

## 服务端接口开发规范

### 1 引言

- 1.1 编写目的
- 1.2 术语、定义和缩略语
  - 1.2.1 术语
  - 1.2.2 定义
  - 1.2.3 缩略语

### 2 接口请求与响应

- 2.1 URL格式
- 2.2 请求方式
- 2.3 响应
- 2.4 JSON响应输出格式
  - 2.4.1 通用字段
  - 2.4.2 响应示例
  - 2.4.3 响应码定义

### 3 RESTful API 设计规范

- 3.1 两个URL
- 3.2 URL中不包含大写字母
- 3.3 推荐用复数名词
- 3.4 非资源请求用动词
- 3.5 用HTTP方法操作资源
- 3.6 API版本
- 3.7 对可选的、复杂的参数，使用查询字符串（? ）。
- 3.8 排序
- 3.9 HTTP状态码
- 3.10 JSON响应中使用小驼峰命名法

### 4 Java RESTful API开发规范

### 5 参考

(2018-05-28, created by [Zhu Yin](#))

# 服务端接口开发规范

---

## 1 引言

---

### 1.1 编写目的

---

编写本文档的目的在于规定服务端提供的REST接口。

本文档适用范围是程序设计人员、开发人员、测试人员。

### 1.2 术语、定义和缩略语

---

### 1.2.1 术语

术语	解释

### 1.2.2 定义

定义	描述

### 1.2.3 缩略语

缩略语	解释

## 2 接口请求与响应

### 2.1 URL格式

```
1 http[s]://{domain}::{port}{base_path}/{resource}[?
  {{query_string}}]
```

说明：

- `{{variable}}` 表示参数 `variable` 的值
- `domain` 为服务端的hostname或ip;
- `port` 为服务端的端口;
- `base_path` 为服务端的基本路径;
- `resource` 为操作的资源;
- `query_string` 由通用参数部分和具体API调用参数部分组成;
- `query_string` 中的key/value对都必须是UTF-8编码。
- 对于GET请求，`query_string` 必须放在QUERY参数中传递，即放在“?”后面;
- 对于POST请求，`query_string` 放在POST参数中传递;
- 本文中同一接口的 HTTP 和 HTTPS 版参数一致，请按需使用。

### 2.2 请求方式

查询类接口为GET方式。

## 2.3 响应

如果不明确指定，响应均为json响应。

## 2.4 JSON响应输出格式

响应数据包的格式为JSON，输出内容为UTF-8编码。

### 2.4.1 通用字段

所有交互正常的响应，都会有 `id`，`succeed`，`code`，`msg`，`data`，`time` 等字段。

服务端实际响应的数据需要放入到 `data` 字段中返回给调用者。

字段	类型	描述
id	long	请求/响应的全局唯一id，由web server生成
succeed	bool	标识HTTP API是否执行成功
code	int	响应码
msg	string	提示信息，如当调用失败时的错误消息
data	json	是一个二级json，由n个包含key和value属性的对象组成，用于封装API返回的数据内容
time	long	web server生成响应消息的unix时间戳, 精确到毫秒

若服务端处理请求时发生异常，则服务端会将异常的错误堆栈写到 `data` 的 `errorStack` 字段，并返回给调用者。

### 2.4.2 响应示例

响应ok

```

1  {
2      "id": "chookin.mac-20180525_162239-0",
3      "succeed": true,
4      "code": 0,
5      "msg": "",
6      "data": {
7          "page": 1,
8          "pageCount": 1,
9          "size": 3,
10         "total": 3
11     },
12     "time": 1527236559855
13 }

```

## 响应错误

```

1  {
2      "id": "chookin.mac-20180525_162336-0",
3      "succeed": false,
4      "code": 40001,
5      "msg": "java.lang.NullPointerException: page",
6      "data": {
7          "method": "GET",
8          "paras": {
9              "month": ["201712"],
10             "step": ["10"]
11         },
12         "path": "/api/app-active"
13     },
14     "time": 1527236616757
15 }

```

## 服务端发生异常时

```

1  {
2      "id": "chookin-imac.local-20180525_162336-3",
3      "succeed": false,
4      "code": -2,
5      "msg": "Timeout waiting for value: waited 2,500 ms. Node status:
Connection Status { localhost/127.0.0.1:11211 active: false, authed:
true, last read: 51,492 ms ago }",
6      "data": {
7          "errorStack": [
8              {
9                  "declaringClass": "net.spy.memcached.MemcachedClient",
10                 "methodName": "get",
11                 "fileName": "MemcachedClient.java",

```

```
12         "lineNumber": 1240
13     },
14     ...
15     {
16         "declaringClass": "java.lang.Thread",
17         "methodName": "run",
18         "fileName": "Thread.java",
19         "lineNumber": 745
20     }
21 ]
22 },
23 "time": 1527236616757
24 }
```

### 2.4.3 响应码定义

响应码	描述
0	响应正确
-1	系统繁忙
-2	服务器内部错误
2xx-5xx	HTTP状态码
20000	文件错误
40000	HTTP方法错误
40001	参数错误
40002	数据过期
40003	没有权限

## 3 RESTful API 设计规范

### 3.1 两个URL

资源集合用一个URL，具体某个资源用一个URL：

```
1  /employees      #资源集合的URL
2  /employees/56   #具体某个资源的URL
```

