目錄

介绍 1.1
身份验证 1.2
    Authorizations 1.2.1
用户 1.3
    Email 1.3.1
    Followers 1.3.2
搜索 1.4
活动（Activity） 1.5
    Starring 1.5.1
    Events 1.5.2
    Watching 1.5.3

介绍说明

这是个人在使用github api v3时，顺带做的笔记，并非是翻译（个人英文水准说实话堪忧）。

基础Api地址

https://api.github.com
直接点击打开，可看到当前所有可用接口。

Api调用方式

大多数api调用都需要经过验证授权，部分可以不经过授权（如搜索api），经过授权的api调用具有可观的调用限制，未授权的api调用，短时间内无法大量调用。

数据分页

大部分的数据获取API，可通过请求参数page以及per_page(默认为30)控制当前数据页以及每页的数据量。

身份验证

介绍web与非web端进行github身份验证的方法。

调用所有api接口需要有验证；验证有以下四种方式：

1. Basic Authentication
   `curl -u "username" https://api.github.com`
2. OAuth2 Token(sent in a header)
   `curl -H "Authorization: token OAUTH-TOKEN" https://api.github.com`
3. OAuth2 Token(sent as a parammter)
4. `curl https://api.github.com/?access_token=OAUTH-TOKEN`
5. OAuth2 Key/Secret
6. `curl 'https://api.github.com/users/whatever?client_id=xxxx&client_secret=yyyy'`

Basic Authentication

基础验证方式，简单的说就是将(username:password)的字符串（不含左右括号），进行Base64编码后(记为code)，拼接成如下: `Basic code` 放入http request的头（"Authorization"）中，随请求一起发送。

OAuth2 Token(sent in a header)

将授权token码，放入头（"Authorization"）中。
token码可以使用Basic Authentication请求/authorizations获得。
也可使用OAuth2 Key/Secret方式获得。

OAuth2 Token(sent as a parammter)

与上一种的区别仅仅是当作请求参数发送。

OAuth2 Key/Secret

使用申请的App的Client_id以及client_secret拼接验证url，直接访问进行授权验证（最简单）。

Authorizations

POST \/authorizations

使用基本身份验证（登录用户名:登录密码）的方式获取一个OAuth2.0的授权。

如果用户有两步验证，需在请求时添加

Header `X-GitHub-OTP:code` (code为两步验证码)

参数使用json形式提交

`请求参数Params`

参数名	类型	含义
note	String	`必要` 用来记录获取授权的目的
client_id	String	20字长的Id，建立Application时获得
client_secret	String	40字长的Secret，建立Application时获得
fingerprint	String	不能重复的字符串，对于同一id和用户，重复请求时，fingerprint不能重复
note_url	String	授权app的介绍地址
scopes	array	`权限` ，可空，但有些api的调用有对应的scope才行，否则调用不成功,详情看Scopes

```
{
  "scopes": [
    "public_repo"
  ],
  "note": "admin script"
  "client_id":"xxxxxxxxxx",
  "client_secret":"xxxxxxxxx"
}
```

`Response` httpCode:201

```
{
  "id": 1,
  "url": "https://api.github.com/authorizations/1",
  "scopes": [
    "public_repo"
  ],
  "token": "abcdefgh12345678",
  "token_last_eight": "12345678",
  "hashed_token": "25f94a2a5c7fbaf499c665bc73d67c1c87e496da89851
31633ee0a95819db2e8",
  "app": {
    "url": "http://my-github-app.com",
    "name": "my github app",
    "client_id": "abcde12345fghij67890"
  },
  "note": "optional note",
  "note_url": "http://optional/note/url",
  "updated_at": "2011-09-06T20:39:23Z",
  "created_at": "2011-09-06T17:26:27Z",
  "fingerprint": ""
}
```

