XML

xml基本结构

xml文件的基本结构　前端请求的msgid　与下面msg标签对应的id 一致　请求接口发送是消息id与之对应

```xml
<?xml version="1.0" encoding="utf-8"?>
<Root>
    <msgs>
        <msg id="" type=" " v=" " perms="">
        </msg>
    </msgs>
</Root>
```

参数解读

1. id　消息id　　也就是前端 $sm(callback,["w_user.editpwd"])
2. type 消息类型　　上面的示例中　类型（type类型）
3. v　消息内容
4. d：数据库类型 默认缺省为potgres数据库
5. did：对应的数据库

> 其中 v、d和did的值允许接收前台传递参数 例如示例中#0#,#1#,#2# 对应sm的数组的第二,第三,第四个参数

管理系统要进行crud　那现有的消息类型 type如下:

注意：这里的**type**　就基本决定了**v**（消息内容）如何写

type类型

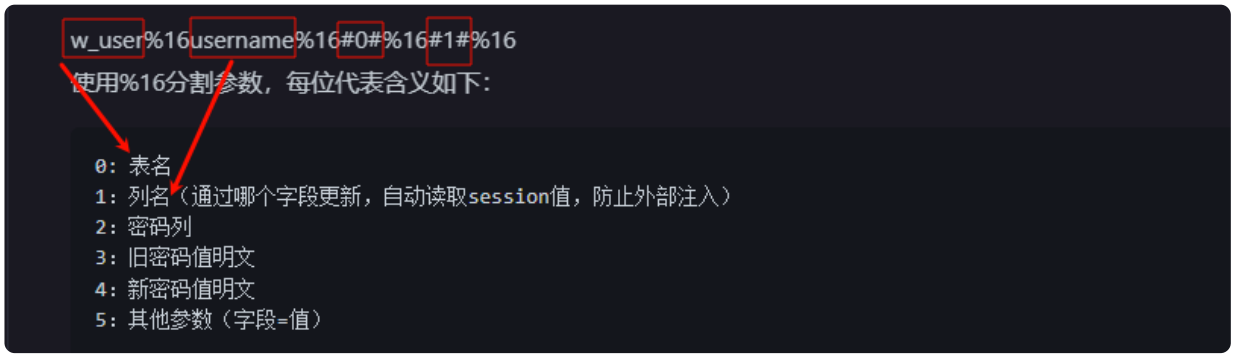
1、alterPwd (修改密码规则)

```xml
<msg id="w_user.editpwd" type="alterpwd"
v="w_user%16username%16#0#%16#1#%16">
</msg>
```

- **参数v 解读**

w_user%16username%16#0#%16#1#%16

这是使用了 %16作为分割符号 用于分割参数



2、spell (全拼简拼规则)

```xml
<msg id="common.spell" type="spell" v="#0#%16#1#">
</msg>
```

- **参数v 解读**

#0#%16#1#
使用%16分割参数，每位代表含义如下：

```
0： 字符串
1：模式：0只取第一项，1返回所有拼音组合情况
```

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["common.spell", "张三", 0]);
```

- **返回数组**

3、update

针对单表更新规则，为update语句，支持where标签传参；

```xml
<msg id="w_group.update" type="update" v="update w_group set
gname='#0#',sp='#1#',spe='#2#',description='#3#' where id=#4# #5#">
    <where idx="5">
        <p> and uid={0}</p>
    </where>
</msg>
```

- **参数v 解读**

update sql语句

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["w_group.update", 1, 2, 3, 4, 5, $.msgwhere()]);
```

- **返回受影响行数**

4、updatejson（推荐使用）

针对单表更新规则，按照前端传参自动拼接语句，用此规则，一个表只需一个更新消息即可满足不同的更新需求；只修改变化字段，减小服务器压力；

```xml
```

```
<msg id="w_user.update" type="updatejson" v="w_user%16#0#%16#1#">
    <!--此处idx固定为columns-->
    <where idx="columns">
        <p key="name" format="string">truename</p>
        <p key="age" format="decimal">age</p>
        <p key="birth" format="date">birthday</p>
        <p key="birth" format="timestamp">birthday</p>
        <p key="sex" format="string"/>
    </where>
    <where idx="1">
        <p key="id">id={0}</p>
        <p key="uname">uname='{0}'</p>
    </where>
</msg>
```

