代码地址

<https://github.com/lilewa/mybatisChicken/tree/master/relation>

mapper文件定义的sql语句中使用的参数值由接口方法提供。Sql语句中使用#加{}方式引用一个参数，如#{name}。

默认地mybatis按照接口方法中参数被定义的顺序提供引用名。接口方法的第一个参数使用#{0}或#{param1}引用，第二个参数使用 #{1}或#{param2}引用

当接口方法只有1个参数时，mybatis提供了一些方便的引用参数的规则

基本类型的参数：可以使用任意的参数名引用这个方法参数，

JavaBean参数：JavaBean包含的字段使得只传入一个参数的情况下，在sql语句中可以使用多个参数值。Sql语句中可以直接使用字段名引用JavaBean的字段提供的值，例如传入如下类型的对象

**public class** SysUser {  
 **private** Long **id**;  
 **private** String **userName**;  
}

sql中可以使用的参数名：#{ **id** }或#{ **userName** }

Map参数：map类型的参数和javabean类似。Sql中通过key 来引用map中该key对应的value值。如#{ **key** }。使用map类型的好处是不用为了两三个参数创建新的javabean类

多个参数

接口方法使用多个参数时，除了使用mybatis提供的默认参数名#{0}或#{param1}，还可以在定义接口方法时使用注解，手动指定方法参数在sql语句中的名字。例如

List<SysRole> selectRolesByUserIdAndRoleEnabled(@Param(**"userId"**) Long userId, @Param(**"enabled"**) Integer enabled);

Sql中使用#{ **userId** }和#{ **enabled** }引用方法参数值

新建两个JavaBean类，使用默认名引用这两个对象参数

SysUser selectBy2JavaBean(SysUserId userId, SysUserName userName);

<**select id="selectBy2JavaBean" resultMap="userMap"**>  
 select *\** from sys\_user where id = #{param1.id} and user\_name=#{param2.userName}  
</**select**>

数据库中的表关联关系可以分为1对1，1对多，多对多

当我们在使用ORM框架时，通常将数据库表映射为javabean类。在javabean层面表示1对1关系时，我们使类A包含1个类B的对象

**public class A{**

**private** B objB;

**}**

在javabean层面表示1对多关系 时。类A包含类B的一个集合

**public class A{**

**private** List<B> listB;

**}**

在第一篇提到过，MyBatis并没有将Java 对象与数据库表关联起来，而是将Java 方法与SQL 语句关联。Mybatis不会窥探数据库的元数据，无法知道表之间的关系。因此javabean之间的关系无需和数据库表完全对应，开发人员可以根据自己的需要任意组合。

假如数据库中存在user表和country表。当程序需要1个用户的所有信息及他的国籍名时。为了方便程序实现，我们在定义user 实体类时，可以将country表，表示国籍名的列**countryname**

添加到user实体类中。

**public class** User {  
 **private** Long **id**;  
 **private** String **userName**;

private String **countryname;**

}

底层的sql语句查询需要关联这两张表作查询。

当业务发生变化，需要更多country表中的列，为了避免造成混乱。将一个country类对象定义在User实体类中，表明1对1关系是更好的做法

当列数据完全来自同一张表时，

Javabean之间的包含只是定义了关联关系。javabean的字段值需要根据它们之间的关系选择恰当的映射方式。

下面演示1对1关系的javabean及如何配置它们的映射

**1 准备数据库**

**1.1添加名为mybatis的数据库**

CREATE DATABASE mybatis DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

**1.2添加table**

CREATE TABLE `sys\_user` (  
 `id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT '用户ID',  
 `user\_name` varchar(50) DEFAULT NULL COMMENT '用户名',  
 `user\_password` varchar(50) DEFAULT NULL COMMENT '密码',  
 `user\_email` varchar(50) DEFAULT 'test@mybatis.tk' COMMENT '邮箱',  
 `user\_info` text COMMENT '简介',  
 `head\_img` blob COMMENT '头像',  
 `create\_time` datetime DEFAULT NULL COMMENT '创建时间',  
 `country\_id` int(11) DEFAULT NULL COMMENT '国籍',  
 PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO\_INCREMENT=1035 DEFAULT CHARSET=utf8 COMMENT='用户表';

**CREATE TABLE** `country` (  
 **`id` int**(11) **NOT NULL** AUTO\_INCREMENT,  
 **`countryname` varchar**(255) **DEFAULT NULL**,  
 **`countrycode` varchar**(255) **DEFAULT NULL**,  
 **PRIMARY KEY** (**`id`**)  
) ENGINE=InnoDB AUTO\_INCREMENT=6 **DEFAULT** CHARSET=utf8;