Response结果中，一般只需将token记录下来即可，注意scopes对应权限。

如果需要二步验证，Response的Header中"X-GitHUb-OTP"的值将是"required"。下次请求带上二步验证码即可。

GET \/applications\/{client_id}\/tokens\/{token}

使用基础身份验证方式（client_id:client_sercret）检查token值是否有效。

{}包裹的是对应的参数值。

`Response` httpCode:200

```
{
  "id": 1,
  "url": "https://api.github.com/authorizations/1",
  "scopes": [
    "public_repo"
  ],
  "token": "abcdefgh12345678",
  "token_last_eight": "12345678",
  "hashed_token": "25f94a2a5c7fbaf499c665bc73d67c1c87e496da89851
31633ee0a95819db2e8",
  "app": {
    "url": "http://my-github-app.com",
    "name": "my github app",
    "client_id": "abcde12345fghij67890"
  },
  "note": "optional note",
  "note_url": "http://optional/note/url",
  "updated_at": "2011-09-06T20:39:23Z",
  "created_at": "2011-09-06T17:26:27Z",
  "fingerprint": "jklmnop12345678",
  "user": {
    "login": "octocat",
    "id": 1,
    "avatar_url": "https://github.com/images/error/octocat_happy
.gif",
    "gravatar_id": "",
    "url": "https://api.github.com/users/octocat",
    "html_url": "https://github.com/octocat",
    "followers_url": "https://api.github.com/users/octocat/follo
wers",
    "following_url": "https://api.github.com/users/octocat/follo
wing{/other_user}",
    "gists_url": "https://api.github.com/users/octocat/gists{/gi
st_id}",
    "starred_url": "https://api.github.com/users/octocat/starred
{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/octocat/s
ubscriptions",
    "organizations_url": "https://api.github.com/users/octocat/o
rgs",
    "repos_url": "https://api.github.com/users/octocat/repos",
    "events_url": "https://api.github.com/users/octocat/events{/
privacy}",
    "received_events_url": "https://api.github.com/users/octocat
```

```
/received_events",
    "type": "User",
    "site_admin": false
  }
}
```

有正常返回，说明token值有效。无效将会返回 `httpCode 404` 。

User

GET /users/{username}

获取中指定用户信息，`username指的是用户的登录名(login)`

`Response` httpCode:200

```
{
  "login": "octocat",
  "id": 1,
  "avatar_url": "https://github.com/images/error/octocat_happy.g
if",
  "gravatar_id": "",
  "url": "https://api.github.com/users/octocat",
  "html_url": "https://github.com/octocat",
  "followers_url": "https://api.github.com/users/octocat/followe
rs",
  "following_url": "https://api.github.com/users/octocat/followi
ng{/other_user}",
  "gists_url": "https://api.github.com/users/octocat/gists{/gist
_id}",
  "starred_url": "https://api.github.com/users/octocat/starred{/
owner}{/repo}",
  "subscriptions_url": "https://api.github.com/users/octocat/sub
scriptions",
  "organizations_url": "https://api.github.com/users/octocat/org
s",
  "repos_url": "https://api.github.com/users/octocat/repos",
  "events_url": "https://api.github.com/users/octocat/events{/pr
ivacy}",
  "received_events_url": "https://api.github.com/users/octocat/r
eceived_events",
  "type": "User",
  "site_admin": false,
  "name": "monalisa octocat",
  "company": "GitHub",
  "blog": "https://github.com/blog",
  "location": "San Francisco",
  "email": "octocat@github.com",
  "hireable": false,
  "bio": "There once was...",
  "public_repos": 2,
  "public_gists": 1,
  "followers": 20,
  "following": 0,
  "created_at": "2008-01-14T04:33:35Z",
  "updated_at": "2008-01-14T04:33:35Z"
}
```

参数名	含义
login	用户的登录名
id	用户Id
avatar_url	头像地址
xxx_url	便捷的一些信息接口，可直接get调用
type	User:用户 、Orgnization:组织
bio	个人介绍

