# 04-MyBatis基于XML的详细使用——高级结果映射



## 1、联合查询

emp.java

```
package cn.tulingxueyuan.pojo;

import java.time.LocalDate;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
public class Emp {
  private Integer id;
  private String username;
  private LocalDate createDate;
  private deptId deptId;


  public Integer getId() {
  return id;
  }

  public void setId(Integer id) {
  this.id = id;
  }
```

```
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>
```

```
 <result column="user_name" property="username"></result>
 <result column="d_id" property="deptId"></result>
 <result column="dept_name" property="deptName"></result>
</resultMap>

<select id="QueryEmp" resultMap="QueryEmp_Map">
 select t1.id as e_id,t1.user_name,t2.id as d_id,t2.dept_name from emp t1
 INNER JOIN dept t2 on t1.dept_id=t2.id
 where t1.id=#{id}
</select>



<!-- 实现表联结查询的方式:  可以映射map -->
<resultMap id="QueryEmp_Map" type="map">
 <id column="e_id" property="id"></id>
 <result column="user_name" property="username"></result>
 <result column="d_id" property="deptId"></result>
 <result column="dept_name" property="deptName"></result>
</resultMap>

<select id="QueryEmp" resultMap="QueryEmp_Map">
 select t1.id as e_id,t1.user_name,t2.id as d_id,t2.dept_name from emp t1
 INNER JOIN dept t2 on t1.dept_id=t2.id
 where t1.id=#{id}
</select>
```

QueryEmpDTO

```
package cn.tulingxueyuan.pojo;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
public class QueryEmpDTO {

 private String deptName;
 private Integer deptId;
 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 + '\'' +
```

```
  '}';
  }
}
```

Test

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

## 2、嵌套结果

### 2.1多对一

EmpMapper.xml

```

<!--嵌套结果 多 对 一 -->
<resultMap id="QueryEmp_Map2" type="Emp">
 <id column="e_id" property="id"></id>
 <result column="user_name" property="username"></result>
 <!--
 association 实现多对一中的“一”
 property 指定对象中的嵌套对象属性
 -->
  <association property="dept">
 <id column="d_id" property="id"></id>
 <id column="dept_name" property="deptName"></id>
 </association>
</resultMap>

<select id="QueryEmp2" resultMap="QueryEmp_Map2">
 select t1.id as e_id,t1.user_name,t2.id as d_id,t2.dept_name from emp t
1
 INNER JOIN dept t2 on t1.dept_id=t2.id
 where t1.id=#{id}
```

```
</select>

```

## 2.2一对多

```
    <!-- 嵌套结果:  一对多 查询部门及所有员工 -->
<resultMap id="SelectDeptAndEmpsMap" type="Dept">
 <id column="d_id" property="id"></id>
 <id column="dept_name" property="deptName"></id>
 <!--
 <collection 映射一对多中的 “多”
 property 指定需要映射的“多”的属性，一般声明为List
 ofType 需要指定list的类型
 -->
  <collection property="emps" ofType="Emp" >
  <id column="e_id" property="id"></id>
  <result column="user_name" property="username"></result>
  <result column="create_date" property="createDate"></result>
  </collection>
</resultMap>

<select id="SelectDeptAndEmps" resultMap="SelectDeptAndEmpsMap">
 select t1.id as d_id,t1.dept_name,t2.id e_id,t2.user_name,t2.create_dat
e from dept t1
  LEFT JOIN emp t2 on t1.id=t2.dept_id
  where t1.id=#{id}
</select>
```

Emp.java

```
package cn.tulingxueyuan.pojo;

import java.time.LocalDate;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
public class Emp {
  private Integer id;
  private String username;
  private LocalDate createDate;
  private Dept dept;

```

```
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lazyLoad
ingEnabled
17   -->
18   <association property="dept" column="dept_id" select="cn.tulingxueyuan.
mapper.DeptMapper.SelectDept">
```

```
 </association>
</resultMap>

<select id="QueryEmp3" resultMap="QueryEmp_Map3">
 select * from emp where id=#{id}
</select>
```

