

04-MyBatis基于XML的详细使用——高级结果映射

04-MyBatis基于XML的详细使用——高级结果映射

1、联合查询

2、嵌套结果

3、嵌套查询

4、延迟查询

5、总结

1、联合查询

emp.java

```
1 package cn.tulingxueyuan.pojo;
2
3 import java.time.LocalDate;
4
5 /**
6  * @Author 徐庶    QQ:1092002729
7  * @Slogan 致敬大师，致敬未来的你
8  */
9 public class Emp {
10     private Integer id;
11     private String username;
12     private LocalDate createDate;
13     private deptId deptId;
14
15
16     public Integer getId() {
17         return id;
18     }
19
20     public void setId(Integer id) {
21         this.id = id;
22     }
23
24     public String getUsername() {
```

```

25         return username;
26     }
27
28     public void setUsername(String username) {
29         this.username = username;
30     }
31
32     public LocalDate getCreateDate() {
33         return createDate;
34     }
35
36     public void setCreateDate(LocalDate createDate) {
37         this.createDate = createDate;
38     }
39
40     public Integer getDeptId() {
41         return dept;
42     }
43
44     public void setDeptId(Integer deptId) {
45         this.dept = dept;
46     }
47
48     @Override
49     public String toString() {
50         return "Emp{" +
51             "id=" + id +
52             ", username='" + username + '\'' +
53             ", createDate=" + createDate +
54             ", deptId=" + deptId +
55             '}';
56     }
57
58 }

```

EmpMapper.xml

```

1  <!-- 实现表联结查询的方式： 可以映射： DTO -->
2  <resultMap id="QueryEmp_Map" type="QueryEmpDTO">
3      <id column="e_id" property="id"></id>

```

```

4      <result column="user_name" property="username"></result>
5      <result column="d_id" property="deptId"></result>
6      <result column="dept_name" property="deptName"></result>
7  </resultMap>
8
9  <select id="QueryEmp" resultMap="QueryEmp_Map">
10      select t1.id as e_id,t1.user_name,t2.id as d_id,t2.dept_name from emp t1
11      INNER JOIN dept t2 on t1.dept_id=t2.id
12      where t1.id=#{id}
13  </select>
14
15
16
17
18  <!-- 实现表联结查询的方式： 可以映射map -->
19  <resultMap id="QueryEmp_Map" type="map">
20      <id column="e_id" property="id"></id>
21      <result column="user_name" property="username"></result>
22      <result column="d_id" property="deptId"></result>
23      <result column="dept_name" property="deptName"></result>
24  </resultMap>
25
26  <select id="QueryEmp" resultMap="QueryEmp_Map">
27      select t1.id as e_id,t1.user_name,t2.id as d_id,t2.dept_name from emp t1
28      INNER JOIN dept t2 on t1.dept_id=t2.id
29      where t1.id=#{id}
30  </select>

```

QueryEmpDTO

```

1  package cn.tulingxueyuan.pojo;
2
3  /**
4   * @Author 徐庶    QQ:1092002729
5   * @Slogan 致敬大师，致敬未来的你
6   */
7  public class QueryEmpDTO {
8
9      private String deptName;
10     private Integer deptId;

```

```
11     private Integer id;
12     private String username;
13
14     public String getDeptName() {
15         return deptName;
16     }
17
18     public void setDeptName(String deptName) {
19         this.deptName = deptName;
20     }
21
22     public Integer getDeptId() {
23         return deptId;
24     }
25
26     public void setDeptId(Integer deptId) {
27         this.deptId = deptId;
28     }
29
30     public Integer getId() {
31         return id;
32     }
33
34     public void setId(Integer id) {
35         this.id = id;
36     }
37
38     public String getUsername() {
39         return username;
40     }
41
42     public void setUsername(String username) {
43         this.username = username;
44     }
45
46     @Override
47     public String toString() {
48         return "QueryEmpDTO{" +
49             "deptName='" + deptName + '\'' +
```

```

50         ", deptId=" + deptId +
51         ", id=" + id +
52         ", username='" + username + '\'' +
53         '}'';
54     }
55 }

```

Test

```

1  /**
2   * 徐庶老师实际开发中的实现方式
3   */
4  @Test
5  public void test01() {
6      try(SqlSession sqlSession = sqlSessionFactory.openSession()){
7          // Mybatis在getMapper就会给我们创建jdk动态代理
8          EmpMapper mapper = sqlSession.getMapper(EmpMapper.class);
9          QueryEmpDTO dto = mapper.QueryEmp(4);
10         System.out.println(dto);
11     }
12 }

