

8 <!--

9 根据班级id查询班级信息(带老师的信息)

10 ##1. 联表查询

11 SELECT \* FROM class c,teacher t WHERE c.teacher\_id=t.t\_id AND c.c\_id=1;

12

13 ##2. 执行两次查询

14 SELECT \* FROM class WHERE c\_id=1; //teacher\_id=1

15 SELECT \* FROM teacher WHERE t\_id=1;//使用上面得到的teacher\_id

16 -->

17

18 <!--

19 方式一：嵌套结果：使用嵌套结果映射来处理重复的联合结果的子集

20 封装联表查询的数据(去除重复的数据)

21 select \* from class c, teacher t where c.teacher\_id=t.t\_id and c.c\_id=1

22 -->

23 <select id="getClass" parameterType="int" resultMap="ClassResultMap">

24 select \* from class c, teacher t where c.teacher\_id=t.t\_id and c.c\_id=#{id}

25 </select>

26 <!-- 使用resultMap映射实体类和字段之间的一一对应关系 -->

27 <resultMap type="me.gacl.domain.Classes" id="ClassResultMap">

28 <id property="id" column="c\_id"/>

29 <result property="name" column="c\_name"/>

30 <association property="teacher" javaType="me.gacl.domain.Teacher">

31 <id property="id" column="t\_id"/>

32 <result property="name" column="t\_name"/>

33 </association>

34 </resultMap>

35

36 <!--

37 方式二：嵌套查询：通过执行另外一个SQL映射语句来返回预期的复杂类型

38 SELECT \* FROM class WHERE c\_id=1;

39 SELECT \* FROM teacher WHERE t\_id=1 //1 是上一个查询得到的teacher\_id的值

40 -->

41 <select id="getClass2" parameterType="int" resultMap="ClassResultMap2">

42 select \* from class where c\_id=#{id}

43 </select>

44 <!-- 使用resultMap映射实体类和字段之间的一一对应关系 -->

45 <resultMap type="me.gacl.domain.Classes" id="ClassResultMap2">

46 <id property="id" column="c\_id"/>

47 <result property="name" column="c\_name"/>

48 <association property="teacher" column="teacher\_id" select="getTeacher"/>

49 </resultMap>

50

51 <select id="getTeacher" parameterType="int" resultType="me.gacl.domain.Teacher">

52 SELECT t\_id id, t\_name name FROM teacher WHERE t\_id=#{id}

53 </select>

54

55 </mapper>

[复制代码](javascript:void(0);)

　　在conf.xml文件中注册classMapper.xml

<mappers>

<!-- 注册classMapper.xml文件，

classMapper.xml位于me.gacl.mapping这个包下，所以resource写成me/gacl/mapping/classMapper.xml-->

<mapper resource="me/gacl/mapping/classMapper.xml"/>

</mappers>

**1.5、编写单元测试代码**

[复制代码](javascript:void(0);)

1 package me.gacl.test;

2

3 import me.gacl.domain.Classes;

4 import me.gacl.util.MyBatisUtil;

5 import org.apache.ibatis.session.SqlSession;

6 import org.junit.Test;

7

8 public class Test3 {

9

10 @Test

11 public void testGetClass(){

12 SqlSession sqlSession = MyBatisUtil.getSqlSession();

13 /\*\*

14 \* 映射sql的标识字符串，

15 \* me.gacl.mapping.classMapper是classMapper.xml文件中mapper标签的namespace属性的值，

16 \* getClass是select标签的id属性值，通过select标签的id属性值就可以找到要执行的SQL

17 \*/

18 String statement = "me.gacl.mapping.classMapper.getClass";//映射sql的标识字符串

19 //执行查询操作，将查询结果自动封装成Classes对象返回

20 Classes clazz = sqlSession.selectOne(statement,1);//查询class表中id为1的记录

21 //使用SqlSession执行完SQL之后需要关闭SqlSession

22 sqlSession.close();

23 System.out.println(clazz);//打印结果：Classes [id=1, name=class\_a, teacher=Teacher [id=1, name=teacher1]]

24 }

25

26 @Test

27 public void testGetClass2(){

28 SqlSession sqlSession = MyBatisUtil.getSqlSession();

29 /\*\*

30 \* 映射sql的标识字符串，

31 \* me.gacl.mapping.classMapper是classMapper.xml文件中mapper标签的namespace属性的值，

32 \* getClass2是select标签的id属性值，通过select标签的id属性值就可以找到要执行的SQL

33 \*/

34 String statement = "me.gacl.mapping.classMapper.getClass2";//映射sql的标识字符串

35 //执行查询操作，将查询结果自动封装成Classes对象返回

36 Classes clazz = sqlSession.selectOne(statement,1);//查询class表中id为1的记录

37 //使用SqlSession执行完SQL之后需要关闭SqlSession

38 sqlSession.close();

