

# 目录

前言	1.1
Postman简介	1.2
Postman下载	1.3
Postman功能: Request	1.4
新建Request	1.4.1
JSON语法检查	1.4.2
GET的Request的多参数	1.4.3
给接口添加描述	1.4.4
Postman功能: Response	1.5
Response数据显示模式	1.5.1
Response其他功能	1.5.2
保存多个Example	1.5.3
Postman功能: 其他工具和功能	1.6
分组Collection	1.6.1
历史记录History	1.6.2
用环境变量实现多服务器版本	1.6.3
代码生成工具	1.6.4
测试接口	1.6.5
Mock Server	1.6.6
Postman功能: 界面和配置	1.7
多Tab分页	1.7.1
界面查看模式	1.7.2
多颜色主题	1.7.3
Postman生成API文档	1.8
预览和发布API文档	1.8.1
附录	1.9
参考资料	1.9.1

# API开发利器：Postman

- 最新版本：`v2.0`
- 更新时间：`20190529`

## 鸣谢

感谢我的老婆陈雪雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

## 简介

在涉及HTTP方面的后台REST API开发时，往往会需要调试API接口。这方面有很多工具，其中最好用的算是接下来要介绍的Postman了。

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### Gitbook源码

- [crifan/api\\_tool\\_postman: API开发利器：Postman](#)

### 如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook\\_template: demo how to use crifan gitbook template and demo](#)

### 在线浏览

- [API开发利器：Postman book.crifan.com](#)
- [API开发利器：Postman crifan.github.io](#)

### 离线下载阅读

- [API开发利器：Postman PDF](#)
- [API开发利器：Postman ePub](#)
- [API开发利器：Postman Mobi](#)

## 版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间： 2019-05-29 21:44:39



# Postman简介

我们在后台开发期间，尤其是HTTP的RESTful API时，往往需要在实现了后台的接口代码后，找合适的工具去调试以验证和确保自己的API接口是可以正常工作的。

这时候，找到合适的API调试工具，就显得很重要。因为合适的工具可以极大地提高工作效率，减少生命的浪费。

之前用过一些工具，最终发现Postman是个非常好用的API调试工具，所以在此推荐之。

关于Postman Postman，其官网有言简意赅的介绍：

Postman helps you develop APIs faster

让（作为后台开发人员的）你开发API接口时更方便和快捷

-» 利用Postman：

- 可以方便的调试API接口
- 还可以把各个API内容发布为文档，方便其他（比如移动端等）人员查看接口详情

下面就来详细的解释Postman的各种功能的用法。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间：2017-12-29 11:59:21

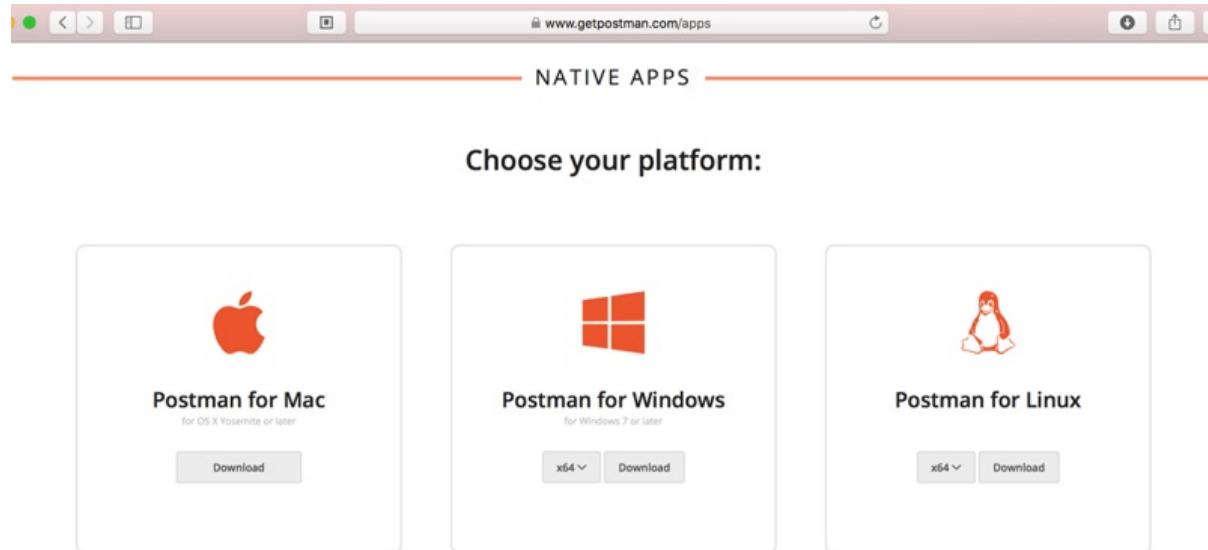
## Postman下载

Postman有两种形式：

- Chrome App: Postman for Chrome
  - 由于Chrome本身快要废弃Chrome App模式了，所以Postman的Chrome App模式也已废弃逐渐不用了
- app软件：建议下载（不同平台的）独立的软件去使用

下面主要介绍下载安装独立版本app软件的Postman的过程：

去主页[Postman | Supercharge your API workflow](#)找到：[Postman | Apps](#)



去下载自己平台的版本：

- Mac
- Windows (x86/x64)
- Linux (x86/x64) 即可。

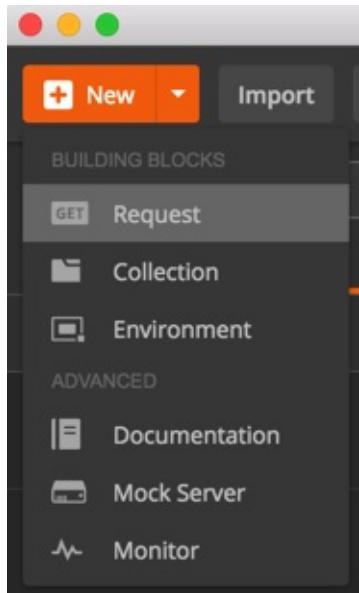
crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 13:50:27

# Postman功能：Request

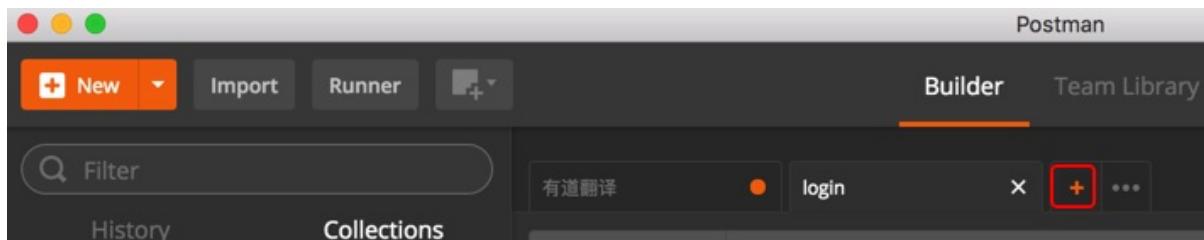
crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-27 17:37:01

## 新建Request

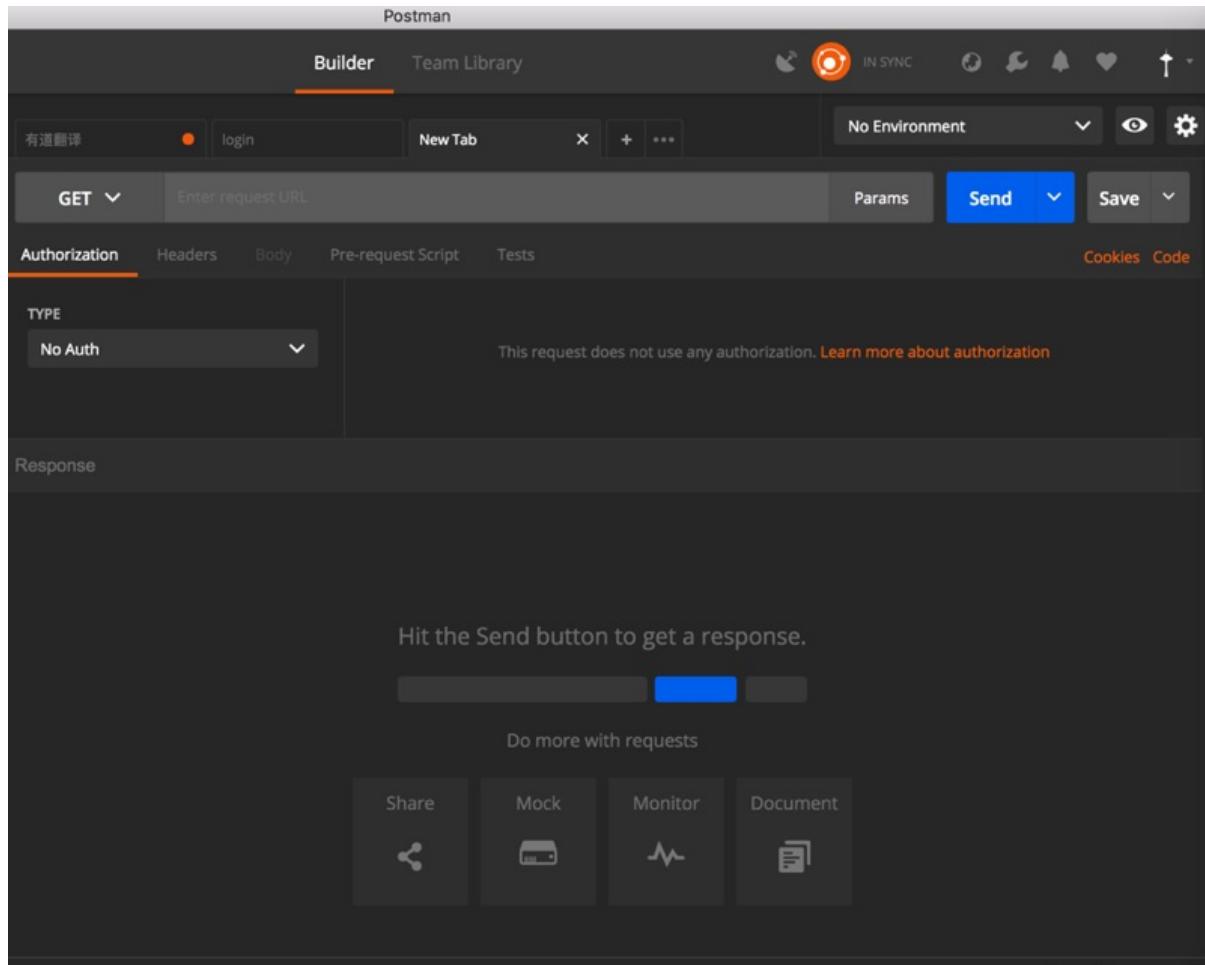
去新建接口，即对应的Request： New -> Request



或，在右边的Tab页面中点击加号+：

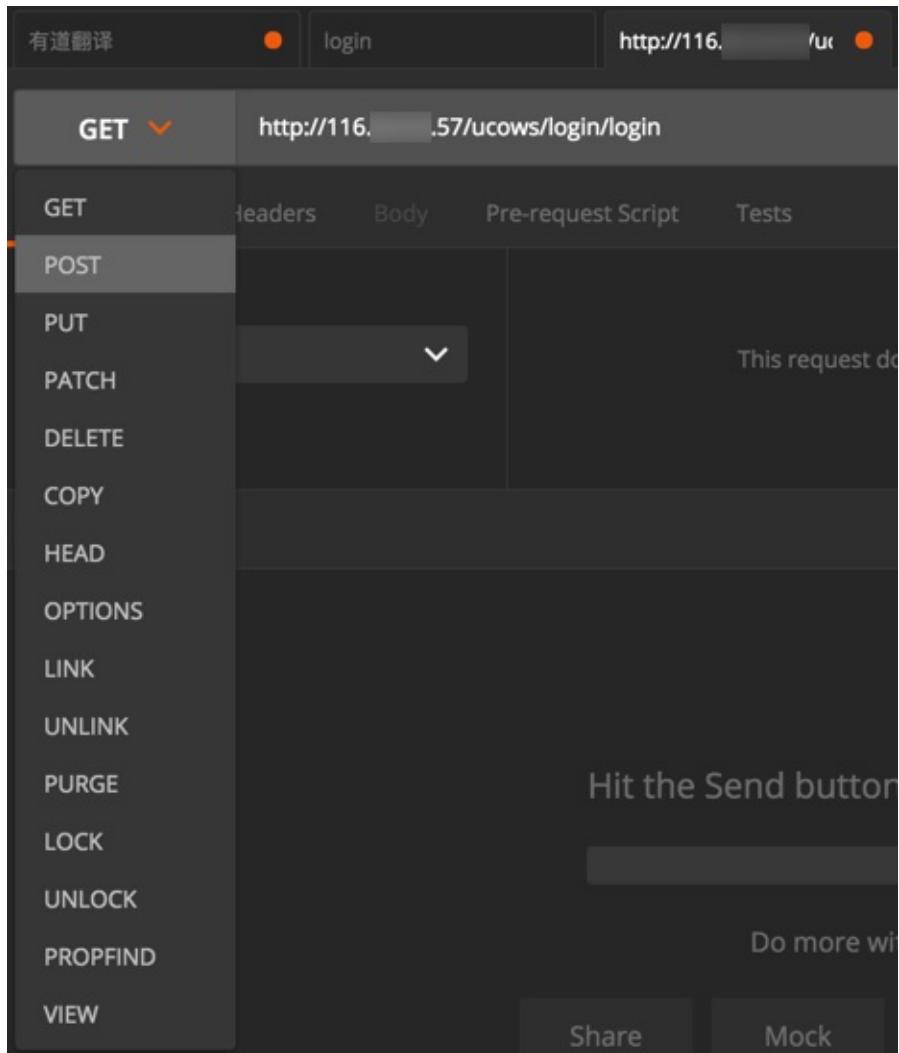


即可看到新建的Tab页：



然后：

- 设置HTTP的Method方法和输入api的地址



- 设置相关头信息

	Key	Value
Resp	content	
	Content-MD5	
	Content-Length	
	Content-Transfer-Encoding	
	Content-Type	

The screenshot shows the Postman interface with a POST request to `http://116.57/ucows/login/login`. The 'Headers' tab is selected, showing a single header `Content-Type` set to `application/json`. A dropdown menu is open over the value field, listing various options: application/atom+xml, application/ecmascript, application/json (selected), application/javascript, application/octet-stream, application/ogg, and application/pdf.

- 设置相关GET或POST等的参数

The screenshot shows the Postman interface with a POST request to `http://116.57/ucows/login/login`. The 'Body' tab is selected, showing the raw JSON payload:

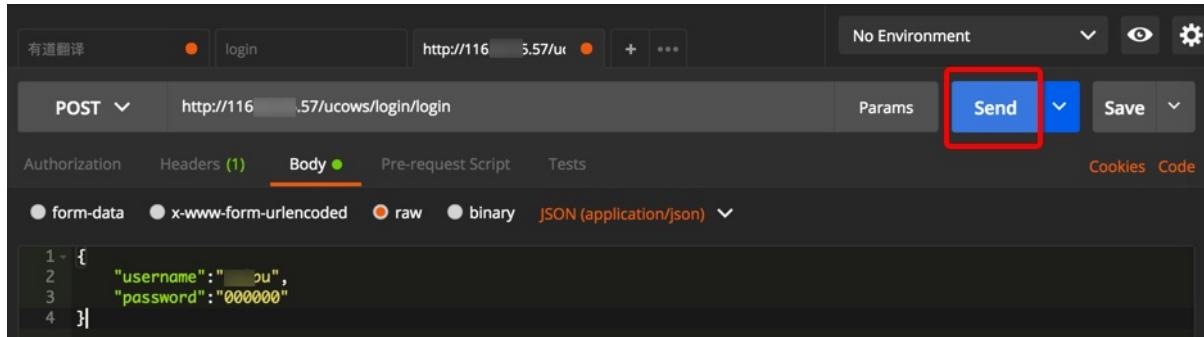
```

1 {
2   "username": "████pu",
3   "password": "000000"
4 }

```

A dropdown menu is open next to the `JSON (application/json)` label, listing other options: Text, Text (text/plain), JSON (application/json) (selected), Javascript (application/javascript), XML (application/xml), XML (text/xml), and HTML (text/html).

都填写好之后，点击Send去发送请求Request：



即可看到返回的响应Response的信息了：

The screenshot shows the Postman interface after sending the request. The response status is 200 OK. The JSON body of the response is displayed:

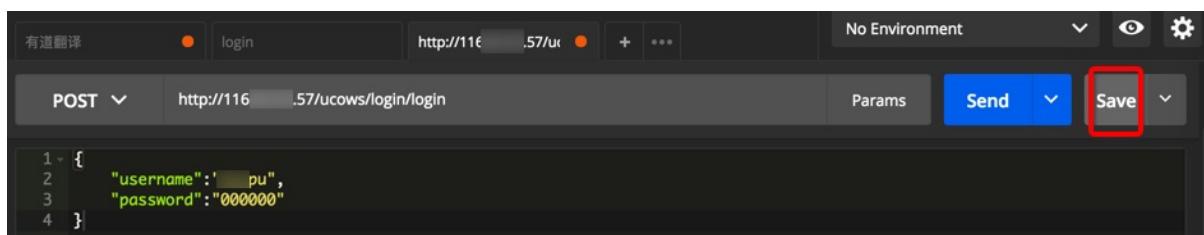
```

1 - {
2   "code": 200,
3   "message": "ok",
4   "data": {
5     "tokenid": "d4a638d",
6     "success": true,
7     "userId": "e5c7bf21",
8     "loginUserType": "cowfarmList",
9     "cowfarmList": [
10       {
11         "name": "乳业",
12         "id": "1",
13         "farm_address": "廊坊市",
14         "email": "weipu@sina.com",
15         "link_man": "甫"
16       }
17     ]
18   }
19 }

```

然后可以重复上述修改Request的参数，点击Send去发送请求的过程，以便调试到API接口正常工作为止。

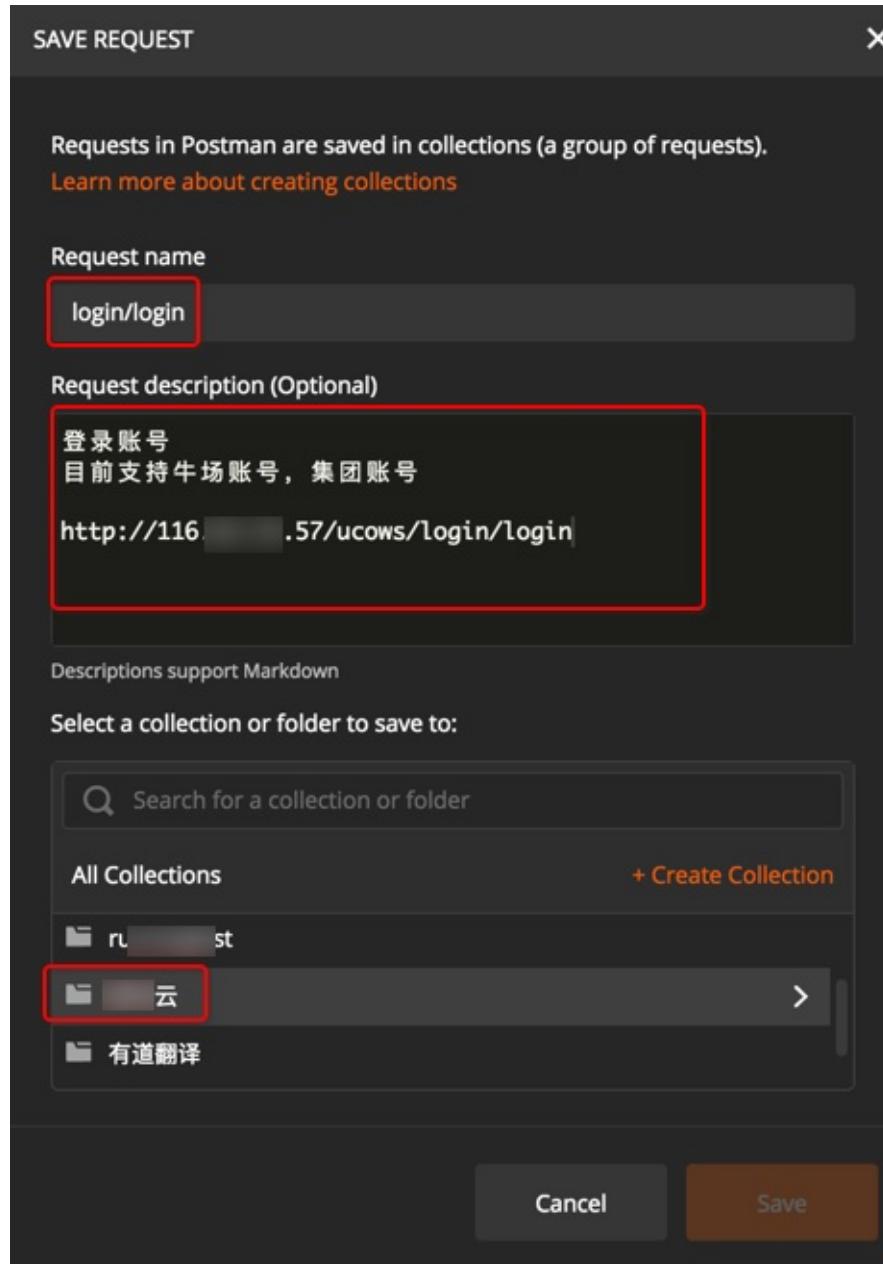
待整个接口都调试完毕后，记得点击Save去保存接口信息：



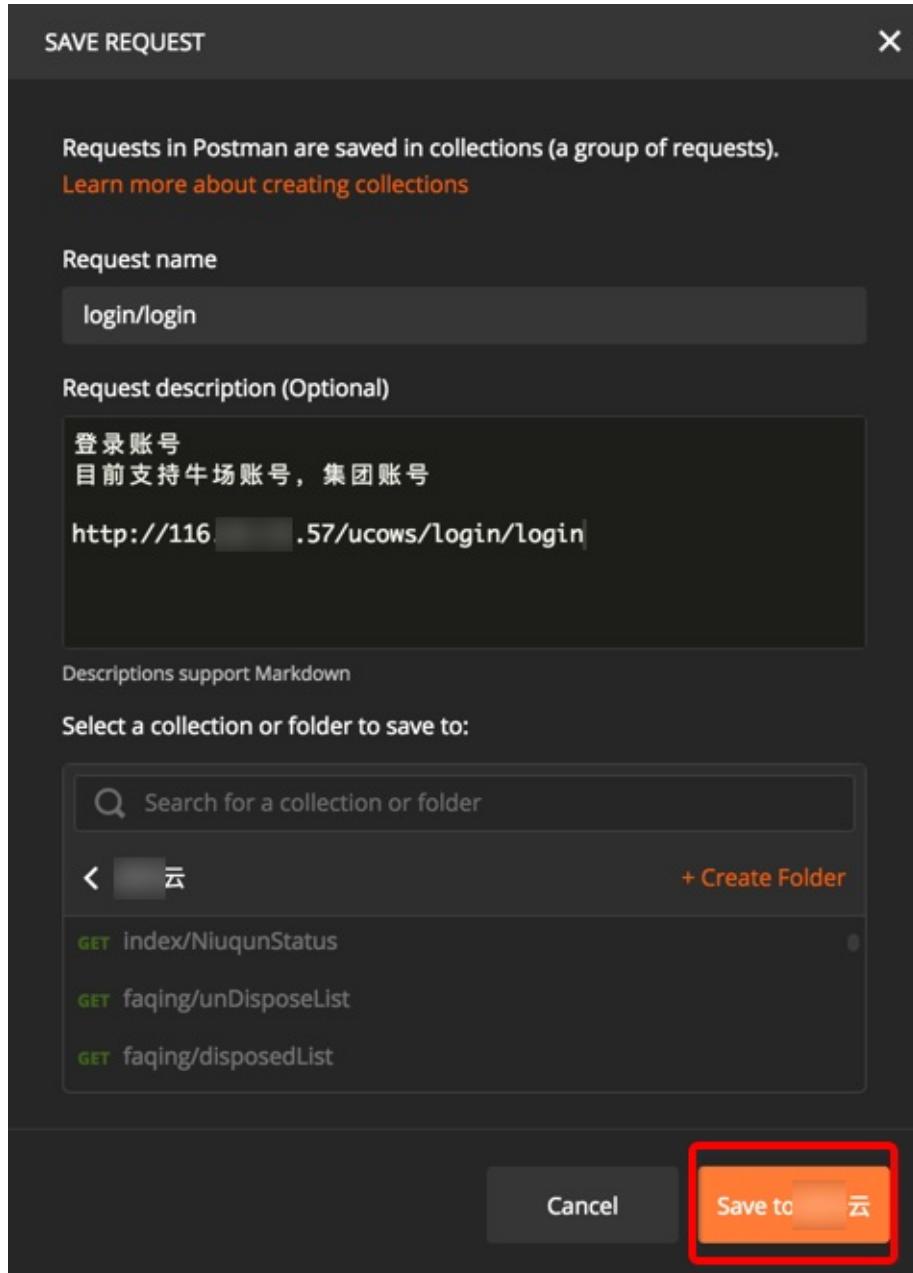
去保存当前API接口，然后需要填写相关的接口信息：

- Request Name: 请求的名字
  - 我一般习惯用保存为 接口的最后的字段名，比如 `http://{{server_address}}/ucows/login/login` 中的 `/login/login`
- Request Description: 接口的描述