```

2、嵌套结果

2.1多对一

EmpMapper.xml

```

1
2  <!--嵌套结果 多 对 一 -->
3  <resultMap id="QueryEmp_Map2" type="Emp">
4      <id column="e_id" property="id"></id>
5      <result column="user_name" property="username"></result>
6      <!--
7      association 实现多对一中的 “一”
8      property 指定对象中的嵌套对象属性
9      -->
10     <association property="dept">
11         <id column="d_id" property="id"></id>
12         <id column="dept_name" property="deptName"></id>
13     </association>
14 </resultMap>

```

```

15
16 <select id="QueryEmp2" resultMap="QueryEmp_Map2">
17     select t1.id as e_id,t1.user_name,t2.id as d_id,t2.dept_name from emp t1
18     INNER JOIN dept t2 on t1.dept_id=t2.id
19     where t1.id=#{id}
20 </select>
21

```

2.2一对多

```

1     <!-- 嵌套结果： 一对多 查询部门及所有员工 -->
2 <resultMap id="SelectDeptAndEmpsMap" type="Dept">
3     <id column="d_id" property="id"></id>
4     <id column="dept_name" property="deptName"></id>
5     <!--
6     <collection 映射一对多中的 “多”
7         property 指定需要映射的“多”的属性，一般声明为List
8         ofType 需要指定list的类型
9     -->
10    <collection property="emps" ofType="Emp" >
11        <id column="e_id" property="id"></id>
12        <result column="user_name" property="username"></result>
13        <result column="create_date" property="createDate"></result>
14    </collection>
15 </resultMap>
16
17 <select id="SelectDeptAndEmps" resultMap="SelectDeptAndEmpsMap">
18     select t1.id as d_id,t1.dept_name,t2.id e_id,t2.user_name,t2.create_date from dept t:
19     LEFT JOIN emp t2 on t1.id=t2.dept_id
20     where t1.id=#{id}
21 </select>

```

Emp.java

```

1 package cn.tulingxueyuan.pojo;
2
3 import java.time.LocalDate;
4
5 /**
6  * @Author 徐庶    QQ:1092002729

```

```
7  * @Slogan 致敬大师，致敬未来的你
8  */
9  public class Emp {
10     private Integer id;
11     private String username;
12     private LocalDate createDate;
13     private Dept dept;
14
15
16     public Integer getId() {
17         return id;
18     }
19
20     public void setId(Integer id) {
21         this.id = id;
22     }
23
24     public String getUsername() {
25         return username;
26     }
27
28     public void setUsername(String username) {
29         this.username = username;
30     }
31
32     public LocalDate getCreateDate() {
33         return createDate;
34     }
35
36     public void setCreateDate(LocalDate createDate) {
37         this.createDate = createDate;
38     }
39
40     public Dept getDept() {
41         return dept;
42     }
43
44     public void setDept(Dept dept) {
45         this.dept = dept;
```

```

46     }
47
48     @Override
49     public String toString() {
50         return "Emp{" +
51             "id=" + id +
52             ", username='" + username + '\'' +
53             ", createDate=" + createDate +
54             ", dept=" + dept +
55             '}';
56     }
57
58 }

```

Dept.java:

```

1  package cn.tulingxueyuan.pojo;
2
3  import java.util.List;
4
5  /**
6   * @Author 徐庶    QQ:1092002729
7   * @Slogan 致敬大师，致敬未来的你
8   */
9  public class Dept {
10     private Integer id;
11     private String deptName;
12     private List<Emp> emps;
13
14     public Integer getId() {
15         return id;
16     }
17
18     public void setId(Integer id) {
19         this.id = id;
20     }
21
22     public String getDeptName() {
23         return deptName;
24     }

```



```

25
26     public void setDeptName(String deptName) {
27         this.deptName = deptName;
28     }
29
30     public List<Emp> getEmps() {
31         return emps;
32     }
33
34     public void setEmps(List<Emp> emps) {
35         this.emps = emps;
36     }
37
38
39     @Override
40     public String toString() {
41         return "Dept{" +
42             "id=" + id +
43             ", deptName='" + deptName + '\'' +
44             ", emps=" + emps +
45             '}';
46     }
47 }

