



作者 顾慎为 (/users/ee974365aea2) 2015.09.20 01:09\*

写了35603字，被25人关注，获得了58个喜欢

(/users)

+ 添加关注 (/sign\_in)

# 用python实现模拟登录人人网

字数4068 阅读1647 评论19 喜欢46

我决定从头说起。懂的人可以快速略过前面理论看最后几张图。

## web基础知识

从OSI参考模型（从低到高：物理层，数据链路层，网络层，传输层，会话层，表示层，应用层）来说，我们的互联网属于应用层。从TCP/IP参考模型（从低到高：物理层，数据链路层，网络层，传输层，应用层）来说，也同样如此。

互联网上有各种各样的资源，包括文本、图片、音频、视频……

通常所见的Web模型需要包括两部分：客户端，服务器。个人电脑上的浏览器就是一种客户端，而保存我们输入的网址所有相关资源的服务器。

客户端通过URL（Uniform Resource Locator，统一资源定位符）来链接至服务器。

`http://www.abcd.com/page.html` (`http://www.abcd.com/page.html`)

通常一个URL如上所示，http是协议名（http、ftp、telnet……），www.abcd.com是页面所在机器的DNS名，page.html是页面文件的名字。

内部细节不表，总之我们在浏览器中输入url，浏览器通过url与服务器经过复杂沟通协调建立联系（TCP、UDP……），将数据从服务器发至浏览器，我们能够通过浏览器看到最终拿到的资源。

通常我们看到的web页面，是使用HTML的语言来编写的。上面除了有之前提到的各种资源外，还有超链接。

# Cookie

新的需求出现了：服务器需要能够认识客户端，比如客户需要登录、需要身份识别、需要权限识别、需要依据不同客户加载不同内容.....

怎么办？当客户端向服务器端请求一个web页面时，服务器端除了提供此页面外，还会提供一些附加信息，其中包括cookie，客户端会将cookie存储在客户机器的本地磁盘上。

当浏览器向该服务器端发起请求时，浏览器会检查它的cookie，确定所请求的目标域是否有存cookie，如果有，会将该cookie包含到请求消息（request）中。服务器得到cookie后，就可以进行识别操作了。

一个cookie可以保函至多5个域：Domain、Path、Content、Expires、Secure。

1. Domain：指示cookie来自什么地方，即域名。每个域至多在客户端存储20个cookie。
2. Path：服务器目录结构中的一个路径，表示服务器文件树的哪些部分可能会用到该cookie。通常是/，表示整个文件树。
3. Content：采用"name = value"的形式，即键值对，熟悉python的话可以得知Dictionary使用相同结构。这是cookie内容存放之处。
4. Expires：过期时间。如果此域不存在的话，浏览器在退出时会将cookie丢弃。否则会一直在客户端保存到过期为止。所以如果服务器想将客户端的cookie删除，只需要再将它发送一次，并且选择一个已经过去的时间作为Expires。
5. Secure：指示浏览器只向安全的服务器（https等等）才返回该cookie。

到这里，我们知道了，后续每次连接人人网的时候，都需要一个正确的cookie来表明我们的身份。

## 静态和动态Web文档

按照之前提到的模型，客户端向服务器端发出Request，服务器端返回Response，一次完成，不管传输的是HTML格式、还是XML格式、还是.....这种都算是静态Web文档，过程非常简单。

然而，社会在进步，需求在增加。越来越多的内容需要根据实际情况来生产，并且内容的生成既可以发生在服务器端，也可发生在客户端。

简  
(/)  
  
☐  
(/collections)  
  
☐  
(/apps)

### 1. 服务器端动态Web页面

用户可以在客户端提交表单（Form）给服务器端，服务器端需要根据表单内容来返回正确的Web页面。

有两种处理表单或类似Request的方法。

传统方法是CGI系统，允许服务器与后端程序及脚本通信，通常用Perl或Python编写。简单的模型是：用户，浏览器，服务器，CGI脚本，磁盘上的数据库，以上几个呈链型。

另外一种方法是在HTML页面中嵌入少量脚本，让服务器来执行这些脚本以便生成最终发送给客户的页面。通常用PHP、JSP、ASP等。简单的模型是：用户，浏览器，服务器，而PHP则作为一个模块存在于服务器内。

