**接口编写规范**

## 知识点

### 材料

**接口编写遵循CMD规范** <https://github.com/cmdjs/specification/blob/master/draft/module.md>

**依赖框架**

jquery框架

seajs框架 <http://seajs.org>

前端验证框架 <https://github.com/chriso/validator.js> 在使用前端验证时，查看此文档

**铭飞通用基础框架**

接口生产使用参考 <http://ms.mingsoft.net/html/86/6507/index.html>

对应java文档参考<http://api.ms.mingsoft.net/people/>

### 具体知识点

## 规范

### 结构规范

1. 接口结构必须和java文档中类的路径结构一致



对应的接口结构如下：

return:{mall:{brand:{/\*接口代码\*/},},}

具体代码[点击这里](#_无参实例)

1. 在同一文件内使用“，”间隔不同类和接口，必须写，少了在编译的时候会报错
2. 接口访问地址统一放在接口结尾处，例如上图的路径：

ajaxCfg.url = ms.base + "/people/mall/orderComment/\*.do";

//\*号那里写具体什么接口

ajaxCfg.params = data;

ms.ajax(ajaxCfg, func);

1. 接口声明规范：
   * 所有的接口都需要先定义。定义代码都一样，代码如下：

define(**function**(require, exports, module) {

**var** ms = require("ms");

**var** ajaxCfg = {

"type" : "post",

"dataType" : "json"

};

1. 对接接口时，不要把冗长的实现代码写在模块加载的方法体内，实现代码写在加载函数外，并且定义成函数然后再在加载函数中调用实现代码。

### 命名规范

1. 所有MS接口文件名为ms.\*的格式
2. 变量命名采用驼峰式
3. 删除接口命名为del

### 注释规范

1. 接口的文档注释必须写，系统会根据注释生成api文档。注释说明如下：

/\*\*

\* 接口名称

------

\* 接口描述

\* @callmethod 接口调用描述

\* @param {{type:类型,have:true为必填参数,false默认可选参数}} 字段名 描述

\* @param N

\* @examples 实例

\* ...

\* 代码片段

\* ...

\* @function 返回值

\* @return {{type:参数名称}} 描述

\* @return N

\*/

1. 代码中的注释使用//，原因是/\*\*/注释，注释内容会生成到api文档中
2. callmethod中的数据参数用data，实例中以实际情况为准

### 其它规范

1. **验证规范：**
   * 统一使用validator验证框架，框架地址上文提过

注意：铭飞基本框架中已经引用了validator框架，所以使用时无需再引用

* + 传过来的所有参数，都必须使用框架判断是否为空
  + 验证参数，先判断是否为空，后再做其它边界验证

1. **弹框规范：**
   * 接口中所有弹框统一使用铭飞基础框架中的ms.alert();
   * 对接接口中需把接口中的弹框处理掉，不能显示出接口提示出的信息
2. **数据传输规范：**
   * 接口的参数采用Key/Value 格式，例如：
     + "foo=bar1&foo=bar2"
3. **数据类型规范：**
   * 编号的都为int型
   * 编码的为字符串型
   * pageNo为分页数量
   * pageSize为每页记录数量
   * 关注类型默认为1
4. 一个接口文档里只声明一次接口版本

## 常见问题

### 规范问题

1. 在使用验证框架的时候，validator是什么意思？
   * Validator是验证框架的名字。同理，ms.alert中“ms”也是这个框架的名字

### 其它问题

1. 接口报错处理方法：
   * 404报错：路径错误，仔细查看
   * 400报错：参数类型错误，仔细核对参数
   * 500报错：查看post请求参数、日志，请求外部帮助具体再分析

## 实例

### 无参实例

define(**function**(require, exports, module) {

**var** ms = require("ms");

**var** ajaxCfg = {

"type" : "post",

"dataType" : "json"

};

**return** {

mall : {

brand : {

"version" : "1.0.0",

/\*\*

\* 品牌列表

------

\* **@callmethod** mall.brand.list(function(json){...});

\* **@param** {{type:function,have:true}} 回调方法 返回值（json）

\* **@examples**

\* ...