GET /user

获取授权用户的信息，返回结果同上。

PATCH /user

更新授权用户的信息。

参数使用json形式提交。

`请求参数Params`

参数	类型	含义
name	String	名字(Name)，不是登录名(login)
email	String	邮件地址
blog	String	博客地址
company	String	所属公司
location	String	所在地
hireable	boolean	是否接受工作雇佣
bio	String	个人介绍

```
{
  "name": "monalisa octocat",
  "email": "octocat@github.com",
  "blog": "https://github.com/blog",
  "company": "GitHub",
  "location": "San Francisco",
  "hireable": true,
  "bio": "There once..."
}
```

Email

`需要user:email scope`

GET /user/emails

列出验证用户的email地址。

`Response 200`

```
[
  {
    "email": "octocat@github.com",
    "verified": true,
    "primary": true
  }
]
```

POST /user/emails

添加验证用户的邮箱地址（可一次添加多个）

`Request param` 以数组形式发送数据：

```
[
  "octocat@github.com",
  "support@github.com"
]
```

`Response 201`

```
[
  {
    "email": "octocat@github.com",
    "primary": false,
    "verified": false
  },
  {
    "email": "support@github.com",
    "primary": false,
    "verified": false
  }
]
```

DELETE /user/emails

删除验证用户的指定邮箱地址。

`Request param` 同添加。

`Response 204`

Followers

GET /users/{username}/followers

列举某个用户的followers。

GET /user/followers

列举验证用户的followers。同上。

搜索

使用API对github资源进行搜索

GET /search/repositories

搜索仓库。
request请求参数：

参数	类型	含义
q	string	搜索关键字
sort	string	排序依据，值可取stars,forks,updated。默认为best match。
order	string	排序顺序，升或降。值可取asc或desc。

以上参数均非必传。
对于q字段，有以下扩展用法，语法格式为 `q=keywords+key:value+key2:value`

参数	类型	含义
in	string	在何字段中对q进行搜索。可用值为repository相关response中的字段
forks	num	forks字段过滤
fork	string	是否返回那些fork的项目。值取true、false（不获取fork的项目到response中）、only（仅获取fork的项目到response中）。
stars	num	同forks，过滤使用
language	string	搜索的语言类型，例java,c,python等

例如 `https://api.github.com/search/repositories?q=tetris+language:assembly&sort=stars&order=desc`

Response

```
{
  "total_count": 40,
  "incomplete_results": false,
  "items": [
    {
      "id": 3081286,
      "name": "Tetris",
      "full_name": "dtrupenn/Tetris",
      "owner": {
        "login": "dtrupenn",
        "id": 872147,
        "avatar_url": "https://secure.gravatar.com/avatar/e79560
84e75f239de85d3a31bc172ace?d=https://a248.e.akamai.net/assets.gi
thub.com%2Fimages%2Fgravatars%2Fgravatar-user-420.png",
        "gravatar_id": "",
        "url": "https://api.github.com/users/dtrupenn",
        "received_events_url": "https://api.github.com/users/dtr
upenn/received_events",
        "type": "User"
      },
      "private": false,
      "html_url": "https://github.com/dtrupenn/Tetris",
      "description": "A C implementation of Tetris using Pennsim
 through LC4",
      "fork": false,
      "url": "https://api.github.com/repos/dtrupenn/Tetris",
      "created_at": "2012-01-01T00:31:50Z",
      "updated_at": "2013-01-05T17:58:47Z",
      "pushed_at": "2012-01-01T00:37:02Z",
      "homepage": "",
      "size": 524,
      "stargazers_count": 1,
      "watchers_count": 1,
      "language": "Assembly",
      "forks_count": 0,
      "open_issues_count": 0,
      "master_branch": "master",
      "default_branch": "master",
      "score": 10.309712
    }
  ]
}
```

total_count为搜索结果总数，因此别忘了使用page和per_page进行结果分组返回。score为搜索匹配度。

GET /search/users

搜索用户。 request请求参数：

参数	类型	含义
q	string	搜索关键字
sort	string	排序依据，值可为followers, repositories, or joined。默认为best match
order	string	排序顺序，desc或asc。默认为desc。

