宗述

首先来看一下他们的区别

urllib和urllib2

urllib 和urllib2都是接受URL请求的相关模块，但是urllib2可以接受一个**Request**类的实例来设置URL请求的headers，urllib仅可以接受URL。

这意味着，你不可以伪装你的User Agent字符串等。

urllib提供**urlencode**方法用来GET查询字符串的产生，而urllib2没有。这是为何urllib常和urllib2一起使用的原因。

目前的大部分http请求都是通过urllib2来访问的

httplib

httplib实现了HTTP和HTTPS的客户端协议，一般不直接使用，在[python](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=c271eaa62c280a3f&k=python&k0=python&kdi0=0&luki=10&n=10&p=baidu&q=72008038_cpr&rb=0&rs=1&seller_id=1&sid=3f0a282ca6ea71c2&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1893960&u=http%3A%2F%2Fwww%2Echengxuyuans%2Ecom%2FPython%2F47674%2Ehtml&urlid=0)更高层的封装模块中（urllib,urllib2）使用了它的http实现。

urllib简单用法

urllib.urlopen(url[, data[, proxies]]) :

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. google = urllib.urlopen('http://www.google.com')
2. **print** 'http header:/n', google.info()
3. **print** 'http status:', google.getcode()
4. **print** 'url:', google.geturl()
5. **for** line **in** google: # 就像在操作本地文件
6. **print** line,
7. google.close()

详细使用方法见

[urllib学习](http://blog.csdn.net/dolphin_h/article/details/23618187)

urllib2简单用法

最简单的形式

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. **import** urllib2
2. response=urllib2.urlopen('http://www.douban.com')
3. html=response.read()

实际步骤：

1、urllib2.Request()的功能是构造一个请求信息，返回的req就是一个构造好的请求

2、urllib2.urlopen()的功能是发送刚刚构造好的请求req，并返回一个文件类的对象response，包括了所有的返回信息。

3、通过response.read()可以读取到response里面的html，通过response.info()可以读到一些额外的信息。

如下：

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. #!/usr/bin/env python
2. **import** urllib2
3. req = urllib2.Request("http://www.douban.com")
4. response = urllib2.urlopen(req)
5. html = response.read()
6. **print** html

有时你会碰到，程序也对，但是服务器拒绝你的访问。这是为什么呢?问题出在请求中的头信息(header)。 有的服务端有洁癖，不喜欢程序来触摸它。这个时候你需要将你的程序伪装成浏览器来发出请求。请求的方式就包含在header中。  
常见的情形：

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. **import** urllib
2. **import** urllib2
3. url = 'http://www.someserver.com/cgi-bin/register.cgi'
4. user\_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'# 将user\_agent写入头信息
5. values = {'name' : 'who','password':'123456'}
6. headers = { 'User-Agent' : user\_agent }
7. data = urllib.urlencode(values)
8. req = urllib2.Request(url, data, headers)
9. response = urllib2.urlopen(req)
10. the\_page = response.read()

values是post数据

GET方法

例如百度：

百度是通过<http://www.baidu.com/s?wd=XXX> 来进行查询的，这样我们需要将{‘wd’:’xxx’}这个字典进行urlencode

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. #coding:utf-8
2. **import** urllib
3. **import** urllib2
4. url = 'http://www.baidu.com/s'
5. values = {'wd':'D\_in'}
6. data = urllib.urlencode(values)
7. **print** data
8. url2 = url+'?'+data
9. response = urllib2.urlopen(url2)
10. the\_page = response.read()
11. **print** the\_page

POST方法

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. **import** urllib
2. **import** urllib2
3. url = 'http://www.someserver.com/cgi-bin/register.cgi'
4. user\_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)' //将user\_agent写入头信息
5. values = {'name' : 'who','password':'123456'}      //post数据
6. headers = { 'User-Agent' : user\_agent }
7. data = urllib.urlencode(values)                   //对post数据进行url编码
8. req = urllib2.Request(url, data, headers)
9. response = urllib2.urlopen(req)
10. the\_page = response.read()

urllib2带cookie的使用

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. #coding:utf-8
2. **import** urllib2,urllib
3. **import** cookielib
5. url = r'http://www.renren.com/ajaxLogin'
7. #创建一个cj的cookie的容器
8. cj = cookielib.CookieJar()
9. opener = urllib2.build\_opener(urllib2.HTTPCookieProcessor(cj))
10. #将要POST出去的数据进行编码
11. data = urllib.urlencode({"email":email,"password":**pass**})
12. r = opener.open(url,data)
13. **print** cj

httplib简单用法

简单示例

**[python]** [view plain](http://blog.csdn.net/dolphin_h/article/details/45296353) [copy](http://blog.csdn.net/dolphin_h/article/details/45296353)

[在CODE上查看代码片](https://code.csdn.net/snippets/653700)

1. #!/usr/bin/env python
2. # -\*- coding: utf-8 -\*-
3. **import** httplib
4. **import** urllib
6. **def** sendhttp():
7. data = urllib.urlencode({'@number': 12524, '@type': 'issue', '@action': 'show'})
8. headers = {"Content-type": "application/x-www-form-urlencoded",
9. "Accept": "text/plain"}
10. conn = httplib.HTTPConnection('bugs.python.org')
11. conn.request('POST', '/', data, headers)
12. httpres = conn.getresponse()
13. **print** httpres.status
14. **print** httpres.reason
15. **print** httpres.read()
17. **if** \_\_name\_\_ == '\_\_main\_\_':
18. sendhttp()