链滴

REST 接口设计规范

作者：washmore

原文链接：https://ld246.com/article/1571037051891

来源网站：链滴

设计原则

- URL即资源，使用名词表示资源
- 接口应直观、简洁、风格命名保持一致
- 必须版本化，具有足够的灵活性来支持上层UI

设计规范

一. 版本号

应该将API的版本号放入URL。

版本号以字符'v'开头，比如：v1

http://api.xxx.com/cuc/sever/v1

HTTP API的版本控制有多种实现方式，更多内容请自行参考其他文献.

二. 路径(Endpoint)

接口路径，也称为端点(Endpoint)，代表一种资源的访问地址。

1. 使用名词表示资源，使用HTTP Method描述操作

因为"资源"表示一种实体，所以应该是名词，URL不应该有动词，动词应该放在HTTP协议中，CRUD操作不要体现在URL中。

反面教材

```
查询员工 GET    /v1/findEmployee?id=1000
新增员工 POST   /v1/addEmployee
更新员工 POST   /v1/updateEmployee?id=1000
删除员工 POST   /v1/deleteEmployee?id=1000
```

正确示例

```
查询员工 GET    /v1/employee/1000
新增员工 POST   /v1/employee
更新员工 PUT    /v1/employee/1000
删除员工 DELETE /v1/employee/1000
```

2. 名词统一使用单数

资源既可以是单个，也可以是一个集合，比如：user是单个资源，users是集合资源。

由于英文名词的复数规则众多，为了保持接口命名的一致性，我们建议URL里描述集合资源的名词统使用单数。

正确示例

查询单个员工 GET /v1/employee/1000
查询一组员工 GET /v1/employee

反面教材

查询单个员工 GET /v1/employees/1000
查询一组员工 GET /v1/employees

3. 使用小写字母，多个单词用"-"分隔，提高URL的可读性

RFC3986定义了URI是大小写敏感的(除了scheme和host部分)，为了避免歧义，接口路径应尽量使用写字母。

对于由多个单词构成的路径，推荐使用'-'(hyphen)分隔，避免使用驼峰命名（camelCase）、下划线名(snake_case)、首字母大写的驼峰命名（PascalCase），以提高可读性，举个例子：

camelCase ： /v1/customer/1000/shippingAddress [Bad]
snake_case : /v1/customer/1000/shipping_address [Bad]
PascalCase : /v1/customer/1000/ShippingAddress [Bad]

hyphen : /v1/customer/1000/shipping-address [OK]

4. 资源嵌套层次避免过深，尽量不超过2层

定义REST接口时，通常使用资源嵌套（多级路径）来标识资源之间的关系，比如：

GET /v1/user/1000/company 查询ID等于1000的用户的公司信息
GET /v1/user/1000/company/10 查询ID等于1000的用户所属公司中ID=10的公司信息，嵌套两层

资源嵌套层次尽量不超过2层，否则很难阅读和使用。

反面教材

GET /v1/company/10/department/20302/employee/10830
GET /v1/zoo/1/area/3/animal/4

三. 使用正确的HTTP Method

编写API接口时，我们通过URL来定义资源的地址，而通过HTTP Method来定义对资源的操作。

CRUD

HTTP协议提供了四种Method：GET、POST、PUT、DELETE，可用于描述对资源的CRUD操作。

<table>
  <thead>
    <tr>
      <th style="text-align:left">HTTP Method(Verb)</th>
      <th style="text-align:left">CRUD操作</th>
      <th style="text-align:left">说明</th>

```
        <th style="text-align:left">幂等性</th>
        <th style="text-align:left" >安全性 </th>
      </tr>
  </thead>
  <tbody>
    <tr>
      <td style="text-align:left">GET</td>
      <td style="text-align:left">Read/Select</td>
      <td style="text-align:left">检索资源</td>
      <td style="text-align:left">幂等</td>
      <td style="text-align:left">安全</td>
    </tr>
    <tr>
      <td style="text-align:left">POST</td>
      <td style="text-align:left">Create</td>
      <td style="text-align:left">新建一个资源</td>
      <td style="text-align:left">不幂等</td>
      <td style="text-align:left">不安全</td>
    </tr>
    <tr>
      <td style="text-align:left">PUT</td>
      <td style="text-align:left">Update</td>
      <td style="text-align:left">更新资源属性</td>
      <td style="text-align:left">幂等</td>
      <td style="text-align:left">不安全</td>
    </tr>
    <tr>
      <td style="text-align:left">DELETE</td>
      <td style="text-align:left">Delete</td>
      <td style="text-align:left">删除资源</td>
      <td style="text-align:left">幂等</td>
      <td style="text-align:left">不安全</td>
    </tr>
  </tbody>
</table>
```