同样q也有扩展字段的用法:

参数	类型	含义
type	string	搜索的用户类型，user或者org
in	string	在何字段中进行q的搜索。通常为user的response中的字段
repos	num	数量过滤字段。根据所拥有的repos数量进行过滤
location	string	所在地过滤
language	string	所拥有的repos语言过滤
followers	num	数量过滤字段

搜索字段匹配结果

以上搜索API，使用`Accept : application/vnd.github.v3.text-match+json`可使返回结果中将多处以下内容，列出所匹配的字段（包括一些不会正常返回的字段，例如搜索user时的email、location字段）。

```
{
  "text_matches": [
    {
      "object_url": "https://api.github.com/repositories/215335/
issues/132",
      "object_type": "Issue",
      "property": "body",
      "fragment": "comprehensive windows font I know of).\n\nIf
we can find a commonly distributed windows font that supports th
em then no problem (we can use html font tags) but otherwise the
 '(21)' style is probably better.\n",
      "matches": [
        {
          "text": "windows",
          "indices": [
            14,
            21
          ]
        },
        {
          "text": "windows",
          "indices": [
            78,
            85
          ]
        }
      ]
    },
    {
      "object_url": "https://api.github.com/repositories/215335/
issues/comments/25688",
      "object_type": "IssueComment",
      "property": "body",
      "fragment": " right after that are a bit broken IMHO :). I
 suppose we could have some hack that maxes out at whatever the
font does...\n\nI'll check what the state of play is on Windows.
\n",
      "matches": [
        {
          "text": "Windows",
          "indices": [
            163,
            170
          ]
        }
      ]
      }
    ]
  }
```

活动（Activity）

Star

仓库收藏相关API

GET /user/starred

列出验证用户star的仓库。

GET /users/{username}/starred

列出某个用户star的仓库。
请求参数request：

参数	类型	含义
sort	string	排序依据，值为created或updated。默认created
direction	string	排序顺序，值为desc或asc