### 2. 客户端动态Web页面

然而CGI、PHP、JSP、ASP这种在部署在服务器端的脚本并不能够对用户的鼠标移动事件进行响应，或者直接与用户交互。

现在经常会谈到“互联网服务，交互界面要友好”，想要达成这一目的，需要在客户端能够实时依据用户的动作来处理，现在最流行的应该是Javascript了吧。简单的模型依旧是：用户，浏览器，服务器，只不过在浏览器多了一个Javascript的模块。

这两种动态web语言并没有谁优谁劣的区分，只不过用途不同罢了。前者就是我们常说的后端，后者就是我们常说的前端。

## HTTP——超文本传输协议

前面提到，URL的第一个部分是协议名，我们用的比较多的是http协议。

### Request

客户端到服务器的消息被称作请求（Request），需要包括应用于资源的方法、资源的标识符和协议的版本号。

HTTP 的 Request 包括 几 种 访 问 方 法：GET/HEAD/PUT/POST/DELETE/TRACE/CONNECT/OPTIONS。

1. GET方法请求最常用，直接以链接形式访问，即链接中包含了所有参数。速度快，内容容量小，不安全。

A  
➡  
(/sign\_in)



简 (/)  
☐☐☐  
☐  
(/apps)

2. POST方法经常和GET进行比较，它是向服务器发送请求，要求服务器接受位于请求后的实体，依据实体内容处理后返回相关内容。速度慢，内容量大，安全。

☐☐☐ (/collections) However，不再以讹传讹，GET和POST的真正区别 (<http://www.nowamagic.net/librarys/veda/detail/1919>)在这篇博客中，作者查证HTTP协议后说：

在HTTP协议中，Method和Data ( URL , Body , Header ) 是正交的两个概念，也就是说，使用哪个Method与应用层的数据如何传输是没有相互关系的。

HTTP没有要求，如果Method是POST数据就要放在BODY中。也没有要求，如果Method是GET，数据（参数）就一定要放在URL中而不能放在BODY中。

那么，网上流传甚广的这个说法是从何而来的呢？我在HTML标准中，找到了相似的描述 (<http://www.w3.org/TR/REC-html40/interact/forms.html#h-17.13.1>)。这和网上流传的说法一致。但是这只是HTML标准对HTTP协议的用法的约定。怎么能当成GET和POST的区别呢？

之后，该博客还对其他几个区别点进行了驳斥。所以，上面所说的是按照HTML标准，按照HTTP标准的话，GET方法请求服务器发送页面，POST方法则是要写入页面（所以POST方法通常要有Content-Type和认证头，通过认证头来证明有权执行所请求的操作）。

## Response

从服务器返回的消息称作响应（Response），包括HTTP协议的版本号、请求的状态（成功与否，等等）和文档的MIME类型。

状态行包括一个3位数字的状态码，如2xx表示成功，常见200表示请求成功；3xx表示进行了重定向；4xx表示客户错误，常见403是禁止的页面，404是页面未找到；5xx是服务器错误。

## 消息头

HTTP的头域包括通用头，请求头，响应头和实体头四个部分。每个头域由一个域名，冒号（:）和域值三部分组成。

A  
➡  
(/sign\_in)

1. 通用头域（即通用头）  
通用头域包含请求和响应消息都支持的头域。通用头域包含Cache-Control、



简  
(/)



(/collections)



(/apps)

Connection、Date、Pragma、Transfer-Encoding、Upgrade、Via。

## 2. 请求消息（请求头）

请求消息的第一行为下面的格式：

*Method Request-URI HTTP-Version*

Host头域指定请求资源的Internet主机和端口号，必须表示请求url的原始服务器或网关的位置。HTTP/1.1请求必须包含主机头域，否则系统会以400状态码返回。

Accept：告诉WEB服务器自己接受什么介质类型，/表示任何类型，type/\*表示该类型下的所有子类型，type/sub-type。

Accept-Charset：浏览器申明自己接收的字符集。

Authorization：当客户端接收到来自WEB服务器的 WWW-Authenticate 响应时，用该头部来回应自己的身份验证信息给WEB服务器。

User-Agent头域的内容包含发出请求的用户信息。

Referer 头域允许客户端指定请求uri的源资源地址，这可以允许服务器生成回退链表，可用来登陆、优化cache等。如果请求的uri没有自己的uri地址，Referer不能被发送。