> 注意p标签内容（value）为字段名，如果不为空用此值，如不传使用key作为字段名；
> json中避免不同字段key和value重名，format为int、float、decimal、boolean、string、date、time、timestamp（如有其他类型可联系新增）
>
> 1.日期类型如果想用数据库默认值，字段类型为date或timestamp、time，前端传入固定值 "now"
> 2.保存密码值，加format="encrypt" 自动加密保存

参数v 解读

按%16分割后p，有三位，缺一不可：

> p[0]: (必选)表名
> p[1]: (必选)字段参数json字符串
> p[2]: (必选)查询条件，按照此条件更新；

前端调用说明

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["w_user.update", JSON.stringify({
    name: "张三",
    age: "5.06",
    birth: "2019-02-16",
    sex: "男"
}), $.msgwhere({id: [Arg("id") || 0]})]);
```

5、insert

针对表单插入规则，按照前端传参数自动拼接语句，支持where标签传参

```xml
<msg id="w_group.adduser" type="insert"
v="w_groupuser%16groupid,userid,isdel%16#0#,#1#,0%16groupid=#0#
#2#">
    <where idx="2">
        <p> and uname='{0}'</p>
    </where>
</msg>
```

参数v 解读

按%16分割后p，有三位 或 四位：

> p[0]: (必选)表名
> p[1]: (必选)插入的字段
> p[2]: (必选)插入的字段对应的值
> p[3]: (可选)插入之前的判断条件，不传为insert；传了先按条件查询，查到update,未查到insert;

前端调用说明

```js
$.sm(function (re, err) {    alert(re, err); }, ["w_group.adduser",
1, 2, $.msgwhere()]);
```

- **执行成功返回id**

6、insertjson（推荐使用）

针对单表插入规则，按照前端传参自动拼接语句，用此规则，一个表只需一个消息即可满足不同的插入需求；只修改变化字段，减小服务器压力；

```xml
<msg id="w_user.insert" type="insertjson" v="w_user%16#0#%16#1#">
    <!--此处idx固定为columns-->
    <where idx="columns">
        <p key="name" format="string">truename</p>
        <p key="age" format="decimal">age</p>
        <p key="birth" format="date">birthday</p>
        <p key="birth" format="timestamp">birthday</p>
        <p key="sex" format="string"/>
    </where>
    <where idx="1">
        <p key="id">id={0}</p>
    </where>
</msg>
```

> 注意p标签内容（value）为字段名，如果不为空用此值，如不传使用key作为字段名；json中避免不同字段key和value重名，format为int、float、decimal、boolean、string、date、time、timestamp（如有其他类型可联系新增）
>
> 1.日期类型如果想用数据库默认值，字段类型为date或timestamp、time，前端传入固定值 "now"
> 2.保存密码值，加**format="encrypt"** 自动加密保存

- **参数v 解读**

按%16分割后p，有二位 或 三位：

> p[0]: (必选)表名
> p[1]: (必选)字段参数json字符串
> p[2]: (可选)插入之前的判断条件，不传为insert；传了先按条件查询，查到update,未查到insert;

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["w_user.insert", JSON.stringify({
    name: "张三",
    age: "5.06",
    birth: "2019-02-16",
    sex: "男"
}), $.msgwhere({id: [Arg("id") || 0]})]);
```

7、insertselect

查询插入规则，支持where标签传参；

```xml
<msg id="w_group.adduser" type="insertselect"
v="w_groupuser%16groupid,userid,isdel%16select groupid,userid,0 from
w_user where isdel=0 #0#">
    <where idx="0">
        <p> and uname='{0}'</p>
    </where>
</msg>
```

