[一篇文章搞定Github API 调用 (v3）](https://segmentfault.com/a/1190000015144126)

[[](https://segmentfault.com/u/solomonxie)**SolomonXie**](https://segmentfault.com/u/solomonxie)发布于 2018-06-02

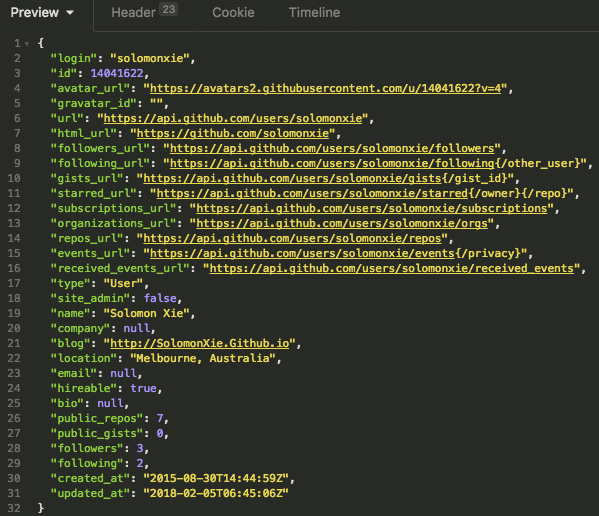
对于常用Github的用户来说，经常有一些自动化的需求。比如我的需求是定时备份Github的issues和comments到本地。以下为Github的API的使用参考。

[v3版API的文档链接](https://developer.github.com/v3/)  
[v3版API的官方教程](https://developer.github.com/v3/guides/getting-started/)

基本访问路径 (Root Endpoints)

一开始读文档的时候，照着它的事例直接在命令行里curl，或者在InSomnia或Postman软件里访问，都完美显示200状态。可是一旦把链接里改写成自己的用户名就各种显示404无页面。还以为是授权问题，然后在页头HEADER中按照各种方式试了username和token密钥，都没用还是404。结果发现，原来不是方法的问题，纯粹是链接地址没写对！  
实际上只是读取的话，完全不用任何授权，可以在命令行、Insomnia、网页等各种情况下直接输入链接访问任何人的所有公开信息。  
然后对照[官方路径列表Root Endpoints](https://api.github.com/)得到的链接，好像怎么访问都不对。反而在Stackoverflow中看到的一个链接，顺藤摸瓜自己发现了各种正确的访问路径，总结如下：

* 首先！访问的链接最后不能有/。如https://api.github.com/users/solomonxie是可以访问到我个人信息的，但是https://api.github.com/users/solomonxie/就不行了，唯一不同是多了一个/.
* 其次！不同于一般URL访问，GIthub的API访问链接是区分大小写的！
* 个人主要信息。 https://api.github.com/users/用户名,得到数据如下图：



* 个人所有repo。https://api.github.com/users/用户名/repos。会得到一个repo的JSON格式列表。
* repo详细信息。https://api.github.com/repos/用户名/仓库名。repo的路径就开始和个人信息不同了。
* 获取某个repo的内容列表。https://api.github.com/repos/solomonxie/gists/contents，注意这只会返回根目录的内容。
* 获取repo中子目录的内容列表。https://api.github.com/repos/solomonxie/gists/contents/目录名。一定要注意这里一定要完全遵循原文件名的大小写，否则无法获得信息。如果是更深层的内容，则在链接列按照顺序逐级写上目录名称。
* 获取repo中某文件信息（不包括内容）。https://api.github.com/repos/solomonxie/gists/contents/文件路径。文件路径是文件的完整路径，区分大小写。只会返回文件基本信息。
* 获取某文件的原始内容（Raw）。1. 通过上面的文件信息中提取download\_url这条链接，就能获取它的原始内容了。2. 或者直接访问：https://raw.githubusercontent.com/用户名/仓库名/分支名/文件路径
* repo中所有的commits列表。https://api.github.com/repos/用户名/仓库名/commits。
* 某一条commit详情。https://api.github.com/repos/用户名/仓库名/commits/某一条commit的SHA
* issues列表。https://api.github.com/repos/用户名/仓库名/issues。
* 某条issue详情。https://api.github.com/repos/用户名/仓库名/issues/序号。issues都是以1,2,3这样的序列排号的。
* 某issue中的comments列表。https://api.github.com/repos/用户名/仓库名/issues/序号/comments。
* 某comment详情。https://api.github.com/repos/用户名/仓库名/issues/comments/评论详情的ID。其中评论ID是从issues列表中获得的。