## 3. 响应消息（响应头）

响应消息的第一行为下面的格式：

*HTTP-Version Status-Code Reason-Phrase*

HTTP-Version表示支持的HTTP版本，例如为HTTP/1.1。

Status-Code是一个三个数字的结果代码。

Reason-Phrase给Status-Code提供一个简单的文本描述。

## 4. 实体消息（实体头和实体）

请求消息和响应消息都可以包含实体信息，实体信息一般由实体头域和实体组成。

# 整理思路，开始动手

1. 我的目的：到人人网将我的各种信息都download到本地。
2. 遇到问题：人人网需要用户登录。
3. 解决办法：只需要获得服务器返回的cookie，以后每次访问页面时将此cookie一并发送至服务器就可以了。
4. 遇到问题：我不会手工构建cookie，怎么获取。
5. 解决办法：先手工模拟浏览器登录人人网，获取第一个cookie。
6. 遇到问题：怎么模拟登录。
7. 解决办法：向登录页面发送http登录请求。

A



(/sign\_in)



简  
(/)

☰

(/collections

8. 遇到问题：登录页面的具体URL是什么；http登录请求需要包含哪些信息。

解决这两个问题后，这篇博客就圆满了。

📱

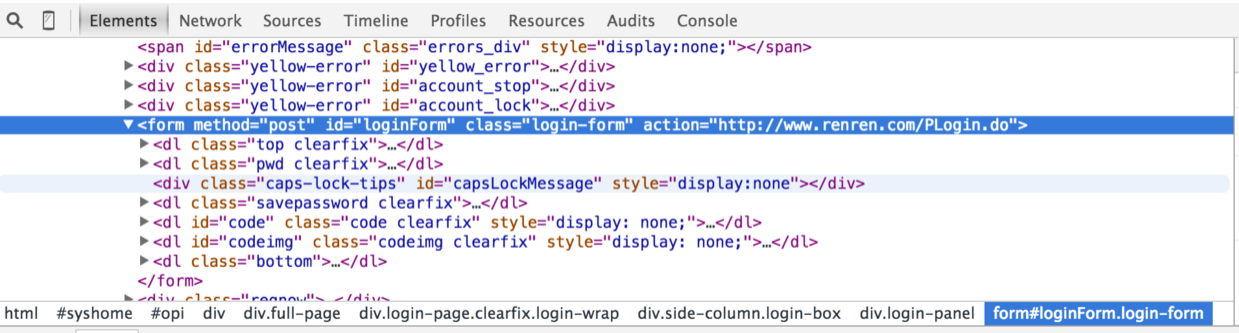
(/apps)

## 登录URL

需要进行抓包，我选用Chrome，因为我的Mac上只有这个。

其实在抓包之前我先看了下网页的编码，如图，打开chrome的开发者工具（option+command+i），点击Elements，然后点击左上方的放大镜图标，再去网页页面上点击“登录”按钮，发现了图中的结构体。

我选中的那一行已经明确表示了，表单方法是“POST”，猜测action属性就是提交表单的对象，也就是登录URL。然而到底是不是呢？这只能算个猜测，具体是什么URL，用抓包去得到的结果比较有说服力。



登录地址猜测.png

点击Network，将左上方的圆圈点击变成红色，这样chrome就会将打开网页的过程抓取下来。

然而这么做我并没有找到登录时的POST Request，我想了好久找了好多参考资料都没写是咋回事。但是当我使用win7下的chrome，以及IE时，均找到了对应的Request。

这是咋回事？经过两者的对比，我发现在Mac下，chrome抓包结果少了很多，尤其是当我在登录界面输入用户名，并切换焦点时，会生成一个“ShowCaptcha”的 Post Request（图中有），结果点击“登录”后就消失不见了。我猜测是由于在登录的过程中页面进行了跳转，使得之前抓取的信息都被覆盖掉了。仔细寻找发现了“Preserve log”，鼠标停在按钮上的解释是“Do not clear log on page reload/navigation”，选取后才一切正常了，结果如图。

A

➡

(/sign\_in)

个人判断，此按钮在mac下才在界面上显示，win下是默认勾选的。

^

☰

✎

简  
(/)

⌘

(/collections)

📱

(/apps)

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline - S
focus?j={%22ID%22:null,%22R%22:... dj.renren.com	GET	200 OK	webp	Other	169 B 0 B	43 ms 41 ms	
click?j={%22ID%22:null,%22R%22:... dj.renren.com	GET	200 OK	webp	Other	208 B 33 B	46 ms 44 ms	
ShowCaptcha /ajax	POST	200 OK	xhr	base-all2.js:7 Script	308 B 1 B	38 ms 37 ms	
login?1=1&uniqueTimestamp=20158113687 /ajaxLogin	POST	200 OK	xhr	base-all2.js:4 Script	1.3 KB 52 B	57 ms 55 ms	
home	GET	302 Found		login.js:1 Script	250 B 0 B	36 ms 36 ms	
284597150	GET	200 OK	document	http://www.renre...	18.1 KB	191 ms	