幂等性

HTTP接口的幂等性，指的是同样的请求被执行一次与连续执行多次的效果是一样的，服务器的状态是一样的。

如果一个接口是幂等的，那么它天然就支持客户端重试，接口自然可用性更高。

常见的幂等性请求:

1. GET

一次或多次查询，不会改变资源状态，返回的都是同一个资源

2. HEAD

只获取某个资源的头部信息，比如获取某个文件的大小、某个资源的修改日期等，一次或多次查询不改变资源状态

3. OPTIONS

应用场景：跨域请求，用于获取Web服务器支持的Http Method，一次或多次查询，返回相同结果

4. DELETE

一次或多次删除，资源的状态都是删除（注意，返回的状态码可能是200或404）

5. PUT

同一个更新请求，一次或多次调用，资源更新后的属性值都一样

安全性

一个HTTP Method如果不更改资源的状态，那么它就是安全的。换句话说，一个安全的HTTP Metho
它一定是只读操作的，常见安全的HTTP Method: GET、HEAD、OPTIONS。

其他说明

PATCH : 更新资源的部分属性，因为 PATCH 比较新，而且规范比较复杂，所以真正实现的比较少，
般都是用 PUT 替代。

[warning] PATCH使用条件

1. HttpClient 4.2 or later
2. Spring 3.2 or later
3. Tomcat8 or later
4.

JDK8 or later

四. 筛选(Filtering)、排序(Sorting)、分页(Pagination)、字段选择

筛选（Filtering)

接口URL尽量保持简短，对于集合资源不同纬度的筛选，可通过组合不同的Query Param来实现。

反面教材

GET /v1/externalEmployees
GET /v1/internalEmployees
GET /v1/internalAndSeniorEmployees

正确示例

GET /v1/employee?state=internal&title=senior

排序(Sorting)

单个字段排序

GET /v1/employee?sort=age&order=asc
GET /v1/employee?sort=age&order=desc

多字段排序

字段名前缀,'+'：表示升序，'-'：降序，多个字段名以逗号分隔。

GET /v1/employee?sort=+age,-userName

分页(Pagination)

对于数据量比较大的集合资源，接口实现一定要支持分页。

常规分页有两个重要的参数：

- pageNo

获取那一页的记录，默认第一页。

- pageSize

每页返回多少条数据，推荐默认值25，必须限定此参数的最大值。

返回的资源列表为：[(pageNo-1)*pageSize, pageNo*pageSize)
请求示例：

GET /v1/employee?pageNo=8&pageSize=20&sort=age&order=desc

返回值可采用以下数据格式：

```
{
   "pageSize":20,      // 页尺寸
   "pageNo":8,         // 页码
   "totalCount":182,  // 总记录数
   "pageList":[
            {
               "id":13483,
               "name":"Hehe",
               "title":"senior"
            },
             {
               "id":13484,
               "name":"haha",
               "title":"junior"
            }
   ]
}
```

分页还有另一种实现方式，适合单页面"加载更多"的交互。

点击"加载更多"，即去请求一页结果追加到当前页面，这种分页也有两个参数：

- sinceId 上次分页查询的最后一条记录的主键ID，第一次分页默认值为0。
- pageSize 每页返回多少条数据，推荐默认值25，必须限定此参数的最大值。

请求示例：

```
GET /v1/employee?sinceId=10284&pageSize=20&sort=age&order=desc
```

接口的返回值就是List结构：

```
[
   {
      "id":13483,
      "name":"Hehe",
      "title":"senior"
   },
    {
      "id":13484,
      "name":"haha",
      "title":"junior"
   }
]
```

由于是根据有序主键ID检索数据，因此性能更高，数据体量越大优势越明显。

字段选择

调用方可能只需要资源的少量属性，接口提供方可支持客户端通过请求参数fields来选择返回的字段多个字段以逗号分隔，示例如下：

```
GET /v1/employee?fields=id,name,age&pageNo=1&pageSize=20&sort=age&order=desc
```

五. 错误处理

根据REST规范URL即资源的原则，我们建议直接返回请求者预期的结果。在Http Status = 2xx 的情下，返回结果即资源，非200(2xx)均为异常，返回标准的异常消息体。对客户端来说，一个接口的响只有两种分支：onSuccess 和 onFail。

请求失败（非200）时，我们使用业务码来返回错误信息。

业务码

业务码(BizCode)是在业务处理过程中不符合预期时给客户端的响应信息，包含code和message。

请求出错时，Response payload 示例：

```
{
    "code": 210000, //业务码
    "message": "员工已离职", //具体的错误信息
    "path": "/v1/users/87812",  //请求路径
    "timestamp": 134212243332  //请求时间戳
}
```

选择合适的状态码

HTTP响应通常会返回一个重要的字段：status code。它说明了请求执行的大致情况，是否正常完成需要进一步处理、出现了什么错误，对于客户端非常重要。

2xx (请求成功处理并返回)