查询参数 (Parameters)

如果在上面基本链接中加入查询条件，那么返回的数据就是filtered，过滤了的。比如要求只返回正在开放的issues，或者让列表数据分页显示。常用如下：

* 分页功能。格式是?page=页数&per\_page=每页包含数量。

如https://api.github.com/users/solomonxie/repos?page=2&per\_page=3

* issues状态。格式是?state=状态。

如https://api.github.com/repos/solomonxie/solomonxie.github.io/issues?state=closed

权限认证 Authentication

首先需要知道都是，到此为止之前所有都查询都是不需要任何权限的，给个地址就返回数据，全公开。  
但是创建文件、更新、删除等就是必须用自己的账号"登录"才能实现的。所以为了下面的增删改做准备，需要先看一下权限问题。  
官网虽然写的很简答，不过如果不熟悉API的话还是不能马上就理解。

常用的认证方法有三种，Basic authentication, OAuth2 token, OAuth2 key/secret  
三种方法效果一样，但是各有其特点和方便之处。选哪种就要看自己哪种方便了。

认证方法一：Basic authentication

这种最简单，如果是用curl的话，就：

curl -u "用户名:密码" https://api.github.com

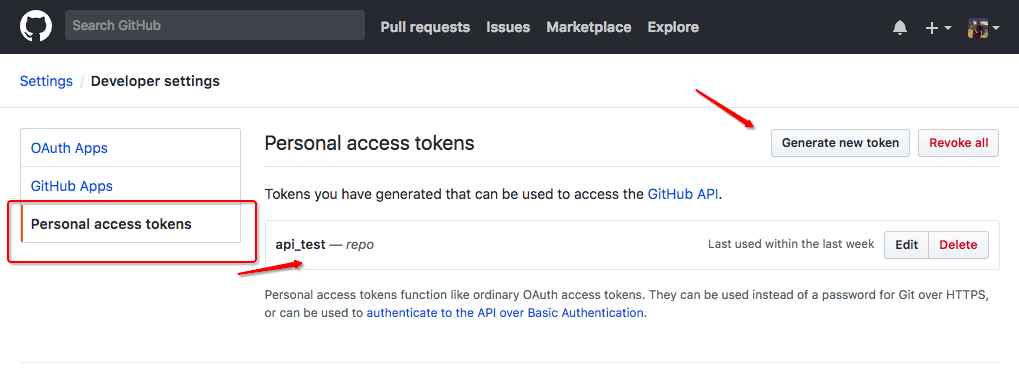
如果是用Insomnia等api调试工具的话，直接在Auth选项栏里选Basic Auth，然后填上用户名密码即可。

认证方法二：OAuth2 token

关于token

这种token方式，说实话如果不是操作过API或深度了解REST的话，是很难理解的东西。  
说白了就是第二个密码，你既不用到处泄露自己的用户名密码，又可以专门给这个"第二密码"设置不同需要的权限，如有的只可读有的还可以写等。而且这个“第二密码”是既包括用户名又包括密码功能的，全站只此一个绝对不会和别人重复。初次之外，你还可以设置很多个token，也就是第三、第四、第五...密码。很方便。

设置token方法

就位于github个人账号设置->开发者设置->个人token里。创建一个新token时，可以选择具体的权限，创建成功时一定要复制到本地哪里保存，只会让你看见一次，如果忘记的话就需要重新生成（其实丢了也不算麻烦）。  


另外！注意：

token字符串不能存储在github的repo中，经过测试，一旦提交的文件中包含这个token字符串，那么github就会自动删除这个token -\_-! 我用了很久才明白过来，创建的Personal Access Token总是自动消失，还以为是有时限的。

