版本号：Python2.7.5，Python3改动较大，各位另寻教程。

所谓网页抓取，就是把URL地址中指定的网络资源从网络流中读取出来，保存到本地。   
类似于使用程序模拟IE浏览器的功能，把URL作为HTTP请求的内容发送到服务器端， 然后读取服务器端的响应资源。

在[**Python**](http://lib.csdn.net/base/python)中，我们使用urllib2这个组件来抓取网页。  
urllib2是[**python**](http://lib.csdn.net/base/python)的一个获取URLs(Uniform Resource Locators)的组件。

它以urlopen函数的形式提供了一个非常简单的接口。

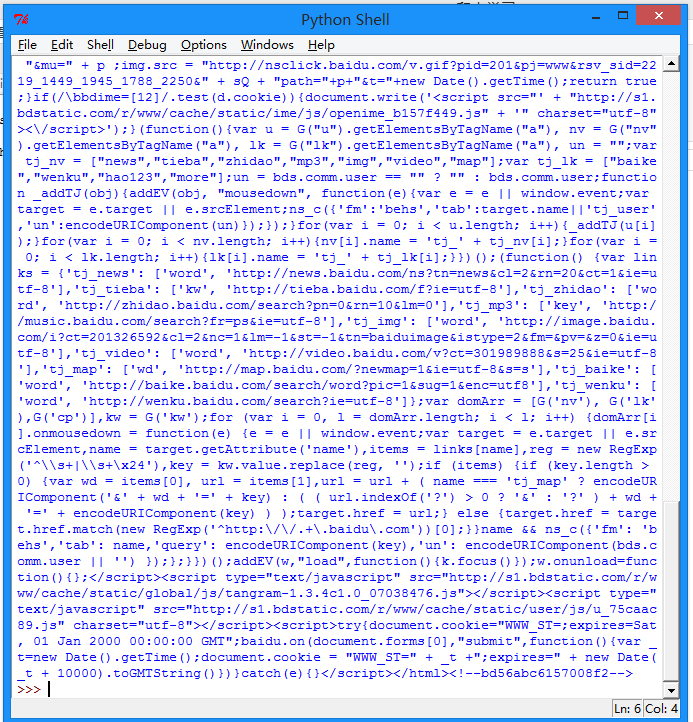
最简单的urllib2的应用代码只需要四行。

我们新建一个文件urllib2\_test01.py来感受一下urllib2的作用：

**[python]** [view plain](http://blog.csdn.net/pleasecallmewhy/article/details/8923067) [copy](http://blog.csdn.net/pleasecallmewhy/article/details/8923067)

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

按下F5可以看到运行的结果：



我们可以打开百度主页，右击，选择查看源代码（火狐OR谷歌浏览器均可），会发现也是完全一样的内容。

也就是说，上面这四行代码将我们访问百度时浏览器收到的代码们全部打印了出来。

这就是一个最简单的urllib2的例子。

除了"http:"，URL同样可以使用"ftp:"，"file:"等等来替代。

HTTP是基于请求和应答机制的：

客户端提出请求，服务端提供应答。

urllib2用一个Request对象来映射你提出的HTTP请求。

在它最简单的使用形式中你将用你要请求的地址创建一个Request对象，

通过调用urlopen并传入Request对象，将返回一个相关请求response对象，

这个应答对象如同一个文件对象，所以你可以在Response中调用.read()。

我们新建一个文件urllib2\_test02.py来感受一下：

**[python]** [view plain](http://blog.csdn.net/pleasecallmewhy/article/details/8923067) [copy](http://blog.csdn.net/pleasecallmewhy/article/details/8923067)

1. **import** urllib2
2. req = urllib2.Request('http://www.baidu.com')
3. response = urllib2.urlopen(req)
4. the\_page = response.read()
5. **print** the\_page

