urllib模块提供的上层接口，使我们可以像读取本地文件一样读取www和ftp上的数据。每当使用这个模块的时候，老是会想起公司产品的客户端，同事用C++下载Web上的图片，那种“痛苦”的表情。我以前翻译过libcurl教程，这是在C/C++环境下比较方便实用的网络操作库，相比起libcurl，Python的urllib模块的使用门槛则低多了。可能有些人又会用效率来批评Python，其实在操作网络，或者在集群交互的时候， 语言的执行效率绝不是瓶颈。这种情况下，一个比较好的方法是，将python嵌入到C/C++中，让Python来完成一些不是核心的逻辑处理。又扯远了，废话少说，开始urllib之旅吧~~ （前几天我用这个模块写了个蜘蛛，感兴趣的同学可以在以前的博客中找到代码）

先看一个例子，这个例子把Google首页的html抓取下来并显示在控制台上：

Python

|  |  |
| --- | --- |
| 1  2  3 | # 别惊讶，整个程序确实只用了两行代码  import urllib  print urllib.urlopen(&#039;http://www.google.com&#039;).read() |

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

创建一个表示远程url的类文件对象，然后像本地文件一样操作这个类文件对象来获取远程数据。参数url表示远程数据的路径，一般是网址；参数data表示以post方式提交到url的数据(玩过web的人应该知道提交数据的两种方式：post与get。如果你不清楚，也不必太在意，一般情况下很少用到这个参数)；参数proxies用于设置代理（这里不详细讲怎么使用代理，感兴趣的看客可以去翻阅Python手册urllib模块）。urlopen返回 一个类文件对象，他提供了如下方法：

* read() , readline() , readlines() , fileno() , close() ：这些方法的使用方式与文件对象完全一样;
* info()：返回一个httplib.HTTPMessage 对象，表示远程服务器返回的头信息；
* getcode()：返回Http状态码。如果是http请求，200表示请求成功完成;404表示网址未找到；
* geturl()：返回请求的url；

下面来扩充一下上面的例子，看官可以运行一下这个例子，加深对urllib的印象：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | google = urllib.urlopen(&#039;http://www.google.com&#039;)  print &#039;http header:/n&#039;, google.info()  print &#039;http status:&#039;, google.getcode()  print &#039;url:&#039;, google.geturl()  for line in google: # 就像在操作本地文件      print line,  google.close() |

**urllib.urlretrieve(url[, filename[, reporthook[, data]]])：**

urlretrieve方法直接将远程数据下载到本地。参数filename指定了保存到本地的路径（如果未指定该参数，urllib会生成一个临时文件来保存数据）；参数reporthook是一个回调函数，当连接上服务器、以及相应的数据块传输完毕的时候会触发该回调。我们可以利用这个回调函 数来显示当前的下载进度，下面的例子会展示。参数data指post到服务器的数据。该方法返回一个包含两个元素的元组(filename, headers)，filename表示保存到本地的路径，header表示服务器的响应头。下面通过例子来演示一下这个方法的使用，这个例子将新浪首页的html抓取到本地，保存在D:/sina.html文件中，同时显示下载的进度。

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | def cbk(a, b, c):      &#039;&#039;&#039;回调函数      @a: 已经下载的数据块      @b: 数据块的大小      @c: 远程文件的大小      &#039;&#039;&#039;      per = 100.0 \* a \* b / c      if per &gt; 100:          per = 100      print &#039;%.2f%%&#039; % per    url = &#039;http://www.sina.com.cn&#039;  local = &#039;d://sina.html&#039;  urllib.urlretrieve(url, local, cbk) |

上面介绍的两个方法是urllib中最常用的方法，这些方法在获取远程数据的时候，内部会使用URLopener或者 FancyURLOpener类。作为urllib的使用者，我们很少会用到这两个类，这里我不想多讲。如果对urllib的实现感兴趣， 或者希望urllib支持更多的协议，可以研究这两个类。在Python手册中，urllib的作者还列出了这个模块的缺陷和不足，感兴趣的同学可以打开 Python手册了解一下。

urllib中还提供了一些辅助方法，用于对url进行编码、解码。url中是不能出现一些特殊的符号的，有些符号有特殊的用途。我们知道以get方式提交数据的时候，会在url中添加key=value这样的字符串，所以在value中是不允许有’=’，因此要对其进行编码；与此同时服务器接收到这些参数的时候，要进行解码，还原成原始的数据。这个时候，这些辅助方法会很有用：

* urllib.quote(string[, safe])：对字符串进行编码。参数safe指定了不需要编码的字符;
* urllib.unquote(string) ：对字符串进行解码；
* urllib.quote\_plus(*string* [ , *safe* ] ) ：与urllib.quote类似，但这个方法用’+’来替换’ ‘，而quote用’%20’来代替’ ‘
* urllib.unquote\_plus(*string* ) ：对字符串进行解码；
* urllib.urlencode(query[, doseq])：将dict或者包含两个元素的元组列表转换成url参数。例如 字典{‘name’: ‘dark-bull’, ‘age’: 200}将被转换为”name=dark-bull&age=200″
* urllib.pathname2url(path)：将本地路径转换成url路径；
* urllib.url2pathname(path)：将url路径转换成本地路径；

用一个例子来体验一下这些方法吧~~：

Python

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | data = &#039;name = ~a+3&#039;    data1 = urllib.quote(data)  print data1 # result: name%20%3D%20%7Ea%2B3  print urllib.unquote(data1) # result: name = ~a+3    data2 = urllib.quote\_plus(data)  print data2 # result: name+%3D+%7Ea%2B3  print urllib.unquote\_plus(data2)    # result: name = ~a+3    data3 = urllib.urlencode({ &#039;name&#039;: &#039;dark-bull&#039;, &#039;age&#039;: 200 })  print data3 # result: age=200&amp;name=dark-bull    data4 = urllib.pathname2url(r&#039;d:/a/b/c/23.php&#039;)  print data4 # result: ///D|/a/b/c/23.php  print urllib.url2pathname(data4)    # result: D:/a/b/c/23.php |

urllib模块的基本使用，就这么简单。