屏幕快照 2015-09-21 下午11.48.43.png

找找为数不多的几个POST Request，不难发现就是图中的倒数第三个。点击可以看到它的头部信息。

获得信息，Request URL：http://www.renren.com/ajaxLogin/login?1=1&uniqueTimestamp=20158113687 (http://www.renren.com/ajaxLogin/login?1=1&uniqueTimestamp=20158113687)

url最后写uniqueTimestamp，说明它是有时间戳的，过段时间此数值就不一样了。

在该界面的最下方可以找到Post Request的Form Data，由于可能泄露隐私，就不截图了。观察可发现其中有两个关键参数：email，是我的登录邮箱；password，是经过处理后的密码。还有其他一些参数，尽量还原，但是够用就好。

Name Path	Headers	Preview	Response	Cookies	Timing
focus?j={%22ID%22:n... dj.renren.com					
click?j={%22ID%22:nul... dj.renren.com					
ShowCaptcha /ajax					
login?1=1&uniqueTim... /ajaxLogin	<b>General</b> Remote Address: 220.181.181.237:80 Request URL: http://www.renren.com/ajaxLogin/login?1=1&uniqueTimestamp=20158113687 Request Method: POST Status Code: 200 OK <b>Response Headers</b> Cache-Control: no-cache Connection: keep-alive Content-Encoding: gzip Content-Type: text/html; charset=UTF-8 Date: Sun, 20 Sep 2015 17:41:35 GMT Expires: Thu, 01 Jan 1970 00:00:00 GMT Pragma: no-cache Server: nginx/1.2.0 Set-Cookie: societyquester=f00738c51b51d822cb471b0c19d65ad20; domain=.renren.com; path=/				
home					
284597150					

屏幕快照 2015-09-22 上午12.04.22.png

A

➔

(/sign\_in)

http登录请求包

⬆

⌘

✎

简  
(/)



(/collections)这也算是最简单的一种了吧。



(/apps)

实际上如何构造http登录请求包是个复杂的过程，若登录的是taobao这类安全要求较高的网站，需要构造很多校验内容。人人网倒是非常简单，直接POST Request即可，并且发现人人并没有校验HTTP头部。

并且请求包很灵活，每个网站的要求还不太一样，所以如果以后需要解析其他请求包时，再单独分析吧。希望不要太复杂。

首先声明一个 cookiejar 的对象来保存 cookie，然后使用 urllib2 的 HTTPCookieProcessor来创建cookie处理器，通过此处理器来构建 opener，并将此 opener设置为urllib2的默认 opener。

从此之后，只要使用此 opener，就可以使用当前的 cookie 了。也可以直接使用 self.opener，效果是一样的，只不过这里是显式调用，之前的install之后使用urllib2看不到具体的 opener 罢了。在下面的第二章截图中，两种方式均有实例。

```
'''
def __init__(self, username = '', password = '', path = ''):
    self.username = username
    self.password = password
    self.path = path
    self.cookie = cookielib.LWPCookieJar()
    self.opener = urllib2.build_opener( urllib2.HTTPCookieProcessor(self.cookie) )
    urllib2.install_opener(self.opener)
```

屏幕快照 2015-09-22 上午12:55:44.png

这里只是简单的构造一个Request。可以看到我试用了两个loginURL，两者最后都返回了cookie。

并且我分别尝试用这两个 cookie 去打开 http://www.renren.com (http://www.renren.com)，发现都成功进入了我的账户！

A



(/sign\_in)





简  
(/)

☰

(/collections)

📱

(/apps)

```
3
4 def login(self):
5     data = {
6         'email' : self.username,
7         'password' : self.password
8     }
9     loginUrl = 'http://www.renren.com/ajaxLogin/login?1=1&uniqueTimestamp=201582021961'
10    # loginUrl = 'http://www.renren.com/PLogin.do'
11    request = urllib2.Request(loginUrl, urllib.urlencode(data))
12    try:
13        response = self.opener.open(request)
14        url = response.geturl()
15        print url
16    # for item in self.cookie:
17    #     print 'Name :', item.name
18    #     print 'Value :', item.value
19    url = urllib2.urlopen('http://www.renren.com').geturl()
20    print url
21    if re.match('http://www.renren.com/[\\d]{9}', url):
22        print 'Login Successfully'
23        return True
24    else:
25        print 'Account/Password is not correct.'
26        return False
27    except Exception,e:
28        print 'Fail to login:', e.message
29        return False
30
```