参数v 解读

按%16分割后p，有三位 或 四位：

> p[0]: (必选)表名
> p[1]: (必选)插入的字段
> p[2]: (必选)查询插入表数据条件；

前端调用说明

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["w_group.adduser", $.msgwhere()]);
```

- **执行成功返回id**

8、select

查询消息，格式为sql查询语句，支持where标签传参

```
<msg id="w_user.select" type="select" v="select
id,truename,age,sex,birthday from w_user where isdel=0 and sex='#0#'
#1#">
    <where idx="1">
        <p> and id={0}</p>
    </where>
</msg>
```

- **参数v 解读**

select sql语句

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["w_user.select", "女", $.msgwhere()]);
```

- **返回二维数组**

```
[
   [1,"张三", 5.01, "女", "2022-02-16"],
   [2,"李四", 5.01, "男", "2022-02-16"],
   [3,"王五", 5.01, "女", "2022-02-16"],
   ......
]
```

9、selectjson

查询消息，格式为sql查询语句，支持where标签传参

```xml
<msg id="w_user.select" type="selectjson" v="select
id,truename,age,sex,birthday from w_user where isdel=0 and sex='#0#'
#1#">
    <where idx="1">
        <p> and id={0}</p>
    </where>
</msg>
```

- **参数v 解读**

select sql语句

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["w_user.select", "女", $.msgwhere()]);
```

- **返回一维数组（对象形式）**

```
[
    {       id: 1,
            truename: "张三",
           age: 5.01,
            sex: "女",
           birthday: "2022-02-16"
    },
    ......
]
```

10、selectonejson

查询消息，格式为sql查询语句，支持where标签传参

```xml
<msg id="w_user.select" type="selectonejson" v="select
id,truename,age,sex,birthday from w_user where isdel=0 and sex='#0#'
#1#">
    <where idx="1">
        <p> and id={0}</p>
    </where>
</msg>
```

- **参数v 解读**

select sql语句

- **前端调用说明**

```js
$.sm(function (re, err, obj) {
    alert(obj.id);
}, ["w_user.select", "女", $.msgwhere()]);
```

- **返回对象 前端接收 使用obj参数(objData)**

```json
{
    id: 1,
    truename: "张三",
    age: 5.01,
    sex: "女",
    birthday: "2022-02-16"
}
```

11、layuitable

针对**layui**表格插件生成消息规则，支持where标签传参；

```xml
<msg id="workcheck.list" type="layuitable"
v="c.id,w.truename,w.cardnumber,to_char(checktime, 'YYYY-MM-DD') as
checktime,to_char(jhchecktime, 'YYYY-MM-DD') as
jhchecktime,wid,ckresult,ispubliccard,cardpath%16tb_workercheck c
left join tb_employee w on c.wid=w.id%16c.isdel=0 #1#">
    <where idx="1"><!--对应消息中#1#-->
        <p key="key1">and isdel=0</p>
    </where>
    <where idx="laywhere"><!--laywhere固定值对应外部where: {swhere:
getwhere()}-->
        <p key="key1">and isdel=0 and id={0}</p>
    </where>
    <where idx="orderwhere"><!--orderwhere固定值对应外部fields: -->
        <p key="key1">id</p>
    </where>
</msg>
```

参数v 解读

按%16分割后p，有二位 或 三位：

> p[0]: 字段
> p[1]: 表
> p[2]: 查询条件
> p[4]: groupby

> laywhere固定值对应外部where: {swhere: getwhere()}

> 新增参数d类型，当d="crossdb"时，支持跨库查询（前提多个库表结构一样），消息d,did参数采用crossdb进行数据库连接，did外部传入格式：
> JSON.stringify([
> ["f", "fuyou_110101_dfdsfsfsfsfsfsfsfsf"],//连接类型、guid
> ["f", "fuyou_110102_dfdsfsfsfsfsfsfsfsf"]
>]);
>
> url: encodeURI(.smurl+"?"+.getSmStr(["zhikonglist.table",
> JSON.stringify(getcrossdb())])),
> 返回数据中会自动增加dbtype，dbguid两个字段，方便定位查看。

前端调用说明