```

EmpMapper.java:

```

1  package cn.tulingxueyuan.mapper;
2
3
4  import cn.tulingxueyuan.pojo.Emp;
5  import cn.tulingxueyuan.pojo.QueryEmpDTO;
6
7  import java.util.Map;
8
9  /**
10   * @Author 徐庶    QQ:1092002729
11   * @Slogan 致敬大师，致敬未来的你
12   */
13  public interface EmpMapper {

```

```

14
15     /*徐庶老师实际开发中的实现方式*/
16     QueryEmpDTO QueryEmp(Integer id);
17
18     /*实用嵌套结果实现联合查询 多对一 */
19     Emp QueryEmp2(Integer id);
20
21
22     /*实用嵌套查询实现联合查询 多对一 */
23     Emp QueryEmp3(Integer id);
24 }

```

DeptMapper.java:

```

1  package cn.tulingxueyuan.mapper;
2
3
4  import cn.tulingxueyuan.pojo.Dept;
5
6  /**
7   * @Author 徐庶    QQ:1092002729
8   * @Slogan 致敬大师，致敬未来的你
9   */
10 public interface DeptMapper {
11     //嵌套查询： 一对多    使用部门id查询员工
12     Dept SelectDeptAndEmps(Integer id);
13
14     // 嵌套查询（异步查询）： 一对多    查询部门及所有员工
15     Dept SelectDeptAndEmps2(Integer id);
16 }

```

3、嵌套查询

在上述逻辑的查询中，是由我们自己来完成sql语句的关联查询的，那么，我们能让mybatis帮我们实现自动的关联查询吗？

3.2、多对一

EmpMapper.xml:

```

1  <!-- 嵌套查询（分步查询）    多 对 一

```

```

2 联合查询和分步查询区别：      性能区别不大
3                                分部查询支持 懒加载（延迟加载）
4 需要设置懒加载，一定要使用嵌套查询的。
5 要启动懒加载可以在全局配置文件中设置 lazyLoadingEnabled=true
6 还可以单独为某个分步查询设置立即加载 <association fetchType="eager"
7  -->
8 <resultMap id="QueryEmp_Map3" type="Emp">
9     <id column="id" property="id"></id>
10    <result column="user_name" property="username"></result>
11    <!-- association 实现多对一中的 “一”
12         property 指定对象中的嵌套对象属性
13         column   指定将哪个字段传到分步查询中
14         select   指定分步查询的 命名空间+ID
15         以上3个属性是实现分步查询必须的属性
16         fetchType 可选， eager|lazy   eager立即加载   lazy跟随全局配置文件中的lazyLoadingEna
17     -->
18    <association property="dept"      column="dept_id"  select="cn.tulingxueyuan.mapper.Dept
19    </association>
20 </resultMap>
21
22 <select id="QueryEmp3" resultMap="QueryEmp_Map3">
23     select  * from emp where id=#{id}
24 </select>

```

DeptMapper.xml

```

1
2 <!-- 根据部门id查询部门-->
3 <select id="SelectDept" resultType="dept">
4     SELECT * FROM dept where id=#{id}
5 </select>

```

3.2、一对多

DeptMapper.xml

```

1
2
3 <!-- 嵌套查询（异步查询）： 一对多 查询部门及所有员工 -->
4 <resultMap id="SelectDeptAndEmpsMap2" type="Dept">

```

```

4      <collection property="emps" ofType="Emp" column="id" select="cn.tulingxueyuan.mapper
5      <id column="d_id" property="id"></id>
6      <id column="dept_name" property="deptName"></id>
7      <!--
8      <collection 映射一对多中的“多”
9          property 指定需要映射的“多”的属性，一般声明为List
10         ofType 需要指定list的类型
11         column 需要将当前查询的字段传递到异步查询的参数
12         select 指定异步查询
13     -->
14     <collection property="emps" ofType="Emp" column="id" select="cn.tulingxueyuan.mapper
15     </collection>
16 </resultMap>
17
18 <select id="SelectDeptAndEmps2" resultMap="SelectDeptAndEmpsMap2">
19     SELECT * FROM dept where id=#{id}
20 </select>

```

EmpMapper.xml

```

1
2 <!-- 根据部门id所有员工 -->
3 <select id="SelectEmpByDeptId" resultType="emp">
4     select * from emp where dept_id=#{id}
5 </select>