屏幕快照 2015-09-22 上午1.23.29.png

最后提出一个问题：登录URL的时间戳是咋回事？

目前测试前天的还可以用。分析一下，首先这个URL是自动生成的，然后和登录相关，猜测是在前端编写的。

很快就找到了疑似的js脚本。

```
Q 📱 Elements Network Sources Timeline Profiles Resources Audits Console
<!DOCTYPE html>
<html class="nx-main980">
  <head>...</head>
  <body id="syshome" class="login">
    <iframe src="http://login.renren.com/ajaxproxy.htm" style="display: none;">...</iframe>
    <div id="header">...</div>
    <div id="opi" class="page-wrapper clearfix">...</div>
    <div class="ft-wrapper clearfix" style="width: 980px;">...</div>
    <script src="http://s.xnimg.cn/a79117/nx/apps/login/login.js" type="text/javascript"></script>
    <div id="jebe_ad_prepare" style="height: 0px; overflow: hidden;"></div>
  </body>
</html>
```

屏幕快照 2015-09-23 上午12.03.30.png

打开，搜“uniqueTimestam”。

A

➡

(/sign\_in)

^

☰

✎



屏幕快照 2015-09-23 上午12.04.36.png

像我一样不懂JS的人可以百度下“js Date”，查看函数说明。按照解释，我截图中“201582021961”的意思依次是：2015年，9月，星期三，0点，21秒，961毫秒（居然没有Minutes！）

也可以直接在Chrome的开发者工具里面点击“Console”，直接输入截图中的命令就可以获得当前的取值。

同时，我们从这段js代码中，又一次确认了登录URL，整个登录也的确是一个简单的Request。其中的变量c存储了POST方法要发送的data（我猜的，毕竟我不懂js）。但是怎么判断这个时间戳是有效的我没看出来，我把代码中的时间戳改成2014年也依然有效。总之，如果之后登录失败了，我们可以选择重新生成一个新的时间戳。

另附带几个chrome开发工具使用说明：

Network界面左上方的禁止图标，点击后可以清楚当前抓取的所有信息。

有时需要清除浏览器的Cache和Cookie，此时在Network右键即可看到选项。

## 参考书目

《计算机网络（第4版）》 Andrew S. Tanenbaum著

简 (/)



(/collections)



(/apps)


如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

¥ 打赏支持

喜欢 | 46

分享到微博 分享到微信 更多分享

19条评论 ( 按时间正序 · 按时间倒序 · 按喜欢排序 ) 添加新评论 (/sign\_in)

 rbe (/users/57f3ef7bd5be)  
(/users/57f3ef7bd5be) 2015.09.22 08:51 (/p/5c9855ca6474/comments/638013#comment-638013)

如果已经登陆过网站，计算机中已经存储了cookie呢，抓cookie还能生效吗


喜欢(0) 回复

顾慎为 (/users/ee974365aea2) : @rbe (/users/57f3ef7bd5be) 可以生效，程序中用urllib2发起request时并没有用到本机已有的cookie。

其实如果本机有cookie，可以直接用LMcookieJar来读取该文件。  
2015.09.22 12:34 (/p/5c9855ca6474/comments/638607#comment-638607) 回复

顾慎为 (/users/ee974365aea2) : @rbe (/users/57f3ef7bd5be) 说错了，是LMPCookieJar  
2015.09.22 12:36 (/p/5c9855ca6474/comments/638621#comment-638621) 回复

添加新回复

 理发客 (/users/0bb0a1416a68)  
(/users/0bb0a1416a68) 2015.09.22 09:59 (/p/5c9855ca6474/comments/638243#comment-638243)

简单的事情，搞的这么麻烦

喜欢(0) 回复

顾慎为 (/users/ee974365aea2) : @理发客 (/users/0bb0a1416a68) 前面理论铺垫多了点  
2015.09.22 12:44 (/p/5c9855ca6474/comments/638643#comment-638643) 回复

A



(/sign\_in)



简 (/)



(/collections)



(/apps)

添加新回复



sparkchen (/users/7f8e0a64993c)

5楼 2015.09.22 23:19 (/p/5c9855ca6474/comments/640575#comment-640575)

一个request就搞定了。。

喜欢(0)

