# Select查询详解

## 返回list

先写一个方法

使用模糊查询，查询lastName的list集合员工

List<Employee> getEmpsForLastName(String LastName);

然后在sql映射文件中编写sql语句

如果要查询一个list集合，返回值类型写list集合中元素类型即可

<select id=*"getEmpsForLastName"* resultType=*"*

*entities.Employee"*>

select \* from tbl\_employee

where last\_name like #{lastName}

</select>

## 返回Map

### 返回单条记录的Map

列名为key，记录值为value

先写一个方法：

获取单条记录的Map，key对应列名，value对应记录值

Map<String, Object> getForOneMap(Integer id);

Sql映射文件中编写sql

返回值类型这里用的时Map类全类名的默认别名

<select id=*"getForOneMap"* resultType=*"map"*>

select \* from tbl\_employee where id = #{id}

</select>

### 返回多条记录的Map

Key为id，值为数据模型对象

先写一个方法

获取由lastName模糊查询的key为id，值为Employee对象的Map集合

这里需要用 **@MapKey** 注解来告诉mybatis返回的Map的key为Employee的哪个属性

@MapKey("id")

Map<Integer, Employee> getForLastNameKeyIdMap(String LastName);

在sql映射文件中编写sql语句

返回值类型写map

<select id=*"getForLastNameKeyIdMap"* resultType=*"map"*>

select \* from tbl\_employee

where last\_name like #{lastName}

</select>

## 自定义封装规则resultMap

### 简单自定义规则

在sql映射文件中，用resultMap标签来写封装规则，**支持级联属性**

resultMap 自定义返回结果与javaBean的映射规则

type： 自定义规则的JavaBean的全类名

id: 自定义规则的唯一标识

<resultMap type=*"entities.Employee"* id=*"MyEmp"*>

主键用id标签，底层会有优化 column为列名，property为属性名

<id column=*"id"* property=*"id"*/>

普通列用result标签

<result column=*"last\_name"* property=*"lastName"* />

其他不指定的列会自动封装，但我们只要写上resultMap就把所有映射规则全写上

<result column=*"gender"* property=*"gender"*/>

<result column=*"email"* property=*"email"*/>

</resultMap>

引用这个封装规则

<select id=*"getEmpsForLastName"* resultMap=*"MyEmp"*>

select \* from tbl\_employee

where last\_name like #{lastName}

</select>

**注意：使用了resultMap属性，就不能再用resultType来指定类型了**

### 关联查询规则

#### 级联属性封装结果

使用<**resultMap**>标签自定义封装规则，通过级联属性封装结果

**Sql映射文件中sql语句：**

<select id=*"getEmpAndDep"* resultMap=*"EmpAndDep"*>

SELECT e.`id`,e.`last\_name`,e.`email`,e.`gender`,

e.`department\_id`,d.`department\_name`

FROM tbl\_employee e JOIN tbl\_department d

ON e.`department\_id`=d.`id`

WHERE e.`id`=#{id}

</select>

查询结果列名：

**id last\_name email gender department\_id department\_name**

**自定义封规则，使用resultMap标签**

<resultMap type=*"entities.Employee"* id=*"EmpAndDep"*>

<id column=*"id"* property=*"id"* />

<result column=*"last\_name"* property=*"lastName"* />

<result column=*"gender"* property=*"gender"* />

<result column=*"email"* property=*"email"* />

<result column=*"department\_id"* property=*"department.id"* />

<result column=*"department\_name"* property=

*"department.departmentName"* />

</resultMap>

#### association定义关联属性的封装规则

**编写resultMap自定义规则：**

<resultMap type=*"entities.Employee"* id=*"EmpAndDep2"*>

<id column=*"id"* property=*"id"* />

<result column=*"last\_name"* property=*"lastName"* />

<result column=*"gender"* property=*"gender"* />

<result column=*"email"* property=*"email"* />

使用association标签封装关联单个对象的封装规则

property javaBean中关联的属性名

javaType 封装的类型全类名【必须指定】

<association property=*"department"* javaType=*"entities.*

*Department"*>

<id column=*"department\_id"* property=*"id"*/>

<result column=*"department\_name"* property=

*"departmentName"*/>

</association>

</resultMap>

**嵌套自定义查询封装**

#### association分步查询关联

**Sql映射文件中sql语句：**

<select id=*"getEmpAndDepStep"* resultMap=*"EmpAndDep3"*>

select \* from tbl\_employee where id = #{id}

</select>

**非常简洁**

**编写resultMap自定义规则：**

<resultMap type=*"entities.Employee"* id=*"EmpAndDep3"*>

