# API 命名规范

## 一、通用

1. 面向资源（领域）设计，从业务中提取概念，按概念间的关联程度定义出领域，API 定义尽量面向领域；
2. 函数名使用动宾结构，命名遵循 “操作+资源” 的风格；

   正例：saveOrder、removeOrder、updateOrderItem
3. 术语统一，对资源的操作用词在相同场景下要统一，资源命名当然也要统一。常见的有：check 、paging、save、remove、query、update、list、get；
4. 为了达到代码自解释的目标，任何自定义编程元素在命名时，使用尽量完整的单词组合来表达；

   反例：int a；
5. 类名、函数、参数命名均不能使用特殊符号，如下划线或美元符等；

   反例：name_ 、$name
6. 所有命名严禁使用拼音和中文；

   正例：aliyun/ taobao / hunan 等国际通用的名称，可视同英文；

   反例：DaZhePromotion [打折] / getPingfenByName() [评分]
7. 类名、接口名使用 UpperCamelCase(大驼峰) 风格，但以下情形例外：DO / BO / DTO / VO / AO / PO / UID 等；

   正例：ForceCode / UserDO / HtmlDTO / XmlService / TcpUdpDeal / TaPromotion

   反例：forcecode / UserDo / HTMLDto / XMLService / TCPUDPDeal / TAPromotion
8. 函数名、参数名统一使用 lowerCamelCase(小驼峰) 风格；

   正例： localValue / getHttpMessage() / inputUserId
9. 枚举属性全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌麻烦；

   正例：SUCCESS / UNKNOWN_REASON
10. POJO 类中的任何布尔类型的变量，都不要加 is 前缀，否则部分框架解析会引起序列化错误；

    反例：定义为基本数据类型 Boolean isDeleted 的属性，它的方法也是 isDeleted()
11. 杜绝完全不规范的缩写，避免望文不知义；

    反例：AbstractClass “缩写” 命名成 AbsClass；condition “缩写” 命名成 condi，此类随意缩写严重降低了代码的可阅读性。

## 二、RESTful

1. RESTful 风格的 API 要求面向资源设计，具体来说，URI 需要遵循如下格式：

   父资源/[子资源]/[参数]/[操作]

不同的请求方式表示不同的操作类型，单独使用GET、POST、PUT这些无法准确表意时，需自定义操作。如：

| 请求方式 | URI | 描述 |
| --- | --- | --- |
| GET | https://api.shiguangkey.com/users | 获取用户列表 |
| GET | https://api.shiguangkey.com/user/{id} | 获取某个用户详情 |
| GET | https://api.shiguangkey.com/goods/paging | 分页获取商品列表 |
| PUT | https://api.shiguangkey.com/goods | 更新某个商品详情 |

备注：使用资源的单复数来区分返回结果是单个还是多个。资源由多个单词组成的话，使用“_”分割

2. Http 请求方式具有特定含义，选择使用合适的请求方式：
    - GET：从服务器取出资源（一项或多项）；
    - POST：在服务器新建一个资源；
    - PUT：在服务器更新资源（客户端提供改变后的完整资源）；
    - DELETE：从服务器删除资源。
3. Http 请求有多种方式来传递参数，需根据不同的请求方式使用响应的参数传递方式。
    - Query 主要用于 GET 方式的简单参数传递，如：?limit=10&offset=10；
    - Body 主要用于表单信息提交（POST、PUT）；
    - Header 用于存放请求数据格式、编码、sign、token等信息。