用token通过权限认证

有两种传送方法，哪种都可以：

1. 作为url中的参数明文传输：

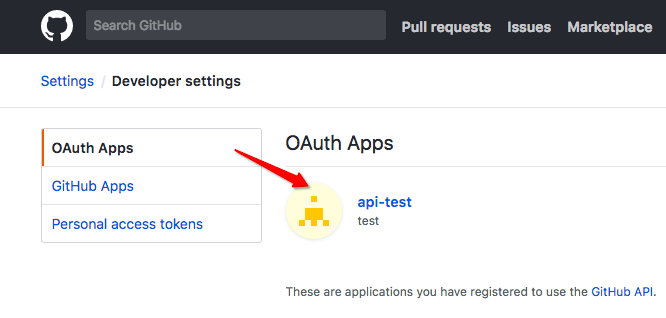
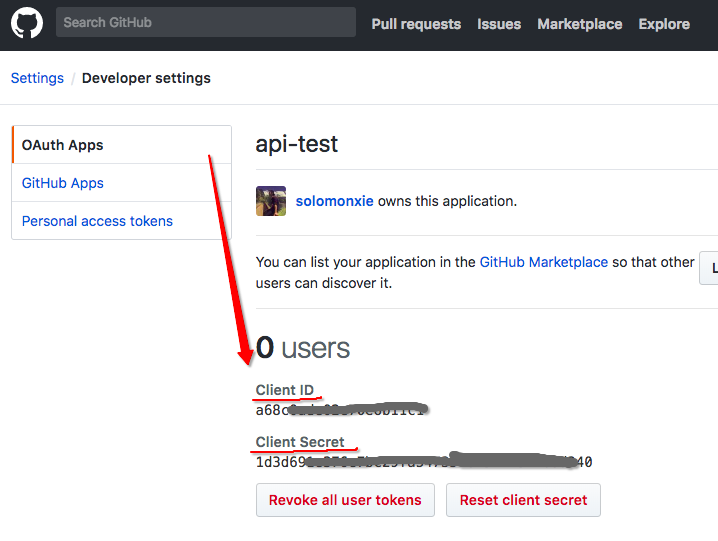
curl https://api.github.com/?access\_token=OAUTH-TOKEN

1. 作为header中的参数传输：

curl -H "Authorization: token OAUTH-TOKEN" https://api.github.com

如果不是用curl而是Insomnia测试的话，和上面basic auth是大同小异的，很容易操作就不复述了。  
到此为止，权限认证就算搞清了，而且也实际验证过有效了。强烈建议用insomnia工具操作，有GUI界面方便理解，成功后再转为curl或python等程序语言。

认证方法三：OAuth2 key/secret

这个是除了Personal Access Token之外的另一种好用的方法，即创建自己的OAuth app，然后得到一对client\_id和client\_secret。如下：  
  
  
得到这两个值之后，直接在访问任何api的url连接后面显性加上这两个参数即可完成认证，如：  
https://api.github.com/users/yourusername?client\_id=YOUR-CLIENT-ID&client\_secret=YOUR-CLIENT-SECRET  
但是：

目前这种认证方式**不支持**查询以外的操作，也就是只能GET获取某些api信息，不能执行request里的任何PUT/PATCH/DELETE操作。

创建新文件 Create content

[Contents操作 官方文档](https://developer.github.com/v3/repos/contents/#get-contents)

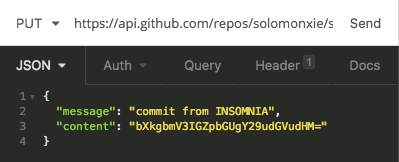
* 传输方法：PUT
* 访问路径：https://api.github.com/repos/用户名/仓库名/contents/文件路径
* JSON格式：

{

"message": "commit from INSOMNIA",

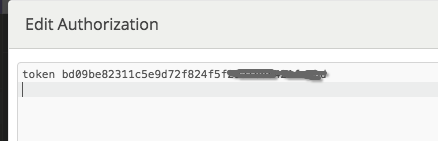
"content": "bXkgbmV3IGZpbGUgY29udGVudHM="

