写在前面url的格式

protocol :// hostname[:port] / path / [;parameters][?query]#fragment

protocol（协议）

hostname（主机名）

是指存放资源的服务器的域名系统(DNS) 主机名或 IP 地址。有时，在主机名前也可以包含连接到服务器所需的用户名和密码（格式：username:password@hostname）。

port（端口号）

整数，可选，省略时使用方案的默认端口，各种传输协议都有默认的端口号，如http的默认端口为80。如果输入时省略，则使用默认端口号。有时候出于安全或其他考虑，可以在服务器上对端口进行重定义，即采用非标准端口号，此时，URL中就不能省略端口号这一项。

path（路径）

由零或多个“/”符号隔开的字符串，一般用来表示主机上的一个目录或文件地址。

parameters（参数）

这是用于指定特殊参数的可选项。

query(查询)

可选，用于给动态网页（如使用CGI、ISAPI、PHP/JSP/ASP/ASP。NET等技术制作的网页）传递参数，可有多个参数，用“&”符号隔开，每个参数的名和值用“=”符号隔开。

fragment（信息片断）

字符串，用于指定网络资源中的片断。例如一个网页中有多个名词解释，可使用fragment直接定位到某一名词解释。

# urllib模块中的方法

## urllib.request

### 1.urllib.request.urlopen(url[,data[,proxies]])

打开一个url的方法，返回一个文件对象，然后可以进行类似文件对象的操作。本例试着打开google

>>> import urllib

>>> f = urllib. request.urlopen('http://www.google.com.hk/')

>>> firstLine = f.readline() #读取html页面的第一行

>>> firstLine

'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"><head><meta content="/images/google\_favicon\_128.png" itemprop="image"><title>Google</title><script>(function(){\n'

urlopen返回对象提供方法：

- 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请求成功完成;404网址未找到

- geturl()：返回请求的url

### 2.urllib. request.urlretrieve(url[,filename[,reporthook[,data]]])

urlretrieve方法将url定位到的html文件下载到你本地的硬盘中。如果不指定filename，则会存为临时文件。

urlretrieve()返回一个二元组(filename,mine\_hdrs)

临时存放：

>>> filename = urllib. request.urlretrieve('http://www.google.com.hk/')

>>> type(filename)

<type 'tuple'>

>>> filename[0]

'/tmp/tmp8eVLjq'

>>> filename[1]

<httplib.HTTPMessage instance at 0xb6a363ec>

存为本地文件:

>>> filename = urllib. request.urlretrieve('http://www.google.com.hk/',filename='/home/dzhwen/python文件/Homework/urllib/google.html')

>>> type(filename)

<type 'tuple'>

>>> filename[0]

'/home/dzhwen/python\xe6\x96\x87\xe4\xbb\xb6/Homework/urllib/google.html'

>>> filename[1]

<httplib.HTTPMessage instance at 0xb6e2c38c>

### 3.urllib. request.urlcleanup()

清除由于urllib. request.urlretrieve()所产生的缓存

### 4.urllib. request.quote(url)和urllib.quote\_plus(url)

将url数据获取之后，并将其编码，从而适用与URL字符串中，使其能被打印和被web服务器接受。

>>> urllib.quote('http://www.baidu.com')

'http%3A//www.baidu.com'

>>> urllib.quote\_plus('http://www.baidu.com')

'http%3A%2F%2Fwww.baidu.com'

5.urllib.unquote(url)和urllib.unquote\_plus(url)

与4的函数相反。

6.urllib.urlencode(query)

将URL中的键值对以连接符&划分

这里可以与urlopen结合以实现post方法和get方法：

GET方法：

>>> import urllib

>>> params=urllib.urlencode({'spam':1,'eggs':2,'bacon':0})

>>> params

'eggs=2&bacon=0&spam=1'

>>> f=urllib.urlopen("http://python.org/query?%s" % params)

>>> print f.read()

POST方法：

>>> import urllib

>>> parmas = urllib.urlencode({'spam':1,'eggs':2,'bacon':0})

>>> f=urllib.urlopen("http://python.org/query",parmas)

>>> f.read()

## Urllib.parse

## urllib.parse.urlparse(urlstring, scheme='', allow\_fragments=True)

将urlstring解析成6个部分，它从urlstring中取得URL，并返回元组 (scheme, netloc, path, parameters, query, fragment)，但是实际上是基于namedtuple，是tuple的子类。它支持通过名字属性或者索引访问的部分URL，每个组件是一串字符，也有可能是空的。组件不能被解析为更小的部分，%后面的也不会被解析，分割符号并不是解析结果的一部分，除非用斜线转义，注意，返回的这个元组非常有用，例如可以用来确定网络协议(HTTP、FTP等等 )、服务器地址、文件路径，等等。

**>>> from** **urllib.parse** **import** urlparse

**>>>** o = urlparse('http://www.cwi.nl:80/*%7E*guido/Python.html')

**>>>** o

ParseResult(scheme='http', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',

params='', query='', fragment='')

**>>>** o.scheme

'http'

**>>>** o.port

80

**>>>** o.geturl()

'http://www.cwi.nl:80/%7Eguido/Python.html'