```

Emp.java

```

1 package cn.tulingxueyuan.pojo;
2
3 import java.time.LocalDate;
4
5 /**
6  * @Author 徐庶    QQ:1092002729
7  * @Slogan 致敬大师，致敬未来的你
8  */
9 public class Emp {

```

```
10     private Integer id;
11     private String username;
12     private LocalDate createDate;
13     private Dept dept;
14
15
16     public Integer getId() {
17         return id;
18     }
19
20     public void setId(Integer id) {
21         this.id = id;
22     }
23
24     public String getUsername() {
25         return username;
26     }
27
28     public void setUsername(String username) {
29         this.username = username;
30     }
31
32     public LocalDate getCreateDate() {
33         return createDate;
34     }
35
36     public void setCreateDate(LocalDate createDate) {
37         this.createDate = createDate;
38     }
39
40     public Dept getDept() {
41         return dept;
42     }
43
44     public void setDept(Department dept) {
45         this.dept = dept;
46     }
47
48     @Override
```

```

49     public String toString() {
50         return "Emp{" +
51             "id=" + id +
52             ", username='" + username + '\'' +
53             ", createDate=" + createDate +
54             ", dept=" + dept +
55             '}';
56     }
57
58 }

```

Dept.java:

```

1  package cn.tulingxueyuan.pojo;
2
3  import java.util.List;
4
5  /**
6   * @Author 徐庶    QQ:1092002729
7   * @Slogan 致敬大师，致敬未来的你
8   */
9  public class Dept {
10     private Integer id;
11     private String deptName;
12     private List<Emp> emps;
13
14     public Integer getId() {
15         return id;
16     }
17
18     public void setId(Integer id) {
19         this.id = id;
20     }
21
22     public String getDeptName() {
23         return deptName;
24     }
25
26     public void setDeptName(String deptName) {
27         this.deptName = deptName;

```

```

28     }
29
30     public List<Emp> getEmps() {
31         return emps;
32     }
33
34     public void setEmps(List<Emp> emps) {
35         this.emps = emps;
36     }
37
38
39     @Override
40     public String toString() {
41         return "Dept{" +
42             "id=" + id +
43             ", deptName='" + deptName + '\'' +
44             ", emps=" + emps +
45             '}';
46     }
47 }

```

EmpMapper.java:

```

1  package cn.tulingxueyuan.mapper;
2
3
4  import cn.tulingxueyuan.pojo.Emp;
5  import cn.tulingxueyuan.pojo.QueryEmpDTO;
6
7  import java.util.Map;
8
9  /**
10   * @Author 徐庶    QQ:1092002729
11   * @Slogan 致敬大师，致敬未来的你
12   */
13  public interface EmpMapper {
14
15      /*徐庶老师实际开发中的实现方式*/
16      QueryEmpDTO QueryEmp(Integer id);

```

```

17
18     /*实用嵌套结果实现联合查询 多对一 */
19     Emp QueryEmp2(Integer id);
20
21
22     /*实用嵌套查询实现联合查询 多对一 */
23     Emp QueryEmp3(Integer id);
24 }

```

DeptMapper.java:

```

1  package cn.tulingxueyuan.mapper;
2
3
4  import cn.tulingxueyuan.pojo.Dept;
5
6  /**
7   * @Author 徐庶    QQ:1092002729
8   * @Slogan 致敬大师，致敬未来的你
9   */
10 public interface DeptMapper {
11     //嵌套查询： 一对多 使用部门id查询员工
12     Dept SelectDeptAndEmps(Integer id);
13
14     // 嵌套查询（异步查询）： 一对多 查询部门及所有员工
15     Dept SelectDeptAndEmps2(Integer id);
16 }

```

4、延迟查询

当我们在进行表关联的时候，有可能在查询结果的时候不需要关联对象的属性值，那么此时可以通过延迟加载来实现功能。在全局配置文件中添加如下属性

mybatis-config.xml

```

1  <!-- 开启延迟加载，所有分步查询都是懒加载 （默认是立即加载）-->
2  <setting name="lazyLoadingEnabled" value="true"/>
3  <!-- 当开启时， 使用pojo中任意属性都会加载延迟查询 ，默认是false
4  <setting name="aggressiveLazyLoading" value="false"/>-->
5  <!-- 设置对象的哪些方法调用会加载延迟查询 默认： equals,clone,hashCode,toString-->