- 可选 最好写上该接口的要实现的基本功能和相关注意事项
- 支持Markdown语法
- Select a collection or folder to save: 选择要保存到哪个分组（或文件夹）
  - 往往保存到某个API接口到所属的该项目名的分组

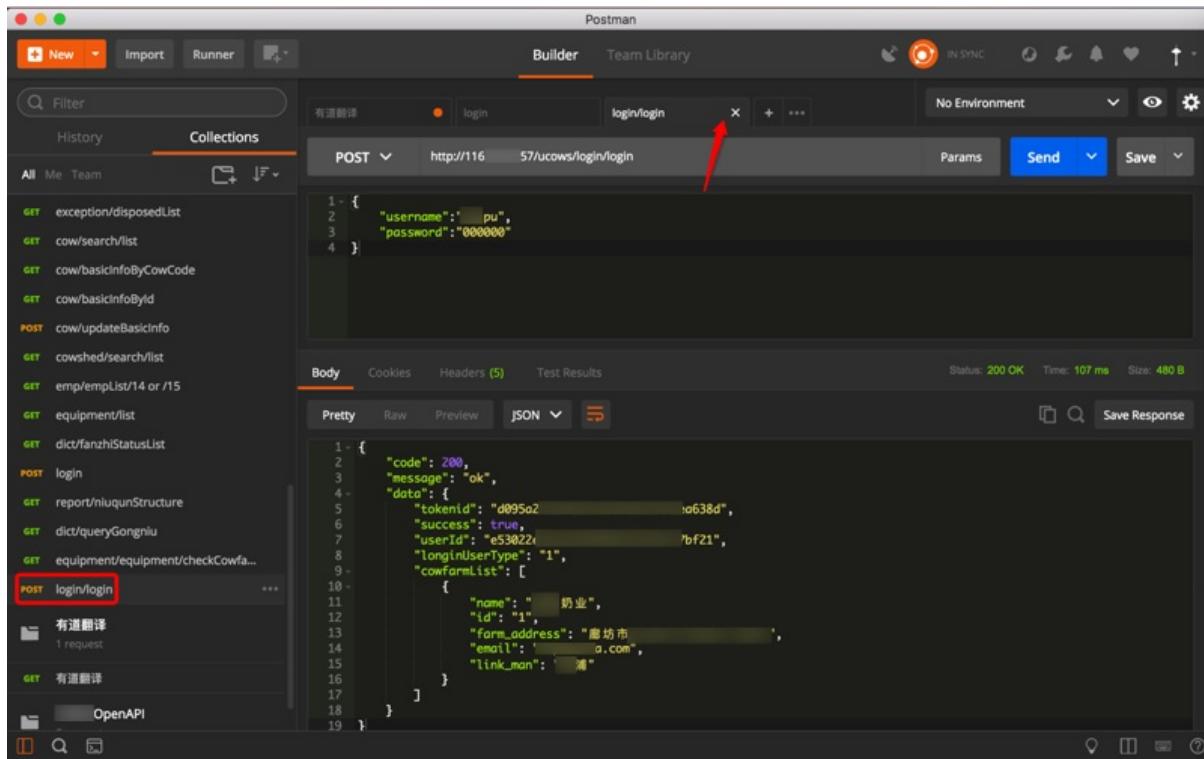


填写好内容，选择好分组，再点击保存：



此时，Tab的右上角的黄色点（表示没有保存）消失了，表示已保存。

且对应的分组中可以看到对应的接口了：



## 默认不保存返回的Response数据

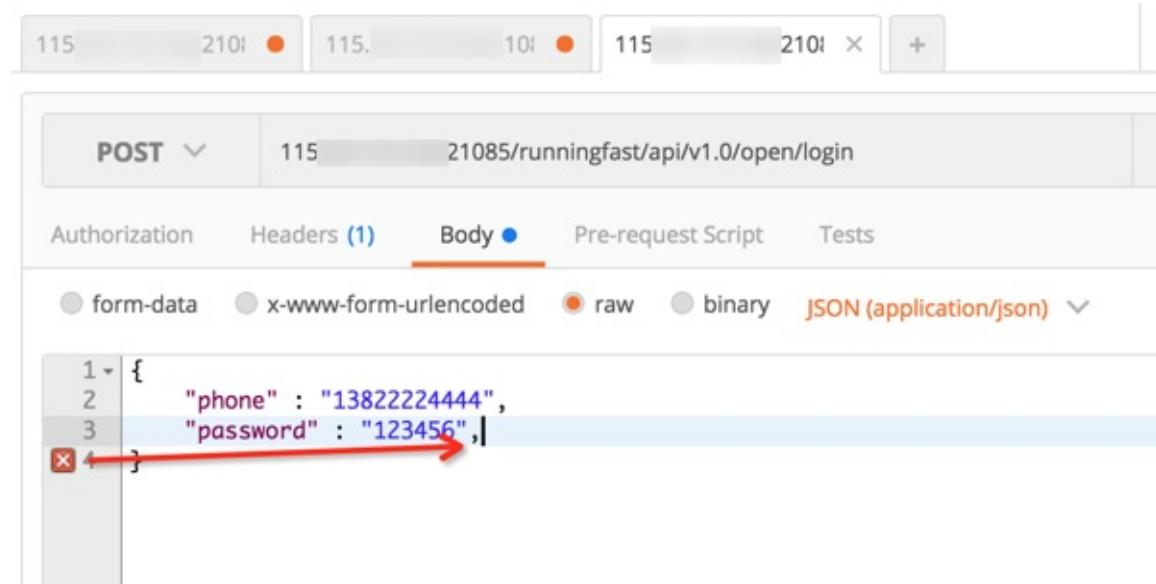
- 直接点击Save去保存，只能保存API本身（的Request请求），不会保存Response的数据
- 想要保存Response数据，需要用后面要介绍的 [多个Example](#)

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2018-06-16 11:13:05

## JSON语法检查

在写POST时的Body中的JSON参数时，如果语法出错会智能提示。

比如json的值的行末多余逗号：



crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间： 2017-12-29 20:09:11

# GET的Request的多参数

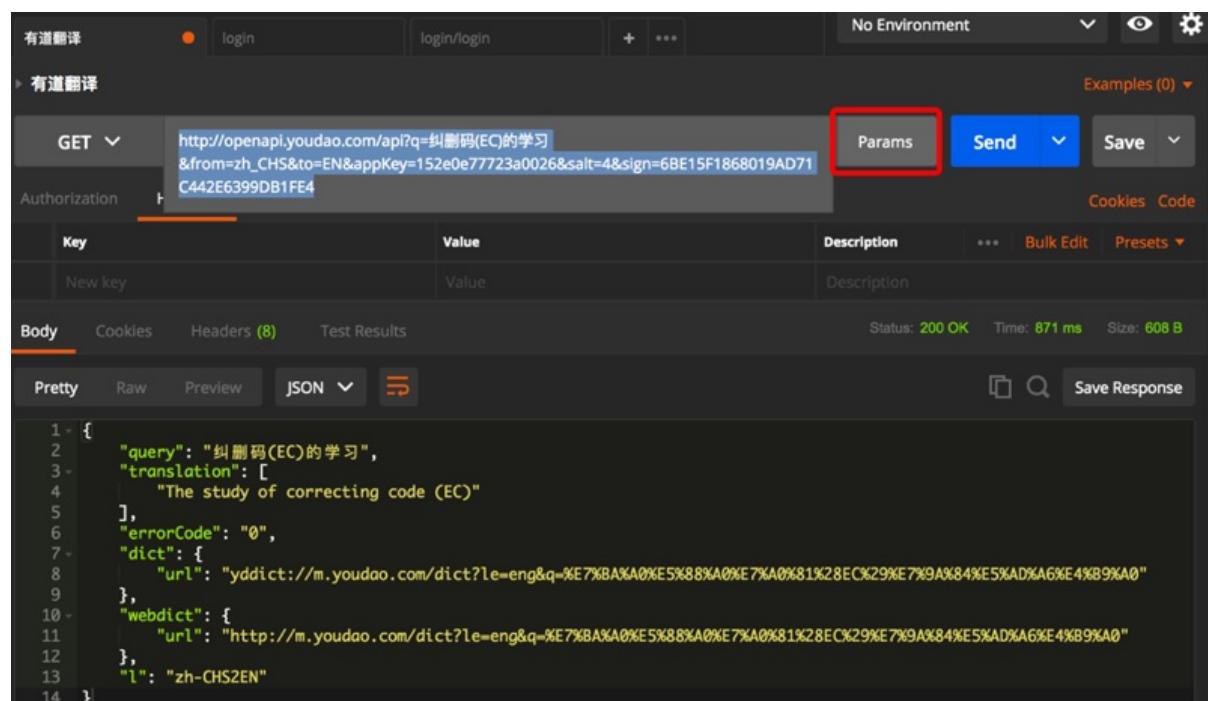
在GET请求中有多个参数需要处理的时候，Postman中有很多方便的手段去操作：

## 自动解析多个参数Params

比如，对于一个GET的请求的url是：  
`http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=152e0e77723a0026&salt=4&sign=6BE15F1868019AD71C442E6399DB1FE4`

对应着其实是 `?key=value` 形式中包含多个Http的GET的query string=query parameters

Postman可以自动帮我们解析出对应参数，可以点击Params：



The screenshot shows the Postman interface with a GET request to `http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=152e0e77723a0026&salt=4&sign=6BE15F1868019AD71C442E6399DB1FE4`. The 'Params' button in the top right is highlighted with a red box. The response body is shown in JSON format:

```

1 - {
2   "query": "纠删码(EC)的学习",
3   "translation": [
4     "The study of correcting code (EC)"
5   ],
6   "errorCode": "0",
7   "dict": {
8     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
9   },
10  "webdict": {
11    "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
12  },
13  "l": "zh-CHS2EN"
14 }

```

看到展开的多个参数：

The screenshot shows the Postman interface with a red box highlighting the 'Params' section. The table lists the following parameters:

Key	Value	Description
<input checked="" type="checkbox"/> q	纠删码(EC)的学习	
<input checked="" type="checkbox"/> from	zh_CHS	
<input checked="" type="checkbox"/> to	EN	
<input checked="" type="checkbox"/> appKey	152e0e77723a0026	
<input checked="" type="checkbox"/> salt	4	
<input checked="" type="checkbox"/> sign	6BE15F1868019AD71C442E6399DB1FE4	

Below the table, the 'Body' tab is selected, showing the JSON response:

```

1 - {
2   "query": "纠删码(EC)的学习",
3   "translation": [
4     "The study of correcting code (EC)"
5   ],
6   "errorCode": "0",

```

如此就可以很方便的修改，增删对应的参数了。

## 不勾选某些参数达到临时禁用的效果

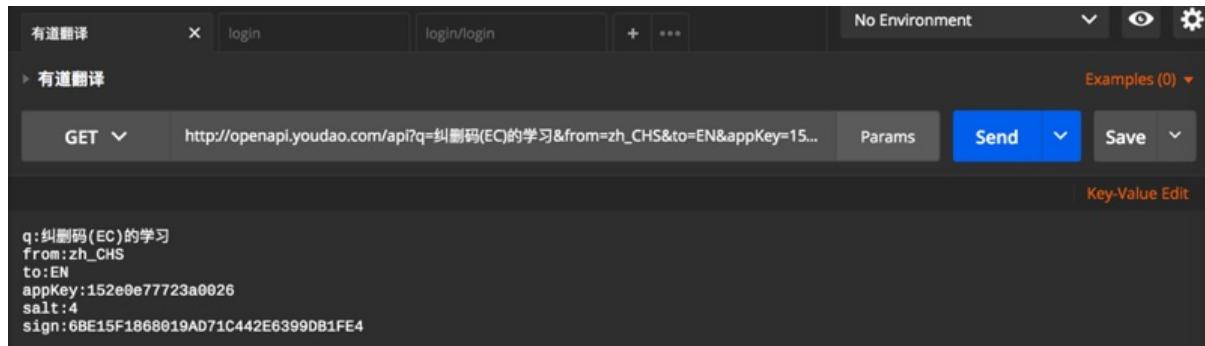
且还支持，在不删除某参数的情况下，如果想要暂时不传参数，可以方便的通过不勾选的方式去实现：

The screenshot shows the Postman interface with the 'Params' section. The table lists the following parameters:

Key	Value	Description
<input type="checkbox"/> q	纠删码(EC)的学习	
<input checked="" type="checkbox"/> from	zh_CHS	
<input type="checkbox"/> to	EN	
<input checked="" type="checkbox"/> appKey	152e0e77723a0026	
<input checked="" type="checkbox"/> salt	4	
<input checked="" type="checkbox"/> sign	6BE15F1868019AD71C442E6399DB1FE4	

## 批量编辑GET的多个参数

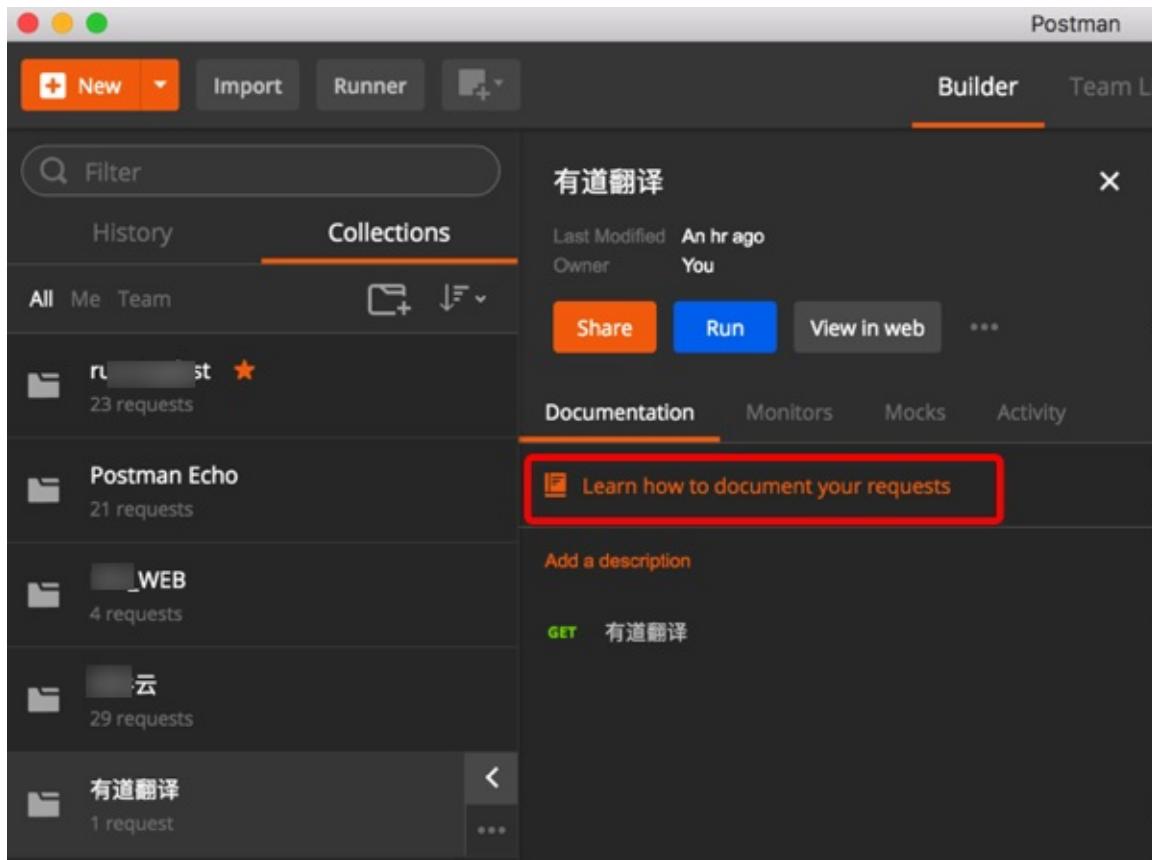
当然，如果想要批量的编辑参数，可以点击右上角的Bulk Edit，去实现批量编辑。



crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 20:19:54

## 给接口添加描述

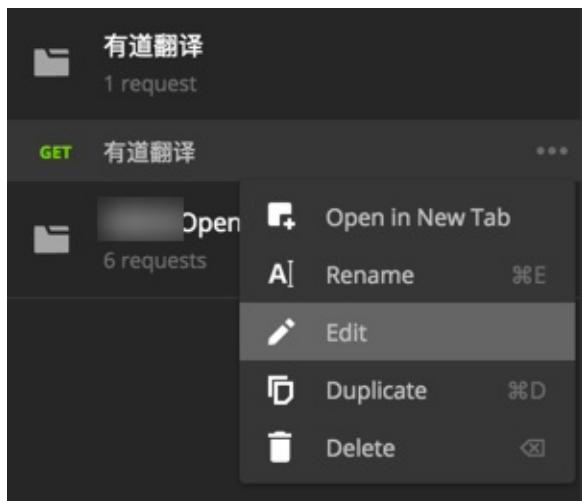
通过看到：



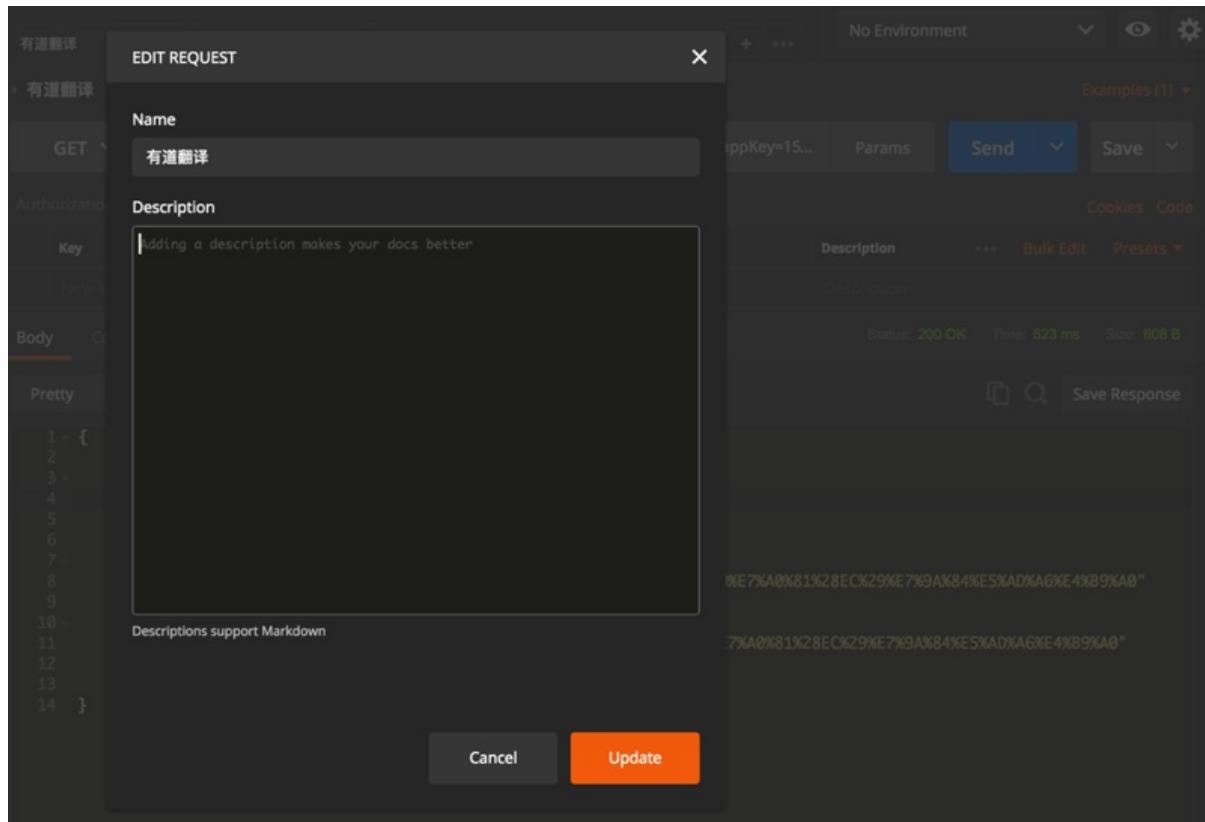
[Intro to API documentation](#)

