# 参数类型

## String类型

1.1对于string类型，我们一般传 在sql中写明parameterType="string" ，例:

<select id=*"getJqBxxByZid"* parameterType=*"java.lang.String"*resultType=*"java.lang.String"*>

<select>

则在获取参数时，如果是要进行非null的判断，则不可在if后直接那变量名进行判空，因为mybatis会默认变量名为\_parameter，否则会报no getter/setter错误。正确的写法如下:

<if test=”\_parameter != null and \_paramter !=’’”>

AND position\_name like concat(‘%’,#{positionName}. ’%’)

</if>

这个地方用 \_paramter 接受String类型的参数，不能用positionName接受参数，除非在dao层传值时加上@Param注解。

就是对应的mapper中，添加这个注解，例如

List<String> findGroup(@Param(“positionName”)String positionName,@Param(“positionId”)String positionId);

## Map类型

要出入多个String类型的参数时，可以放在一个map中传入mybatis。

## List类型

若是List<String>

<foreach collection=”groupNoList” index=”index” item=”item” open=”(” separator=”,” close=”)”>

#{item}

</foreach>

若是List<Map>

<foreach collection=”groupNoList” index=”index” item=”item” open=”(” separator=”,” close=”)”>

(#{item.typeId},#{item.scope})

</foreach>

# 标签用法

## <trim>

insert into DB\_YZGL.dbo.T\_YZGL\_GJBZBD

<trim prefix="(" suffix=")" suffixOverrides="," >

<if test="cSfyx != null" >

C\_SFYX,

</if>

<trim>

<trim prefix="values (" suffix=")" suffixOverrides="," >

<trim>

Prefix:这个标签的意思是以 value( 开头

Suffix：这个标签的意思是以 ）结尾

prefixoverride="AND |OR" 去掉第一个 and 或者 or

suffixOverrides：表示去掉最后一个 ,

就是拼接为了拼接sql。

## <foreach>

select name,hobby

　　 from student where id in

<foreach item="item" index="index" collection="list" open="(" separator="," close=")">

#{item}

</foreach>

如果传入的参数为List<Map>

<if test=”groupNoList != null and groupNoList .size()>0”>

And a.group\_no in

<foreach collection= “” index=”” item=”” open=”(” separator=”,” close=”)”>

(#{item.typeId},#{item.scope})

</foreach>

</if>

* collection：collection属性的值有三个分别是list、array、map三种，分别对应的参数类型为：List、数组、map集合。
* item ：表示在迭代过程中每一个元素的别名
* index ：表示在迭代过程中每次迭代到的位置（下标）
* open ：前缀
* close ：后缀
* separator ：分隔符，表示迭代时每个元素之间以什么分隔

## <where>

<select id="getStudentListWhere" parameterType="Object" resultMap="BaseResultMap">

SELECT \* from STUDENT

WHERE

<if test="name!=null and name!='' ">

NAME LIKE CONCAT(CONCAT('%', #{name}),'%')

</if>

<if test="hobby!= null and hobby!= '' ">

AND hobby = #{hobby}

</if>

</select>

这种写法会出现的问题是 如果第一个if为空，第二个不为空，sql拼接起来出现 where and

<select id="getStudentListWhere" parameterType="Object" resultMap="BaseResultMap">

SELECT \* from STUDENT

<where>

<if test="name!=null and name!='' ">

NAME LIKE CONCAT(CONCAT('%', #{name}),'%')

</if>

<if test="hobby!= null and hobby!= '' ">

AND hobby = #{hobby}

</if>

</where>

</select>

Where标签作用：第一，如果他的子标签中有返回值的话，他就会自动拼接一个where。第二，如果他的子标签返回的内容是以AND或OR开头的，就会剔除掉。

## <set>

<update id="updateStudent" parameterType="Object">

UPDATE STUDENT SET

<if test="name!=null and name!='' ">

NAME = #{name},

</if>

<if test="hobby!=null and hobby!='' ">

MAJOR = #{major},

</if>

<if test="hobby!=null and hobby!='' ">

HOBBY = #{hobby}

</if>

WHERE ID = #{id};

</update>

如果最后一个if没有执行，结果就是最后多一个逗号，使用set标签有两个作用：1.动态添加set关键字，2.剔除追加到条件末尾的任何不相关的逗号。

# 别名

Mybatis中支持别名

1. parameterType中的java.lang.String,可以由string代替
2. Javabean也可以起一个别名

<typeAliases>标签中的<typeAlias>标签和<package>标签。

# #和$的区别

#{}是实现的是向prepareStatement中的预处理语句中设置参数值，sql语句中#{}表示一个占位符。将参数的内容添加到sql语句中指定的位置。

#{}读取实体类对象属性的内容。

#{}读取map集合中关键字的值。

#{}采用预编译方式，可以防止sql注入。

${}采用直接赋值的方法，无法阻止sql注入攻击。

如果传入的值：name= “zhangsan”;

两种方式解析之后的结果都是select \* from user where name = 'zhangsan';

但是在预编译的处理是不一样的，#{}会把参数部分用一个占位符？代替：

select \* from user where name = ?;

${}只是简单的字符串拼接，在动态解析阶段解析为：

select \* from user where name = 'zhangsan';

但是如果需要动态的指定查询中的排序字段。此时只能使用${}。