```



```
6 <setting name="lazyLoadTriggerMethods" value=""/>
```

如果设置了全局加载，但是希望在某一个sql语句查询的时候不使用延时策略，可以添加fetchType下属性：

```
1 <association property="dept" fetchType="eager" column="dept_id" select="cn.tulingxu
2 </association>
```

5、总结

在Java中,通过对象也可以进行关联关系描述,如图下图所示:

```
class A{
    B b;
}
```

```
class B{
    A a;
}
```

一对一

```
class A{
    List<B> b;
}
```

```
class B{
    A a;
}
```

一对多

```
class A{
    List<B> b;
}
```

```
class B{
    List<A> a;
}
```

多对多

https://blog.csdn.net/z_ssy

三种关联关系都有两种关联查询的方式，嵌套查询，嵌套结果

*Mybatis的延迟加载配置

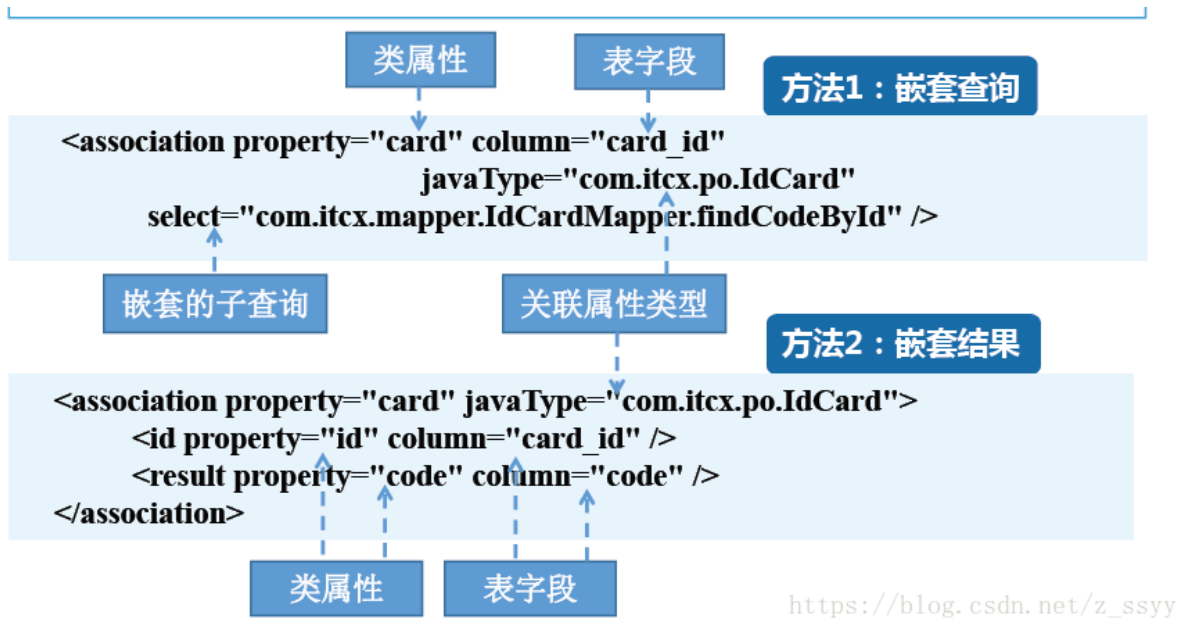
在全局配置文件中加入下面代码

```
1 <settings>
2 <setting name="lazyLoadingEnabled" value="true" />
3 <setting name="aggressiveLazyLoading" value="false"/>
4 </settings>
```

在映射文件中,<association>元素和<collection>元素中都已默认配置了延迟加载属性,即默认属性fetchType=" lazy" (属性fetchType=" eager" 表示立即加载),所以在配置文件中开启延迟加载后,无需在映射文件中再做配置

1.一对一

使用<association>元素进行一对一关联映射非常简单,只需要参考如下两种示例配置即可



2. 一对多

`<resultMap>`元素中,包含了一个`<collection>`子元素,MyBatis就是通过该元素来处理一对多关联关系的

`<collection>`子元素的属性大部分与`<association>`元素相同,但其还包含一个特殊属性`ofType`。`ofType`属性与`javaType`属性对应,它用于指定实体对象中集合类属性所包含的元素类型。

`<collection>`元素的使用也非常简单,同样可以参考如下两种示例进行配置,具体代码如下:



3. 多对多

多对多的关联关系查询,同样可以使用前面介绍的`<collection>`元素进行处理 (其用法和一对多关联关系查询语句用法基本相同)