DeptMapper.xml

```

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

### 3.2、一对多

DeptMapper.xml

```


<!-- 嵌套查询（异步查询）： 一对多 查询部门及所有员工 -->
<resultMap id="SelectDeptAndEmpsMap2" type="Dept">
 <id column="d_id" property="id"></id>
 <id column="dept_name" property="deptName"></id>
 <!--
 <collection 映射一对多中的 “多”
 property 指定需要映射的“多”的属性，一般声明为List
  ofType 需要指定list的类型
  column 需要将当前查询的字段传递到异步查询的参数
  select 指定异步查询
  -->
 <collection property="emps" ofType="Emp" column="id" select="cn.tulingxueyuan.mapper.EmpMapper.SelectEmpByDeptId" >
 </collection>
</resultMap>

<select id="SelectDeptAndEmps2" resultMap="SelectDeptAndEmpsMap2">
 SELECT * FROM dept where id=#{id}
</select>
```

EmpMapper.xml

```

<!-- 根据部门id所有员工 -->
<select id="SelectEmpByDeptId" resultType="emp">
```

```
 select * from emp where dept_id=#{id}
</select>
```

Emp.java

```
package cn.tulingxueyuan.pojo;

import java.time.LocalDate;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
public class Emp {
  private Integer id;
  private String username;
  private LocalDate createDate;
  private Dept dept;


  public Integer getId() {
  return id;
  }

  public void setId(Integer id) {
  this.id = id;
  }

  public String getUsername() {
  return username;
  }

  public void setUsername(String username) {
  this.username = username;
  }

  public LocalDate getCreateDate() {
  return createDate;
  }
```

```
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;
```

```
    }

    public void setId(Integer id) {
    this.id = id;
    }

    public String getDeptName() {
    return deptName;
    }

    public void setDeptName(String deptName) {
    this.deptName = deptName;
    }

    public List<Emp> getEmps() {
    return emps;
    }

    public void setEmps(List<Emp> emps) {
    this.emps = emps;
    }


    @Override
    public String toString() {
    return "Dept{" +
    "id=" + id +
    ", deptName='" + deptName + '\'' +
    ", emps=" + emps +
    '}';
    }
}
```

EmpMapper.java:

```
package cn.tulingxueyuan.mapper;


import cn.tulingxueyuan.pojo.Emp;
import cn.tulingxueyuan.pojo.QueryEmpDTO;

import java.util.Map;

```

```java
/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
public interface EmpMapper {

 /*徐庶老师实际开发中的实现方式*/
 QueryEmpDTO QueryEmp(Integer id);

 /*实用嵌套结果实现联合查询 多对一 */
 Emp QueryEmp2(Integer id);


 /*实用嵌套查询实现联合查询 多对一 */
 Emp QueryEmp3(Integer id);
}
```

DeptMapper.java:

```java
package cn.tulingxueyuan.mapper;


import cn.tulingxueyuan.pojo.Dept;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
public interface DeptMapper {
 //嵌套查询： 一对多 使用部门id查询员工
 Dept SelectDeptAndEmps(Integer id);