Response:

```
[
  {
    "id": 1296269,
    "owner": {
      "login": "octocat",
      "id": 1,
      "avatar_url": "https://github.com/images/error/octocat_hap
py.gif",
      "gravatar_id": "",
      "url": "https://api.github.com/users/octocat",
      "html_url": "https://github.com/octocat",
      "followers_url": "https://api.github.com/users/octocat/fol
lowers",
      "following_url": "https://api.github.com/users/octocat/fol
lowing{/other_user}",
      "gists_url": "https://api.github.com/users/octocat/gists{/
gist_id}",
      "starred_url": "https://api.github.com/users/octocat/starr
ed{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/octocat
/subscriptions",
      "organizations_url": "https://api.github.com/users/octocat
/orgs",
      "repos_url": "https://api.github.com/users/octocat/repos",
      "events_url": "https://api.github.com/users/octocat/events
{/privacy}",
      "received_events_url": "https://api.github.com/users/octoc
at/received_events",
      "type": "User",
      "site_admin": false
    },
    "name": "Hello-World",
    "full_name": "octocat/Hello-World",
    "description": "This your first repo!",
    "private": false,
    "fork": false,
    "url": "https://api.github.com/repos/octocat/Hello-World",
    "html_url": "https://github.com/octocat/Hello-World",
    "archive_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/{archive_format}{/ref}",
    "assignees_url": "http://api.github.com/repos/octocat/Hello-
World/assignees{/user}",
    "blobs_url": "http://api.github.com/repos/octocat/Hello-Worl
d/git/blobs{/sha}",
    "branches_url": "http://api.github.com/repos/octocat/Hello-W
orld/branches{/branch}",
    "clone_url": "https://github.com/octocat/Hello-World.git",
    "collaborators_url": "http://api.github.com/repos/octocat/He
llo-World/collaborators{/collaborator}",
    "comments_url": "http://api.github.com/repos/octocat/Hello-W
orld/comments{/number}",
    "commits_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/commits{/sha}",
    "compare_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/compare/{base}...{head}",
    "contents_url": "http://api.github.com/repos/octocat/Hello-W
orld/contents/{+path}",
    "contributors_url": "http://api.github.com/repos/octocat/Hel
```

```
lo-World/contributors",
    "deployments_url": "http://api.github.com/repos/octocat/Hell
o-World/deployments",
    "downloads_url": "http://api.github.com/repos/octocat/Hello-
World/downloads",
    "events_url": "http://api.github.com/repos/octocat/Hello-Wor
ld/events",
    "forks_url": "http://api.github.com/repos/octocat/Hello-Worl
d/forks",
    "git_commits_url": "http://api.github.com/repos/octocat/Hell
o-World/git/commits{/sha}",
    "git_refs_url": "http://api.github.com/repos/octocat/Hello-W
orld/git/refs{/sha}",
    "git_tags_url": "http://api.github.com/repos/octocat/Hello-W
orld/git/tags{/sha}",
    "git_url": "git:github.com/octocat/Hello-World.git",
    "hooks_url": "http://api.github.com/repos/octocat/Hello-Worl
d/hooks",
    "issue_comment_url": "http://api.github.com/repos/octocat/He
llo-World/issues/comments{/number}",
    "issue_events_url": "http://api.github.com/repos/octocat/Hel
lo-World/issues/events{/number}",
    "issues_url": "http://api.github.com/repos/octocat/Hello-Wor
ld/issues{/number}",
    "keys_url": "http://api.github.com/repos/octocat/Hello-World
/keys{/key_id}",
    "labels_url": "http://api.github.com/repos/octocat/Hello-Wor
ld/labels{/name}",
    "languages_url": "http://api.github.com/repos/octocat/Hello-
World/languages",
    "merges_url": "http://api.github.com/repos/octocat/Hello-Wor
ld/merges",
    "milestones_url": "http://api.github.com/repos/octocat/Hello
-World/milestones{/number}",
    "mirror_url": "git:git.example.com/octocat/Hello-World",
    "notifications_url": "http://api.github.com/repos/octocat/He
llo-World/notifications{?since, all, participating}",
    "pulls_url": "http://api.github.com/repos/octocat/Hello-Worl
d/pulls{/number}",
    "releases_url": "http://api.github.com/repos/octocat/Hello-W
```

```
orld/releases{/id}",
    "ssh_url": "git@github.com:octocat/Hello-World.git",
    "stargazers_url": "http://api.github.com/repos/octocat/Hello
-World/stargazers",
    "statuses_url": "http://api.github.com/repos/octocat/Hello-W
orld/statuses/{sha}",
    "subscribers_url": "http://api.github.com/repos/octocat/Hell
o-World/subscribers",
    "subscription_url": "http://api.github.com/repos/octocat/Hel
lo-World/subscription",
    "svn_url": "https://svn.github.com/octocat/Hello-World",
    "tags_url": "http://api.github.com/repos/octocat/Hello-World
/tags",
    "teams_url": "http://api.github.com/repos/octocat/Hello-Worl
d/teams",
    "trees_url": "http://api.github.com/repos/octocat/Hello-Worl
d/git/trees{/sha}",
    "homepage": "https://github.com",
    "language": null,
    "forks_count": 9,
    "stargazers_count": 80,
    "watchers_count": 80,
    "size": 108,
    "default_branch": "master",
    "open_issues_count": 0,
    "has_issues": true,
    "has_wiki": true,
    "has_pages": false,
    "has_downloads": true,
    "pushed_at": "2011-01-26T19:06:43Z",
    "created_at": "2011-01-26T19:01:12Z",
    "updated_at": "2011-01-26T19:14:43Z",
    "permissions": {
      "admin": false,
      "push": false,
      "pull": true
    }
  }
]
```

```
通过添加Header(Accept:application/vnd.github.v3.star+json)可获取何
时star了该仓库，返回格式略有改变。
```

```
[
  {
    "starred_at": "2011-01-16T19:06:43Z",
    "repo": {
      "id": 1296269,
