

浏览器到Node端接口约束文档

1、背景说明

网络应用程序，分为前端和后端两个部分。当前的发展趋势，就是前端设备层出不穷（手机、平板、桌面电脑、其他专用设备.....）。因此，必须有一种统一的机制，方便不同的前端设备与后端进行通信。这导致API构架的流行，甚至出现"APIFirst"的设计思想。RESTful API是目前比较成熟的一套互联网应用程序的API设计理论。

REST（Representational State Transfer）表述性状态转换，REST指的是一组架构约束条件和原则。如果一个架构符合REST的约束条件和原则，我们就称它为RESTful架构。REST本身并没有创造新的技术、组件或服务，而隐藏在RESTful背后的理念就是使用Web的现有特征和能力，更好地使用现有Web标准中的一些准则和约束。虽然REST本身受Web技术的影响很深，但是理论上REST架构风格并不是绑定在HTTP上，只不过目前HTTP是唯一与REST相关的实例。

2、规范说明

2.1、URI规范

- 1、不用大写；
- 2、用中杠 - 不用下杠 _；
- 3、get参数列表要urlencode；
- 4、在RESTful架构中，每个网址代表一种资源（resource），所以网址中尽量不使用动词，只能有名词（特殊情况可以使用动词），而且所用的名词往往与数据库的表格名对应。

避免层级过深的URI

在url中表达层级，用于按实体关联关系进行对象导航，一般根据id导航。

过深的导航容易导致url膨胀，不易维护，如 GET /zoos/1/areas/3/animals/4，尽量使

查询参数代替路径中的实体导航，如 GET/animals?zoo=1&area=3；

2.2、版本规范

统一将版本号放入URI中,统一在域后面第二节填充版本。

Eg: <http://ip:port/api/v1/vehicle>

2.3、请求方式规范

目前HTTP协议中有8种方法，我们系统中目前常用的主要有4种，目前着重先说明这四种，至于其他的后续扩展有需要再进行补充说明

请求方式	场景	说明
GET	查询	从服务器获取资源（一项/多项）
POST	添加/查询/终端交互	在服务器创建一个资源，或者查询，在参数过多的情况下可以使用
PUT	更新	在服务器更新资源，客户端提供更新属性等信息
DELETE	删除	从服务器删除资源信息

GET请求：

	示例	备注
过滤条件	?type=1&age=16	允许一定的uri冗余，如 /zoos/1 与 /zoos?id=1
排序	?sort=age&order=asc	指定返回结果按照哪个属性排序，以及排序顺序
投影	?whitelist=id,name,emai	
分页	?startIndex=2&pageSize=100	指定第几页，以及每页的记录数

如果请求参数过长，get请求满足不了的情况下建议直接用json封装，通过body直接post提交

POST请求：

提交方式为POST时，交互参数统一封装为json的方式进行提交

Content-type : application/json

2.4、内容命名规范

- 1、参数名称和返回字段的命名统一采用驼峰，如：userId,companyName,page等。
- 2、Header 大写开头，分词采用‘-’和http标准请求头保持一致，如 **Accept-Encoding**、**Connection** 等。

2.5、权限验证规范

为了方便在整个应用交互中权限控制和访问域控制，需要在请求中携带认证信息，请求头标示统一为Token。Token的构成形式为 userID,roleID,clientID,utcTimeStamp

总共分为4部分

userID	系统用户id，由web统一管理分配，是系统用户唯一标示符
roleID	角色ID，用户所属角色的ID，是角色唯一标示符，通过角色ID能够界定用户操作权限范围
clientID	业务模块编码，用于标示当前请求是来源于哪个模块进程，方便服务端区分消费业务进程，如DST
utcTimeStamp	token生成时的时间戳，方便控制token有效期

将4部分通过半角逗号分隔，组装成字符串用过des加密。在用户登录阶段由服务端返回给客户端供后续业务接口使用。

在业务查询中，如果调用方没有指定查询范围时，默认采用访问域内的所有范围按照参数规则检索并返回应答信息

2.6、应答规范

目前红树林系统中应答统一为json数据格式，内容命名规范同2.4

应答结果封装

现针对红树林业务，为了方便整体在上下级网元间的正确信息传导，针对应答数据进行统一封装。封装结构主要由两部分构成，code 用于描述服务器处理情况，是否正常处理（参数格式是否正确、权限信息是否满足、服务器是否出现异常等情况），

Msg:内部信息，主要用于内部信息提示，不用于用户界面信息展现，比如内部验重失败的原因，参数错误的原因，服务器异常的原因等，方便使用方明确问题

result部分则为数据应答部分内容。所有结果在第一层级只有这两个信息。

```
Eg: {"code":200,"msg":"xxxxxxx","result":false}

{"code":200,"msg":"xxxxxxx","result":{"dataCount":500,"dataList":[{"...},{...}]}}
```

2.6.1、常用true/false应答

此应答通常用于保存、删除、修改、指令下发等场景中，返回应答只需要告知调用方执行结果即可，应答格式为

```
{"code":200,"msg":"xxxxxxx","result":false}
```

2.6.2、分页应答

此应答主要用于分页查询的场景下用于数据返回，其中主要包括了当前查询数据的总条数、当前分页获取的信息

```
{"code":200,"msg":"xxxxxxx","result":{"dataCount":500,"dataList":[{"...},{...}]}}
```

dataCount:目标资源记录个数

dataList:当前页数据项

2.6.2、创建应答

此应答作为创建数据时的通用应答，主要包括了当前创建信息的唯一标识和创建结果

`{"code":200,"msg":"xxxxxxx","result":{"id":1,"saveResult":true}}`

id:创建数据的唯一标识

saveResult:保存结果，true表示保存成功

2.7、常见错误码定义

模块	错误码	描述
通用	200	服务器正常处理，没有出现异常、参数错误等情况
201	权限校验失败，当前权限不能操作响应请求	
202	服务器异常，服务器处理过程中出现异常，整个流程未能正常进行	
203	请求参数异常	
204	唯一校验不通过	
205-400	预留	
终端状态	401	终端不在线
402	终端响应超时	
403	终端应答异常	
404-500	预留	
用户	1000-1099	该模块自定义预留
角色	1100-1199	该模块自定义预留
车组	1200-1299	该模块自定义预留
车辆	1300-1399	该模块自定义预留
终端	1400-1499	该模块自定义预留
sim卡	1500-1599	该模块自定义预留
驾驶员	1600-1699	该模块自定义预留
报警	1700-1799	该模块自定义预留
证据	1800-1899	该模块自定义预留