得知，API的描述中，也支持Markdown的。

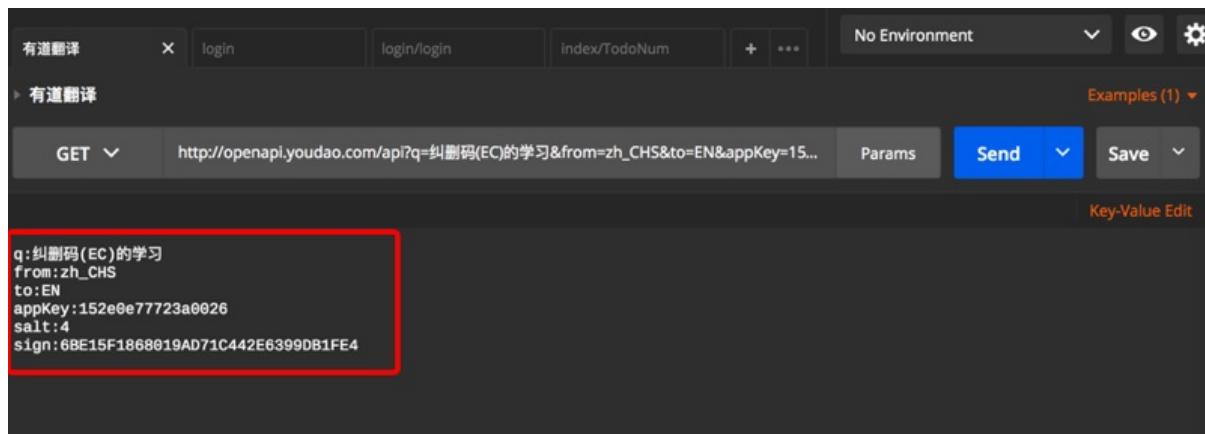
所以，可以很方便的添加有条理的接口描述，尤其是参数解释了：



可以看到 Descriptions support Markdown



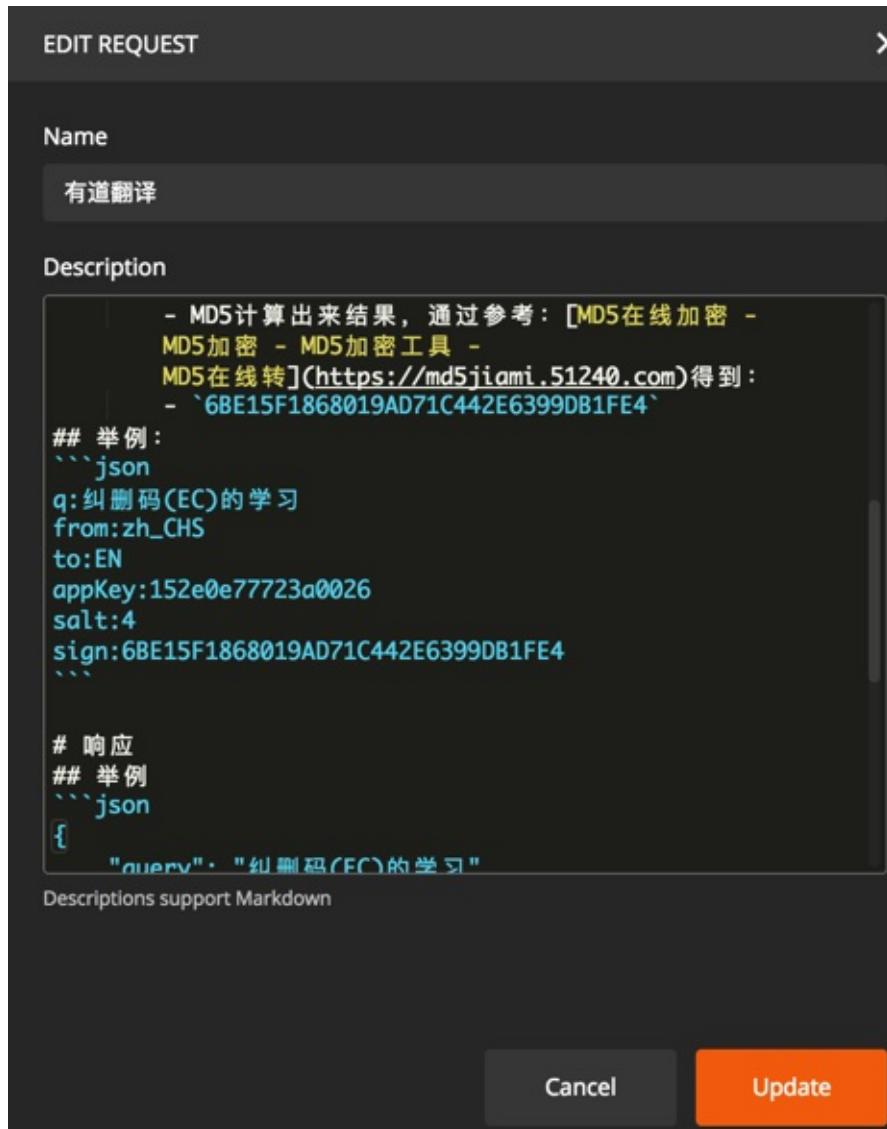
而对于要解释的参数，可以通过之前的 Param -> Bulk Edit 的内容：



拷贝过来，再继续去编辑：



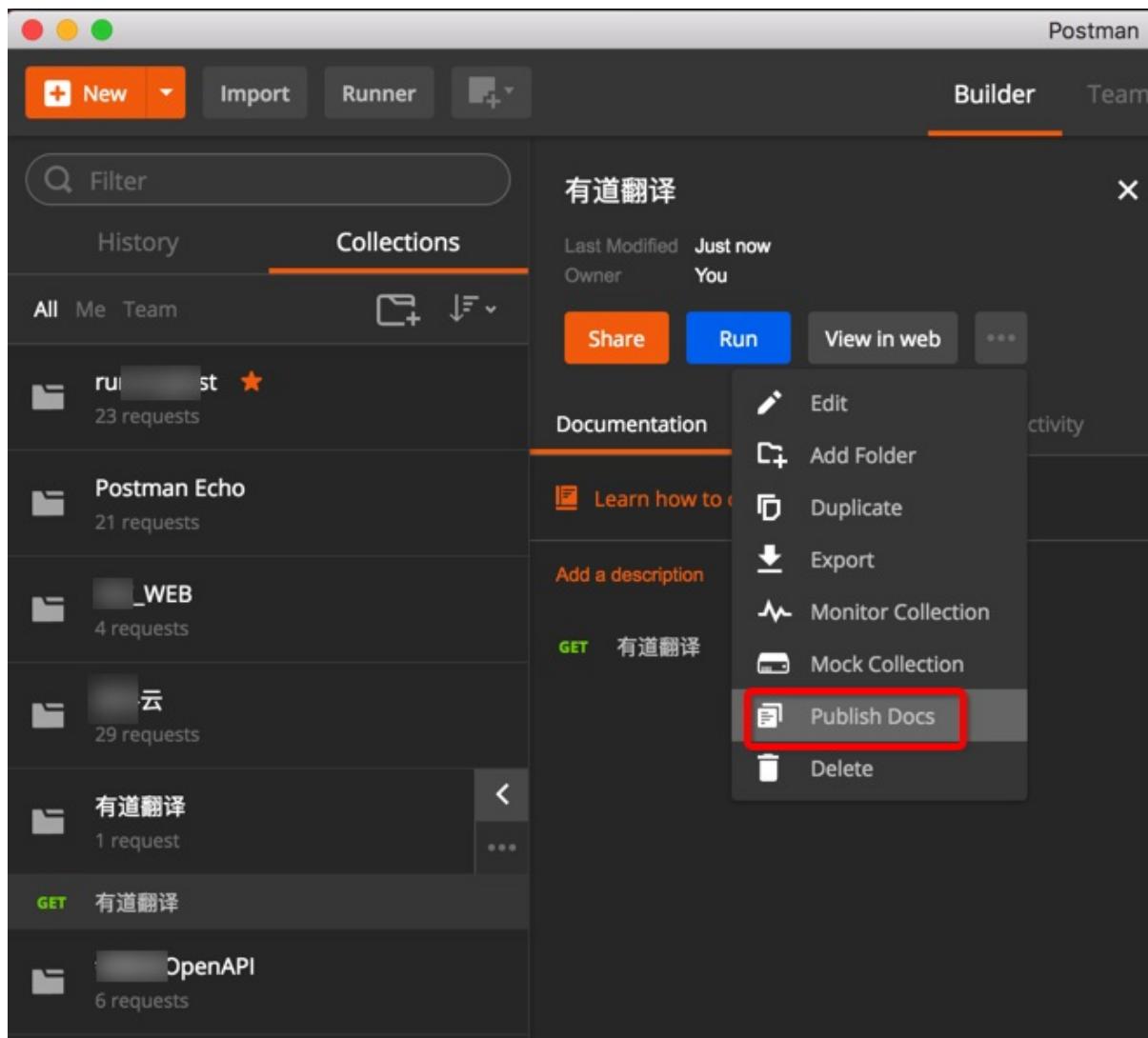
以及添加更多解释信息：



点击Update后，即可保存。

## 发布后带Markdown描述的API的效果

去发布后：



对应的效果: [有道翻译](#)

The screenshot shows the POSTMAN application interface. At the top, there are tabs for 'Mock resp', '奶牛云', 'Supercharge...', 'collection-ru...', '奶牛云', 'Documenter', 'Intro to API d...', '有道翻译', 'Documenter', and '有道翻译'. The main window title is 'documenter.getpostman.com/view/669382/collection/77fd4ek'. The collection name '有道翻译' is displayed. The collection description is '调用 有道 的API实现将 中文 翻译为 英文' (Call the Youdao API to translate Chinese to English). A sample request is provided:

```
curl --request GET \
--url "http://openapi.youdao.com/api?q=%E7%BA%A0%E5%88%A0%E7%A0%DB1FE4"
```

The sample response is:

```
{
  "query": "纠删码(EC)的学习",
  "translation": [
    "The study of correcting code (EC)"
  ],
  "errorCode": "0",
  "dict": {
    "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%DB1FE4"
  }
}
```

A red box highlights the '请求参数' (Request Parameters) section, which lists the following parameters:

- **q**: query string, 查询字符串
- **from**: 从什么语言, 原始语言
- **to**: 翻译为, 目标语言
- **appKey**: 自己 有道智云 账号中创建的应用的 应用ID
- **salt**: 给MD5算法加的 盐 , 一般用任意随机数即可
- **sign**: MD5加密后的值=md5(appKey+q+salt+密钥)
  - 经过验证, 好像大小写均可
    - 举例:
      - appKey=应用ID=152e0e77723a0026
      - q=纠删码(EC)的学习

有道翻译

Introduction

GET 有道翻译

```
q:纠删码(EC)的学习
from:zh_CHS
to:EN
appKey:152e0e77723a0026
salt:4
sign:6BE15F1868019AD71C442E6399DB1FE4
```

## 响应

### 举例

```
{
  "query": "纠删码(EC)的学习",
  "translation": [
    "The study of correcting code (EC)"
  ],
  "errorCode": "0",
  "dict": {
    "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%AE%A9%E6%8B%8D%E6%95%AF"
  },
  "webdict": {}Click to Expand
```

PARAMS

<b>q</b>	纠删码(EC)的学习
<b>from</b>	zh_CHS
<b>to</b>	EN
<b>appKey</b>	152e0e77723a0026
<b>salt</b>	4
<b>sign</b>	6BE15F1868019AD71C442E6399DB1FE4

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2018-01-02 11:01:53

## Postman功能：Response

Postman中对于Response响应，也有很多方便好用的功能。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 20:20:38

# Response数据显示模式

Postman对于返回的Response数据，支持三种显示模式：

## 默认 格式化后的Pretty模式

The screenshot shows the Postman interface with a request to `http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=15...`. The response body is displayed in Pretty mode, which formats the JSON data with indentation and line breaks. The JSON output is:

```

1 - {
2   "query": "纠删码(EC)的学习",
3   "translation": [
4     "The study of correcting code (EC)"
5   ],
6   "errorCode": "0",
7   "dict": {
8     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
9   },
10  "webdict": {
11    "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
12  },
13  "l": "zh-CHS2EN"
14 }

```

## Raw原始模式

点击Raw，可以查看到返回的没有格式化之前的原始数据：

The screenshot shows the Postman interface with the same request and response. The response body is displayed in Raw mode, which shows the raw JSON data without any formatting. The JSON output is identical to the Pretty mode output:

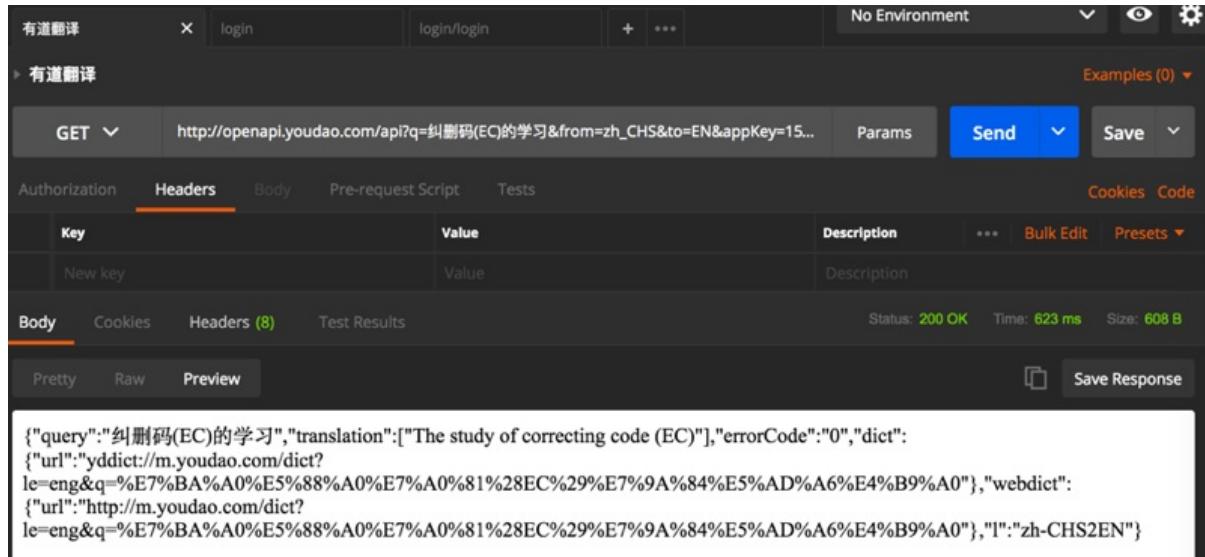
```

{"query":"纠删码(EC)的学习","translation":["The study of correcting code (EC)"],"errorCode":"0","dict":{"url":"yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"},"webdict":{"url":"http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"},"l":"zh-CHS2EN"}

```

## Preview预览模式

以及Preview，是对应Raw原始格式的预览模式：



The screenshot shows the Postman interface with a request to `http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=15...`. The Headers tab is selected, and the Body tab is selected under the Preview section. The response body is displayed as:

```
{"query":"纠删码(EC)的学习","translation":["The study of correcting code (EC)"],"errorCode":"0","dict": {"url":"yddict./m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"}, "webdict": {"url":"http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"}, "l":"zh-CHS2EN"}
```

Preview这种模式的显示效果，好像是对于返回的是html页面这类，才比较有效果。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间： 2019-03-27 17:30:01

# Response其他功能

## Response的Cookies

很多时候普通的API调用，倒是没有Cookie的：

The screenshot shows the Postman application interface. At the top, there's a header bar with tabs for 'login' and 'login/login'. Below the header, the URL is set to `http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=15...`. The main workspace has tabs for 'Body', 'Headers (8)', and 'Test Results', with 'Headers' currently selected. On the right side, there are buttons for 'Send', 'Save', and environment dropdowns. Below the tabs, there's a table for managing cookies, with columns for 'Key', 'Value', 'Description', and 'Bulk Edit'. A note below the table says 'No cookie for you'. At the bottom right, the response details are shown: Status: 200 OK, Time: 623 ms, Size: 608 B.

## Response的Headers头信息

举例，此处返回的是有Headers头信息的：

The screenshot shows a Postman interface with the following details:

- Request Method: GET
- URL: [http://openapi.youdao.com/api?q=纠删码\(EC\)的学习&from=zh\\_CHS&to=EN&appKey=15...](http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=15...)
- Headers tab selected
- Headers listed:
  - Connection → keep-alive
  - Content-Encoding → gzip
  - Content-Type → application/json;charset=UTF-8
  - Date → Thu, 02 Nov 2017 02:23:06 GMT
  - Server → nginx
  - Transfer-Encoding → chunked
  - Vary → Accept-Encoding
  - X-Application-Context → application:8686
- Status: 200 OK
- Time: 623 ms
- Size: 608 B

可以从中看到服务器是Nginx的。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 20:30:27

## 保存多个Example

之前想要实现，让导出的API文档中能看到接口返回的Response数据。后来发现是Example这个功能去实现此效果的。

### 如何添加Example

The screenshot shows the Postman application interface. At the top, there's a header bar with tabs for 'login' and 'login/login'. Below the header, the main workspace shows a GET request to 'http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh\_CHS&to=EN&appKey=15...'. The 'Headers' tab is selected, showing a single entry 'New key' under 'Key' with an empty 'Value'. The 'Body' tab is also selected, displaying a JSON response with code highlighting. On the right side, a sidebar titled 'Examples (0)' is open, containing a message 'No examples added' and a button labeled 'Add Example' which is highlighted with a red box.

```
1 - {
2   "query": "纠删码(EC)的学习",
3   "translation": [
4     "The study of correcting code (EC)"
5   ],
6   "errorCode": "0",
7   "dict": {
8     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
9   },
10  "webdict": {
11    "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
12  },
13  "l": "zh-CHS2EN"
14 }
```

继续点击Save Example：

The screenshot shows the Postman interface with a saved example. The top right corner has a red box around the "Save Example" button. The "EXAMPLE REQUEST" section shows a GET request to [http://openapi.youdao.com/api?q=纠删码\(EC\)的学习&from=zh\\_CHS&to=EN&appKey=152e0e77723a0026&salt=4&sig...](http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=152e0e77723a0026&salt=4&sig...). The "EXAMPLE RESPONSE" section shows a 200 OK status with a JSON response body:

```

1 - {
2 -   "query": "纠删码(EC)的学习",
3 -   "translation": [
4 -     "The study of correcting code (EC)"
5 -   ],
6 -   "errorCode": "0",
7 -   "dict": {
8 -     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%AA%81%28EC%29%E7%9A%84%E5%AD%A6%E4%89%A0"
9 -   },
10 -   "webdict": {
11 -     "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%AA%81%28EC%29%E7%9A%84%E5%AD%A6%E4%89%A0"
12 -   },
13 -   "l": "zh-CHS2EN"
14 - }

```

保存后，就能看到Example(1)了：

The screenshot shows the Postman interface with a single example selected. A red arrow points to the "Examples (1)" dropdown in the top right. The "EXAMPLE REQUEST" section shows a GET request to [http://openapi.youdao.com/api?q=纠删码\(EC\)的学习&from=zh\\_CHS&to=EN&appKey=15...](http://openapi.youdao.com/api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=15...). The "EXAMPLE RESPONSE" section shows a 200 OK status with a JSON response body identical to the one above:

```

1 - {
2 -   "query": "纠删码(EC)的学习",
3 -   "translation": [
4 -     "The study of correcting code (EC)"
5 -   ],
6 -   "errorCode": "0",
7 -   "dict": {
8 -     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%AA%81%28EC%29%E7%9A%84%E5%AD%A6%E4%89%A0"
9 -   },
10 -   "webdict": {
11 -     "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%AA%81%28EC%29%E7%9A%84%E5%AD%A6%E4%89%A0"
12 -   },
13 -   "l": "zh-CHS2EN"
14 - }

```

单个Example在导出的API文档中的效果

然后再去导出文档，导出文档中的确能看到返回数据的例子：

The screenshot shows the Youdao Translate API documentation on the left and the Postman interface on the right.

**Youdao Translate API Documentation:**

- Introduction:** Shows a "有道翻译" (Youdao Translation) logo.
- GET 有道翻译:** Describes the API to translate Chinese to English.
- 请求 (Request):** Shows the "请求参数 (Request Parameters)" section with the following details:
  - q: query string, query字符串
  - from: 从什么语言, 原始语言
  - to: 翻译为, 目标语言
  - appKey: 自己有道智云账号中创建的应用的应用ID
  - salt: 给MD5算法加的盐, 一般用任意随机数即可
  - sign: MD5加密后的值=MD5(appKey+q+salt+密钥)
    - 经过验证, 好像大小写均可
    - 举例:
      - appKey=应用ID=152e0e77723a0026
      - q=纠正码(EC)的学习
      - salt=4
      - 密钥=应用密钥
      - =sYmnnOaisQgZzrlrBFozWAtsaRyylg4N

**Postman Interface:**

- Sample Request:** Shows a Node.js script for a GET request to the Youdao API.
- Sample Response:** Shows a JSON response object with fields like "query", "translation", "errorCode", "dict", and "url". A red box highlights the "url" field: "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%8A&".

## 多个Example在导出的API文档中的效果

The screenshot shows the NiuNiuCloud API documentation on the left and the Postman interface on the right.

**NiuNiuCloud API Documentation:**

- Introduction:** Shows a "牛牛云" (NiuNiuCloud) logo.
- POST login/login:** Describes the login endpoint.
- HEADERS:** Shows "Content-Type: application/json".
- BODY:** Shows a JSON object with "username" and "password" fields.

**Postman Interface:**

- Sample Request:** Shows a Node.js script for a POST request to the login endpoint. A red box highlights the "server\_address" field in the URL: "http://116.191.57.157/ucows/login/login".
- Sample Response:** Shows a JSON response object with fields like "code", "message", "data", and "tokenid". A red box highlights the "tokenid" field: "Ba68bd4553447379a1ac14387b2d477".

The screenshot shows the Postman application interface. At the top, there are buttons for 'Published' (green), 'No environment' (grey), and '季茂 (crifan)' (dropdown). The main area has a title 'One-click live docs for your teams collections Try for 7 days and share with your team'. On the left, a sidebar titled 'Introduction' lists various API endpoints under 'Niu Yun'. The central part shows a 'POST login/login' endpoint with a sample request and response.