\* mmall.mall.brand.list(function(json){

\* alert(JSON.stringify(json));

\* });

\* ...

\* **@function**

\* {"categoryTitle":"乐视","categoryDescription":"","categoryCategoryId":"6053"}

\* **@return** {{type:categoryTitle}} 品牌名

\* **@return** {{type:categoryDescription}} 类别描述

\* **@return** {{type:categoryCategoryId}} 商品分类编号

\*/

list : **function**(func){

ajaxCfg.url = ms.base + "/mall/brand/list.do";

ms.ajax(ajaxCfg, func);

},

},

},

}

});

### 有参实例

define(**function**(require, exports, module) {

**var** ms = require("ms");

**var** ajaxCfg = {

"type" : "post",

"dataType" : "json"

};

**return** {

"version" : "1.0.0",

/\*\*

\* 支付宝支付接口

------

\* **@callmethod** bank.pay.alipay(data,function(json){...});

\* **@param** {{type:URL,have:true}} notifyUrl 接口异步请求地址

\* **@param** {{type:URL,have:true}} returnUrl 返回地址

\* **@param** {{type:int,have:true}} orderNo 订单编号

\* **@param** {{type:string,have:true}} orderName 订单标题

\* **@param** {{type:string,have:true}} orderPrice 订单价格

\* **@param** {{type:string}} orderDesc 订单描述

\* **@param** {{type:URL,have:true}} showUrl 商品显示地址

\* **@param** {{type:function}} 回调方法 返回值（json）

\* **@examples**

\* ...

\* mmall.bank.pay.alipay($("form").serialize(),function(json){

\* alert(JSON.stringify(json));

\* })

\* ...

\* **@function**

\* {"resultMsg":"","result":true}

\* **@return** {{type:resultMsg}} 提示信息

\* **@return** {{type:result}} true成功、false失败

\*/

bank : {

pay : {

alipay : **function**(data,func){

**if** (ms.isEmpty(data)){

**return**;

};

//将序列化数据转换为json数据

**var** newDataJson = ms.turnJson(data);

//判断newDataJson.notifyUrl是否为空

**if**(validator.isNull(newDataJson.notifyUrl)){

ms.alert("接口请求地址不能为空");

**return**;

}

//判断newDataJson.notifyUrl是否为地址

**if** (!validator.isURL(newDataJson.notifyUrl)){

ms.alert("接口请求地址无效");

**return**;

}

**if** (validator.isNull(newDataJson.returnUrl)){

ms.alert("返回地址不能为空");

**return**;

}

**if** (!validator.isURL(newDataJson.returnUrl)){

ms.alert("返回地址无效");

**return**;

}

**if** (validator.isNull(newDataJson.orderNo)){

ms.alert("订单编号不能为空");

**return**;

}

//判断newDataJson.orderNo是否为整型

**if** (!validator.isInt(newDataJson.orderNo)){

ms.alert("订单编号必须为整型");

**return**;

}

**if** (validator.isNull(newDataJson.orderName)){

ms.alert("订单标题不能为空");

**return**;

}

**if** (validator.isNull(newDataJson.orderPrice)){

ms.alert("订单价格不能为空");

**return**;

}

**if** (!validator.isCurrency(newDataJson.orderPrice)){

ms.alert("订单价格无效");

}

**if** (validator.isNull(newDataJson.showUrl)){

ms.alert("商品显示地址不能为空");

**return**;

};

**if** (!validator.isURL(newDataJson.showUrl)){

ms.alert("商品显示地址无效");

**return**;

}

ajaxCfg.url = ms.base + "/bank/pay/alipay.do";

ajaxCfg.params = data;

ms.ajax(ajaxCfg, func);

},

},

},

}

});