39 System.out.println(clazz);//打印结果：Classes [id=1, name=class\_a, teacher=Teacher [id=1, name=teacher1]]

40 }

41 }

[复制代码](javascript:void(0);)

**1.6、MyBatis一对一关联查询总结**

　　MyBatis中使用association标签来解决一对一的关联查询，association标签可用的属性如下：

* property:对象属性的名称
* javaType:对象属性的类型
* column:所对应的外键字段名称
* select:使用另一个查询封装的结果

**二、一对多关联**

**2.1、提出需求**

　　根据classId查询对应的班级信息,包括学生,老师

**2.2、创建表和数据**

　　在上面的一对一关联查询演示中，我们已经创建了班级表和教师表，因此这里再创建一张学生表

[复制代码](javascript:void(0);)

CREATE TABLE student(

s\_id INT PRIMARY KEY AUTO\_INCREMENT,

s\_name VARCHAR(**20**),

class\_id INT

);

INSERT INTO student(s\_name, class\_id) VALUES('student\_A', **1**);

INSERT INTO student(s\_name, class\_id) VALUES('student\_B', **1**);

INSERT INTO student(s\_name, class\_id) VALUES('student\_C', **1**);

INSERT INTO student(s\_name, class\_id) VALUES('student\_D', **2**);

INSERT INTO student(s\_name, class\_id) VALUES('student\_E', **2**);

INSERT INTO student(s\_name, class\_id) VALUES('student\_F', **2**);

[复制代码](javascript:void(0);)

**2.3、定义实体类**

　　1、Student类

[复制代码](javascript:void(0);)

1 package me.gacl.domain;

2

3 /\*\*

4 \* @author gacl

5 \* 定义student表所对应的实体类

6 \*/

7 public class Student {

8

9 //定义属性，和student表中的字段对应

10 private int id; //id===>s\_id

11 private String name; //name===>s\_name

12

13 public int getId() {

14 return id;

15 }

16

17 public void setId(int id) {

18 this.id = id;

19 }

20

21 public String getName() {

22 return name;

23 }

24

25 public void setName(String name) {

26 this.name = name;

27 }

28

29 @Override

30 public String toString() {

31 return "Student [id=" + id + ", name=" + name + "]";

32 }

33 }

[复制代码](javascript:void(0);)

　　2、修改Classes类，添加一个List<Student> students属性，使用一个List<Student>集合属性表示班级拥有的学生，如下：

[复制代码](javascript:void(0);)

1 package me.gacl.domain;

2

3 import java.util.List;

4

5 /\*\*

6 \* @author gacl

7 \* 定义class表对应的实体类

8 \*/

9 public class Classes {

10

11 //定义实体类的属性，与class表中的字段对应

12 private int id; //id===>c\_id

13 private String name; //name===>c\_name

14

15 /\*\*

16 \* class表中有一个teacher\_id字段，所以在Classes类中定义一个teacher属性，

17 \* 用于维护teacher和class之间的一对一关系，通过这个teacher属性就可以知道这个班级是由哪个老师负责的

18 \*/

19 private Teacher teacher;

20 //使用一个List<Student>集合属性表示班级拥有的学生

21 private List<Student> students;

22

23 public int getId() {

24 return id;

25 }

26

27 public void setId(int id) {

28 this.id = id;

29 }

30

31 public String getName() {

32 return name;

33 }

34

35 public void setName(String name) {

36 this.name = name;

37 }

38

39 public Teacher getTeacher() {

40 return teacher;

41 }

42

43 public void setTeacher(Teacher teacher) {

44 this.teacher = teacher;

45 }

46

47 public List<Student> getStudents() {

48 return students;

49 }

50

51 public void setStudents(List<Student> students) {

52 this.students = students;

53 }

54

55 @Override

56 public String toString() {

57 return "Classes [id=" + id + ", name=" + name + ", teacher=" + teacher

58 + ", students=" + students + "]";

59 }

60 }

[复制代码](javascript:void(0);)

**2.4、修改sql映射文件classMapper.xml**

　　添加如下的SQL映射信息

[复制代码](javascript:void(0);)

1 <!--

2 根据classId查询对应的班级信息,包括学生,老师

3 -->

4 <!--

5 方式一: 嵌套结果: 使用嵌套结果映射来处理重复的联合结果的子集

6 SELECT \* FROM class c, teacher t,student s WHERE c.teacher\_id=t.t\_id AND c.C\_id=s.class\_id AND c.c\_id=1

7 -->

8 <select id="getClass3" parameterType="int" resultMap="ClassResultMap3">

9 select \* from class c, teacher t,student s where c.teacher\_id=t.t\_id and c.C\_id=s.class\_id and c.c\_id=#{id}