**Sample Request:**

```
var http = require("http");
var options = {
  "method": "POST",
  "hostname": [
    "{{server_address}}"
  ],
  "path": [
    "ucows",
    "ihui"
  ]
}
```

**Sample Response:**

```
{
  "code": 200,
  "message": "ok",
  "data": {
    "tokenId": "40a0bf1caf8247ebbac16ddc7942cbeb",
    "success": true,
    "userId": "B61cf863d454471b7e75af69e018794",
    "loginUserType": "1"
  }
}
```

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 21:38:10

## Postman功能：其他工具和功能

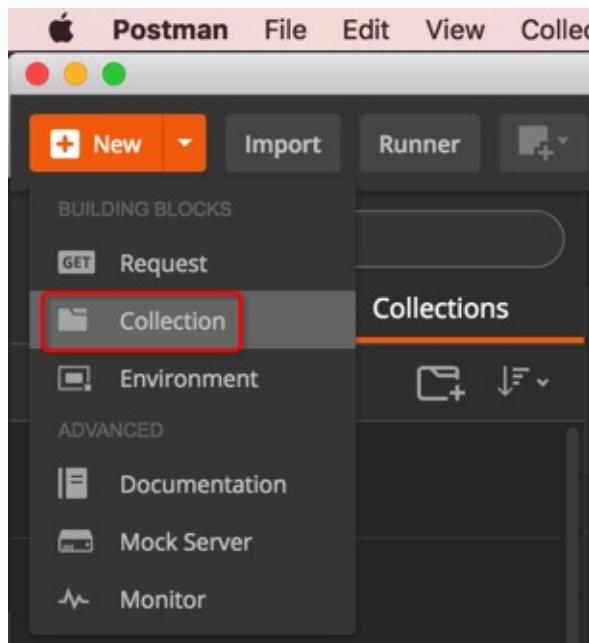
此处接着介绍Postman的其他一些功能或有用的工具。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 13:51:38

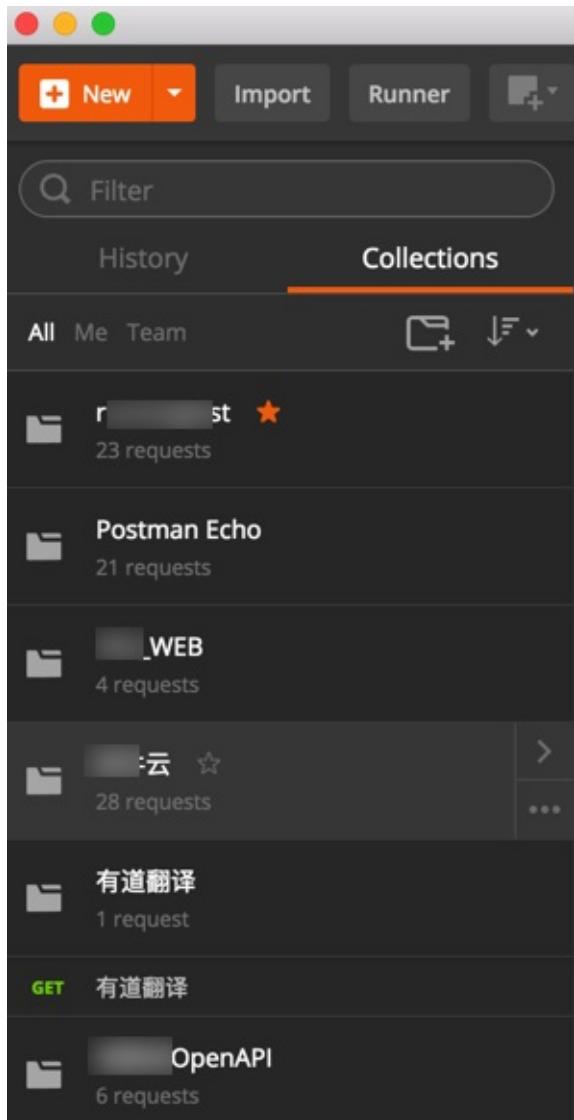
## 分组Collection

在刚开始一个项目时，为了后续便于组织和管理，把同属该项目的多个API，放在一组里

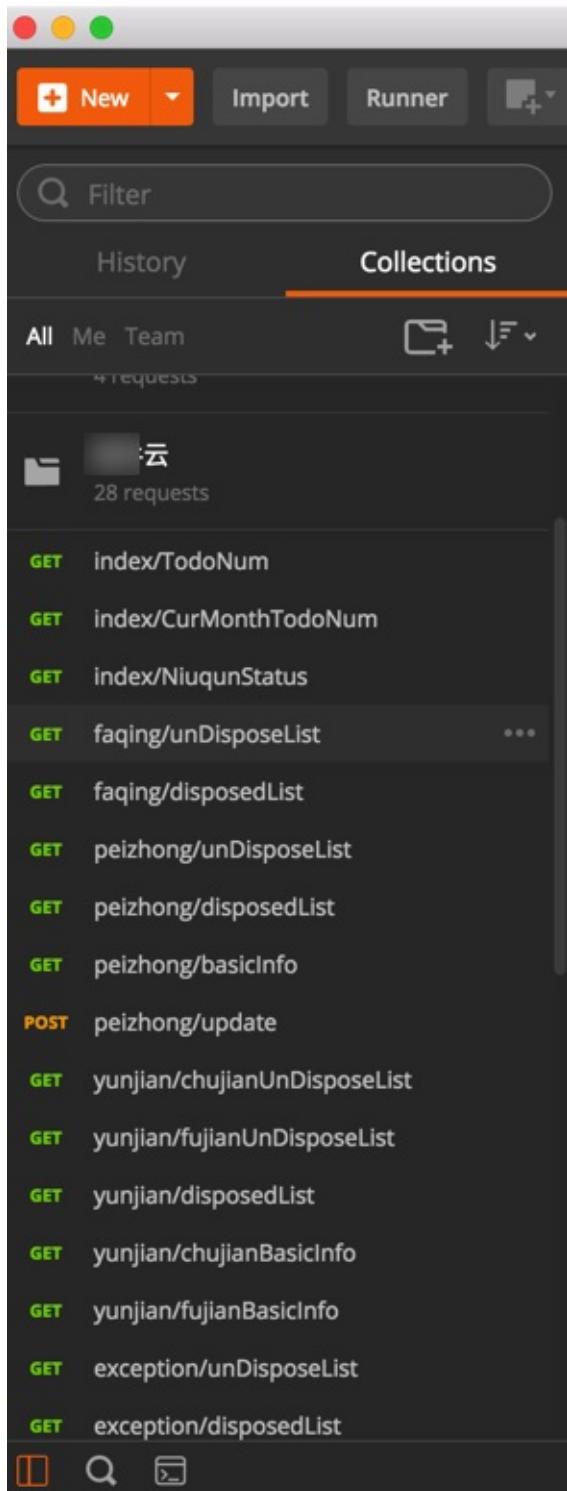
所以要先去新建一个Collection: New -> Collection



使用了段时间后，建了多个分组的效果：



单个分组展开后的效果：



crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 13:56:46

# 历史记录History

Postman支持history历史记录，显示出最近使用过的API：

The screenshot shows the Postman interface with the 'History' tab selected in the left sidebar. A red arrow points to the 'History' tab. The main workspace displays a recent API request for a POST method to '21084/runningfast/api/v1.0/open/register'. The 'Body' tab is selected, showing JSON input:

```
1 - {  
2   "phone": "13811118888",  
3   "smsCode": "505033",  
4   "email": "register",  
5   "firstName": "crifan",  
6   "lastName": "Li",  
7   "password": "123456",  
8   "facebookUserId": "11 56"  
9 }
```

Below the body, the response status is 200 OK and the time is 926 ms. The response body is:

```
1 - {  
2   "code": 10303,  
3   "message": "sms code is wrong"  
4 }
```

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间： 2017-12-29 15:22:24

# 用环境变量实现多服务器版本

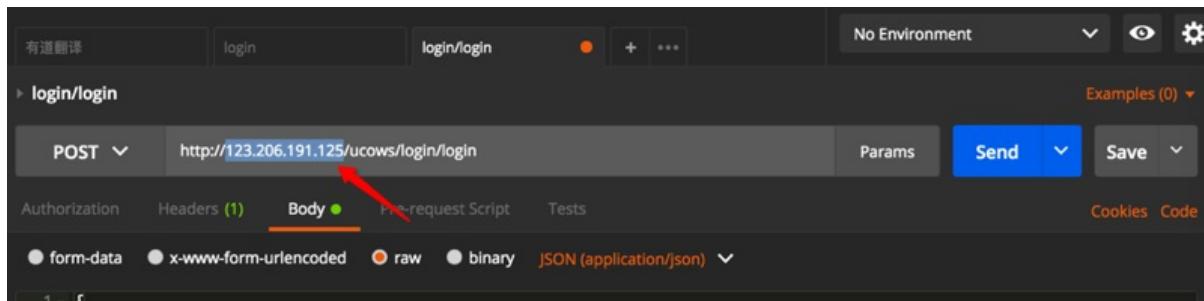
## 现存问题

在测试API期间，往往存在多种环境，对应IP地址（或域名也不同）

比如：

- Prod: `http://116.62.25.57/ucows`
  - 用于开发完成发布到生产环境
- Dev: `http://123.206.191.125/ucows`
  - 用于开发期间的线上的Development的测试环境
- LocalTest: `http://192.168.0.140:80/ucows`
  - 用于开发期间配合后台开发人员的本地局域网内的本地环境，用于联合调试API接口

而在测试API期间，往往需要手动去修改API的地址：



效率比较低，且地址更换后之前地址就没法保留了。

另外，且根据不同IP地址（或者域名）也不容易识别是哪套环境。

## 解决办法

### 小幺鸡的线上环境和本机环境的切换

之前得知[小幺鸡，简单好用的接口文档管理工具 -》发送JSON-演示项目](#)中有个好用的功能：

支持不同环境：

- 线上环境
- 本地环境

等，当时以为Postman不支持呢

### Postman支持用Environment环境变量去实现多服务器版本

后来发现Postman中，有Environment和Global Variable，用于解决这个问题，实现不同环境的管理：

The screenshot shows the Postman interface with the 'Environment' tab highlighted by a red box. In the 'Environment' panel, the 'Globals' section is also highlighted with a red box. The code in the 'Body' tab is as follows:

```

1 {
2   "query": "纠删码(EC)的学习",
3   "translation": [
4     "The study of correcting code (EC)"
5   ],
6   "errorCode": "0",
7   "dict": {
8     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28ECK29%E7%9A%84%E5%AD%A6%E4%B9%A0"
9   },
10  "webdict": {
11    "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28ECK29%E7%9A%84%E5%AD%A6%E4%B9%A0"
12  },
13  "l": "zh-CHS2EN"
14 }

```

-》很明显，就可以用来实现不用手动修改url中的服务器地址，从而动态的实现，支持不同服务器环境：

- Production 生产环境
- Development 开发环境
- Local 本地局域网环境

## 如何使用Environment实现多服务器版本

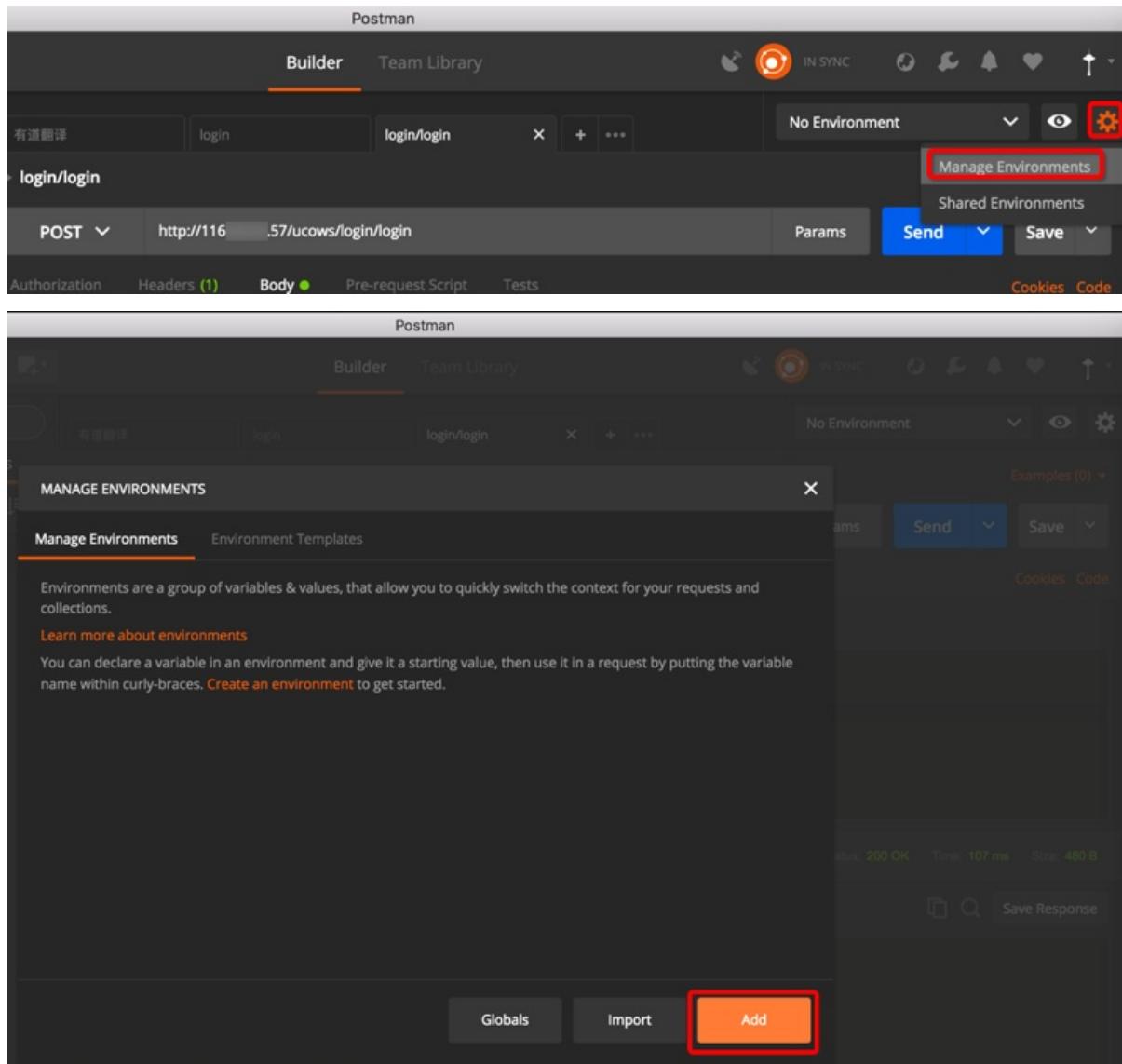
The screenshot shows the Postman interface with a 'login/login' request. The 'Environment' tab is highlighted by a red box. The 'Globals' section is also highlighted with a red box. The code in the 'Body' tab is as follows:

```

1 {
2   "username": "ipu",
3   "password": "000000"
4 }

```

或者：



Environments are a group of variables & values, that allow you to quickly switch the context for your requests and collections.

[Learn more about environments](#)

You can declare a variable in an environment and give it a starting value, then use it in a request by putting the variable name within curly-braces. [Create an environment](#) to get started.

输入Key和value：

MANAGE ENVIRONMENTS

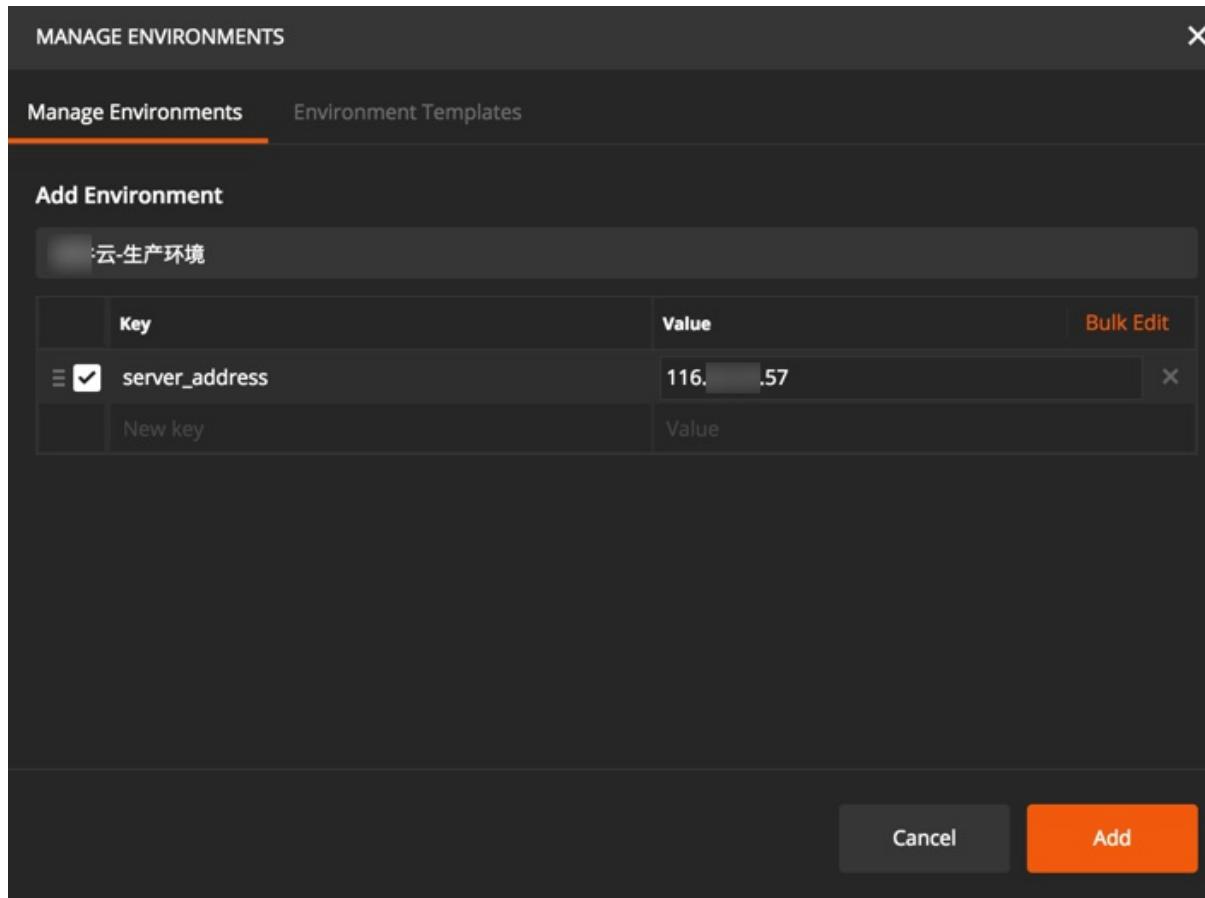
Manage Environments Environment Templates

Add Environment

云-生产环境

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> server_address	116. .57	X
New key	Value	

Cancel Add



点击Add后：

MANAGE ENVIRONMENTS

Manage Environments Environment Templates

Environments are a group of variables & values, that allow you to quickly switch the context for your requests and collections.

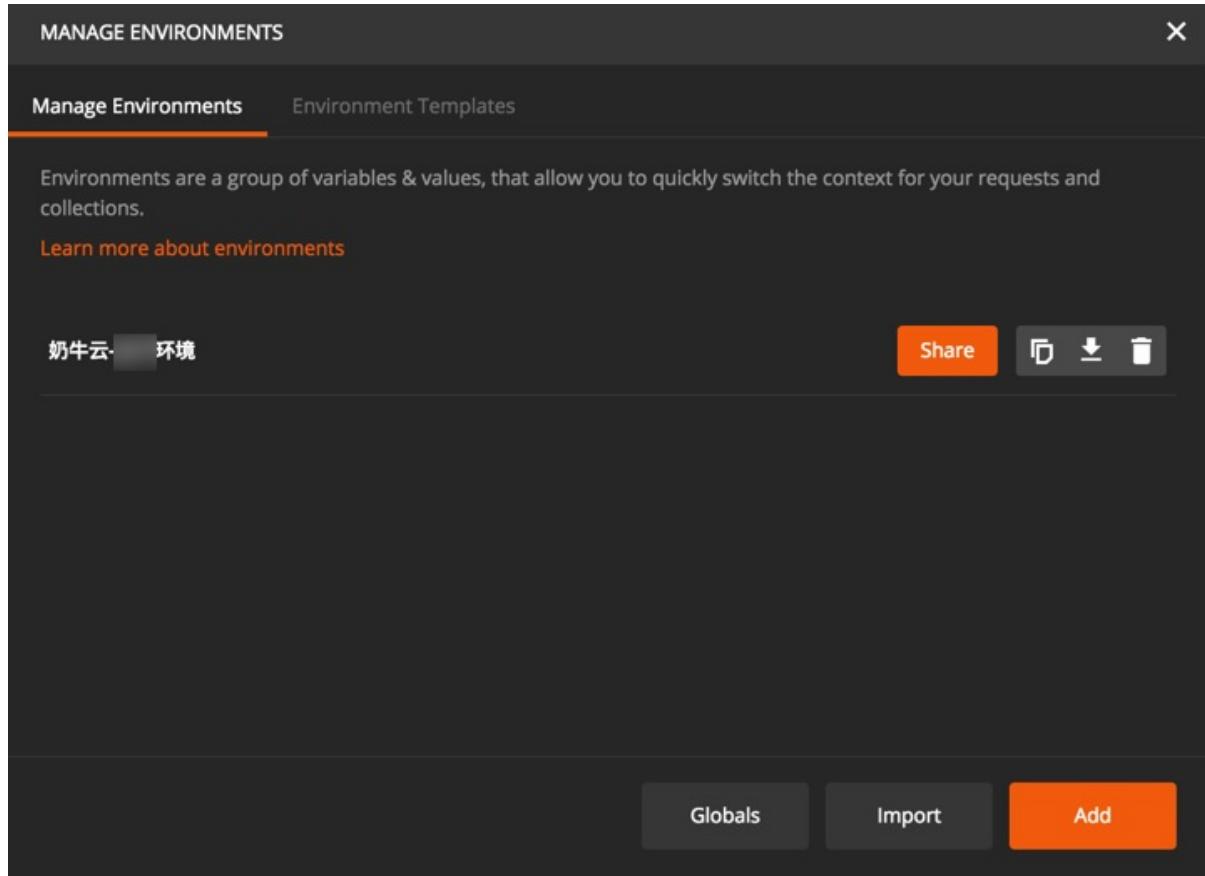
Learn more about environments

奶牛云- 环境

Share

Import

Add



## 环境变量可以使用的地方

- URL
- URL params
- Header values
- form-data/url-encoded values
- Raw body content
- Helper fields
- 写test测试脚本中
  - 通过postman的接口，获取或设置环境变量的值。

此处把之前的在url中的IP地址（或域名）换成环境变量：

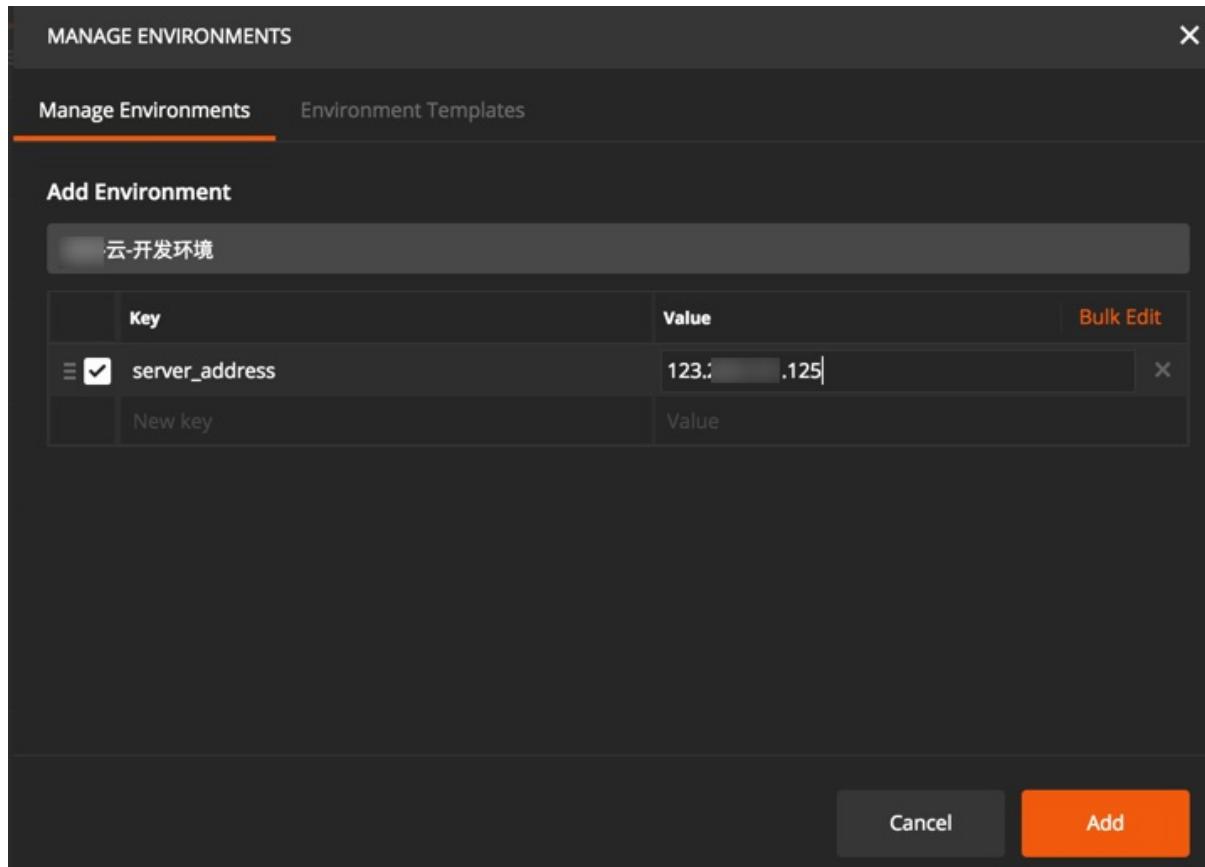
The screenshot shows the Postman Builder interface. In the 'Body' tab, there is a JSON payload:

```
1 {  
2   "username": "pu",  
3   "password": "000000"  
4 }
```