 // 嵌套查询（异步查询）： 一对多 查询部门及所有员工
 Dept SelectDeptAndEmps2(Integer id);
}
```

## 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

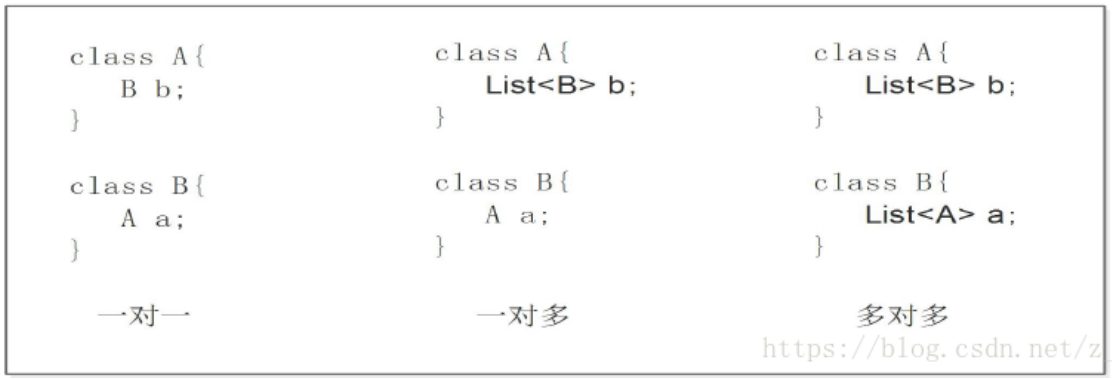
```
<!-- 开启延迟加载，所有分步查询都是懒加载 （默认是立即加载）-->
<setting name="lazyLoadingEnabled" value="true"/>
<!--当开启式， 使用pojo中任意属性都会加载延迟查询 ,默认是false
<setting name="aggressiveLazyLoading" value="false"/>-->
<!--设置对象的哪些方法调用会加载延迟查询 默认：
equals,clone,hashCode,toString-->
<setting name="lazyLoadTriggerMethods" value=""/>
```

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：

```
<association property="dept" fetchType="eager" column="dept_id"
select="cn.tulingxueyuan.mapper.DeptMapper.SelectDept">
</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：




三种关联guan关系都有两种关联查询的方式，嵌套查询，嵌套结果

*Mybatis的yanc延迟加载配置

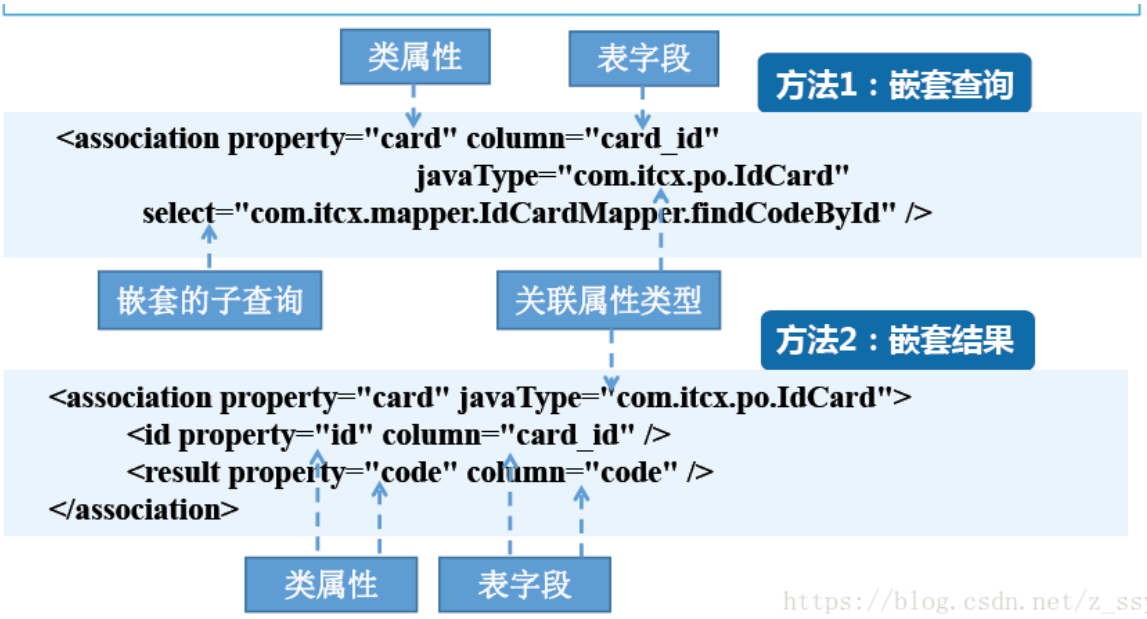
在全局配置文件中加入下面代码

```
<settings>
<setting name=”lazyLoadingEnabled” value=”true” />
<setting name=”aggressiveLazyLoading” value=”false”/>
</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）