```js
url: encodeURI($.smurl + "?" + $.getSmStr(["workcheck.list"])),
```

12、flexgrid

针对**Flexigrid**表格插件生成消息规则，支持where标签传参；

```xml
<msg id="alleadmin.gridlist" type="flexigrid"
v="c.id,w.truename,w.cardnumber,to_char(checktime, 'YYYY-MM-DD') as
checktime,to_char(jhchecktime, 'YYYY-MM-DD') as
jhchecktime,wid,ckresult,ispubliccard,cardpath%16tb_workercheck c
left join tb_employee w on c.wid=w.id%16c.isdel=0 #1#">
    <where idx="1"><!--对应消息中#1#-->
        <p key="key1">and isdel=0</p>
    </where>
    <where idx="gridwhere"><!--laywhere固定值对应外部where: {swhere:
getwhere()}-->
        <p key="key1">and isdel=0 and id={0}</p>
    </where>
    <where idx="orderwhere"><!--orderwhere固定值对应外部fields: -->
        <p key="key1">id</p>
    </where>
</msg>
```

参数v 解读

按%16分割后p，有二位 或 三位：

> p[0]: 字段
> p[1]: 表
> p[2]: 查询条件
> p[4]: groupby

> gridwhere固定值对应外部where: {swhere: getwhere()}

- **前端调用说明**

```js
msg: {
    pm: ["alleadmin.gridlist", parent.tongsetting.cursysdate]
},
```

13、time

针对单表插入规则，按照前端传参自动拼接语句，用此规则，一个表只需一个消息即可满足不同的插入需求；

```xml
<msg id="common.getdate" type="time" v=""/>
```

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["common.getdate"]);
```

14、myrule（复杂业务场景）

扩展规则，对于复杂页面，普通消息满足不了实现要求，特开放接口，后台实现方法

```xml
<msg id="wxyey.zajitasksetup" type="myrule"
v="#0#%16#1#%16#2#%16#3#"
classMethod="com.btkj.tbyypt.atte.AtteRule.taskSetup"/>
```

```java
/**
 * 对应类要实现Rule接口
 */
public class AtteRule implements Rule {
    /**
```

```
     * 对应方法，需要有示例参数
     * @param p
     * @param cp
     * @param request
     * @return
     */
    public String taskSetup(String[] p,String[] cp,
HttpServletRequest request) {
        String finaltask = "";
        return "{\"re\":"+finaltask+"}";
    }
```

- **参数v 解读**

按%16分割后p：

> 多位参数，后台按需使用

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["wxyey.zajitasksetup", 1, 2, 3, 4]);
```

15、SqlToExcel规则

导出Excel消息，拼接sql消息，支持where标签传参

```xml
<msg id="w_user.select" type="sqltoexcel"
v="userfiles/enrolllottery%16select er.oncerandom,name,idcard,age
from tb_enrolllottery_baby eb left join tb_enrolllottery_random er
on eb.lotteryid=er.lotteryid and eb.id=er.lotterybabyid where
eb.isdel=0 and eb.lotteryid=#0# #1# order by
er.creatime,eb.id%16#2#%16序号,名称,身份证号码,年龄:double:0.0">
     <where idx="1">
         <p key="yes"> and lotterybabyid is not null</p>
         <p key="no"> and lotterybabyid is null</p>
    </where>
    <where idx="2">
         <p key="yes">中号名单</p>
         <p key="no">未中号名单</p>
         <p key="all">全部摇号名单</p>
    </where>
</msg>
```

- **参数v 解读**

> 使用%16分割参数，每位代表含义如下：
> 1.文件路径
> 2.查询语句；支持多sheet，每列对应excel中列，多个sheet语句用;分隔
> 3.sheet名称；支持多sheet，多个sheet用;分隔
> 4.列名，对应2中列数目；支持设置字段格式用:分隔，默认String类型，支持double、date类型，后面可设置format格式；支持多sheet,使用;分隔

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, ["w_user.select", "女", $.msgwhere()]);
```