**2 创建实体类**

**public class** Country {  
 **private** Long **id**;  
 **private** String **countryname**;  
 **private** String **countrycode**;

**}**

**public class** SysUser {**private** Long **id**;**private** String **userName**;**private** String **userPassword**;**private** String **userEmail**;**private** String **userInfo**;**private byte**[] **headImg**;**private** Date **createTime**;  
 **private** Long **countryId**;  
 **private** Country **country**;

}

**3 创建接口类**

UserMapper.java

**3.1添加接口方法**

在UserMapper 接口中添加selectUserCountry 方法，此方法根据用户id查询用户信息及国籍信息

SysUser selectUserCountry(Long id);

**4 配置UserMapper接口对应的mapper文件**

**4.3 增加查询sql语句**

<**select id="selectUserCountry" resultMap="userCountryMap"**>  
 SELECT u.id,user\_name,user\_password,user\_email,user\_info,head\_img,create\_time,country\_id,  
 countryname,countrycode  
 FROM sys\_user u,country c where c.id=u.country\_id and c.id=#{id}  
</**select**>

**4.4使用<resultMap>元素配置查询结果映射规则**

<**resultMap id="userCountryMap" type="mybatisChicken.relation.model.SysUser"**>  
 <**id property="id" column="id"**/>  
 <**result property="userName" column="user\_name"**/>

。。。。。。  
 <**result property="country.countryname" column="countryname"**/>  
 <**result property="country.countrycode" column="countrycode"**/>  
</**resultMap**>

注意到，在映射**SysUser类型包含的country对象时，使用了country.countryname格式的字符串。指示将countryname列的值映射到country对象的countryname字段上**

**5 增加测试类方法**

testselectUserCountry

增加table

**CREATE TABLE** `sys\_role` (  
 **`id`** bigint(20) **NOT NULL** AUTO\_INCREMENT COMMENT **'角色ID'**,  
 **`role\_name` varchar**(50) **DEFAULT NULL** COMMENT **'角色名'**,  
 **`enabled` int**(11) **DEFAULT NULL** COMMENT **'有效标志'**,  
 **`create\_by`** bigint(20) **DEFAULT NULL** COMMENT **'创建人'**,  
 **`create\_time`** datetime **DEFAULT NULL** COMMENT **'创建时间'**,  
 **PRIMARY KEY** (**`id`**)  
) ENGINE=InnoDB AUTO\_INCREMENT=3 **DEFAULT** CHARSET=utf8 COMMENT=**'角色表'**;

创建实体类

**public class** CreateInfo {**private** String **createBy**;// *创建人* **private** Date **createTime**;// *创建时间*

}

**public class** SysRole {  
 **private** Long **id**;  
 **private** String **roleName**;  
 **private** String **createBy**;  
 **private** Date **createTime**;  
 **private** SysUser **user**;  
 **private** CreateInfo **createInfo**;

}

**列create\_by和create\_time来自于同一张表**sys\_role。我们在实体类SysRole中额外定义了一个1对1 关系CreateInfo。**createBy包含字段createBy和createTime**

**配置sql语句及resultMap映射规则**

<**select id="selectRoleById" resultMap="roleAutoMap"**>  
 select id, role\_name roleName,  
 create\_by createBy,  
 create\_time createTime  
 from sys\_role  
 where id = #{id}  
</**select**>

<**resultMap id="roleAutoMap" autoMapping="true" type="mybatisChicken.relation.model.SysRole"**>  
 <**id property="id" column="id"** />  
 <**result property="createInfo.createBy" column="createBy"**/>  
 <**result property="createInfo.createTime" column="createTime"**/>  
</**resultMap**>

此处使用了**autoMapping特性。当autoMapping特性值为true时，没有指定的列会被自动映射到同名对象字段中。打断点调试时可以看到role对象的**roleName字段已经被自动映射。而**role对象自身的createBy和createTime没有被自动映射，因为这两列已经映射到createInfo的字段上。**

**使用association元素显示配置1对1映射**

association 标签包含以下属性。

property ：对应实体类中的属性名，必填项。实体类中的实体类属性名

javaType ： 属性对应的Java 类型。

resultMap ： 可以直接使用现有的resultMap ，而不需要在这里配置。

columnPrefix ：查询列的前缀，配置前缀后，在子标签配置result 的column 时可以省略前缀。