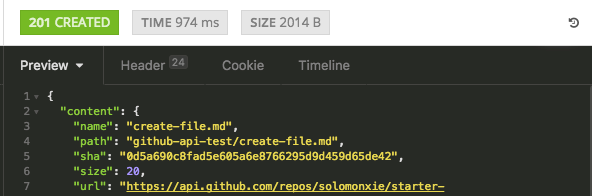
}

JSON填写如下图：  


* 注意：1.必须添加权限验证（上面有写） 2. 数据传送格式选择JSON 3. 文件内容必须是把文件整体转为Base64字符串再存到JSON变量中 4. 文件路径中如果有不存在的文件夹，则会自动创建

起初不管怎么尝试都一直报同样都错误，400 Invalid JSON，如下图：  
[图片上传失败...(image-884e71-1527903120996)]

最后发现原来是犯了很小很小都错误才导致如此：  
  
原来，我的token看似是正常的，唯独错误的是，多了一个空行！也就是说，标明都是invalid JSON，结果没注意竟然是invalid Token!

增加文件成功后返回的消息：  


更新文件 Update content

主要这几点： 1. 传送方式用PUT 和创建文件一样 2. 需要权限验证，3. 传输内容数据用JSON 4. 需要指定该文件的SHA码 4. 路径和访问content时一样 5. 文件内容必须是把文件整体转为Base64字符串再存到JSON变量中

* 传输方法：PUT
* 访问路径：https://api.github.com/repos/用户名/仓库名/contents/文件路径
* JSON格式：

{

"message": "update from INSOMNIA",

"content": "Y3JlYXRlIGZpbGUgZnJvbSBJTlNPTU5JQQoKSXQncyB1cGRhdGVkISEhCgpJdCdzIHVwZGF0ZWQgYWdhaW4hIQ==",

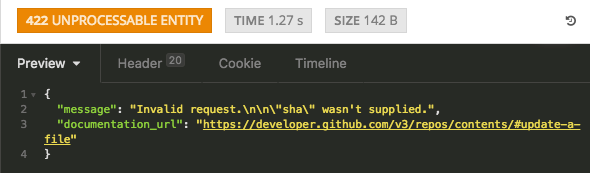
"sha": "57642f5283c98f6ffa75d65e2bf49d05042b4a6d"

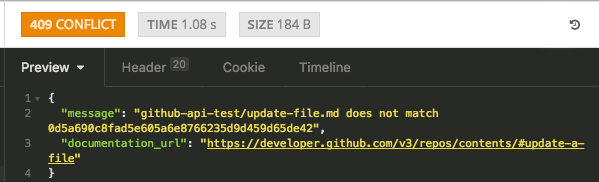
}

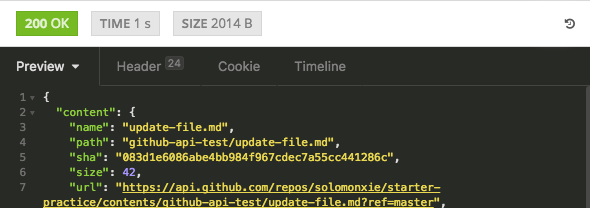
* 注意：必须指定该文件的SHA码，相当于文件的ID。

SHA虽然是对文件的唯一识别码，相当于ID，但是它是会随着文件内容变化而变化的！所以必须每次都重新获取才行。

至于获取方式，验证后发现，目前最靠谱的是用前面的get content获取到该文件的信息，然后里面找到sha。

对上传时的JSON格式另有要求，如果没有按照要求把必填项输入，则会出现422错误信息：  


或者如果用错了SHA，会出现409错误消息：  


如果正确传送，就会显示200完成更新：  


删除文件 Delete content

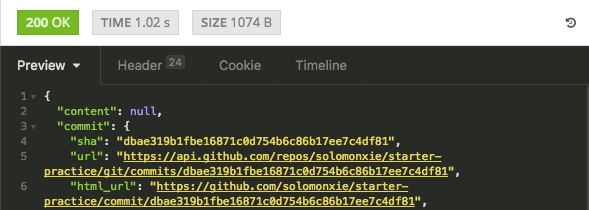
* 传输方法：DELETE
* 访问路径：https://api.github.com/repos/用户名/仓库名/contents/文件路径
* JSON格式：