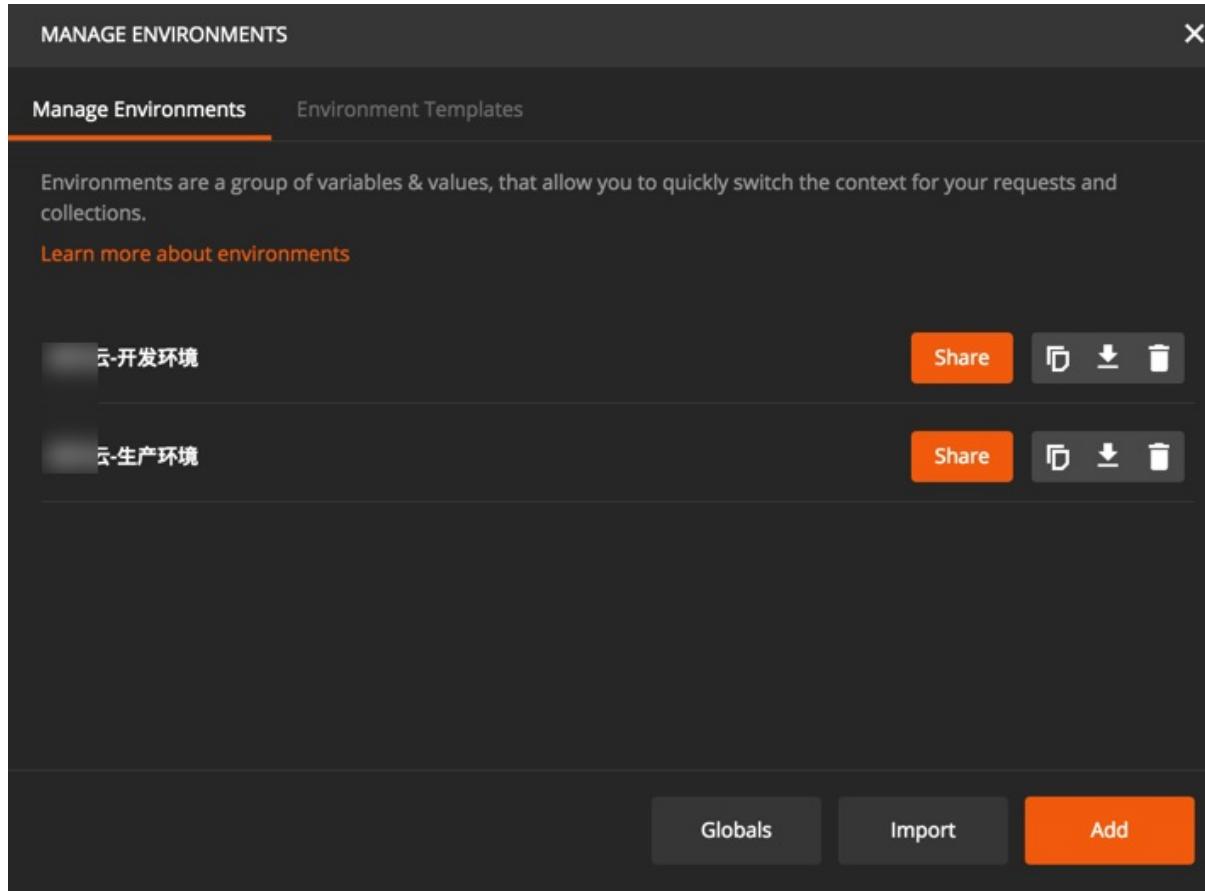
鼠标移动到环境变量上，可以动态显示出具体的值：

The screenshot shows the Postman Builder interface. A tooltip is displayed over the placeholder {{server\_address}} in the URL field, showing the value '116.57' and the scope 'Environment'.

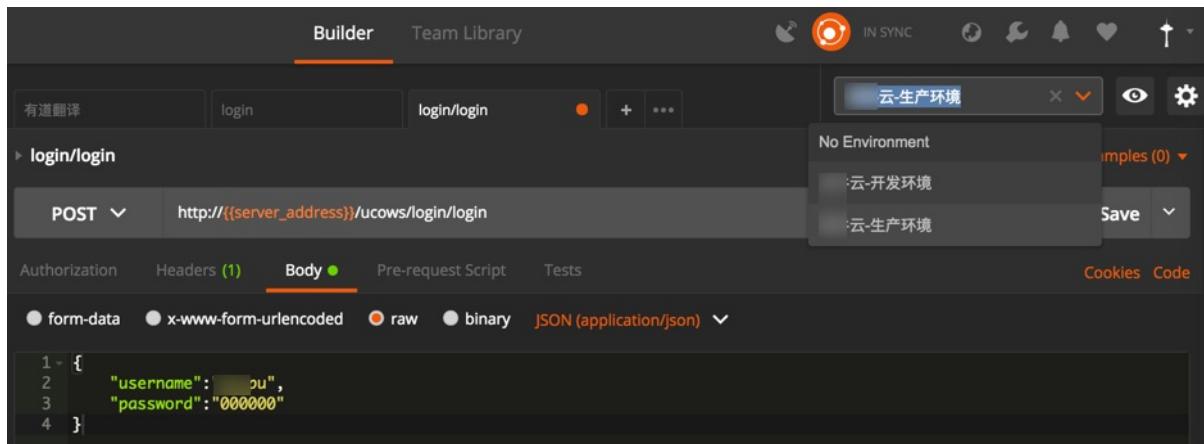
再去添加另外一个开发环境：



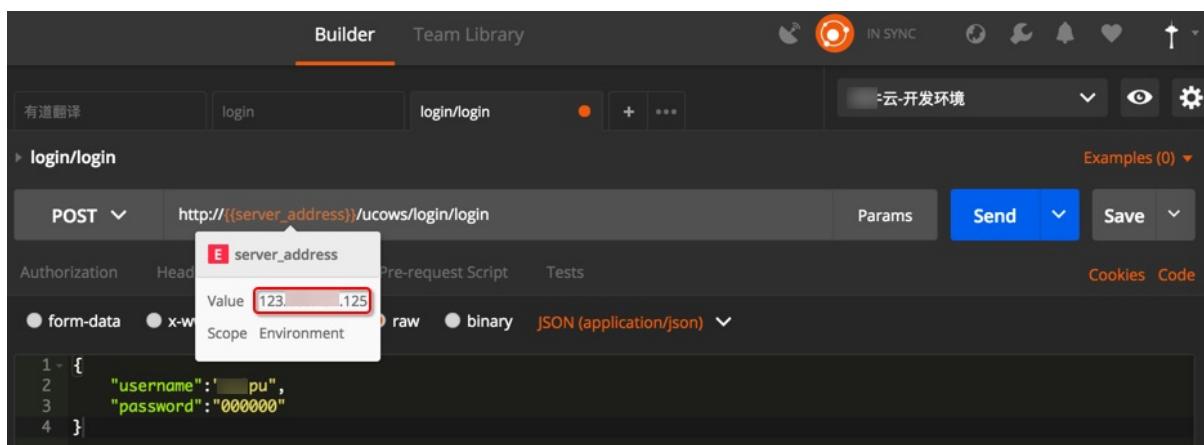
则可添加完2个环境变量，表示两个服务器地址，两个版本：



然后就可以切换不同服务器环境了：



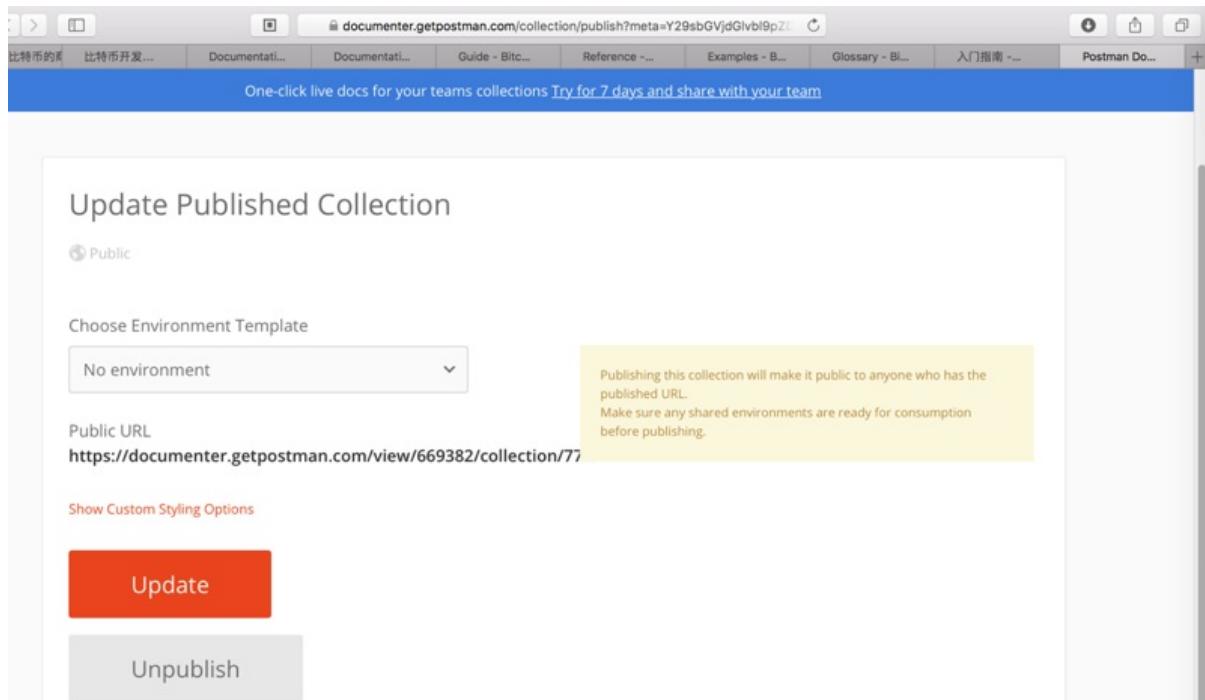
可以看到，同样的变量server\_address，在切换后对应IP地址就变成希望的开发环境的IP了：



## Postman导出API文档中多个环境变量的效果

顺带也去看看，导出为API文档后，带了这种Environment的变量的接口，文档长什么样子：

发现是在发布之前，需要选择对应的环境的：



## Update Published Collection

Public

### Choose Environment Template

No environment

No environment

一线 开发环境

2/c

云-开发环境

云-生产环境

云-英文版

Unpublish

# Update Published Collection

Public

## Choose Environment Template

云-开发环境

## Public URL

<https://documenter.getpostman.com/view/669382/collection/77>

Show Custom Styling Options

Update

Unpublish

发布后的文档，可以看到所选环境和对应服务器的IP的：

The screenshot shows the Postman interface with a published collection named '奶牛云'. The environment dropdown at the top is set to '云-开发环境' (highlighted with a red box). The collection page lists several API endpoints, and a sample POST request for the 'login/login' endpoint is shown with its URL highlighted with a red box.

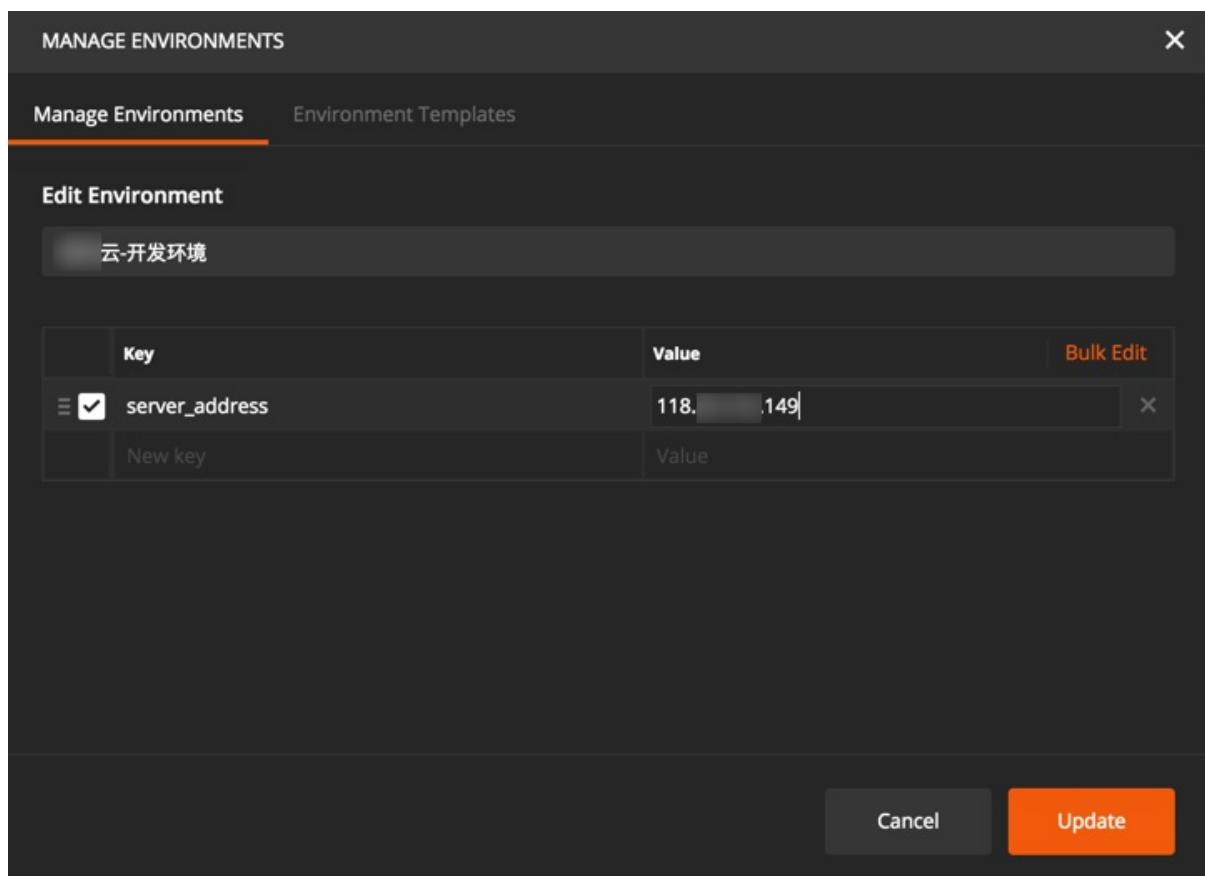
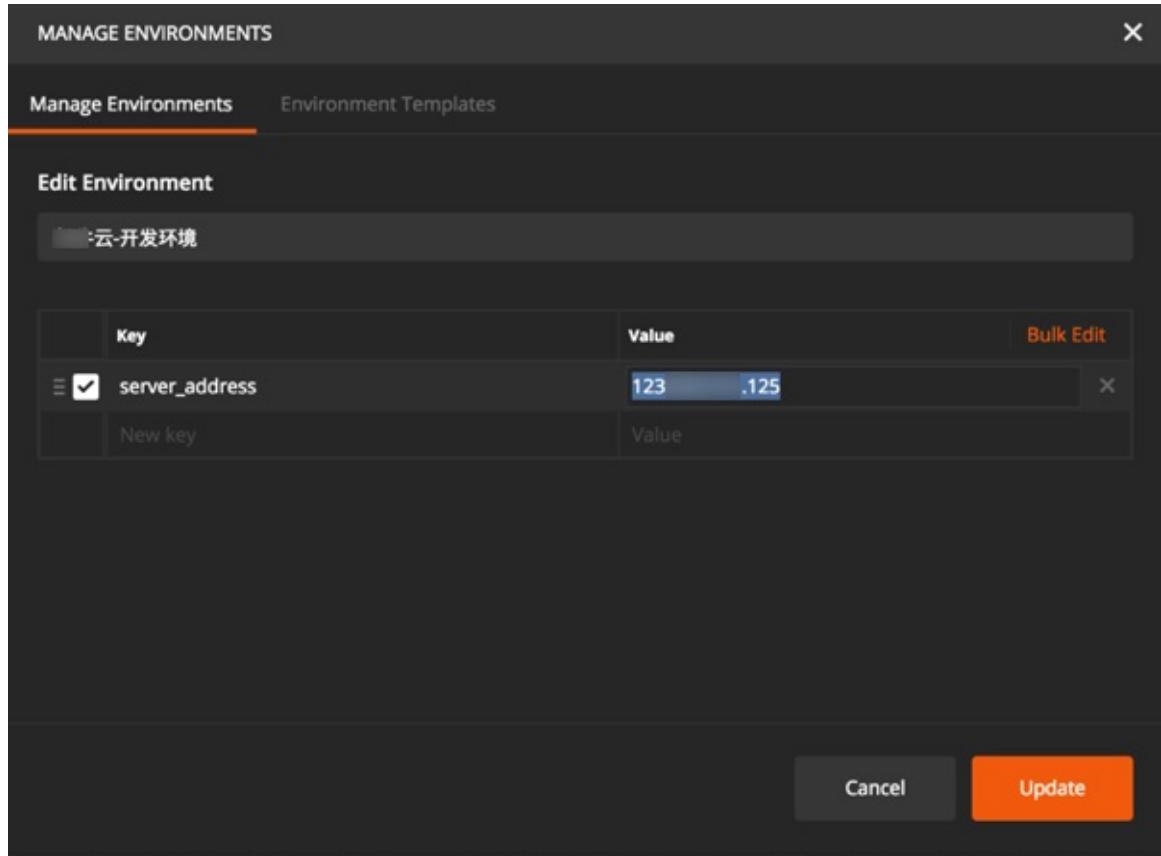
当然发布文档后，也可以实时切换环境：

The screenshot shows the Postman interface. On the left, there's a sidebar with 'POST /login/login' and a list of other API endpoints. The main area shows a 'Sample Request' for 'POST /login/login' with a curl command. Above the curl command, there's a dropdown menu for 'Environment'. The menu is open, showing several environments: 'No environment', 'SHARED TEMPLATES', 'No shared environments', 'PRIVATE ENVIRONMENTS', and four local environments: ':云-开发环境', ':一线 开发环境', ':云-生产环境', and ':云-英文版'. The ':云-开发环境' option is highlighted. At the top right, there's a dropdown for '李茂 (crifan)' and another for 'Public'.

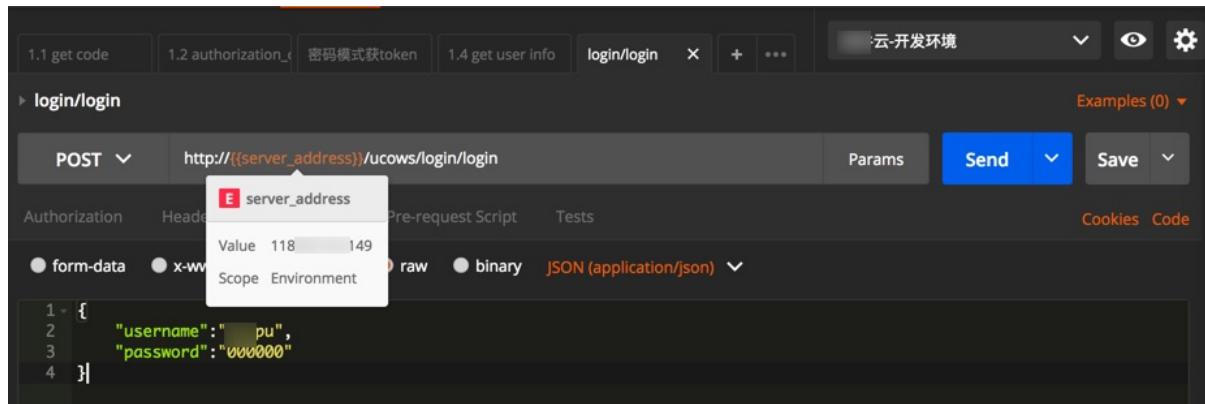
Below the main interface, there's a separate window or tab showing the generated API documentation. It includes the same 'POST /login/login' endpoint with its URL and headers, and the same curl command with environment variables.