<id column=*"id"* property=*"id"* />

<result column=*"last\_name"* property=*"lastName"* />

<result column=*"gender"* property=*"gender"* />

<result column=*"email"* property=*"email"* />

使用association分步查询，

select 表示需要调用的查询sql，命名空间.id

column 表示查询调用查询需要传入的参数

propert 表示查出的结果需要赋值的属性名

<association column=*"department\_id"*

select=*"mapper.DepartmentMapper.getDepForId"*

property=*"department"*>

</association>

</resultMap>

**规则中调用了mapper.DepartmentMapper.getDepForId方法：**

<select id=*"getDepForId"* resultMap=*" mapper.*

*DepartmentMapper "*>

select \* from tbl\_department where id=#{id}

</select>

**先使用DepartmentMapper获取department对象，在赋值到此Employee对象中，分两步走，简化sql，合理利用现有资源**

#### 延迟加载

我们在获取一个Employee对象时，并一定不需要他的部门信息，每次从数据库中获取这个对象时，都自动获取部门信息，有些浪费数据库资源。我们希望，在使用员工部门信息的时候在把部门信息查询并加载进去，按需加载，这就用到了延迟加载。

**在分布查询的基础之上，在全局配置文件中设置两个属性即可**

**全局启用或禁用延迟加载。当禁用时，所有关联对象都会即时加载**

<setting name=*"lazyLoadingEnabled"* value=*"true"*/>

**当启用时，有延迟加载属性的对象在被调用时将会完全加载任意属性。否则，每种属性将会按需要加载。**

<setting name=*"aggressiveLazyLoading"* value=*"false"*/>

#### 关联属性为集合类型

例如，在部门中有部门所有员工的信息，是一个list，这样就不能使用association来自定义集合对象的封装规则（association是封装单个对象规则的标签），我们可以使用collection来指定集合的内元素的封装规则。

**封装关联的集合属性**

<resultMap type=*"entities.Department"* id=*"MyDepPlus"*>

<id column=*"id"* property=*"id"*/>

<result column=*"department\_name"* property=*"departmentName"*/>

如果不是分布查询用这个 ofType写封装类型的全类名 ，在子标签中定义列和值的对应属性（类似于单个对象），这里省略详细步骤

<!-- <collection property="employees" ofType="entities.Employee"></collection> -->

**分布查询，select：告诉需要调用哪个方法进行查询**

**Property：需要封装到的属性名**

**Column：传入调用方法参数的属性名**

<collection property=*"employees"*

select=*"mapper.EmployeeMapper.getEmpForDepId"*

column=*"id"* >

</collection>

</resultMap>

**Sql语句映射**

<select id=*"getDepForIdPlus"* resultMap=*"MyDepPlus"*>

select \* from tbl\_department where id=#{id}

</select>

**mapper.EmployeeMapper.getEmpForDepId方法：**

<select id=*"getEmpForDepId"* resultMap=*"MyEmps"*>

select \* from tbl\_employee

where department\_id = #{dep\_id}

</select>

**支持延迟加载**

#### 补充

1. 如果分布查询调用方法需要传的值为多个

Column的值可以改成键值对

column=*"{key1=column1,key2=column2}"*

Key为方法#{}里的内容，比如#{depId},key就是depId

1. 如果全局中为延迟加载，而这个集合不想设置延迟，可以collection中设置fetchType属性

<collection fetchType=*"lazy|eager"*></collection

*延迟|即可*

#### 鉴别器

**<discriminator javaType=""></discriminator>**

鉴别器：mybatis可以使用discriminator判断某列的值，然后根据某列的值改变封装行为

封装Employee：

如果查出的是女生：就把部门信息查询出来，否则不查询；

如果是男生，把last\_name这一列的值赋值给email;

<resultMap type=*"com.atguigu.mybatis.bean.Employee"* id=*"MyEmpDis"*>

<id column=*"id"* property=*"id"*/>

<result column=*"last\_name"* property=*"lastName"*/>

<result column=*"email"* property=*"email"*/>

<result column=*"gender"* property=*"gender"*/>

**column：指定判定的列名**

**javaType：列值对应的java类型**

<discriminator javaType=*"string"* column=*"gender"*>

**女生 resultType:指定封装的结果类型；不能缺少。或resultMap**

<case value=*"0"* resultType=*"com.atguigu.mybatis.*

*bean.Employee"*>

<association property=*"dept"*

select=*"com.atguigu.mybatis.dao.DepartmentMapper.getDeptById"*

column=*"d\_id"*>

</association>

</case>

**男生 ;如果是男生，把last\_name这一列的值赋值给email;**

<case value=*"1"* resultType=*"com.atguigu.mybatis.bean.*

*Employee"*>

<id column=*"id"* property=*"id"*/>

<result column=*"last\_name"* property=*"lastName"*/>

<result column=*"last\_name"* property=*"email"*/>

<result column=*"gender"* property=*"gender"*/>

</case>

</discriminator>

</resultMap>