{

"message": "delete a file",

"sha": "46d2b1f2ef54669a974165d0b37979e9adba1ab2"

}

删除成功后，会返回200消息：  


增删改issues

如果做过了上面文件的增删改，这里大同小异，不同的访问路径和JSON的格式而已。唯一不同的是，issues是不用把内容转为Base64码的。

参考链接：[github官方文档](https://developer.github.com/v3/issues/#create-an-issue)

增加一条issue

* 传输方法：POST
* 访问路径：https://api.github.com/repos/用户名/仓库名/issues
* JSON格式：

{

"title": "Creating issue from API",

"body": "Posting a issue from Insomnia"

}

* 注意：issue的数据里面是可以加label，milestone和assignees的。但是必须注意milestone和assignees必须是已有的名次完全对应才行，否则无法完成创建。

更改某条issue

* 传输方法：PATCH
* 访问路径：https://api.github.com/repos/用户名/仓库名/issues/序号
* JSON格式：

{

"title": "Creating issue from API ---updated",

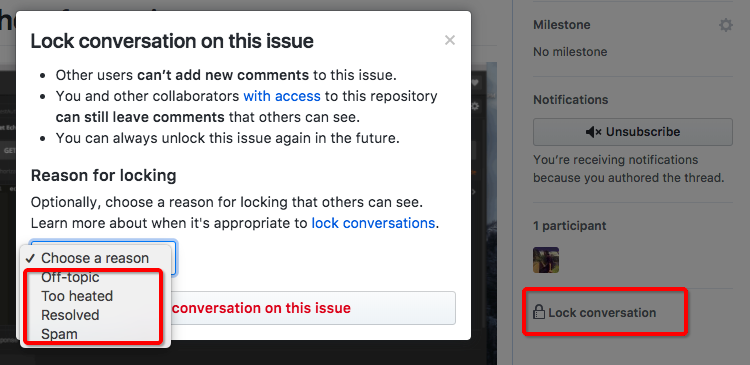
"body": "Posting a issue from Insomnia \n\n Updated from insomnia.",

"state": "open"

}

* 注意：如果JSON中加入空白的labels或assignees，如"labels": []，作用就是清空所有的标签和相关人。

锁住某条issue

不允许别人评论（自己可以）  


* 传输方法：PUT
* 访问路径：https://api.github.com/repos/用户名/仓库名/issues/序号/lock
* JSON格式：

{

"locked": true,

"active\_lock\_reason": "too heated"

}

* 注意：active\_lock\_reason只能有4种值可选：off-topic, too heated, resolved, spam，否则报错。

另外，成功锁住，会返回204 No Content信息。

解锁某条issue

* 传输方法：DELETE
* 访问路径：https://api.github.com/repos/用户名/仓库名/issues/序号/lock
* 无JSON传输

增删改comments

参考[官方文档](https://developer.github.com/v3/issues/comments/#create-a-comment)

增加comment

* 传输方法：POST
* 访问路径：https://api.github.com/repos/用户名/仓库名/issues/序号/comments
* JSON格式：

{

"body": "Create a comment from API"

}

更改comment

* 传输方法：PATCH
* 访问路径：https://api.github.com/repos/用户名/仓库名/issues/comments/评论ID
* JSON格式：

{

"body": "Create a comment from API \n\n----Updated"

}

* 注意：地址中，issues后不用序号了，因为可以通过唯一的评论ID追查到。查看评论ID的方法，直接在上面查询链接中找。

删除comment

* 传输方法：DELETE
* 访问路径：https://api.github.com/repos/用户名/仓库名/issues/comments/评论ID
* 无传输数据