## 环境变量的好处

当更换服务器时，直接修改变量的IP地址：



即可实时更新，当鼠标移动到变量上即可看到效果：



crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2018-06-16 12:44:52

# 代码生成工具

## 查看当前请求的HTTP原始内容

对于当前的请求，还可以通过点击Code

The screenshot shows the Postman interface with a GET request to `http://openapi.youdao.com/api?q=纠删码(EC)的学习`. The 'Code' button in the top right corner is highlighted with a red box. The response body is displayed in JSON format:

```

1 {
2   "query": "纠删码(EC)的学习",
3   "translation": [
4     "The study of correcting code (EC)"
5   ],
6   "errorCode": "0",
7   "dict": {
8     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
9   },
10  "webdict": {
11    "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
12  },
13  "l": "zh-CHS2EN"
14 }

```

去查看对应的符合HTTP协议的原始的内容：

The screenshot shows the Postman interface with the raw HTTP request copied to the clipboard. The 'Copy to Clipboard' button in the top right corner is highlighted with a red box. The raw request is as follows:

```

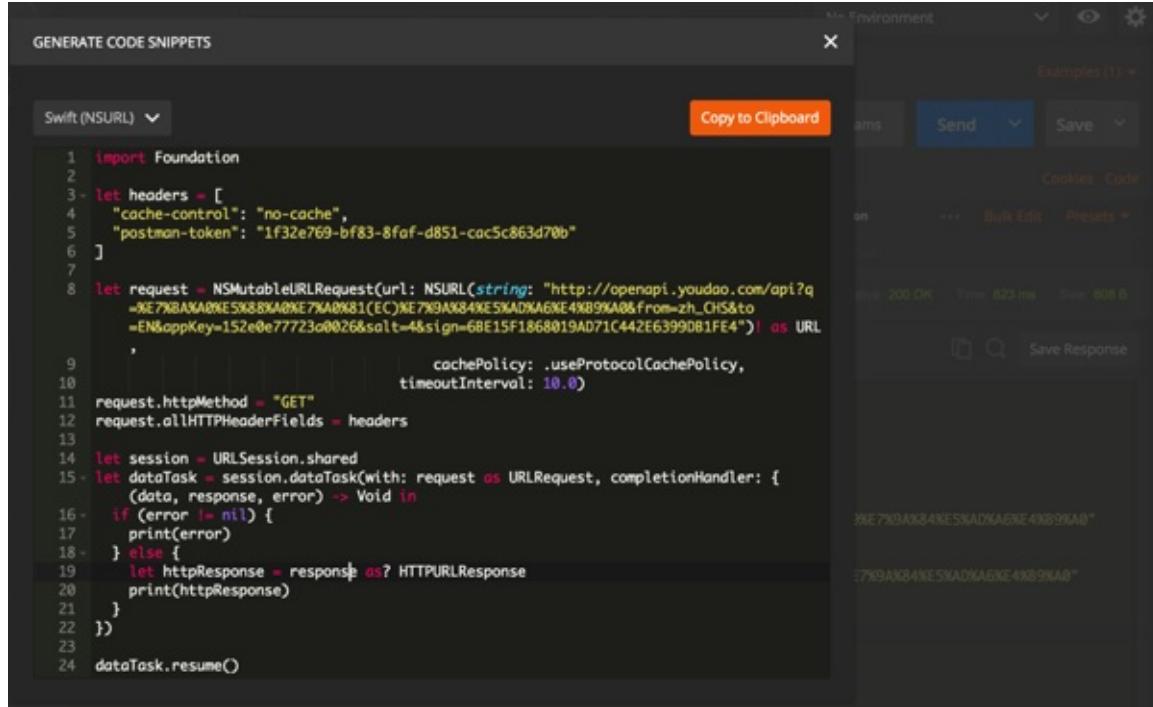
1 GET /api?q=纠删码(EC)的学习&from=zh_CHS&to=EN&appKey=152e0e77723a0026&salt=4&sign=6BE15F1868019AD71C442E6399D81FE4 HTTP/1.1
2 Host: openapi.youdao.com
3 Cache-Control: no-cache
4 Postman-Token: e9f5a7d8-b5fb-ef9b-c4b4-4293805285b7
5

```

## 各种语言的示例代码 Code Generation Tools

比如：

### Swift语言



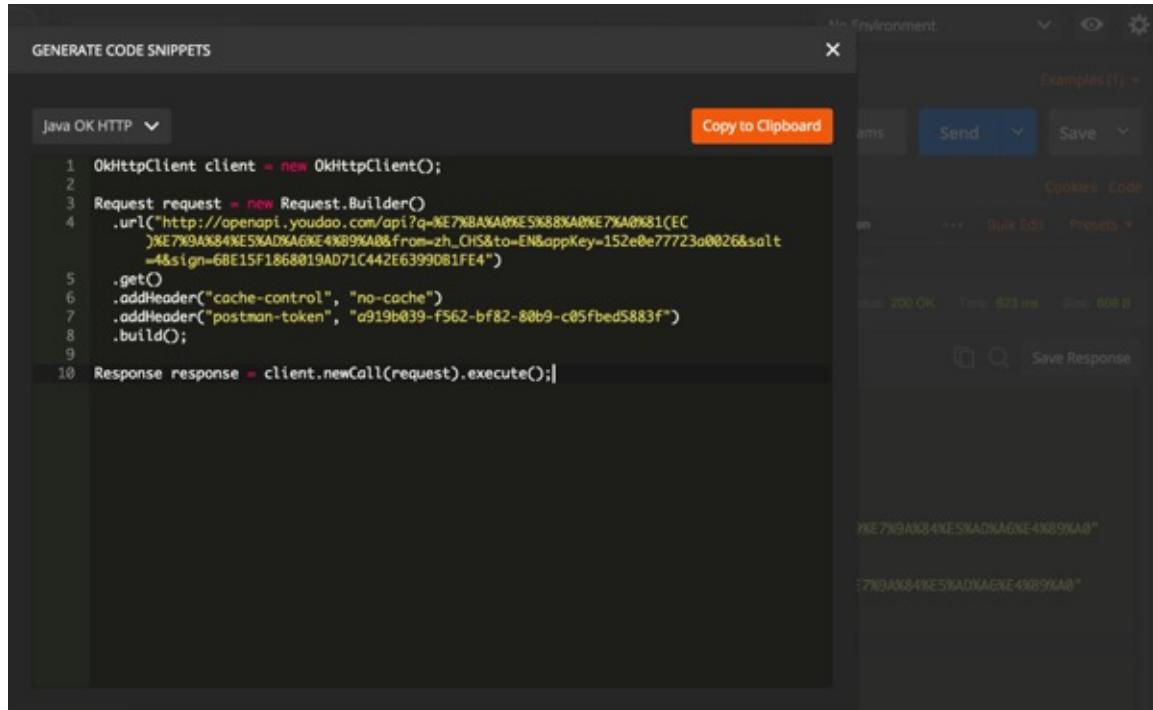
The screenshot shows the Postman interface with the "GENERATE CODE SNIPPETS" tab selected. The language dropdown is set to "Swift (NSURL)". The code block contains Swift code for creating an NSURLConnection:

```

1 import Foundation
2
3 let headers = [
4     "cache-control": "no-cache",
5     "postman-token": "1f32e769-bf83-8faf-d851-cac5c863d70b"
6 ]
7
8 let request = NSMutableURLRequest(url: NSURL(string: "http://openapi.youdao.com/api?q=%E7%BA%A0%E5%88%A0&to=%E7%9A%84%EC%8E%AD%6E4%8B%9A&from=zh_CHS&to=EN&appKey=152e0e77723a0026&salt=4&sign=68E15F1868019AD71C442E63990B1FE4")) as URLRequest
9
10 request.cachePolicy = .useProtocolCachePolicy,
11         timeoutInterval: 10.0
12 request.httpMethod = "GET"
13 request.allHTTPHeaderFields = headers
14
15 let session = URLSession.shared
16 let dataTask = session.dataTask(with: request as URLRequest, completionHandler: {
17     (data, response, error) -> Void in
18     if (error != nil) {
19         print(error)
20     } else {
21         let httpResponse = response as? HTTPURLResponse
22         print(httpResponse)
23     }
24 })
25 dataTask.resume()

```

### Java语言



The screenshot shows the Postman interface with the "GENERATE CODE SNIPPETS" tab selected. The language dropdown is set to "Java OK HTTP". The code block contains Java code using OkHttp to make a request:

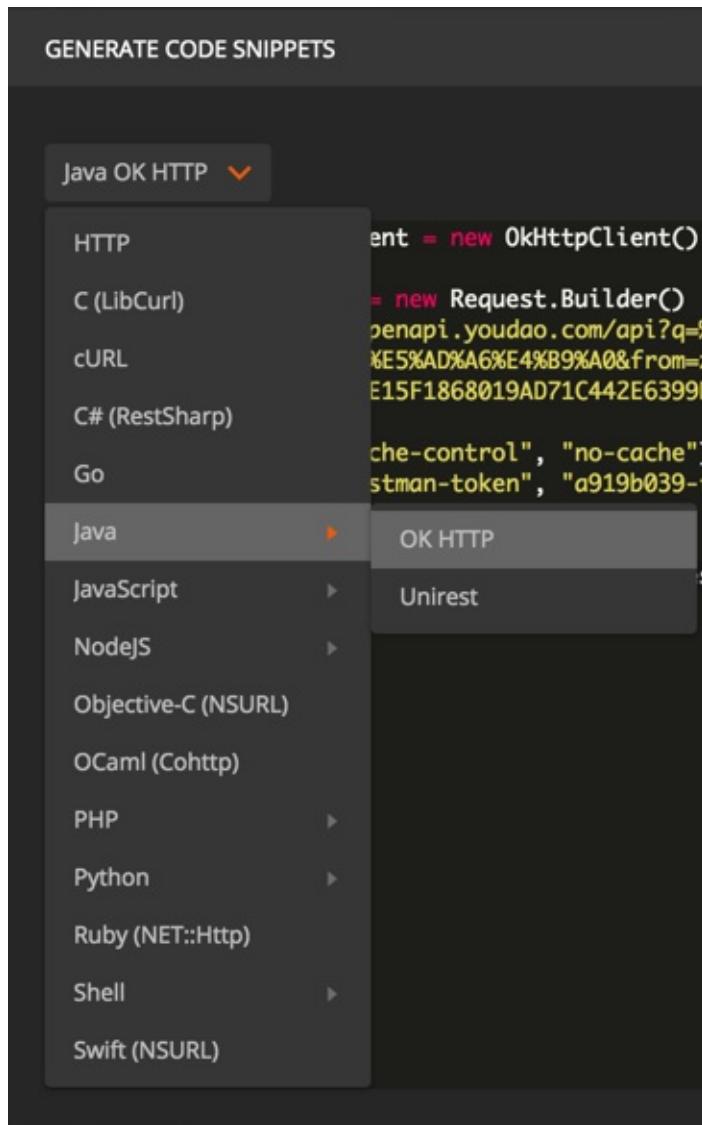
```

1 OkHttpClient client = new OkHttpClient();
2
3 Request request = new Request.Builder()
4     .url("http://openapi.youdao.com/api?q=%E7%BA%A0%E5%88%A0&to=%E7%9A%84%EC%8E%AD%6E4%8B%9A&from=zh_CHS&to=EN&appKey=152e0e77723a0026&salt=4&sign=68E15F1868019AD71C442E63990B1FE4")
5     .get()
6     .addHeader("cache-control", "no-cache")
7     .addHeader("postman-token", "a919b039-f562-bf82-80b9-c05fbed5883f")
8     .build();
9
10 Response response = client.newCall(request).execute();

```

### 其他各种语言

还支持其他各种语言：



目前支持的语言有：

- HTTP
- C (LibCurl)
- cURL
- C#(RestSharp)
- Go
- Java
  - OK HTTP
  - Unirest
- Javascript
- NodeJS
- Objective-C(NSURLConnection)
- OCaml(Cohttp)
- PHP
- Python
- Ruby(NET::Http)
- Shell

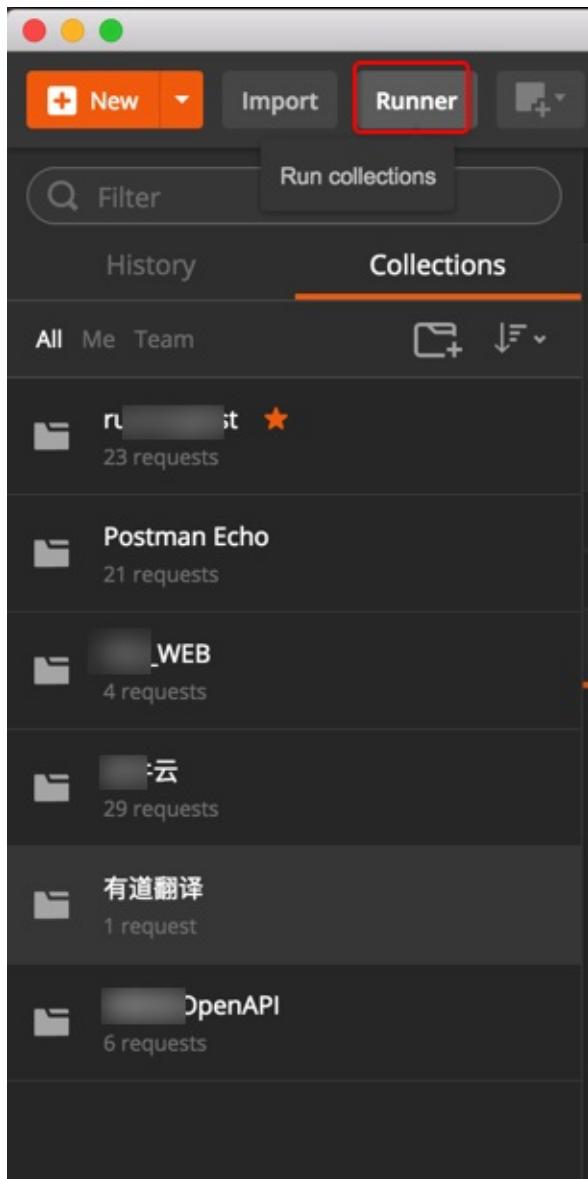
- Swift(NSURL)

代码生成工具的好处是：在写调用此API的代码时，就可以参考对应代码，甚至拷贝粘贴对应代码，即可。

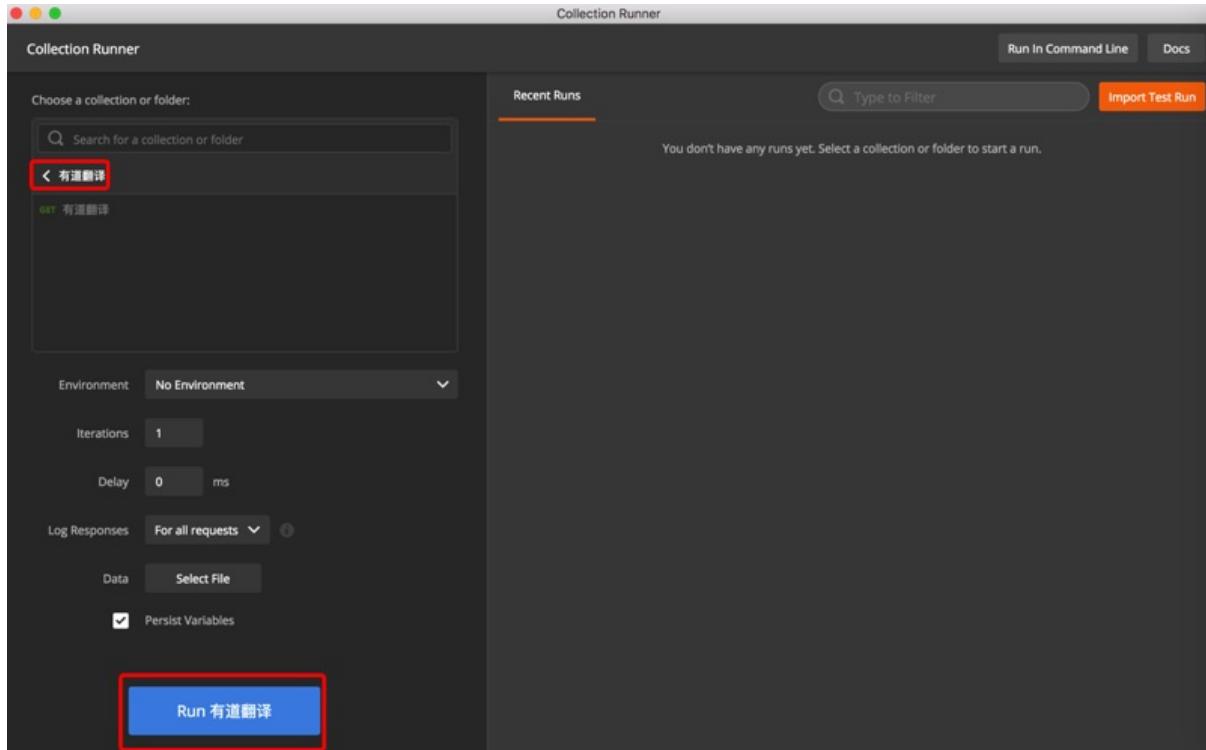
crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间：2017-12-29 20:46:55

## 测试接口

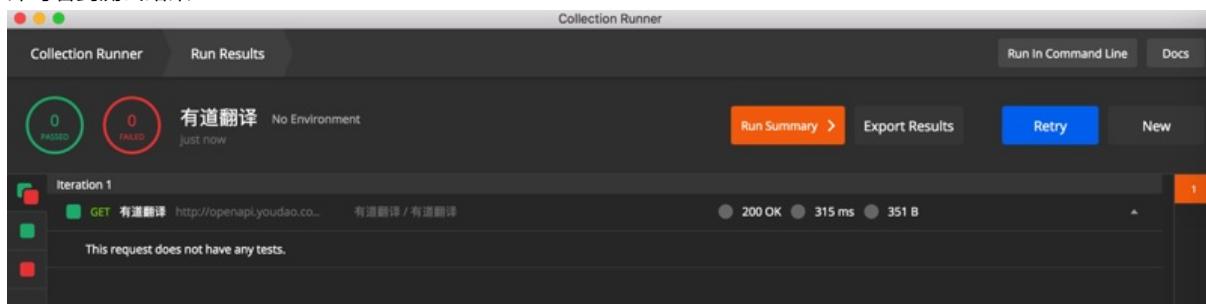
选中某个分组后，点击Runner



选中某个分组后点击Run



即可看到测试结果：



-» 好像是需要自己预先去添加test，然后才能测试的。

关于此功能的介绍可参考[Postman官网的git图](#)

## TODO

待后续有空继续完善此处Postman测试接口的内容。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间： 2018-06-16 12:44:59

# Mock Server

TODO:

关于Mock功能，抽空去参考：

[What is Postman Pro](#)

-»

[Mock responses in Postman by using Examples – Postman Blog](#)

试试。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间： 2018-01-02 13:52:49

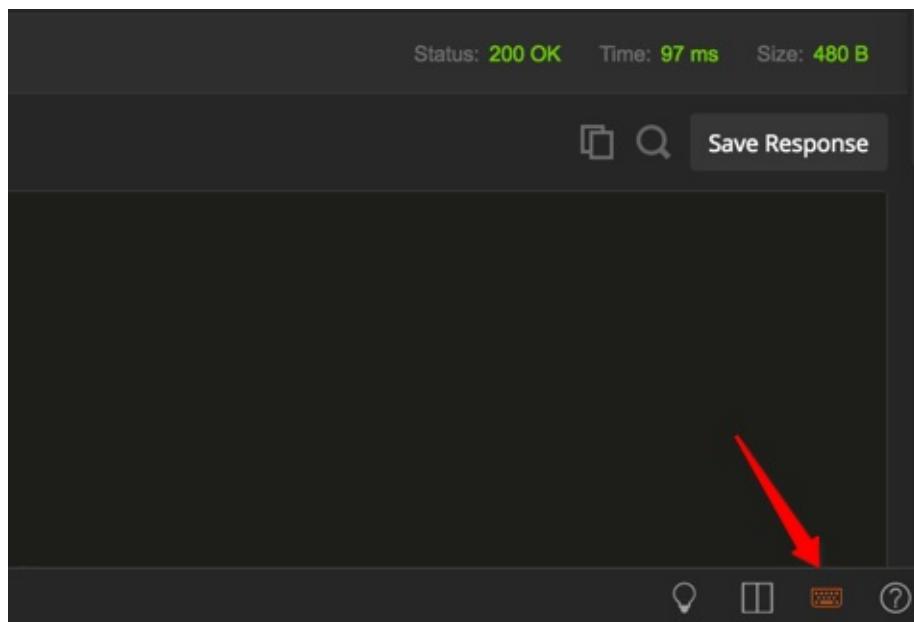
## Postman功能：界面和配置

打开Postman的设置有两种方式：

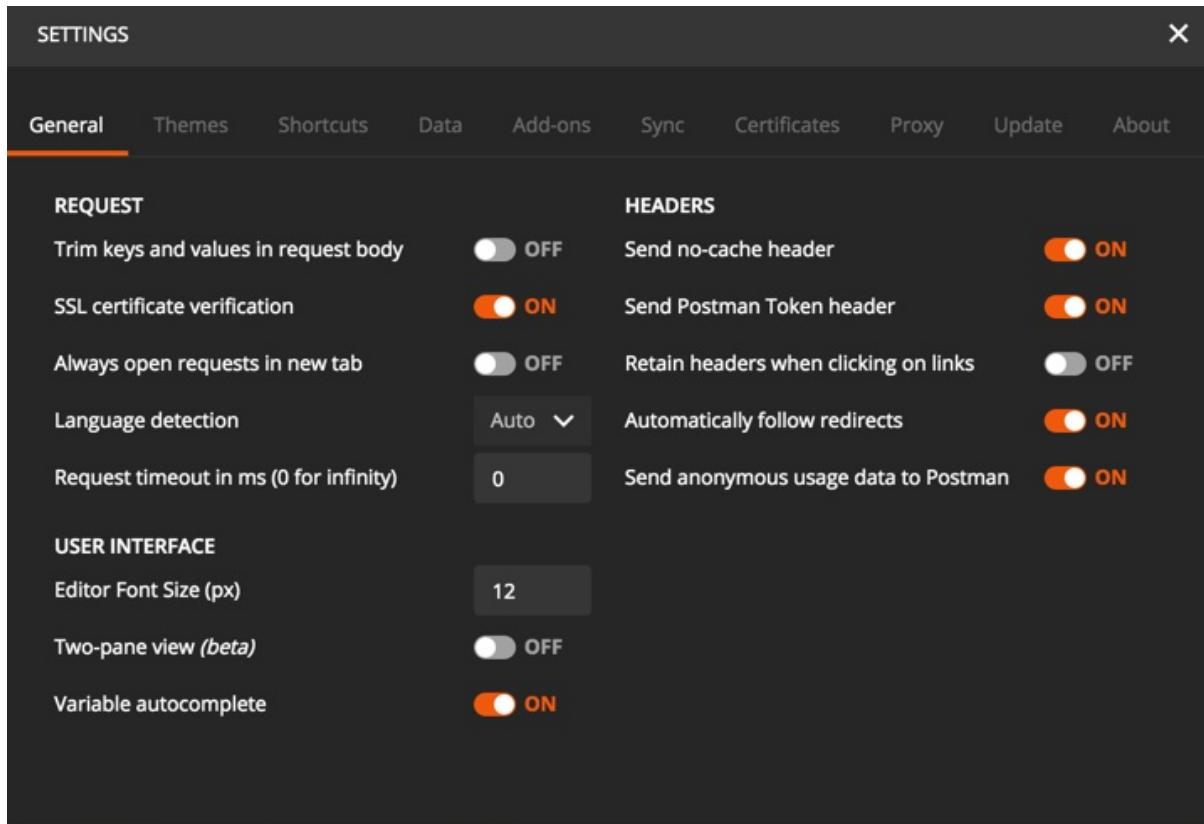
Postman -> Preferences



点击右下角的键盘按钮



都可以打开配置界面：



接下来介绍一些常见界面和功能的设置。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2018-01-02 11:49:45

# 多Tab分页

Postman支持多tab页，于此对比之前有些API调试工具就不支持多Tab页，比如 Advanced Rest Client

多tab的好处：

方便在一个tab中测试，得到结果后，复制粘贴到另外的tab中，继续测试其它接口

比如此处tab1中，测试了获取验证码接口后，拷贝手机号和验证码，粘贴到tab2中，继续测试注册的接口

POST 115 126.210.1.115.126.21084/runningfast/api/v1.0/open/smscode

```

1 {
2   "phone": "13811118888",
3   "type": "register"
4 }
    
```

Status: 200 OK Time: 880 ms

POST 115 126.210.1.115.126.21084/runningfast/api/v1.0/open/register

```

1 {
2   "phone": "13811118888",
3   "smsCode": "505033",
4   "email": "register",
5   "firstName": "crifan",
6   "lastName": "li",
7   "password": "123456",
8   "facebookUserId": "113156"
9 }
    
```

Status: 200 OK Time: 926 ms

12-29 15:29:47

# 界面查看模式

Postman的默认的Request和Response是上下布局：

The screenshot shows the Postman interface in its default mode. At the top, there's a navigation bar with tabs for 'Builder' and 'Team Library'. Below the navigation bar, there's a search bar containing 'login' and a dropdown menu showing 'login/login'. To the right of the search bar are several icons: a gear for settings, a bell for notifications, a heart for favorites, and a refresh symbol. The main workspace is titled 'No Environment'. On the left, there's a sidebar with a 'Builder' section containing a 'login' entry and a 'Examples (1)' dropdown. The main content area has tabs for 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Headers' tab is currently selected. Below the tabs, there's a table for managing headers. The 'Body' tab is also visible. On the right side of the main content area, there are status indicators: 'Status: 200 OK', 'Time: 623 ms', and 'Size: 608 B'. At the bottom of the main content area, there's a large JSON response block. The JSON response is as follows:

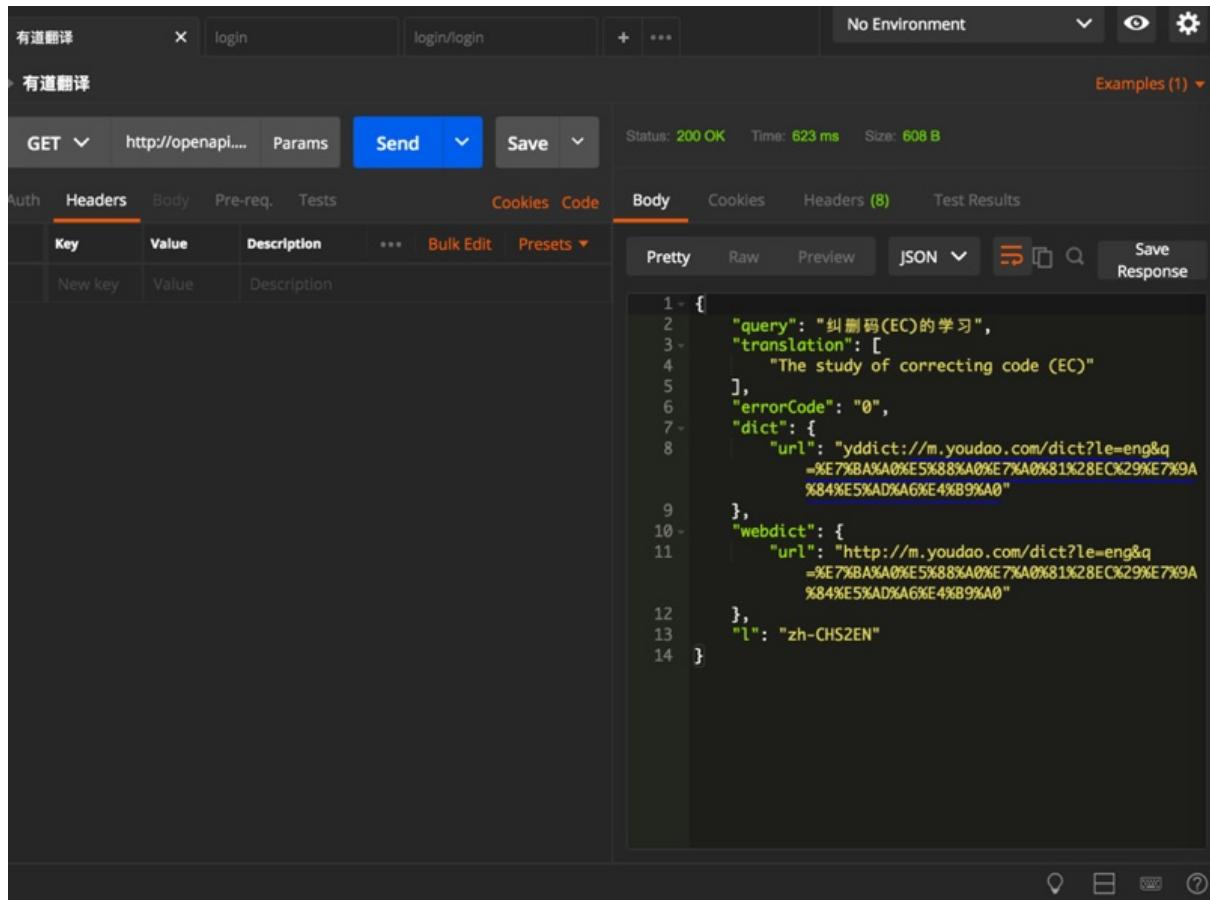
```

1 - {
2   "query": "纠删码(EC)的学习",
3   "translation": [
4     "The study of correcting code (EC)"
5   ],
6   "errorCode": "0",
7   "dict": {
8     "url": "yddict://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
9   },
10  "webdict": {
11    "url": "http://m.youdao.com/dict?le=eng&q=%E7%BA%A0%E5%88%A0%E7%A0%81%28EC%29%E7%9A%84%E5%AD%A6%E4%B9%A0"
12  },
13  "l": "zh-CHS2EN"
14 }

```

In the bottom right corner of the main content area, there's a button labeled 'Two pane view (⌘⌘V)' with a red box drawn around it. Below the main content area, there's a toolbar with icons for location, refresh, and help.

此处点击右下角的 Two pane view , 就变成左右的了：



## 左右布局的用途

对于数据量很大，又想要同时看到请求和返回的数据的时候，应该比较有用。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2018-06-16 12:45:05

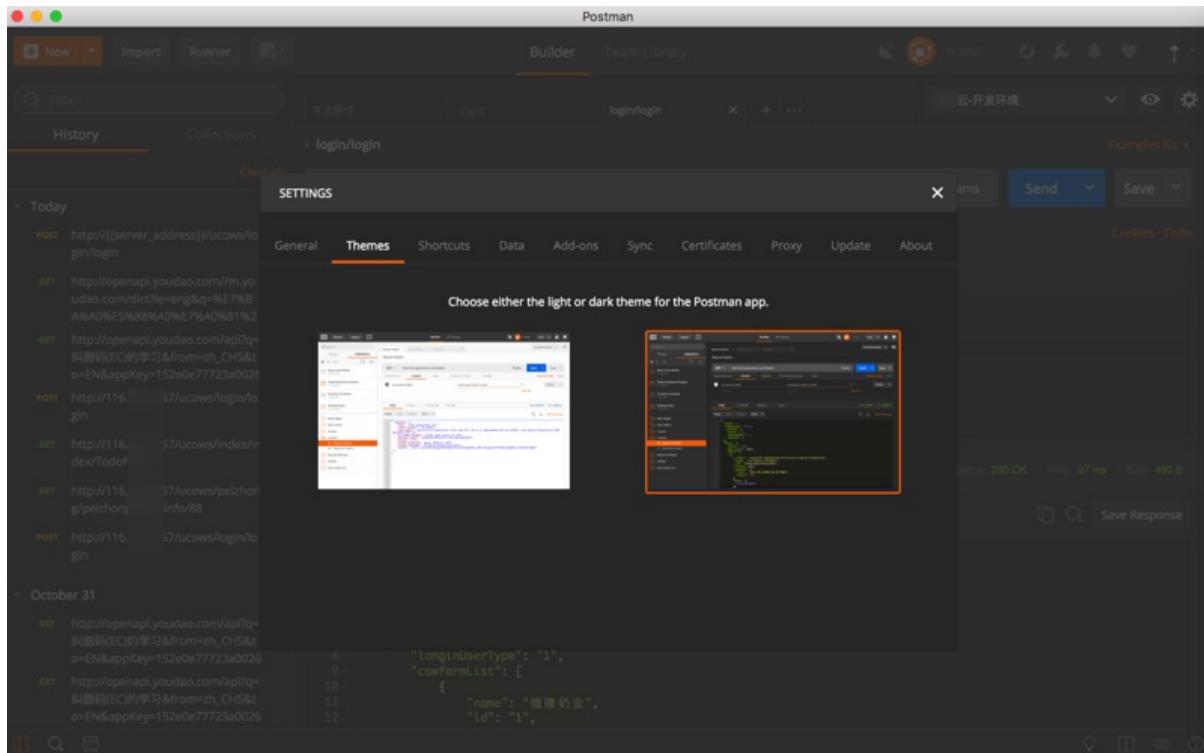
# 多颜色主题

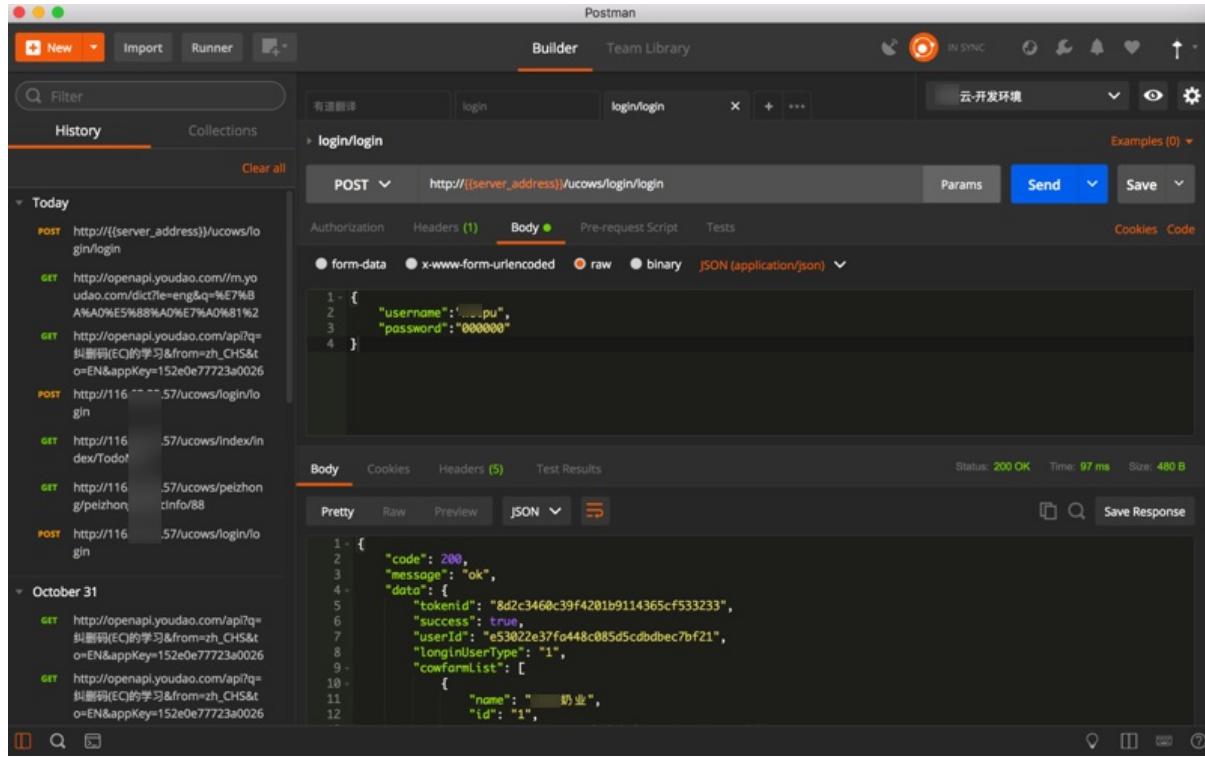
Postman支持两种主题：

## 深色主题

深色：

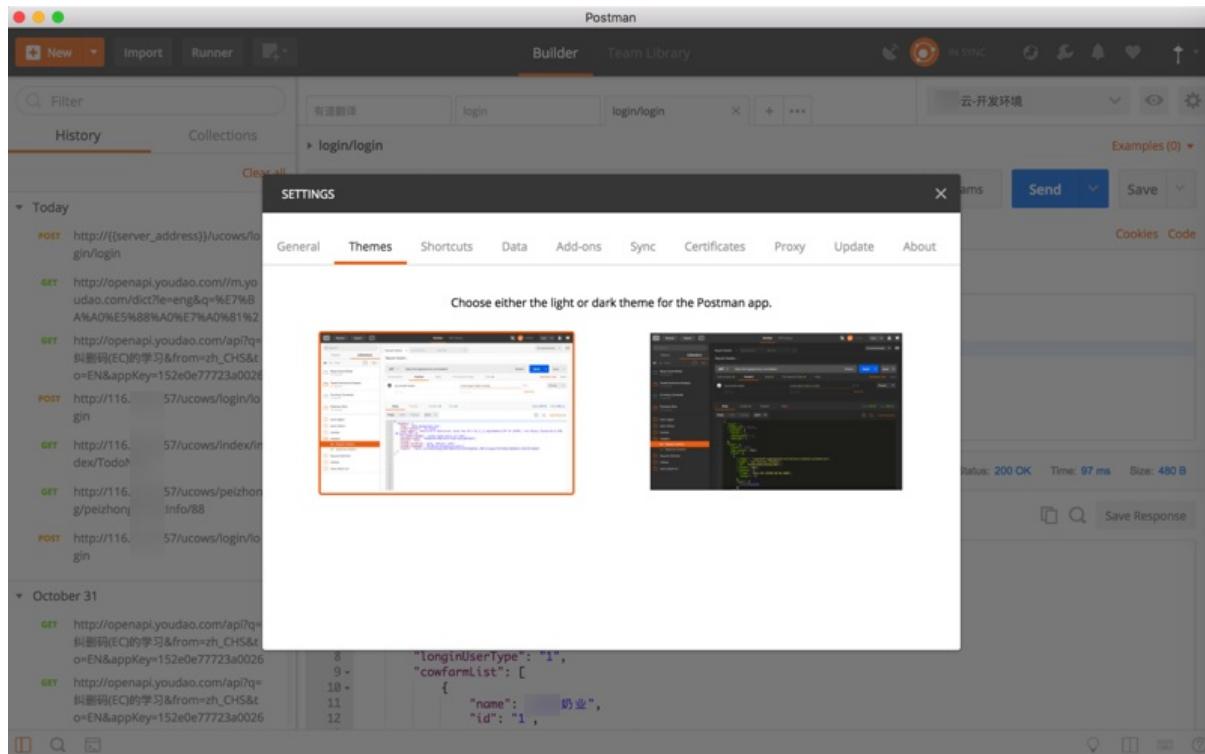
当前是深色主题，效果很不错：

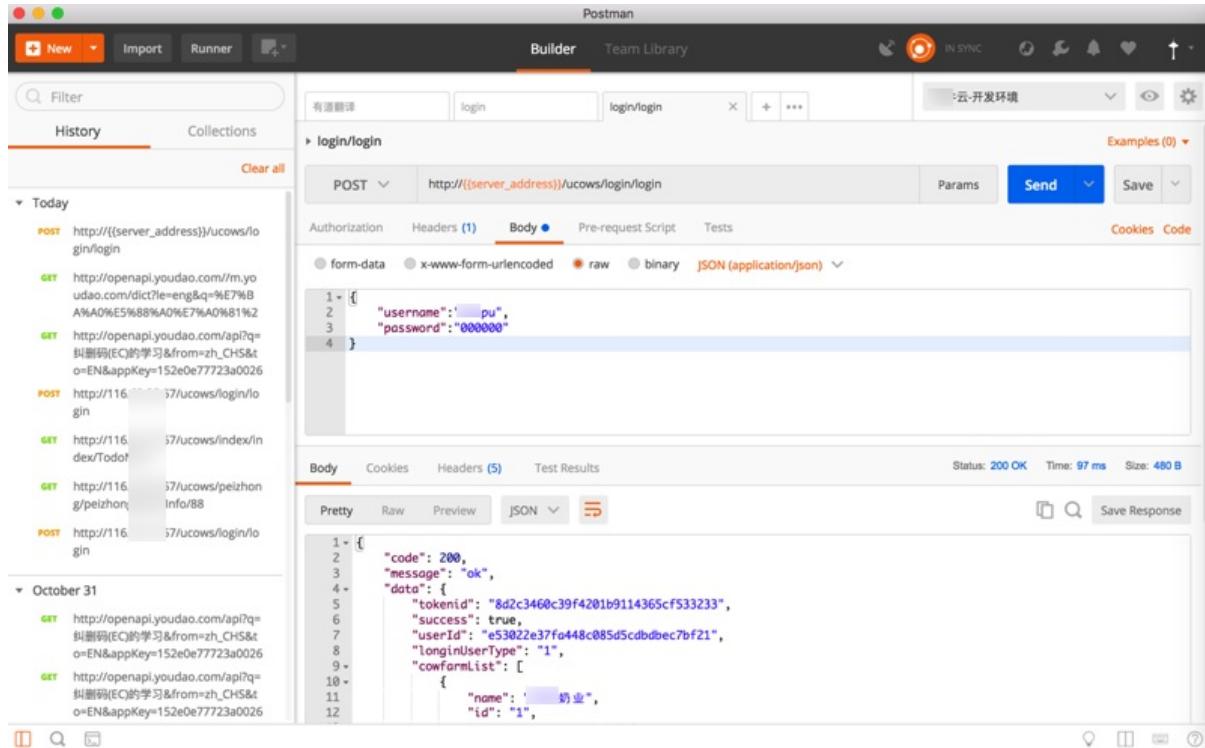




## 浅色主题

可以切换到 浅色主题：





crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2018-01-02 13:51:30

## Postman生成API文档

在服务端后台的开发人员测试好了接口后，打算把接口的各种信息发给使用此API的前端的移动端人员时，往往会遇到：

- 要么是用复制粘贴 -> 格式不友好
- 要么是用Postman中截图 -> 方便看，但是不方便获得API接口和字段等文字内容
- 要么是用Postman中导出为JSON -> json文件中信息太繁杂，不利于找到所需要的信息
- 要么是用文档，比如去编写Markdown文档 -> 但后续API的变更需要实时同步修改文档，也会很麻烦

这都会导致别人查看和使用API时很不方便。

-> 对此，Postman提供了发布API文档的功能，可以很好的解决此类问题。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间：2017-12-29 21:02:40