回复

顾慎为 (/users/ee974365aea2) : @sparkchen (/users/7f8e0a64993c) 人人的登录的确简单。

2015.09.22 23:31 (/p/5c9855ca6474/comments/640609#comment-640609)

回复

添加新回复



D\_shine (/users/ecd5eb1286bf)

5楼 2015.09.22 23:54 (/p/5c9855ca6474/comments/640695#comment-640695)

最近正好在学这个，正好复习一下。说的挺清楚的

喜欢(0)

回复

顾慎为 (/users/ee974365aea2) : @D\_shine (/users/ecd5eb1286bf) 最后更新了下时间戳的解释。欢迎再看。

2015.09.23 00:21 (/p/5c9855ca6474/comments/640779#comment-640779)

回复

添加新回复



风过无痕3 (/users/575d6d0ecd73)

6楼 2015.09.23 06:46 (/p/5c9855ca6474/comments/640924#comment-640924)

赞

喜欢(0)

回复

顾慎为 (/users/ee974365aea2) : @风过无痕3 (/users/575d6d0ecd73) 谢谢

2015.09.23 08:13 (/p/5c9855ca6474/comments/641019#comment-641019)

回复

风过无痕3 (/users/575d6d0ecd73) : @顾慎为 (/users/ee974365aea2) 握手

2015.09.23 10:30 (/p/5c9855ca6474/comments/641449#comment-641449)

回复



(/sign\_in)



简  
(/)

添加新回复



(/collections)



生姜头 (/users/4f6773840e51)

(/users/4f6773840e51)

7楼 2015-09-25 00:08 (/p/5c9855ca6474/comments/648593#comment-648593)



(/apps)

hao dongxi

喜欢(0)

回复

顾慎为 (/users/ee974365aea2) : @生姜头 (/users/4f6773840e51) 多谢 !

2015.09.25 09:05 (/p/5c9855ca6474/comments/649176#comment-649176)

回复

添加新回复



生姜头 (/users/4f6773840e51)

(/users/4f6773840e51)

7楼 2015-09-25 00:08 (/p/5c9855ca6474/comments/648594#comment-648594)

hao dongxi

喜欢(0)

回复



晓龙酱 (/users/62f51c390c19)

(/users/62f51c390c19)

8楼 2015-09-25 13:15 (/p/5c9855ca6474/comments/650187#comment-650187)

专业 !

喜欢(1)

回复

顾慎为 (/users/ee974365aea2) : @晓龙酱 (/users/62f51c390c19) 过奖啊

2015.09.25 17:44 (/p/5c9855ca6474/comments/651084#comment-651084)

回复

添加新回复



浓眉大眼小胖子 (/users/ed93d8efa5b6)

(/users/ed93d8efa5b6)

10楼 2015-10-07 21:46 (/p/5c9855ca6474/comments/701472#comment-701472)

请教一下,我在qq登陆时抓取包,只有get,那我模拟登陆的时候,不能提交表单,应该靠的是cookies吧,那怎么构建呢

喜欢(0)

回复

A



(/sign\_in)



简 (/)



(/collections)



(/apps)

顾慎为 (/users/ee974365aea2)：@浓眉大眼小胖子 (/users/ed93d8efa5b6) cookie肯定是服务器返回的，获取cookie需要模拟登录的流程。具体怎么模拟你可以搜一下，我粗略看了下基于webQQ的模拟的，会涉及到加密，验证码等等，比较复杂。

2015.10.11 23:34 (/p/5c9855ca6474/comments/722958#comment-722958)

回复

添加新回复

登录后发表评论 (/sign\_in)

被以下专题收入，发现更多相似内容：



**程序员** (/collection/NEt52a )

简书程序员大本营 投稿须知： 1.本专题仅收录与程序有关的文章。 2.请在代码框里写代码，尽量保证可看性。

5701篇文章 (/collection/NEt52a) · 53513人关注



添加关注 (/sign\_in)



**@IT** (/collection/V2CqjW )

简书上的IT大社群。 管理员由 简书@IT社群 不定期轮换当班。 入选本专题的文章，会通过首页投稿审核，则会进入简书首页 ...

3040篇文章 (/collection/V2CqjW) · 16413人关注



添加关注 (/sign\_in)



**代码改变世界** (/collection/Of5e015fc36c )

Write the Code, Change the World, 改变世界从编程开始, 收集编程相关的干货

1323篇文章 (/collection/Of5e015fc36c) · 8732人关注



添加关注 (/sign\_in)

A



(/sign\_in)

