|  |
| --- |
| 重庆紫光华山智安科技有限公司技术规范 |
| Restful接口规范 |
|  |
|  |
|  |

目 录

[1 简介 1-１](#_Toc522121564)

[2 URI 2-１](#_Toc522121565)

[2.1 URI命名规范 2-１](#_Toc522121566)

[2.2 版本 2-１](#_Toc522121567)

[2.3 过滤信息 2-１](#_Toc522121568)

[如果记录数量很多，服务器不可能都将它们返回给用户。API应该提供参数，过滤返回结果。 下面是一些常见的参数。 2-１](#_Toc522121569)

[3 Request 3-２](#_Toc522121570)

[3.1 公共参数 3-２](#_Toc522121571)

[3.2 HTTP方法 3-２](#_Toc522121572)

[4 Response 4-１](#_Toc522121573)

[4.1 各HTTP方法成功处理后的数据格式 4-１](#_Toc522121574)

[4.2 不需要包装 4-１](#_Toc522121575)

[4.3 JSON格式约定 4-１](#_Toc522121576)

[4.4 分页response 4-１](#_Toc522121577)

[4.5 状态码 4-２](#_Toc522121578)

[4.5.1 成功 4-２](#_Toc522121579)

[4.5.2 失败 4-２](#_Toc522121580)

[5 服务型资源 5-４](#_Toc522121581)

[6 异步任务 6-５](#_Toc522121582)

# 简介

rest是一种软件架构风格，rest接口是围绕“资源”展开的，利用HTTP的协议，其实rest本也可以和HTTP无关，但是现在大家普遍的使用rest都是依托于HTTP协议。HTTP 的url即资源。

# URI

URI 表示资源，资源一般对应服务器端领域模型中的实体类。

## URI命名规范

为规范URI的使用，请遵从以下几点规范：

* + - * 1. 全部使用小写；
        2. 单词之前用中杠’-’不用下划线’\_’；
        3. 不能含有动词；
        4. URI中的名词表示资源集合，使用复数形式；
        5. 结尾不应该包含斜杠“/”。

## 版本

应该将API的版本号放入URL。如：

https://api.example.com/api/vms/v1

## 过滤信息

### 如果记录数量很多，服务器不可能都将它们返回给用户。API应该提供参数，过滤返回结果。 下面是一些常见的参数。

?limit=10：指定返回记录的数量

?offset=10：指定返回记录的开始位置。

?page\_number=2&page\_size=100：指定第几页，以及每页的记录数。

?sortby=name&order=asc：指定返回结果按照哪个属性排序，以及排序顺序。

?animal\_type\_id=1：指定筛选条件 参数的设计允许存在冗余，即允许API路径和URL参数偶尔有重复。

比如， GET /zoo/ID/animals 与 GET /animals?zoo\_id=ID 的含义是相同的。

## 字符集

统一使用utf-8

# Request

## 公共参数

公共请求参数是指每个接口都可能需要传递的参数，示例如下：

公共参数header数据格式说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 参数 | 是否必须 | header格式 | 说明 |
|  | app | 是 | us-app:$app | 请求客户端应用标识，示例:xxxx-ios、xxxx-android、xxxx-pc、xxxx-h5 |
|  | user\_id | 是 | Authorization:base64(user\_id:token) | 用户标识 |
|  | token | 是 | Authorization:base64(user\_id:token) | 用户令牌 |
|  | date | 可选 | Date: | 用于客户端与云端的时间同步 |
|  | local\_address | 可选 | Local Address:$local\_address | 网络标识 |

## HTTP方法

* GET: 获取资源
* POST： 新建资源
* PUT：在服务器更新资源（向客户端提供改变后的所有资源）
* PATCH: 在服务器更新资源（向客户端提供改变的属性）
* DELETE：删除资源

示例：

GET /zoos：列出所有动物园

POST /zoos：新建一个动物园

GET /zoos/ID：获取某个指定动物园的信息

PUT /zoos/ID：更新某个指定动物园的信息（提供该动物园的全部信息）

PATCH /zoos/ID：更新某个指定动物园的信息（提供该动物园的部分信息）

DELETE /zoos/ID：删除某个动物园

GET /zoos/ID/animals：列出某个指定动物园的所有动物

DELETE /zoos/ID/animals/ID：删除某个指定动物园的指定动物

## 接口调用

restful接口调用，使用HTTP短连接

# Response

## 各HTTP方法成功处理后的数据格式

HTTP方法成功处理后的数据格式说明

|  |  |  |
| --- | --- | --- |
|  | 方法 | 返回格式 |
|  | GET | 单个对象、集合 |
|  | POST | 新增成功的资源对象 |
|  | PUT/PATCH | 更新成功的资源对象 |
|  | DELETE | 空 |

