**Mybatis笔记 - SQL标签方法**

      Mpper.xml映射文件中定义了操作数据库的sql，并且提供了各种标签方法实现动态拼接sql。每个sql是一个statement，映射文件是mybatis的核心。

**一、内容标签**

**1、NamePlace**

       NamePlace命名空间作用就是对sql进行**分类化管理**。若使用Dao开发方式，映射文件的nameplace可以任意命名；但如果采用的是Mapper接口代理的方式开发，Mapper的映射文件中namespace必须为接口的全名。

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="Mapper.EmpMapper">

//CURD操作标签

//if片段

</mapper>

**2、CRUD标签**

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="Mapper.EmpMapper">

<!-- 查询 -->

<select id="" parameterType="" resultType=""></select>

<!-- 添加 -->

<insert id="" parameterType=""></insert>

<!-- 删除 -->

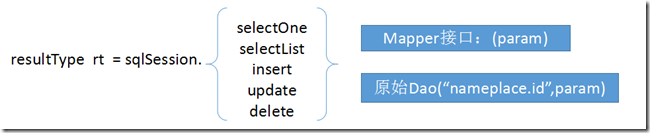
<delete id="" parameterType=""></delete>

<!-- 更新 -->

<update id="" parameterType=""></update>

</mapper>

**3、标签调用方法**

[](http://images2017.cnblogs.com/blog/1246180/201801/1246180-20180130140427406-531329286.png)

（1）selectOne与selectList方法

        selectOne表示查询出一条结果集进行映射，使用selectOne查询多条记录会抛出异常。selectList表示查询出一个列表（多条记录）进行映射，对于使用selectOne可以实现的查询，使用selectList必然也可以实现（list中只有一个对象）。

（3）代理对象内部调用

       动态代理对象调用sqlSession.selectOne()和sqlSession.selectList()是根据mapper接口方法的返回值决定，如果mapper方法返回单个pojo对象（非集合对象），代理对象内部通过selectOne查询数据库。如果mapper方法返回集合对象，代理对象内部通过selectList查询数据库。

**二、动态SQL标签**

**1、if标签**

//进行空字符串校验

<select id="findUserList" parameterType="user" resultType="user">

select \* from user where 1=1

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

and id=#{id}

</if>

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

and username like '%${username}%'

</if>

</select>

**2、where标签**

//<where/>可以自动处理第一个and

<select id="findUserList" parameterType="user" resultType="user">

select \* from user

<where>

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

and id=#{id}

</if>

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

and username like '%${username}%'

</if>

</where>

</select>

**3、sql片段**

       Sql中可将重复的sql提取出来，使用时用include引用即可，最终达到sql重用的目的，如下：

//建立sql片段

<sql id="query\_user\_where">

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

and id=#{id}

</if>

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

and username like '%${username}%'

</if>

</sql>

//使用include引用sql片段

<select id="findUserList" parameterType="user" resultType="user">

select \* from user

<where>

<include refid="query\_user\_where"/>

</where>

</select>

//引用其它mapper.xml的sql片段

<include refid="namespace.sql片段"/>

**三、foreach标签**

**1、通过pojo类传递list**

        向sql传递数组或List，mybatis使用foreach解析，foreach参数定义如下：collection指定输入 对象中集合属性，  item每个遍历生成对象中，open开始遍历时拼接的串，close结束遍历时拼接的串，separator：遍历的两个对象中需要拼接的串。

（1）sql语句

SELECT \* FROM USERS WHERE username LIKE '%张%' AND (id =10 OR id =89 OR id=16)

SELECT \* FROM USERS WHERE username LIKE '%张%' id IN (10,89,16)

（2）vo类

public class QueryVo{

private User user;

private UserCustom userCustom;

//传递多个用户id

private List<Integer> ids;

set()/get() ...

}

（3）映射文件

<select id="findUserList" parameterType="UserQueryVo" resultType="UserCustom">  
     SELECT \* FROM USER  
     <where>  
        <!-- 使用实现下边的sql拼接： AND (id=1 OR id=10 OR id=16) -->

        <if test="ids!=null and ids.size>0">

           <foreach collection="ids" item="user\_id" open="AND (" close=")" separator="or">

                  id=#{user\_id}

           </foreach>

       </if>  
     </where>   
</select>

<!-- 使用实现下边的sql拼接： and id IN(1,10,16)—>

<foreach collection="ids" item="user\_id" open="and id IN(" close=")" separator=",">

     #{user\_id}

</foreach>

（4）测试代码

List<Integer> ids = new ArrayList<Integer>();

ids.add(1);//查询id为1的用户

ids.add(10); //查询id为10的用户

queryVo.setIds(ids);

List<User> list = userMapper.findUserList(queryVo);

**2、传递单个list**

（1）Mapper映射文件

<select id="selectUserByList" parameterType="java.util.List" resultType="user">

select \* from user

<where>

<!-- 传递List，List中是pojo -->

<if test="list!=null">

<foreach collection="list" item="item" open="and id in( "separator="," close=")">

**#{item.id}**

</foreach>

</if>

</where>

</select>

（2）Mapper接口

public List<User> selectUserByList(List userlist);

（3）测试程序

//构造查询条件List

List<User> userlist = new ArrayList<User>();

User user = new User();

user.setId(1);

userlist.add(user);

user = new User();

user.setId(2);

userlist.add(user);

//传递userlist列表查询用户列表

List<User>list = userMapper.selectUserByList(userlist);

**3、传递pojo类数组**

（1）Mapper映射文件

      参数含义：index为数组的下标，item为数组每个元素的名称，名称随意定义，open循环开始,close循环结束，separator中间分隔输出。

<select id="selectUserByArray" parameterType="Object[]" resultType="user">

select \* from user

<where>

<!-- 传递pojo类数组 -->

<if test="array!=null">

<foreach collection="array" index="index" item="item"

open="and id in("separator=","close=")">

**#{item.id}**

</foreach>

</if>

</where>

</select>

（2）Mapper接口

public List<User> selectUserByArray(Object[] userlist)

（3）测试程序

//构造查询条件List

Object[] userlist = new Object[2];

User user = new User();

user.setId(1);

userlist[0]=user;

user = new User();

user.setId(2);

userlist[1]=user;

//传递user对象查询用户列表

List<User>list = userMapper.selectUserByArray(userlist);

**4、传递字符串类数组**

（1）Mapper映射文件

<select id="selectUserByArray" parameterType="Object[]" resultType="user">

select \* from user

<where>

<!-- 传递字符串数组 -->

<if test="array!=null">

<foreach collection="array"index="index"item="item"

open="and id in("separator=","close=")">

**#{item}**

</foreach>

</if>

</where>

</select>

        如果数组中是简单类型则写为#{item}，不用再通过ognl获取对象属性值了。

（2）Mapper接口

public List<User> selectUserByArray(Object[] userlist)

（3）测试程序

//构造查询条件List

Object[] userlist = new Object[2];

userlist[0]=”1”;

userlist[1]=”2”;

//传递user对象查询用户列表

List<User>list = userMapper.selectUserByArray(userlist);

分类: [Spring+Mybatis](https://www.cnblogs.com/sh086/category/1206195.html" \t "_blank)