- **返回二维数组**

```
[
    [1,"张三", 5.01, "女", "2022-02-16"],
    [2,"李四", 5.01, "男", "2022-02-16"],
    [3,"王五", 5.01, "女", "2022-02-16"],
    ......
]
```

16、batch 批量消息

```xml
<msg id="common.batch" type="batch" title="" perms="sys:user:list">
    <include id="w_user.select"/>
    <include id="w_role.select"/>
    <include id="w_menu.select"/>
</msg>
```

参数解读

批量消息也是一个消息，最大的改变就是需要定义，否则不允许组合发送，原则上一个消息对应一个后台接口，可以单独控制权限、操作日志等。

include标签

可引用多个单个消息，**include**个数可以与外部调用消息个数不同，但必须包含外部调用消息

前端调用说明

```js
$.sm(function (re, err) {
    alert(re, err);
}, [
    ["w_user.select"],
    ["w_role.select"],
    ["w_menu.select"],
], {msgid: "common.batch"});
```

注意：**msgid**必须写

- **返回数组**

17、batchtrans (batch 批量事务消息)

```xml
<msg id="w_user.insert" type="insert" v="w_user%16name,sex,pwd"/>
<msg id="w_role.insert" type="insert" v=""/>
<msg id="w_menu.insert" type="insert" v=""/>

<msg id="common.batch.insert" type="batchtrans" title=""
perms="sys:user:list">
    <include id="w_user.insert"/>
    <include id="w_role.insert"/>
    <include id="w_menu.insert"/>
</msg><msg id="common.spell" type="spell" v="#0#%16#1#">
</msg>
```

- **参数解读**

批量事务执行，定义一组**insert**消息，批量事务执行，要么成功，要么都失败；代替了之前trans参数传递方式；也是一个消息，最大的改变就是需要定义，否则不允许组合发送，原则上一个消息对应一个后台接口，可以单独控制权限、操作日志等。

- **include标签**

可引用多个单个消息，include个数可以与外部调用消息个数不同，但必须包含外部调用消息

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, [
    ["w_user.insert", 1, 2, 3, 4],
    ["w_user.insert", 11, 22, 33, 44],
    ["w_user.insert", 111, 222, 333, 444],
    ["w_role.insert", 5, 6, 7, 8],
    ["w_menu.insert", 9, 10, 11, 12],
], {msgid: "common.batch.insert"});
```

> 注意：msgid必须写

- **返回数组**

18、batchtransdb (batch 批量事务消息)

```xml
<msg id="w_user.insert" type="insert" v="w_user%16name,sex,pwd"
d="w" did=""/>
<msg id="w_role.insert" type="insert" v="" d="y" did="{yguid}"/>
<msg id="w_menu.insert" type="insert" v="" d="y" did="{yguid}"/>

<msg id="common.batch.insert" type="batchtransdb" title=""
perms="sys:user:list">
    <include id="w_user.insert"/>
    <include id="w_role.insert"/>
    <include id="w_menu.insert"/>
</msg>
```

- **参数解读**

批量事务执行，定义一组insert消息，批量事务执行，要么成功，要么都失败；代替了之前trans参数传递方式；也是一个消息，最大的改变就是需要定义，否则不允许组合发送，原则上一个消息对应一个后台接口，可以单独控制权限、操作日志等。

- **include标签**

可引用多个单个消息，include个数可以与外部调用消息个数不同，但必须包含外部调用消息

- **前端调用说明**

```js
$.sm(function (re, err) {
    alert(re, err);
}, [
    ["w_user.insert", 1, 2, 3, 4],
    ["w_user.insert", 11, 22, 33, 44],
    ["w_user.insert", 111, 222, 333, 444],
    ["w_role.insert", 5, 6, 7, 8],
    ["w_menu.insert", 9, 10, 11, 12],
], { msgid: "common.batch.insert" });
```

> 注意：msgid必须写

- **返回数组**