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Index | Value | Value if not present |
| scheme | 0 | URL scheme specifier | scheme parameter |
| netloc | 1 | Network location part | empty string |
| path | 2 | Hierarchical path | empty string |
| params | 3 | Parameters for last path element | empty string |
| query | 4 | Query component | empty string |
| fragment | 5 | Fragment identifier | empty string |
| username |  | User name | None |
| password |  | Password | None |
| hostname |  | Host name (lower case) | None |
| port |  | Port number as integer, if present | None |

urlparse.urlunparse(parts)

从一个元组构建一个url，元组类似urlparse返回的，它接收元组(scheme, netloc, path, parameters, query, fragment)后，会重新组成一个具有正确格式的URL，以便供Python的其他HTML解析模块使用。

**>>> import** urlparse

**>>>** url=urlparse.urlparse('http://www.baidu.com/index.php?username=guol')

**>>>** print url

ParseResult(scheme='http', netloc='www.baidu.com', path='/index.php', params='', query='username=guol', fragment='')

**>>>** u=urlparse.urlunparse(url)

**>>>** print u

http://www.baidu.com/index.php?username=guol

## urllib.parse.urlunparse

urlparse的反向结果

## urllib.parse.parse\_qs(qs, keep\_blank\_values=False, strict\_parsing=False, encoding='utf-8', errors='replace')

主要分析由url解析得来的query键值，把‘=’前后的值作为键、值存储到一个字典中，query包含多个参数时会以‘&’字符分割。

**>>>** a=urllib.parse.urlparse('http://localhost/test.py?a=hello&b=world')

**>>>** a

ParseResult(scheme='http', netloc='localhost', path='/test.py', params='', query='a=hello&b=world', fragment='')

**>>>** b=urllib.parse.parse\_qs(a.query,True)

**>>>** b

{'a': ['hello'], 'b': ['world']}

**>>>** b=urllib.parse.parse\_qs('a=hh&b=aa&c=11',True)

**>>>** b

{'a': ['hh'], 'b': ['aa'], 'c': ['11']}

**>>>** b=urllib.parse.parse\_qsl('a=hh&b=aa&c=11',True)

**>>>** b

[('a', 'hh'), ('b', 'aa'), ('c', '11')]

## urllib.parse.parse\_qsl(qs, keep\_blank\_values=False, strict\_parsing=False, encoding='utf-8', errors='replace')

和urllib.parse.parse\_qs类似，转换为一个列表

**>>>** b=urllib.parse.parse\_qsl('a=hh&b=aa&c=11',True)

**>>>** b

[('a', 'hh'), ('b', 'aa'), ('c', '11')]

## urlparse.urlsplit(urlstring[, scheme[, allow\_fragments]])

主要是分析urlstring，返回一个包含5个字符串项目的元组：协议、位置、路径、查询、片段。allow\_fragments为False时，该元组的组后一个项目总是空，不管urlstring有没有片段，省略项目的也是空。urlsplit()和urlparse()差不多。不过它不切分URL的参数。适用于遵循RFC2396的URL，每个路径段都支持参数。这样返回的元组就只有5个元素。

**>>>** **import** urlparse

**>>>** url=urlparse.urlparse('http://www.baidu.com/index.php?username=guol')

**>>>** print url

ParseResult(scheme='http', netloc='www.baidu.com', path='/index.php', params='', query='username=guol', fragment='')

**>>>** url=urlparse.urlsplit('http://www.baidu.com/index.php?username=guol')

**>>>** print url

SplitResult(scheme='http', netloc='www.baidu.com', path='/index.php', query='username=guol', fragment='') urlparse.urlunsplit(parts)

urlunsplit使用urlsplit()返回的值组合成一个url

## urllib.parse.urlunsplit(parts)

urlsplit的反向结果。

## urllib.parse.urljoin(base, url, allow\_fragments=True)

urljoin主要是拼接URL，它以base作为其基地址，然后与url中的相对地址相结合组成一个绝对URL地址。函数urljoin在通过为URL基地址附加新的文件名的方式来处理同一位置处的若干文件的时候格外有用。需要注意的是，如果基地址并非以字符/结尾的话，那么URL基地址最右边部分就会被这个相对路径所替换。如果希望在该路径中保留末端目录，应确保URL基地址以字符/结尾。

allow\_fragments默认为true，当url是一个绝对的URL（that is, starting with // or scheme://），那么url将取代base的URL出现在结果中

**>>>** **import** urlparse

**>>>** urlparse.urljoin('http://www.oschina.com/tieba','index.php')

'http://www.oschina.com/index.php'

**>>>** urlparse.urljoin('http://www.oschina.com/tieba/','index.php')

'http://www.oschina.com/tieba/index.php'

**>>>** urljoin('http://www.cwi.nl/*%7E*guido/Python.html',

**...**  '//www.python.org/*%7E*guido')

'http://www.python.org/%7Eguido'

## urllib.parse.urldefrag(url)

如果URL中包含的片段识别，返回修正版本URL with no片段片段的识别，确定了与单独的字符串。如果没有确定在URL片段，unmodified and an empty返回URL字符串。