10 </select>

11 <resultMap type="me.gacl.domain.Classes" id="ClassResultMap3">

12 <id property="id" column="c\_id"/>

13 <result property="name" column="c\_name"/>

14 <association property="teacher" column="teacher\_id" javaType="me.gacl.domain.Teacher">

15 <id property="id" column="t\_id"/>

16 <result property="name" column="t\_name"/>

17 </association>

18 <!-- ofType指定students集合中的对象类型 -->

19 <collection property="students" ofType="me.gacl.domain.Student">

20 <id property="id" column="s\_id"/>

21 <result property="name" column="s\_name"/>

22 </collection>

23 </resultMap>

24

25 <!--

26 方式二：嵌套查询：通过执行另外一个SQL映射语句来返回预期的复杂类型

27 SELECT \* FROM class WHERE c\_id=1;

28 SELECT \* FROM teacher WHERE t\_id=1 //1 是上一个查询得到的teacher\_id的值

29 SELECT \* FROM student WHERE class\_id=1 //1是第一个查询得到的c\_id字段的值

30 -->

31 <select id="getClass4" parameterType="int" resultMap="ClassResultMap4">

32 select \* from class where c\_id=#{id}

33 </select>

34 <resultMap type="me.gacl.domain.Classes" id="ClassResultMap4">

35 <id property="id" column="c\_id"/>

36 <result property="name" column="c\_name"/>

37 <association property="teacher" column="teacher\_id" javaType="me.gacl.domain.Teacher" select="getTeacher2"></association>

38 <collection property="students" ofType="me.gacl.domain.Student" column="c\_id" select="getStudent"></collection>

39 </resultMap>

40

41 <select id="getTeacher2" parameterType="int" resultType="me.gacl.domain.Teacher">

42 SELECT t\_id id, t\_name name FROM teacher WHERE t\_id=#{id}

43 </select>

44

45 <select id="getStudent" parameterType="int" resultType="me.gacl.domain.Student">

46 SELECT s\_id id, s\_name name FROM student WHERE class\_id=#{id}

47 </select>

[复制代码](javascript:void(0);)

**2.5、编写单元测试代码**

[复制代码](javascript:void(0);)

1 package me.gacl.test;

2

3 import me.gacl.domain.Classes;

4 import me.gacl.util.MyBatisUtil;

5 import org.apache.ibatis.session.SqlSession;

6 import org.junit.Test;

7

8 public class Test4 {

9

10 @Test

11 public void testGetClass3(){

12 SqlSession sqlSession = MyBatisUtil.getSqlSession();

13 /\*\*

14 \* 映射sql的标识字符串，

15 \* me.gacl.mapping.classMapper是classMapper.xml文件中mapper标签的namespace属性的值，

16 \* getClass3是select标签的id属性值，通过select标签的id属性值就可以找到要执行的SQL

17 \*/

18 String statement = "me.gacl.mapping.classMapper.getClass3";//映射sql的标识字符串

19 //执行查询操作，将查询结果自动封装成Classes对象返回

20 Classes clazz = sqlSession.selectOne(statement,1);//查询class表中id为1的记录

21 //使用SqlSession执行完SQL之后需要关闭SqlSession

22 sqlSession.close();

23 //打印结果：Classes [id=1, name=class\_a, teacher=Teacher [id=1, name=teacher1], students=[Student [id=1, name=student\_A], Student [id=2, name=student\_B], Student [id=3, name=student\_C]]]

24 System.out.println(clazz);

25 }

26

27 @Test

28 public void testGetClass4(){

29 SqlSession sqlSession = MyBatisUtil.getSqlSession();

30 /\*\*

31 \* 映射sql的标识字符串，

32 \* me.gacl.mapping.classMapper是classMapper.xml文件中mapper标签的namespace属性的值，

33 \* getClass4是select标签的id属性值，通过select标签的id属性值就可以找到要执行的SQL

34 \*/

35 String statement = "me.gacl.mapping.classMapper.getClass4";//映射sql的标识字符串

36 //执行查询操作，将查询结果自动封装成Classes对象返回

37 Classes clazz = sqlSession.selectOne(statement,1);//查询class表中id为1的记录

38 //使用SqlSession执行完SQL之后需要关闭SqlSession

39 sqlSession.close();

40 //打印结果：Classes [id=1, name=class\_a, teacher=Teacher [id=1, name=teacher1], students=[Student [id=1, name=student\_A], Student [id=2, name=student\_B], Student [id=3, name=student\_C]]]

41 System.out.println(clazz);

42 }

43 }

[复制代码](javascript:void(0);)

**2.6、MyBatis一对多关联查询总结**

　　MyBatis中使用collection标签来解决一对多的关联查询，ofType属性指定集合中元素的对象类型。