- **200 OK**

请求成功接收并处理，一般响应中都会有 body。

- **201 Created**

请求已完成，并导致了一个或者多个资源被创建，最常用在 POST 创建资源的时候

- **202 Accepted**

请求已经接收并开始处理，但是处理还没有完成。一般用在异步处理的情况，响应 body 中应该告诉户端去哪里查看任务的状态。

- **204 No Content**

请求已经处理完成，但是没有信息要返回，经常用在 PUT 更新资源的时候（客户端提供资源的所有性，因此不需要服务端返回）。如果有重要的 metadata，可以放到头部返回。

3xx (重定向)

- **304 Not Modified**

请求的资源和之前的版本一样，没有发生改变。用来缓存资源，和条件性请求（conditional request 一起出现

4xx (客户端错误)

- **400 Bad Request**

客户端发送的请求有错误（请求语法错误、非法参数、body数据格式有误、body 缺少必须的字段等，导致服务端无法处理。

- **401 Unauthorized**

无操作权限，请求的资源需要认证，客户端没有提供认证信息或者认证信息不正确。

- **403 Forbidden**

服务器已经理解请求，但是拒绝执行它。与401响应不同的是，身份验证并不能提供任何帮助，而且个请求也不应该被重复提交。

- **404 Not Found**

客户端要访问的资源不存在

- **405 Method Not Allowed**

请求行中指定的请求方法不能被用于请求相应的资源。例如，GET访问只接受POST请求的接口。

- **415 Unsupported Media Type**

服务端不支持客户端请求的资源格式，一般是因为客户端在 Content-Type 或者 Content-Encoding 中申明了希望的返回格式，但是服务端没有实现。比如，客户端希望收到 xml返回，但是服务端只支持 son。

- **417 Expectation Failed**

未满足期望值

- **429 Too Many Requests**

客户端在规定的时间里发送了太多请求，在进行限流的时候会用到。

5xx (服务器端错误)

- **500　Internal Server Error**

通用错误消息，服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。

- **502 Bad Gateway**

作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。

- **503　Service Unavailable**

由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是暂时的，并且将在一段时间后恢复。如果能够预计延迟时间，那么响应中可以包含一个Retry-After头用以标明这个延迟时间。如没有给出这个Retry-After信息，那么客户端应当以处理500响应的方式处理它。

- **504 Gateway Timeout**

作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例HTTP、FTP、LDAP）或者辅助服务器（例如DNS）收到响应。

更多HTTP状态码，请参考Wikipedia: Http状态码

六. 限流 Ratelimit

客户端的请求触发接口限流之后，要有方法告诉用户请求的使用情况，我们推荐返回以下Reponse Heder:

- X-RateLimit-Limit 每分钟/小时/天允许的请求量，不同接口限流的时间窗口和允许次数不一样
- X-RateLimit-Remaining 当前时间窗口里，还剩余多少请求次数
- X-RateLimit-Reset 多少秒后时间窗口重置，可选

通常情况下，客户端触发限流时，接口应返回状态码：429 too many quests，但特殊情况下，也可不响应此状态码。

举个例子，发送邮件接口，为了防止恶意软件检测到请求限流时自动切换账号，我们可以伪造正常Response，返回status code 200，但是不发送邮件。

七. 不符合 CRUD 的情况

在实际资源操作中，总会有一些不符合CRUD的情况，一般有几种处理方法。

使用 POST

为需要的动作增加一个 endpoint，使用 POST 来执行动作，比如 POST /resend 重新发送邮件。

增加控制参数

添加动作相关的参数，通过修改参数来控制动作。

比如一个博客网站，会有把写好的文章“发布”的功能，可以用上面的 POST /article/{:id}/publish 法，也可以在文章中增加 published:boolean 字段，发布的时候就是更新该字段 PUT /article/{:id}?published=true。

把动作转换成资源

把动作转换成可以执行 CRUD 操作的资源， github 就是用了这种方法。

比如"喜欢"一个 gist，就增加一个 /gists/:id/star 子资源，然后对其进行操作： “喜欢”使用 PUT /gists/:id/star，"取消喜欢"使用 DELETE /gists/:id/star 。

另外一个例子是 Fork，这也是一个动作，但是在 gist 下面增加 forks资源，就能把动作变成 CRUD 容的： POST /gists/:id/forks 可以执行用户 fork 的动作。

八. 提供完善的接口文档

API是面向开发者的，完善的接口文档，可以显著降低使用者的接入成本。

扩展阅读

- JSON API规范
- OpenAPI Specification
- Web API CheckList
- Azure - API design
- 微软 API开发指南
- Paypal API开发指南
- Best Practices for a pragmatic restful api
- Restful API Guidelines
- api-design-guidelines
- RFC7231 - Http Semantics and Content
- rest-api get resource with different field
- StackOverflow URI中单词分隔符的一些讨论