## 不需要包装

response 的 body 直接就是数据，不要做多余的包装。错误示例：

{

"success":true,

"data":{"id":1,"name":"xiaotuan"},

}

## JSON格式约定

* + - * 1. 时间用13位数长整形(毫秒数)，客户端自己按需解析；
        2. 不传null字段；
        3. JONS 中的 json-name 采用小写、数字及下划线进行命名，如 access\_token；
        4. json消息体中的图片文件数据、文件URL地址等，推荐转换为Base64编码

## 分页response

分页对象需要组装分页参数对象，示例如下：

{

"paging":{"page\_num":1,"page\_size":10,"total":729},

"data":[{},{},{}...]

}

## 状态码

### 成功

首先应遵循 HTTP 中关于 status code的规范

|  |  |  |  |
| --- | --- | --- | --- |
|  | 状态码 | 方法 | 说明 |
|  | 200 | ALL | 资源响应成功 |
|  | 201 | POST | 新资源被创建 |
|  | 204 | DELETE | 资源被删除 |

### 失败

说明

不要发生了错误但给2xx响应，客户端可能会缓存成功的http请求；

正确设置http状态码，不要自定义；

Response body 提供 1) 错误的代码（日志/问题追查）；2) 错误的描述文本（展示给用户）。

当用户访问接口出错时，服务器会返回给一个合适的4××或者5××的HTTP状态码；以及一个application/json格式的消息体，消息体结构示例如下：

{

“error\_code”:”01000001”,

"message":"{error message}",

"request\_id":"01234567-89ab-cdef-0123-456789abcdef",

"host\_id":"{server identity}",

"server\_time":"2014-01-01T12:00:00Z"

}

其中，

**error\_code** 业务错误码,为8位字符串，前两位表示业务组件id，后六位为错误码编号，详情请参照《异常处理及错误码规范2.0》。

**message**为错误的摘要信息，并且应该包含对用户处理该错误有指导意义的信息

**request\_id** 为错误的uuid，用于帮助技术人员在日志系统中获得错误的详细

**host\_id**为发生错误的服务器

**server\_time**为发生错误时的服务器时间

提示

Java 服务器端一般用异常表示 RESTful API 的错误。API 可能抛出两类异常：业务异常和非业务异常。

业务异常由自己的业务代码抛出，表示一个用例的前置条件不满足、业务规则冲突等，比如参数校验不通过、权限校验失败。

非业务类异常表示不在预期内的问题，通常由类库、框架抛出，或由于自己的代码逻辑错误导致，比如数据库连接失败、空指针异常、除0错误等等。

业务类异常必须提供2种信息：

如果抛出该类异常，HTTP 响应状态码应该设成什么；

异常的文本描述；

在Controller层使用统一的异常拦截器：

设置 HTTP 响应状态码：对业务类异常，用它指定的 HTTP code；对非业务类异常，统一500；

Response Body 的错误码：异常类名

Response Body 的错误描述：对业务类异常，用它指定的错误文本；对非业务类异常，线上可以统一文案如“服务器端错误，请稍后再试”，开发或测试环境中用异常的 stacktrace，服务器端提供该行为的开关。

# 服务型资源

除了资源简单的CRUD，服务器端经常还会提供其他服务，这些服务无法直接用上面提到的URI映射。可以把这些服务看成资源，计算的结果是资源的presentation，按服务属性选择合适的HTTP方法。

如：

GET /search?q=filter?category=file //按关键字搜索

GET /distance-calc?lats=47.480&lngs=-122.389&late=37.108&lnge=-122.448 //计算地球上两点间的距离

POST /batch-publish-msg

[{"from":0,"to":1,"text":"abc"},{},{}...] //批量向用户推送消息

# 异步任务

#### 对耗时的异步任务，服务器端接受客户端传递的参数后，应返回创建成功的任务资源，其中包含了任务的执行状态。客户端可以轮训该任务获得最新的执行进度。

提交任务：

POST /batch-publish-msg

[{"from":0,"to":1,"text":"abc"},{},{}...]

返回：

{"taskId":3,"createBy":"Anonymous","status":"running"}

GET /task/3

{"taskId":3,"createBy":"Anonymous","status":"success"}

#### 如果任务的执行状态包括较多信息，可以把“执行状态”抽象成组合资源，客户端查询该状态资源了解任务的执行情况。

提交任务：

POST /batch-publish-msg

[{"from":0,"to":1,"text":"abc"},{},{}...]

返回：

{"taskId":3,"createBy":"Anonymous"}

GET /task/3/status

{"progress":"50%","total":18,"success":8,"fail":1}