**>>>** url=”https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse”

**>>>** urllib.parse.urldefrag(url)

DefragResult(url='https://docs.python.org/3/library/urllib.parse.html', fragment='module-urllib.parse')

# 21.6. [urllib.request](http://python.usyiyi.cn/documents/python_352/library/urllib.request.html#module-urllib.request) — 打开URL的拓展类

**Source code:** [Lib/urllib/request.py](https://hg.python.org/cpython/file/3.5/Lib/urllib/request.py)

[urllib.request](http://python.usyiyi.cn/documents/python_352/library/urllib.request.html#module-urllib.request) 模块定义了在基本和摘要式身份验证、 重定向、 cookies等应用中打开 Url (主要是 HTTP) 的函数和类。

See also

[Requests 包](https://requests.readthedocs.org/) 建议用作高级的非底层的 HTTP 客户端接口。

[urllib.request](http://python.usyiyi.cn/documents/python_352/library/urllib.request.html#module-urllib.request) 模块定义了以下函数︰

urllib.request.urlopen(url, data=None, [timeout, ]\*, cafile=None, capath=None, cadefault=False, context=None)

打开url参数传进来的网址，既可以是一个字符串，又可以是一个[Request](http://python.usyiyi.cn/documents/python_352/library/urllib.request.html#urllib.request.Request)对象。

当你有额外的数据要发送到服务器，那data参数必须是一个有数据的bytes对象，否则为None。其实data 也可以是一个可迭代的对象，若是，则headers中必须带 Content-Length 参数。若本次HTTP请求要用POST方法，data必须有数据；若为GET方法时， data写None就行。

data参数应该是一个标准的application/x-www-form-urlencoded 格式的缓冲区.[urllib.parse.urlencode()](http://python.usyiyi.cn/documents/python_352/library/urllib.parse.html#urllib.parse.urlencode) 函数接受一个映射或者一个内含一个两元素元组的序列，并且返回一个该格式的ASCII字符串。data参数在使用之前应该被编码成bytes。

urllib.request 模块参照 HTTP/1.1 标准， 并且在HTTP请求中，头信息可以有Connection:close。

timeout 参数是可选的。它指定一个超时时间。若超过该时间，任何操作都会被阻止，如链接超时（如果没有指定，则会默认值为socket.\_GLOBAL\_DEFAULT\_TIMEOUT）。This actually only works for HTTP, HTTPS and FTP connections.

如果指定了 context 参数，则它必须是一个描述了各种SSL选项的 [ssl.SSLContext](http://python.usyiyi.cn/documents/python_352/library/ssl.html#ssl.SSLContext) 实例。更详细内容，请点击 [HTTPSConnection](http://python.usyiyi.cn/documents/python_352/library/http.client.html#http.client.HTTPSConnection) 。

cafile 和 capath 参数都是可选的。他们应该被指定一组“HTTPS请求”所需要的受信任的CA 证书。cafile 参数应该指向文件，该文件包含了一大堆CA证书，而capath 参数应该指向一个目录，该目录包含了散列过的证书。更多的信息，可以从 [ssl.SSLContext.load\_verify\_locations()](http://python.usyiyi.cn/documents/python_352/library/ssl.html#ssl.SSLContext.load_verify_locations) 里找到。

cadefault 参数不能用了，所以不用管它了，吼吼~~....................

这个urlopen函数总是返回一个对象，能被[context manager](http://python.usyiyi.cn/documents/python_352/glossary.html#term-context-manager)（上下文管理器）的一些方法使用，如：

* geturl() — 返回一个资源索引的URL，通常重定向后的URL照样能get到。
* info() — 返回页面的元信息，如头信息。像 [email.message\_from\_string()](http://python.usyiyi.cn/documents/python_352/library/email.parser.html#email.message_from_string) 实例的格式一样。（请看 [Quick Reference to HTTP Headers](https://www.cs.tut.fi/%7Ejkorpela/http.html)）
* getcode() – 返回响应后，HTTP的状态码。

对于HTTP和HTTPS的URL，这个urlopen函数返回一个稍微修正过的[http.client.HTTPResponse](http://python.usyiyi.cn/documents/python_352/library/http.client.html#http.client.HTTPResponse)对象除了上述的三种新方法外，还有一个叫msg的属性，包含着和[reason](http://python.usyiyi.cn/documents/python_352/library/http.client.html#http.client.HTTPResponse.reason)属性一样的信息 — 原因短语从服务器上返回 — 代替了[HTTPResponse](http://python.usyiyi.cn/documents/python_352/library/http.client.html#http.client.HTTPResponse)文档里指定的那些响应头。

对于 FTP，文件，和数据的URL，都显式地由以前的 [URLopener](http://python.usyiyi.cn/documents/python_352/library/urllib.request.html#urllib.request.URLopener) 和 [FancyURLopener](http://python.usyiyi.cn/documents/python_352/library/urllib.request.html#urllib.request.FancyURLopener) 类处理，处理它们的函数，返回一个 urllib.response.addinfourl 对象。