可以看到输出的内容和test01是一样的。

urllib2使用相同的接口处理所有的URL头。例如你可以像下面那样创建一个ftp请求。

**[python]** [view plain](http://blog.csdn.net/pleasecallmewhy/article/details/8923067) [copy](http://blog.csdn.net/pleasecallmewhy/article/details/8923067)

1. req = urllib2.Request('ftp://example.com/')

在HTTP请求时，允许你做额外的两件事。

**1.发送data表单数据**

这个内容相信做过Web端的都不会陌生，

有时候你希望发送一些数据到URL(通常URL与CGI[通用网关接口]脚本，或其他WEB应用程序挂接)。

在HTTP中,这个经常使用熟知的POST请求发送。

这个通常在你提交一个HTML表单时由你的浏览器来做。

并不是所有的POSTs都来源于表单，你能够使用POST提交任意的数据到你自己的程序。

一般的HTML表单，data需要编码成标准形式。然后做为data参数传到Request对象。

编码工作使用urllib的函数而非urllib2。

我们新建一个文件urllib2\_test03.py来感受一下：

**[python]** [view plain](http://blog.csdn.net/pleasecallmewhy/article/details/8923067) [copy](http://blog.csdn.net/pleasecallmewhy/article/details/8923067)

1. **import** urllib
2. **import** urllib2
4. url = 'http://www.someserver.com/register.cgi'
6. values = {'name' : 'WHY',
7. 'location' : 'SDU',
8. 'language' : 'Python' }
10. data = urllib.urlencode(values) # 编码工作
11. req = urllib2.Request(url, data)  # 发送请求同时传data表单
12. response = urllib2.urlopen(req)  #接受反馈的信息
13. the\_page = response.read()  #读取反馈的内容

如果没有传送data参数，urllib2使用GET方式的请求。

GET和POST请求的不同之处是POST请求通常有"副作用"，

它们会由于某种途径改变系统状态(例如提交成堆垃圾到你的门口)。

Data同样可以通过在Get请求的URL本身上面编码来传送。

**[python]** [view plain](http://blog.csdn.net/pleasecallmewhy/article/details/8923067) [copy](http://blog.csdn.net/pleasecallmewhy/article/details/8923067)

1. **import** urllib2
2. **import** urllib
4. data = {}
6. data['name'] = 'WHY'
7. data['location'] = 'SDU'
8. data['language'] = 'Python'
10. url\_values = urllib.urlencode(data)
11. **print** url\_values
13. name=Somebody+Here&language=Python&location=Northampton
14. url = 'http://www.example.com/example.cgi'
15. full\_url = url + '?' + url\_values
17. data = urllib2.open(full\_url)

这样就实现了Data数据的Get传送。

**2.设置Headers到http请求**

有一些站点不喜欢被程序（非人为访问）访问，或者发送不同版本的内容到不同的浏览器。

默认的urllib2把自己作为“Python-urllib/x.y”(x和y是Python主版本和次版本号,例如Python-urllib/2.7)，  
  
这个身份可能会让站点迷惑，或者干脆不工作。

浏览器确认自己身份是通过User-Agent头，当你创建了一个请求对象，你可以给他一个包含头数据的字典。

下面的例子发送跟上面一样的内容，但把自身模拟成Internet Explorer。

（多谢大家的提醒，现在这个Demo已经不可用了，不过原理还是那样的）。

**[python]** [view plain](http://blog.csdn.net/pleasecallmewhy/article/details/8923067) [copy](http://blog.csdn.net/pleasecallmewhy/article/details/8923067)

1. **import** urllib
2. **import** urllib2
4. url = 'http://www.someserver.com/cgi-bin/register.cgi'
6. user\_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'
7. values = {'name' : 'WHY',
8. 'location' : 'SDU',
9. 'language' : 'Python' }
11. headers = { 'User-Agent' : user\_agent }
12. data = urllib.urlencode(values)
13. req = urllib2.Request(url, data, headers)
14. response = urllib2.urlopen(req)
15. the\_page = response.read()

顶