接着来看一下id 和result 标签包含的属性。

column ： 从数据库中得到的列名， **或者是列的别名。**

property ：映射到列结果的属性。可以映射简单的如“ username ”这样的属性，也

可以映射一些**复杂对象中的属性**， 例如“ address.street.number ”，这会通过“ ．”

方式的属性嵌套赋值。

javaType ： 一个Jav a 类的完全限定名，或一个类型别名（通过typeAlias 配置或者

默认的类型）。如果映射到一个JavaB ean, M yB at is 通常可以自动判断属性的类型。如

果映射到HashMap ，则需要明确地指定j av a Type 属性。

jdbcType ： 列对应的数据库类型。JDB C 类型仅仅需要对插入、更新、删除操作可能

为空的列进行处理。这是JDB C j dbcType 的需要，而不是MyB atis 的需要。

typeHandler ：使用这个属性可以覆盖默认的类型处理器。这个属性值是类的完全限

定名或类型别名。

接口中定义的返回值类型必须和泊位中配置的resultType 类型一致。返回值类型是由XM L 中的resul tType （或resultMap 中的type ）决

定的，不是由接口中写的返回值类型决定的

如果使用result Type 来设置返回结果的类型，需要在SQL 中为所有列

名和属性名不一致的列设置别名，通过设置别名使最终的查询结果列和result Type 指定对象

的属性名保持一致，进而实现自动映射。

property 属性或别名委和对象中属性的名字相同，但是实际匹配时， MyBatis 会先将两

者都转换为大写形式，然后再判断是否相同，即property ＝” userName ”和property=

” username ”都可以匹配到对象的userName 属性上。判断是否相同的时候要使用USERNAME,

因此在设直property 属性或别名的时候，不需要考虑大小写是否一致，但是为了便于阅读，

要尽可能按照统一的规则来设直。

由于大多数数据库设置不区分大小写，因此下画线方式的命名很常见，如user\_name 、user\_email 。在Java 中， 一般都使用驼峰式命名，如userName 、userEmail 。

因为数据库和Java 中的这两种命名方式很常见，因此MyBatis 还提供了一个全局属性

mapUnderscoreToCamelCase ，通过配置这个属性为true 可以自动将以下画线方式命名的

数据库列映射到Java 对象的驼峰式命名属性中。

<settings>

〈！－－其他面己直一〉

<setting name= ” mapUnderscoreToCamelCase ” value=” true ” />

</ settings >

多表联合查询

1，虽然这个多表关联的查询中涉及了3 个表，但是返回的结果只有sys role 表中的信息，  
 所以直接使用SysRole 作为返回值类型即可

2.查询的结果不仅要包含sys\_role 中的信息，还要包含当前用户的部分信息（不考虑嵌套的情

况），例如增加查询列u .user\_name as userName 。这时resultType 该如何设置呢？

第一种方法就是在SysRole 对象中直接添加userName 属性，

或者也可以创建一个如下所示的对象。将resultType 设置为扩展属性后的SysRoleExtend 对象

public class SysRoleExtend extends SysRole {

private String userName ;

public String getUserName() {

return userName;

public void setUserName(String userName) {

this . userName = userName ;

}

}

第二种方法，直接在SysRole 中增加SysUser 对象， 字段名为user ，增加这个字段后，修改XML

中的selectRolesByUserid 方法。。

/\*\*

角色表

\*/

public class SysRole {

//其他原有字段． ．．

private SysUser user ;

}

<**select id="selectRolesByUserId" resultType="tk.mybatis.simple.model.SysRole"**>  
 select   
 r.id,   
 r.role\_name roleName,   
 r.enabled,  
 r.create\_by createBy,  
 r.create\_time createTime,  
 u.user\_name as "user.userName",  
 u.user\_email as "user.userEmail"  
 from sys\_user u  
 inner join sys\_user\_role ur on u.id = ur.user\_id  
 inner join sys\_role r on ur.role\_id = r.id  
 where u.id = #{userId}  
</**select**>

查询列增加的两行，如下所示。

u.user\_name as "user.userName",  
u.user\_email as "user.userEmail"  
这里在设置别名的时候，使用的是“ user . 属性名”， user 是SysRole 中刚刚增加的SysUser字段

， userName 和userEmail 是SysUser 对象中的属性， 通过这种方式可以直接将值赋给

user 字段中的属性。

先看＜insert>元素，这个标签包含如下属性。

• id：命名空间中的唯一标识符，可用来代表这条语句。

parameterType ：即将传入的语句参数的完全限定类名或别名。这个属性是可边的，

因为MyBatis 可以推断出传入语句的具体参数，因此不建议配置该属性。

flushCache ：默认值为true ，任何时候只要语句被调用，都会清空一级缓存和二级

缓存。

timeout ：设置在抛出异常之前，驱动程序等待数据库返回请求结果的秒数。

stateme 口tType ：对于STATEMENT 、PREPARED 、CALLABLE, MyBatis 会分别使用

对应的Statement 、PreparedStatement 、Callable S tatement ，默认值为

PREPARED 。

useGeneratedKeys ：默认值为false 。如果设置为true, MyBatis 会使用JDB C

的getGeneratedKeys 方法来取出由数据库内部生成的主键

keyProperty: MyBatis 通过getGeneratedKeys 获取主键值后将要赋值的属性名。

如果希望得到多个数据库自动生成的列，属性值也可以是以逗号分隔的属性名称列表。

keyColumn ：仅对INSERT 和UPDATE 有用。通过生成的键值设置表中的列名， 这个

设置仅在某些数据库（如PostgreSQL ）中是必须的， 当主键列不是表中的第一列时需

要设置。如果希望得到多个生成的列，也可以是逗号分隔的属性名称列表。

databaseId ：如果配置了databaseidProvider (4.6 节有详细配置方法） ， MyBatis

会加载所有的不带databaseid 的或匹配当前databaseid 的语句。如果同时存在带

databaseid 和不带databaseid 的语句，后者会被忽略。

由于数据库区分date 、time 、datetime 类型，但是Java 中一般都使用j ava.ut 工l. Date

类型。因此为了保证数据类型的正确，需要手动指定日期类型， date 、time 、datetime 对

应的JDBC 类型分别为DATE 、TIME 、TIMESTAMP 。

<**insert id="insert"**>  
 insert into sys\_user(  
 user\_name, user\_password, user\_email,  
 user\_info, head\_img, create\_time)  
 values(  
 #{userName}, #{userPassword}, #{userEmail},  
 #{userInfo}, #{headImg, jdbcType=BLOB}, #{createTime, jdbcType=TIMESTAMP})  
</**insert**>

#{createTime, jdbcType=TIMESTAMP} 插入的create\_time 的格式是 2019-05-08 23:41:51.602(Timestamp)

#{createTime, jdbcType=DATE} 插入的create\_time 的格式是 2019-05-08(Date)