这套模式本是稳定的，遇到问题，先检查sql是否写错，配置是否没配，如果还没有解决，

直接联系王振，电话15951918630。

# js调用场景

下面几个场景基本可以包括大部分情况，

如果不能满足条件，请走管控系统之前的方式。

## dataGrid分页查询

**参考： genPayReceipt.html页面的***qryContractList*方法

//1必填参数，表示对应的sql语句

param.businessCode = **'contract.qryContractList'**;

//2下面为组装传递到后台的参数，为sql中的参数，要和后台配置的sql中的参数名称一致

**var** param = *getFormJson*(**'qryCondition'**);  
param.**contract\_status** = statusArrayStr;  
param = **jqueryUtil**.deleteEmptyNode(param);

//3调用*initDataGrid方法，***'tabContractList'为dataGrid的组件ID。**  
*initDataGrid*(**'tabContractList'**, param);

## dataGrid不分页查询

参考： genPayReceipt.html页面的queryContractBatchInfo方法

//1必填参数，表示对应的sql语句

param.**businessCode**=**'contract.selectBatchInfo'**;

//2下面为sql中的参数，要和sql中的参数名称一致  
param.**begin\_time**=**$**(**'#begin\_time'**).**val**();  
param.**end\_time**=**$**(**'#end\_time'**).**val**();  
param.**vendor\_no**=**$**(**'#shipper\_party'**).**val**();  
param.**coal\_no**= **$**(**'#coal\_type\_batch'**).**val**();**var** tmp\_mine\_no = **document**.getElementById(**'mine\_no\_batch'**).**value**;  
param.**batch\_no\_type**=**$**(**'#transport\_type'**).**val**();  
param.**contract\_type** = **contractTypeGlobal**;

//3使用*invokeMis方法，*

*invokeMis*(  
 param,  
 **function**(data) {//拿到后台返回的data，即json，进行后面一步处理。

//比如加载dagagrid，具体可以见svn代码，比较长，不贴了。  
 },  
 **function**(XmlHttpRequest,textStatus) {  
 *showErr*(XmlHttpRequest,textStatus);  
 }  
);

## js执行对应的sql，前台返回查询结果的json

参考： genPayReceipt.html页面的queryContractBatchInfo方法

//1必填参数，表示对应的sql语句

param.**businessCode**=**'contract.selectBatchInfo'**;

//2下面为sql中的参数，要和sql中的参数名称一致  
param.**begin\_time**=**$**(**'#begin\_time'**).**val**();  
param.**end\_time**=**$**(**'#end\_time'**).**val**();  
param.**vendor\_no**=**$**(**'#shipper\_party'**).**val**();  
param.**coal\_no**= **$**(**'#coal\_type\_batch'**).**val**();**var** tmp\_mine\_no = **document**.getElementById(**'mine\_no\_batch'**).**value**;  
param.**batch\_no\_type**=**$**(**'#transport\_type'**).**val**();  
param.**contract\_type** = **contractTypeGlobal**;

//3使用*invokeMis方法，*

*invokeMis*(  
 param,  
 **function**(data) {//拿到后台返回的data，即json，进行后面一步处理。json的数据结构，通过$.messager.alert('ret',JSON.stringify(data));  
 },  
 **function**(XmlHttpRequest,textStatus) {  
 *showErr*(XmlHttpRequest,textStatus);  
 }  
 );

## 保存或更新数据

参考： genPayReceipt.html页面的calPay方法

**var** param = {};

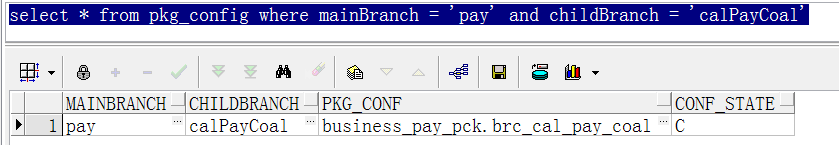
//1必填参数  
param.**public** = {};  
param.**public**.**mainBranch** = **"pay"**;  
param.**public**.**childBranch** = **"calPayCoal"**;

param.**public**.**type** = ***typeComplexPrc***;

说明：

在pkg\_config中配置**mainBranch和childBranch**，这样就会自动调用到你编写的存储过程。

比如：



且特别注意，存储过程必须是：business\_pay\_pck. brc\_cal\_pay\_coal一样的参数，比如：

procedure brc\_cal\_pay\_coal(

v\_json\_in json,

v\_json\_public json,

v\_json\_out out json,

v\_res\_code out varchar2,

v\_res\_msg out varchar2,

v\_op\_id in varchar2

)