RESTful API 中的url是指向资源的，而不是描述行为的，因此设计API时，应使用名词而非动词来描述语义。

```
1 GET /employees
2 GET /employees?state=external
3 POST /employees
4 PUT /employees/56
```

## 3.2 URL中不包含大写字母

URL路径是对大小写敏感的，为与惯例一致，要求：

- `resource` 使用小写+中划线格式，如 `app-active`。

## 3.3 推荐用复数名词

推荐：

```
1 /employees
2 /employees/21
```

不推荐：

```
1 /employee
2 /employee/21
```

复数形式更为常见、直观，例如 `GET /employees?state=external`、`POST /employees`、`PUT /employees/56`。避免复数和单数名词混合使用，这显得非常混乱且容易出错。

## 3.4 非资源请求用动词

有时API调用并不涉及资源（如计算，翻译或转换）。例：

```
1 GET /translate?from=de_DE&to=en_US&text=Hallo
2 GET /calculate?para2=23&para2=432
```

在这种情况下，API响应不会返回任何资源。而是执行一个操作并将结果返回给客户端。因此，您应该在URL中使用动词而不是名词，来清楚的区分资源请求和非资源请求。

## 3.5 用HTTP方法操作资源

使用URL指定你要用的资源。使用HTTP方法来指定怎么处理这个资源。使用四种HTTP方法POST，GET，PUT，DELETE可以提供CRUD功能（创建，获取，更新，删除）。

- **获取**：使用GET方法获取资源。GET请求从不改变资源的状态。无副作用。GET方法是幂等的。GET方法具有只读的含义。因此，你可以完美的使用缓存。
- **创建**：使用POST创建新的资源。
- **更新**：使用PUT更新现有资源。
- **删除**：使用DELETE删除现有资源。

2个URL乘以4个HTTP方法就是一组很好的功能。看看这个表格：

	POST（创建）	GET（读取）	PUT（更新）	DELETE（删除）
/employees	创建一个新员工	列出所有员工	批量更新员工信息	删除所有员工
/employees/56	（错误）	获取56号员工的信息	更新56号员工的信息	删除56号员工

## 3.6 API版本

关于[API的版本是否应该包含在URL或者请求头中](#) 莫衷一是。

有两种方式：

1，版本号拼接在 URL 中。如：

```
1 api.example.com/v1/users
```

2，版本信息放在 Header 中：

```
1 api.example.com/users
2
3 version=v1
```

本规范要求，URL中默认不包含版本号。对于明确使用老版本的api，采用方式1，即版本号拼接在URL中。

## 3.7 对可选的、复杂的参数，使用查询字符串（?）。

不推荐做法：

```
1 GET /employees
2 GET/external_employees
3 GET /internal_employees
4 GET /internal_and_senior_employees
```

为了让你的URL更小、更简洁。为资源设置一个基本URL，将可选的、复杂的参数用查询字符串表示。

```
1 GET /employees?state=internal&maturity=senior
```

## 3.8 排序

排序参数采取逗号分隔的字段列表的形式，`-`表示降序。例如：

- `GET /tickets?sort=-priority` 获取票据列表，按 `priority` 降序排序；
- `GET /tickets?sort=-priority,created_time` 获取票据列表，按 `priority` 字段降序排序，并进一步按照 `created_time` 升序排序。

## 3.9 HTTP状态码

HTTP定义了很多有意义的[状态码](#)，你可以在你的API中使用。这些状态码可以帮助API消费者用来路由它们获取到的响应内容。整理了一个你肯定会用到的状态码列表：

- 200 OK - 对成功的GET、PUT、PATCH或DELETE操作进行响应。也可以被用在不创建新资源的POST操作上
- 201 Created - 对创建新资源的POST操作进行响应。应该带着指向新资源地址的[Location header](#)
- 204 No Content - 对不会返回响应体的成功请求进行响应（比如DELETE请求）
- 304 Not Modified - HTTP缓存header生效的时候用
- 400 Bad Request - 请求异常，比如请求中的body无法解析
- 401 Unauthorized - 没有进行认证或者认证非法。当API通过浏览器访问的时候，可以用来弹出一个认证对话框
- 403 Forbidden - 当认证成功，但是认证过的用户没有访问资源的权限
- 404 Not Found - 当一个不存在的资源被请求
- 405 Method Not Allowed - 所请求的HTTP方法不允许当前认证用户访问
- 410 Gone - 表示当前请求的资源不再可用。当调用老版本API的时候很有用
- 415 Unsupported Media Type - 如果请求中的内容类型是错误的
- 422 Unprocessable Entity - 用来表示校验错误
- 429 Too Many Requests - 由于请求频次达到上限而被拒绝访问
- 500 internal server error - 通用错误响应
- 503 service unavailable - 服务当前无法处理请求。

## 3.10 JSON响应中使用小驼峰命名法

使用小驼峰命名法作为属性标识符。

```
1 { "yearOfBirth": 1982 }
```

# 4 Java RESTful API开发规范



1. 对于必须参数，使用 `@RequestParam` 注解明确说明该参数是required的；
2. java程序中的参数采用驼峰命名；
3. 若参数包含大写字母，使用 `@RequestParam` 注解做转换，例如 `@RequestParam("utype")`。

## 5 参考

---

- [微信公众平台开发者文档](#)
- [RESTful API 设计最佳实践](#)
- [跟着 Github 学习 Restful HTTP API 设计](#)