## 预览和发布API文档

下面介绍Postman中如何预览和发布API文档。

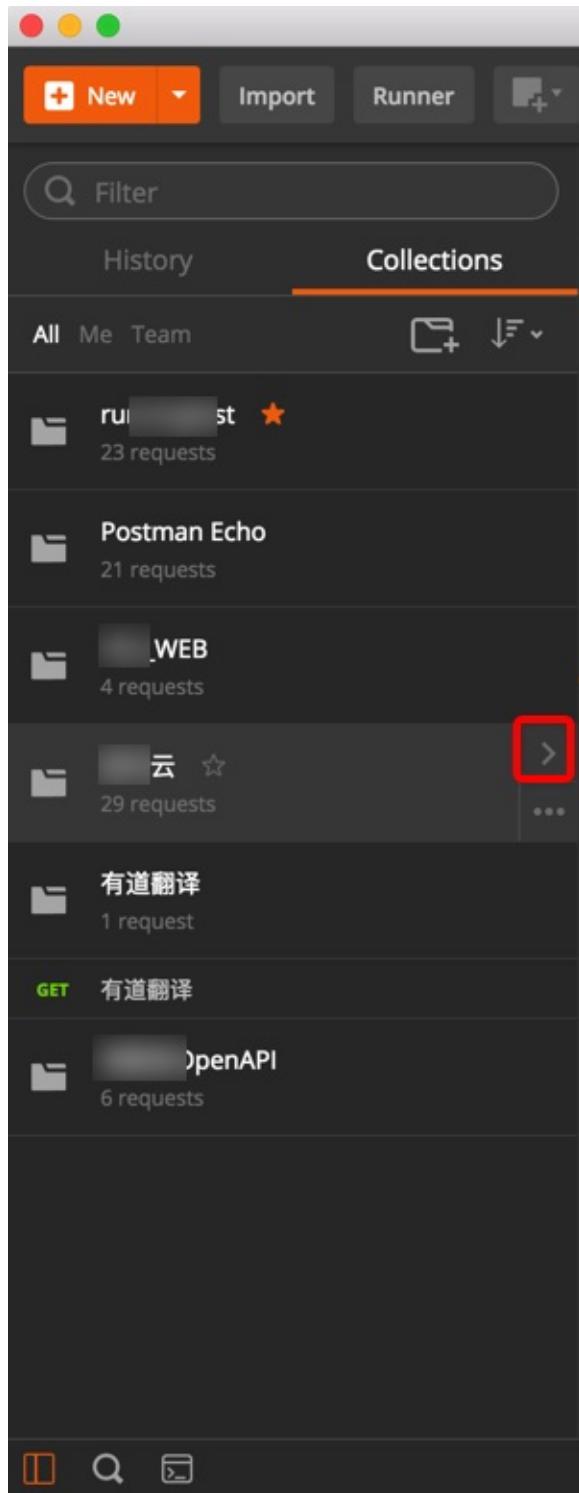
### 简要概述步骤

1. Collection
2. 鼠标移动到某个Collection，点击三个点
3. Publish Docs
4. Publish
5. 得到Public URL
6. 别人打开这个Public URL，即可查看API文档

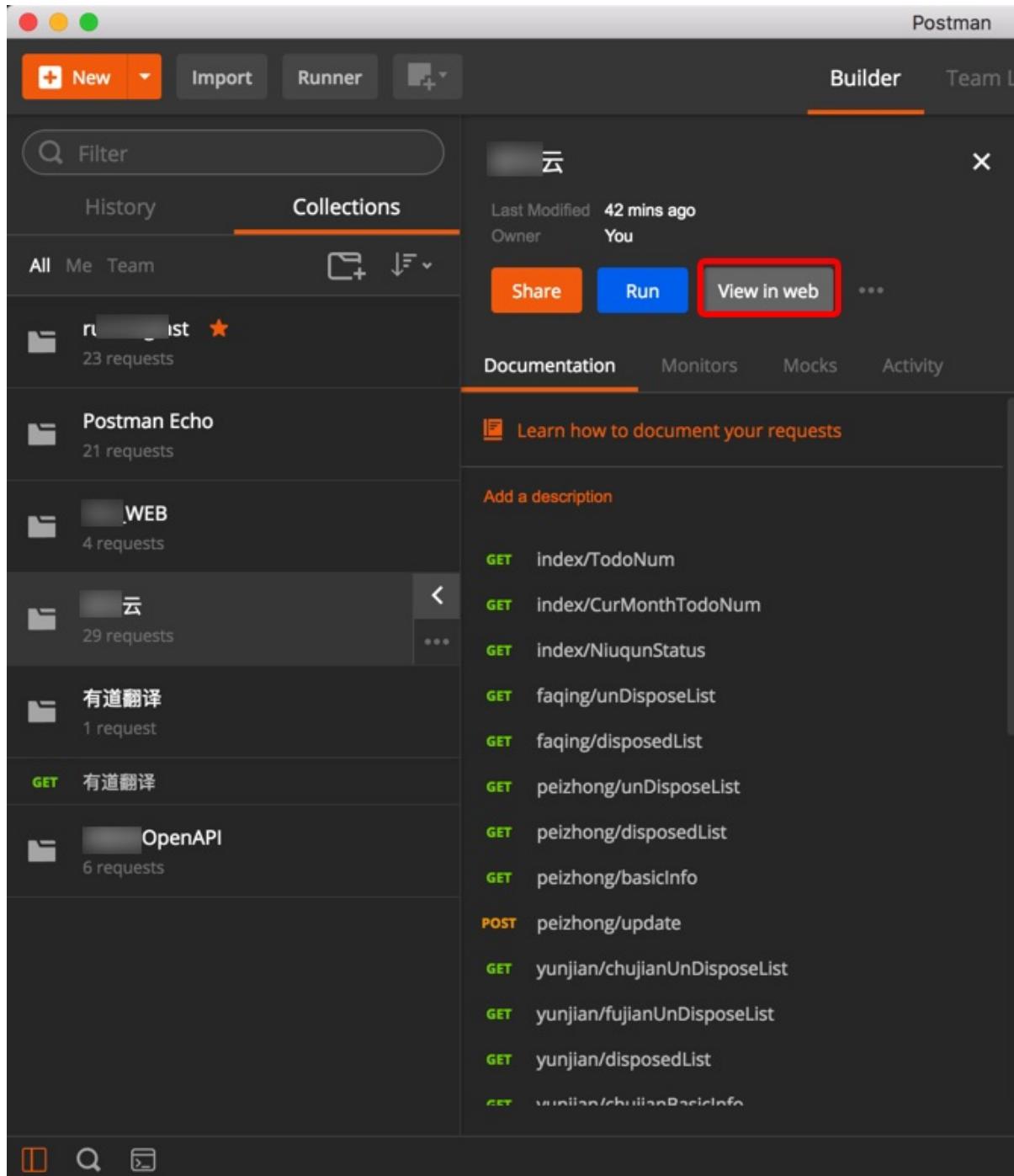
### 详细解释具体操作

#### 预览API文档

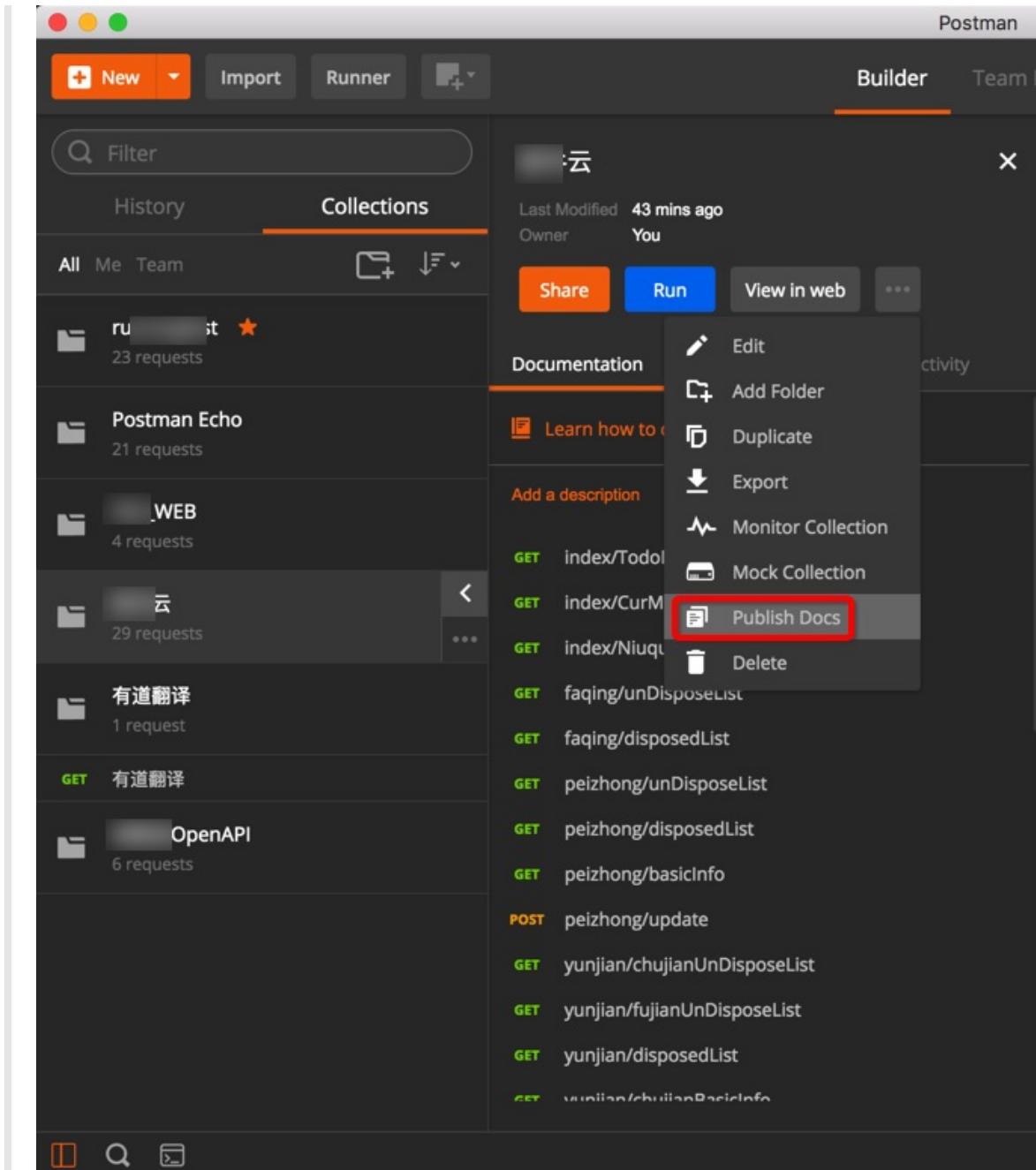
点击分组右边的大于号>



如果只是预览，比如后台开发员自己查看API文档的话，可以选择：View in web



等价于点击Publish Docs去发布：



View in Web后，有 Publish的选项（见后面的截图）

View in Web后，会打开预览页面：

比如：

奶牛云

<https://documenter.getpostman.com/collection/view/669382-42273840-6237-dbae-5455-26b16f45e2b9>

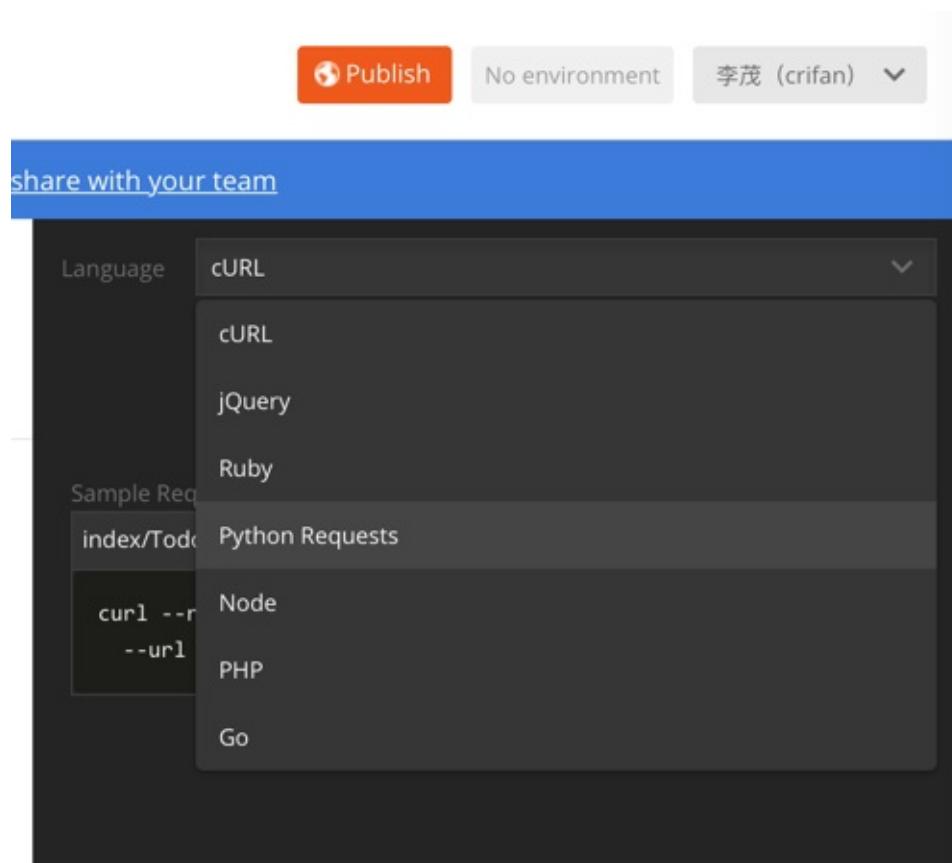
The screenshot shows the Postman application interface. On the left, there is a sidebar with a list of API endpoints categorized under 'Introduction'. The main area displays two API requests:

- GET index/TodoNum**: Shows a sample request URL: `http://116.57/ucows/index/index/TodoNum`. To its right is a cURL command: `curl --request GET \ --url http://116.62.25.57/ucows/index/index/TodoNum`.
- GET index/CurMonthTodoNum**: Shows a sample request URL: `http://116.57/ucows/index/index/CurMonthTodoNum`. To its right is a cURL command: `curl --request GET \ --url http://116.62.25.57/ucows/index/index/CurMonthTodoNum`.

Below these, another section shows a POST request:

- POST peizhong/update**: Shows a sample request URL: `http://116.57/ucows/peizhong/peizhong/update`. To its right is a cURL command: `curl --request POST \ --url http://116.57/ucows/peizhong/peizhong/update \ --header 'authorization: Bear xxx xxx' \ --header 'content-type: application/json' \ --data '{ "id": "88", "cow_code": "16-6963", "isPeizhong": "1", "gongniucow_id": "101202303", "jingyenum": 100, "peizhong_date": "2017-06-30", "isSexControl": 1, "peizhongyuan": "101" }'`.

而右边的示例代码，也可以从默认的cURL换成其他的：



The screenshot shows the Postman application interface with the following details:

- Left Sidebar (API Endpoints):**
  - GET yunjian/fujianUnDisposeList
  - GET yunjian/disposedList
  - GET yunjian/chujianBasicInfo
  - GET yunjian/fujianBasicInfo
  - GET exception/unDisposeList
  - GET exception/disposedList
  - GET cow/search/list
  - GET cow/basicInfoByCowCode
  - GET cow/basicInfoByBd
  - POST cow/updateBasicInfo
  - GET cowshed/search/list
  - GET emp/empList/14 or /15
  - GET equipment/list
  - GET dict/fanzhiStatusList
  - POST login
  - GET report/niuqunStructure
  - GET dict/queryGongnju
  - GET equipment/equipment/checkCowMbind
  - POST login/login
- Middle Section (API Requests):**
  - GET index/TodoNum**
  - http://116 .57/ucows/index/index/TodoNum
  - http://116 .57/ucows/index/index/TodoNum
- Right Section (Code Samples):**
  - Sample Request: index/TodoNum**

```
import requests

url = "http://116 .57/ucows/index/index/TodoNum"

response = requests.request("GET", url)

print(response.text)
```

  - Sample Request: index/CurMonthTodoNum**

```
import requests

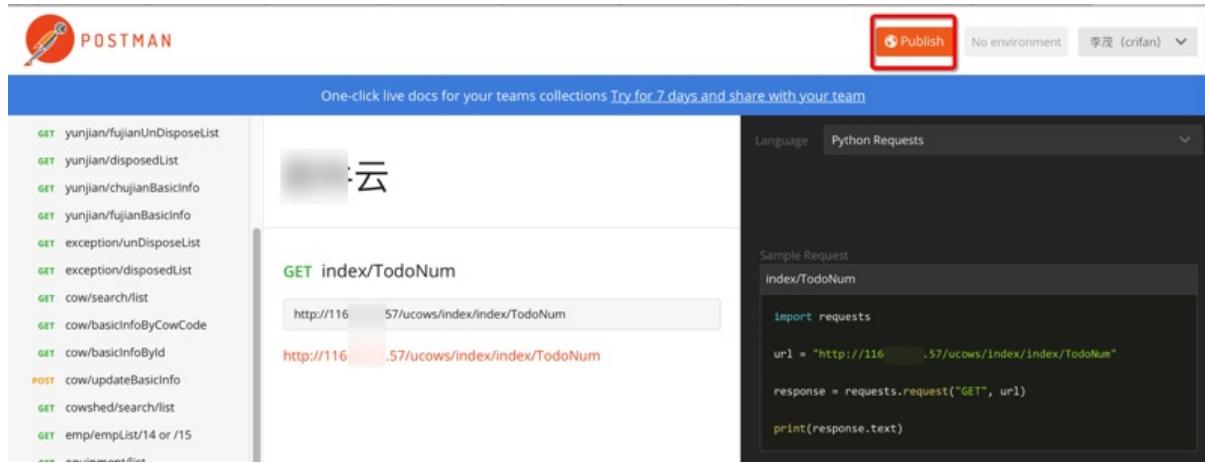
url = "http://116 .57/ucows/index/index/CurMonthTodoNum"

response = requests.request("GET", url)

print(response.text)
```

## 发布API文档

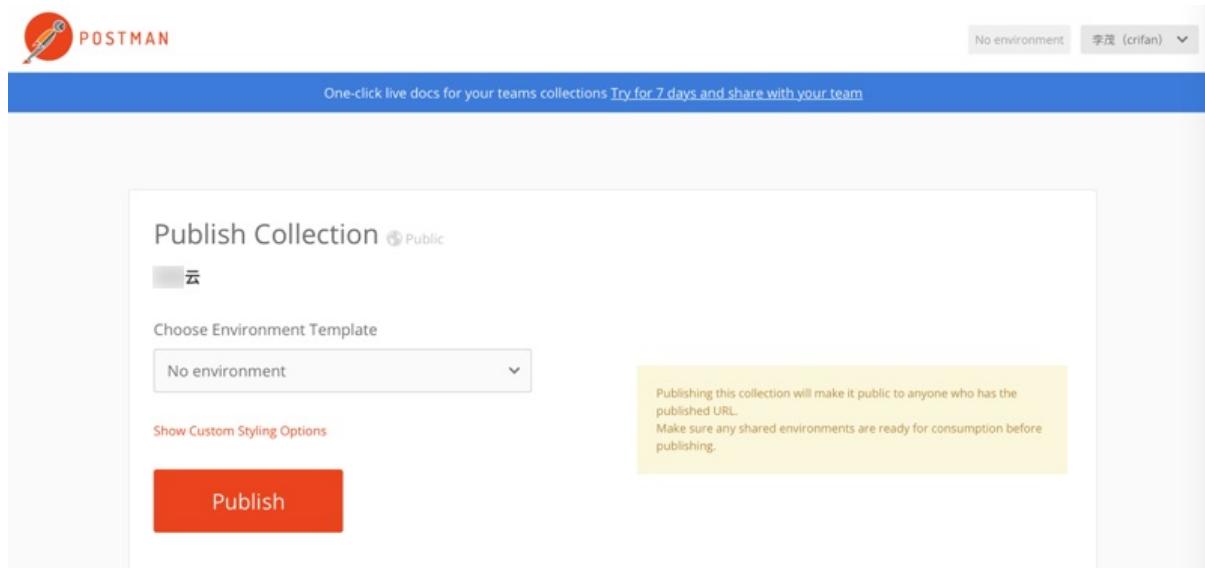
如果想要让其他人能看到这个文档，则点击 Publish：



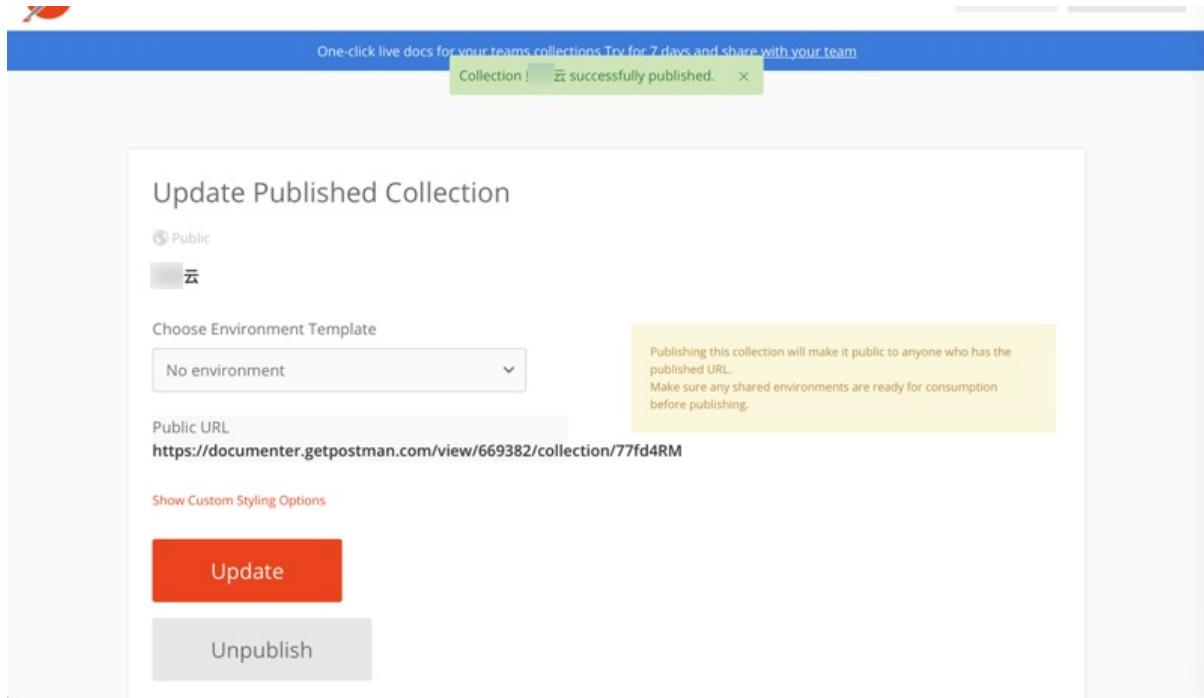
然后会打开类似于这样的地址：

#### Postman Documenter

[https://documenter.getpostman.com/collection/publish?](https://documenter.getpostman.com/collection/publish?meta=Y29sbGVjdG1vb19pZD00MjI3Mzg0MC02MjM3LWRiYWUtNTQ1NS0yNmIxNmY0NWUyYjKmb3duZXI9NjY5MzgyJmNvbGx1Y3RpB25fbmFtZT01RTU1QTU1QjY1R Tc1ODk1OU1RTQ1QkE1OTE=)



点击Publish后，可以生成对应的公开的网页地址：



打开API接口文档地址：

<https://documenter.getpostman.com/view/669382/collection/77fd4RM>

即可看到（和前面预览一样效果的API文档了）：

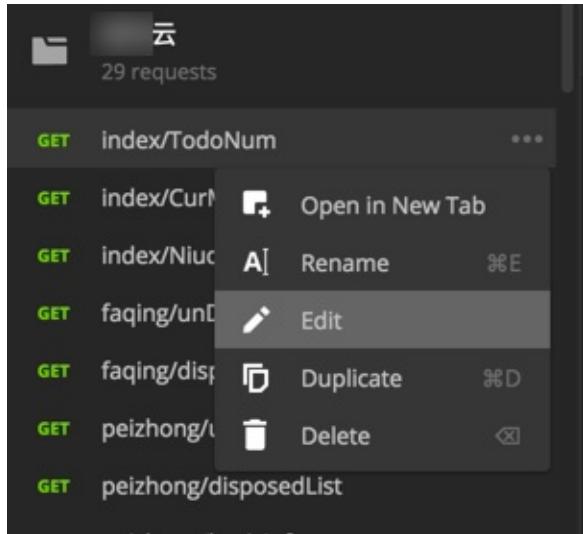
The screenshot shows a detailed API documentation page. On the left, a sidebar lists various endpoints under 'Introduction' and '云'. The main content area shows two examples for the 'GET index/TodoNum' endpoint. Each example includes a URL field (e.g., 'http://116.57/ucows/index/index/TodoNum') and a preview window showing the response. To the right, there are two code snippets in Python Requests for generating these requests. The first snippet is for 'Index/TodoNum' and the second for 'Index/CurMonthTodoNum'. Both snippets include imports, URLs, requests, and print statements for the response text.

如此，别人即可查看对应的API接口文档。

## 已发布的API文档支持自动更新

后续如果自己的API接口修改后：

比如：



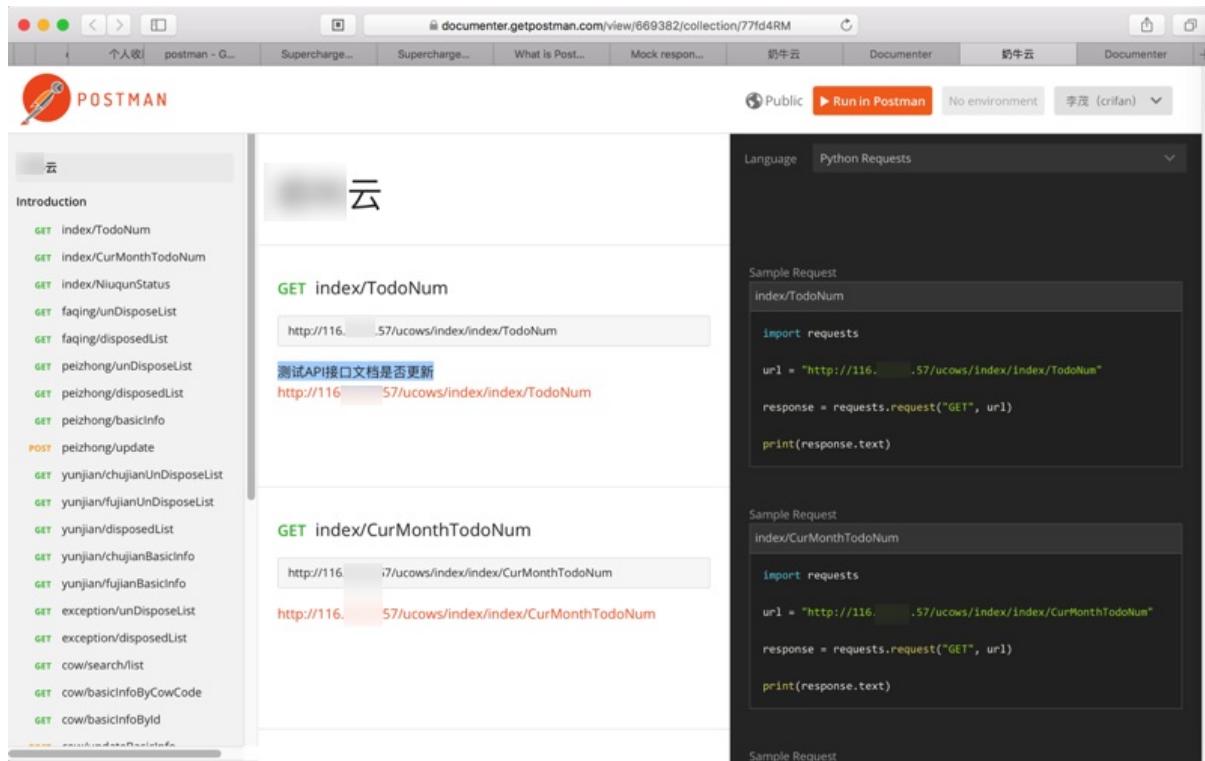
A screenshot of the Postman application interface. On the left, the sidebar shows collections like 'History', 'Collections', and 'Index/TodoNum'. The main area is titled 'EDIT REQUEST' for the 'index/TodoNum' endpoint. The 'Name' field is set to 'index/TodoNum'. In the 'Description' field, there is updated documentation: '测试API接口文档是否更新| http://116.57/ucows/index/index/TodoNum'. The 'Authorization' section shows 'No Auth'. At the bottom right of the dialog are 'Cancel' and 'Update' buttons.

(后来发现，不用再去进入此预览和发布的流程，去更新文档，而是Postman自动支持)

别人去刷新该文档的页面：

<https://documenter.getpostman.com/view/669382/collection/77fd4RM>

即可看到更新后的内容：



crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-29 21:24:55

## 附录

下面列出相关参考资料。

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2017-12-14 11:21:48

## 参考资料

- Manage environments
- postman-变量/环境/过滤等 - 简书
- Postman使用手册3——环境变量 - 简书
- [postman使用之四：切换环境和设置读取变量 - 乔叶叶 - 博客园](#)
- 

crifan.com, 使用[知识署名-相同方式共享4.0协议](#)发布 all right reserved, powered by Gitbook该文件修订时间: 2018-01-02 11:23:28