//2业务参数（离散值部分）

param.**public**.**isAlter** = **'N'**;  
param.**public**.**receiptType** = **'J'**;*//结算单*  
param.**public**.**crontactId** = ***contractIdGlobal***;  
param.**public**.**payBeginTime** = **$**(**'#begin\_time'**).**val**();  
param.**public**.**payEndTime** = **$**(**'#end\_time'**).**val**();

//3业务参数(LIST部分，可以有多个LIST)，oracle中根据list名称解析即可。  
**var** batchList = [];  
**for**(**var** row = 0;row < selectRows.**length**;row++){  
 **var** boIdJson = {};  
 boIdJson.**batchNo** = selectRows[row].batch\_no;  
 boIdJson.**labReportId** = **null**;  
 batchList.push(boIdJson);  
}  
param.**batchList** = batchList;

//4调用

*invokeMisBusiness*(  
 param,  
 **function**(data) {  
 //拿到后台返回的data，即json，进行后面一步处理。

//  
 },  
 **function**(XmlHttpRequest,textStatus) {  
 *showErr*(XmlHttpRequest,textStatus);  
 }  
);

说明：

1. 新增被java调用的存储过程中不要进行commit 或 rollback ，java会根据v\_res\_code进行处理。

当v\_res\_code为0时，会自动commit;

当v\_res\_code不为0时，或存储过程抛出异常时，会自动rollback;

且特别注意，存储过程必须是：business\_pay\_pck. brc\_cal\_pay\_coal一样的参数，比如：

procedure brc\_cal\_pay\_coal(

v\_json\_in json,

v\_json\_public json,

v\_json\_out out json,

v\_res\_code out varchar2,

v\_res\_msg out varchar2,

v\_op\_id in varchar2

)

1. 调用的过程，参考下存储过程，参数都是固定的,

business\_pay\_pck.brc\_cal\_pay\_coal(

v\_json\_in json, //前台js传入的json参数

v\_json\_public json, //前台js传入的json参数的public部分

v\_json\_out out json, //返回前台的json

v\_res\_code out varchar2, //调用状态，0成功，1失败

v\_res\_msg out varchar2, //执行结果说明

v\_op\_id in varchar2 //这个前台不用传，后台从session中自动获取操作员ID传入

1. 当编写存储过程，可以复制business\_pay\_pck.brc\_sample，该名，这个里面基础代码已经有了，只需要

编写业务代码。

# 配置sql

查询，才在config\_sql中配置。

（注意区分： 当保存或更新数据，使用1.4中方式，在pkg\_config中配置，不在config\_sql中配置）

config\_sql 表

SQL\_KEY 对应前台js传过来的 param.businessCode参数

SQL\_STR 执行的查询语句

SQL\_STR\_CNT 如果分页，则查询数量的语句

IS\_PAGE 是否分页 Y / N

SQL\_VERSION 语句版本号

SQL\_AREA 所属电厂，比如库车为 KC

SQL\_FUNCTION 语句功能，比如：根据合同ID查询奖罚条件

SQL\_LAST\_CHANGE\_DATE 语句上一次更新时间

SQL\_STATE 语句状态，N在用，normal含义。D作废不在用。

DATA\_SOURCE 目前不需要用到，填空

## 分页

参考： select \* from config\_sql cf where cf.sql\_key = ‘vendor.qryVendorEvaluateInfo’

1）IS\_PAGE = Y

2）SQL\_STR：

正常写好语句后，外面套上

select f.\*,rownum rownumber from (

sql语句

) f

比如：

select f.\*,rownum rownumber from (

select

t.evaluate\_id,

(select vendor\_name from vendor\_info a where a.vendor\_id = t.vendor\_id) vendor\_name,

decode(t.evaluate\_type,'M','月报','Y','年报') evaluate\_type\_name，

t.total\_score,

(select b.level\_name from vendor\_level b where b.level\_cd = t.vendor\_level\_id) level\_name,

to\_char(t.insert\_time,'YYYY-MM-DD hh24:mi:ss') evaluate\_date,

(select c.realname from security\_user c where c.id = t.op\_code) op\_name,

(select c.realname from security\_user c where c.id = t.update\_code) update\_name,