      "owner": {
        "login": "octocat",
        "id": 1,
        "avatar_url": "https://github.com/images/error/octocat_h
appy.gif",
        "gravatar_id": "",
        "url": "https://api.github.com/users/octocat",
        "html_url": "https://github.com/octocat",
        "followers_url": "https://api.github.com/users/octocat/f
ollowers",
        "following_url": "https://api.github.com/users/octocat/f
ollowing{/other_user}",
        "gists_url": "https://api.github.com/users/octocat/gists
{/gist_id}",
        "starred_url": "https://api.github.com/users/octocat/sta
rred{/owner}{/repo}",
        "subscriptions_url": "https://api.github.com/users/octoc
at/subscriptions",
        "organizations_url": "https://api.github.com/users/octoc
at/orgs",
        "repos_url": "https://api.github.com/users/octocat/repos
",
        "events_url": "https://api.github.com/users/octocat/even
ts{/privacy}",
        "received_events_url": "https://api.github.com/users/oct
ocat/received_events",
        "type": "User",
        "site_admin": false
      },
      "name": "Hello-World",
      "full_name": "octocat/Hello-World",
      "description": "This your first repo!",
      "private": false,
      "fork": false,
      "url": "https://api.github.com/repos/octocat/Hello-World",
      "html_url": "https://github.com/octocat/Hello-World",
      "archive_url": "http://api.github.com/repos/octocat/Hello-
World/{archive_format}{/ref}",
      "assignees_url": "http://api.github.com/repos/octocat/Hell
o-World/assignees{/user}",
      "blobs_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/git/blobs{/sha}",
      "branches_url": "http://api.github.com/repos/octocat/Hello
-World/branches{/branch}",
      "clone_url": "https://github.com/octocat/Hello-World.git",
      "collaborators_url": "http://api.github.com/repos/octocat/
Hello-World/collaborators{/collaborator}",
      "comments_url": "http://api.github.com/repos/octocat/Hello
-World/comments{/number}",
      "commits_url": "http://api.github.com/repos/octocat/Hello-
World/commits{/sha}",
      "compare_url": "http://api.github.com/repos/octocat/Hello-
World/compare/{base}...{head}",
      "contents_url": "http://api.github.com/repos/octocat/Hello
-World/contents/{+path}",
      "contributors_url": "http://api.github.com/repos/octocat/H
ello-World/contributors",
      "deployments_url": "http://api.github.com/repos/octocat/He
llo-World/deployments",
      "downloads_url": "http://api.github.com/repos/octocat/Hell
o-World/downloads",
      "events_url": "http://api.github.com/repos/octocat/Hello-W
orld/events",
      "forks_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/forks",
      "git_commits_url": "http://api.github.com/repos/octocat/He
llo-World/git/commits{/sha}",
      "git_refs_url": "http://api.github.com/repos/octocat/Hello
-World/git/refs{/sha}",
      "git_tags_url": "http://api.github.com/repos/octocat/Hello
```

```
-World/git/tags{/sha}",
      "git_url": "git:github.com/octocat/Hello-World.git",
      "hooks_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/hooks",
      "issue_comment_url": "http://api.github.com/repos/octocat/
Hello-World/issues/comments{/number}",
      "issue_events_url": "http://api.github.com/repos/octocat/H
ello-World/issues/events{/number}",
      "issues_url": "http://api.github.com/repos/octocat/Hello-W
orld/issues{/number}",
      "keys_url": "http://api.github.com/repos/octocat/Hello-Wor
ld/keys{/key_id}",
      "labels_url": "http://api.github.com/repos/octocat/Hello-W
orld/labels{/name}",
      "languages_url": "http://api.github.com/repos/octocat/Hell
o-World/languages",
      "merges_url": "http://api.github.com/repos/octocat/Hello-W
orld/merges",
      "milestones_url": "http://api.github.com/repos/octocat/Hel
lo-World/milestones{/number}",
      "mirror_url": "git:git.example.com/octocat/Hello-World",
      "notifications_url": "http://api.github.com/repos/octocat/
Hello-World/notifications{?since, all, participating}",
      "pulls_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/pulls{/number}",
      "releases_url": "http://api.github.com/repos/octocat/Hello
-World/releases{/id}",
      "ssh_url": "git@github.com:octocat/Hello-World.git",
      "stargazers_url": "http://api.github.com/repos/octocat/Hel
lo-World/stargazers",
      "statuses_url": "http://api.github.com/repos/octocat/Hello
-World/statuses/{sha}",
      "subscribers_url": "http://api.github.com/repos/octocat/He
llo-World/subscribers",
      "subscription_url": "http://api.github.com/repos/octocat/H
ello-World/subscription",
      "svn_url": "https://svn.github.com/octocat/Hello-World",
      "tags_url": "http://api.github.com/repos/octocat/Hello-Wor
ld/tags",
      "teams_url": "http://api.github.com/repos/octocat/Hello-Wo
```

```
rld/teams",
      "trees_url": "http://api.github.com/repos/octocat/Hello-Wo
rld/git/trees{/sha}",
      "homepage": "https://github.com",
      "language": null,
      "forks_count": 9,
      "stargazers_count": 80,
      "watchers_count": 80,
      "size": 108,
      "default_branch": "master",
      "open_issues_count": 0,
      "has_issues": true,
      "has_wiki": true,
      "has_pages": false,
      "has_downloads": true,
      "pushed_at": "2011-01-26T19:06:43Z",
      "created_at": "2011-01-26T19:01:12Z",
      "updated_at": "2011-01-26T19:14:43Z",
      "permissions": {
        "admin": false,
        "push": false,
        "pull": true
      }
    }
  }
]
```

GET /user/starred/{owner}/{repo}

检查是否验证用户是否已经star了某个仓库。

请求参数request:身份验证。

Response：

返回码	含义
204	该仓库已经被star
404	未被star

PUT /user/starred/{owner}/{repo}

star某个仓库。

请求参数request：身份验证。

`Header(Content-Length : 0)`

Response：

返回码	含义
204	star成功
404	star失败

DELETE /user/starred/{owner}/{repo}

取消某个仓库的star状态。

请求参数reqeust：身份验证。

Reponse：

返回码	含义
204	取消star成功
404	取消star失败

Events

events表示所有事务。包括push，star，fork，watch，download，create，delete等等，具体事务类型可见Events Types，针对不同event，获取的数据Json结构不同。

GET /events

获取所有公开的events

GET /repos/{owner}/{repo}/events

获取某仓库相关的事务。

GET /repos/{owner}/{repo}/issues/events

获取某仓库issues相关的事务。

GET /networks/{owner}/{repo}/events

获取某仓库网络相关的事务。

GET /orgs/{org}/events

获取某组织的事务。组织相关api的操作，请查阅scope以及登录organization管理页面确认权限问题。

GET /users/{username}/received_events

获取某个用户接收到的事务，`username为登录账户名而不是昵称。`

Watch

Watching一个Repository将会使用户接收到这个Repository相关的讨论以及一些活动。与Star的区别在于，star不会接收到repository的相关通知。

Get \/repos\/{owner}\/{repo}\/subscribers

获取该repository的所有watcher。`注意分页`。

httpcode 200

`Response`