(select d.name from dictionary\_table d where d.type = 'AUDIT\_STATUS' and d.status = '0' and d.id = t.status ) status\_name

from vendor\_evaluate\_info t

where 1 = 1

and t.evaluate\_type = #{evaluate\_type}--notnull(evaluate\_type)

and t.vendor\_id = #{vendor\_id}--notnull(vendor\_id)

and t.insert\_time > to\_date(#{insert\_time\_min},'YYYY-MM-DD HH24:mi:ss')--notnull(insert\_time\_min)

and t.insert\_time <= to\_date(#{insert\_time\_max},'YYYY-MM-DD HH24:mi:ss') + 1 - 0.00001--notnull(insert\_time\_max)

)f

3）SQL\_STR\_CNT：

查询数量的语句，比如：

select count(1) from vendor\_evaluate\_info t

where 1 = 1

and t.evaluate\_type = #{evaluate\_type}--notnull(evaluate\_type)

and t.vendor\_id = #{vendor\_id}--notnull(vendor\_id)

and t.insert\_time > to\_date(#{insert\_time\_min},'YYYY-MM-DD HH24:mi:ss')--notnull(insert\_time\_min)

and t.insert\_time <= to\_date(#{insert\_time\_max},'YYYY-MM-DD HH24:mi:ss') + 1 - 0.00001--notnull(insert\_time\_max)

4）其他字段参考config\_sql 表的说明。

## 非分页

同分页类似，

1）IS\_PAGE = N

只需要填写SQL\_STR，不需要填写SQL\_STR\_CNT

2）SQL\_STR只需要填写正常的sql语句，不需要再套一层 select f.\*,rownum rownumber from ( 。。。。。。) f

3）其他字段参考config\_sql 表的说明。

## sql例子

类似mybatis ， ${} 表示字符串方式，#{} 表示参数方式，like , 日期参数

如果参数为空，需要分支语句不出来，则

比如： and bi.insert\_time > to\_date(**#{begin\_time}**,'yyyy-mm-dd')-1/24/60/60 **--notnull(begin\_time)**

查询返回中如果有日期字段，请自行先按照业务需要的格式转换，比如to\_char(c.begin\_dt, 'yyyy-mm-dd')

参考一）

select \* from ${tableName} a where a.cust\_name = #{cust\_name}

and a.age = #{age} --notnull(age,cust\_name)

and a.so\_date > to\_date(#{so\_date}, 'yyyymmdd') and a.desp like #{despLike}||'%' and ${columnName} = '198983983' --(${age} > 30 && '${cust\_name}'=='wangz')

参考二）

调用一个函数返回json

select business\_pay\_pck.qry\_reward\_info(#{payReceiptId}) reward\_infos from dual

参考三)in 参数方式

select bi.batch\_no,to\_char(bi.insert\_time,'yyyy-mm-dd') as batch\_date,bi.coal\_no,bi.mine\_no,get\_coal\_name(bi.coal\_no) coal\_name,get\_mine\_name(bi.mine\_no) mine\_name,get\_vendor\_name(bi.ven\_no) vendor\_name,

sum(nvl(qy.tick\_qty,0)) tick\_qty,sum(nvl(qy.kd\_qty,0)) kd\_qty,sum(nvl(qy.net\_qty,0)) net\_qty,(sum(nvl(qy.tick\_qty,0))-sum(nvl(qy.net\_qty,0))) loss\_qty

from batch\_no\_info bi,rlrecordmstqy qy

where qy.car\_batch\_no = bi.batch\_no

and bi.insert\_time > to\_date(#{begin\_time},'yyyy-mm-dd')-1/24/60/60 --notnull(begin\_time)

and bi.insert\_time < to\_date(#{end\_time},'yyyy-mm-dd')+1/24/60/60 --notnull(end\_time)

and bi.ven\_no = #{vendor\_no}--notnull(vendor\_no)

and bi.coal\_no = #{coal\_no}--notnull(coal\_no)

**and bi.mine\_no in ( ${mine\_no} )--notnull(mine\_no)**

and bi.batch\_no\_type = #{batch\_no\_type}--notnull(batch\_no\_type)

and not exists (

select 1 from pay\_2\_batch pb,pay\_receipt pr

where pb.pay\_receipt\_id = pr.pay\_receipt\_id

and pr.contract\_type = #{contract\_type}

and pb.batch\_no = bi.batch\_no

and pr.pay\_status in ('1','2','11','8')

)

group by bi.batch\_no,to\_char(bi.insert\_time,'yyyy-mm-dd'),bi.coal\_no,bi.mine\_no,get\_coal\_name(bi.coal\_no),get\_mine\_name(bi.mine\_no),get\_vendor\_name(bi.ven\_no)