利用urllib爬虫

# 注意： python2 与python3 中的urllib有了改变。

**Python2.x** 中 引入**import urllib**  **import urllib2** 包

而**Python3.x**中引入import urllib import urllib.request

最新的：直接引入 import urllib3

**urllib/urllib2/httplib http库，httplib底层一点，这些库都是Python自带的内置模块，但是推荐第三方的库requests。**

# urllib库

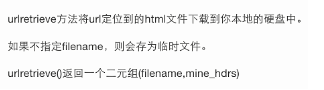




## urlopen



## urlretrieve()函数



**urlretrieve**函数可以指定两个参数，一个是文件的url，另外一个是下载到本地的目录及新文件名称。

如果只指定url，则默认放到临时目录下，重启电脑，就会消失。

Python2 中：

**import** urllib  
imgUrl = **'http://rpic.douyucdn.cn/a1707/19/12/1710060\_170719124107.jpg'**;  
ret = urllib.urlretrieve(imgUrl,**'D://beauty.jpg'**);  
**print**(ret);

返回的：



# Python2与python3中urllib的改进

## 改进

在**Python 3.x**版本中，urllib2这个模块已经不单独存在（也就是说当你**import urllib2**时，系统提示你没这个模块），urllib2被合并到了urllib中，叫做**urllib.request 和 urllib.error** 。

urllib整个模块分为**urllib.request, urllib.parse, urllib.error**。

例： 其中**urllib2.urlopen()变成了urllib.request.urlopen()**

**urllib2.Request()变成了urllib.request.Request()**

## python2.x 中：

**import urllib**

url = 'http://www.baidu.com';

ret = urllib.urlopen(url);

str = ret.read();

print(str);

## python3.x 中

**import urllib.request**

url = 'http://www.baidu.com';

f = **urllib.request.urlopen(url);**

ff = f.read();

print(ff);

# urllib和urllib2模块之间的区别

### 在python中，urllib和urllib2不可相互替代的。

### 整体来说，urllib2是urllib的增强，但是urllib中有urllib2中所没有的函数。

### urllib2可以用urllib2.openurl中设置Request参数，来修改Header头。如果你访问一个网站，想更改User Agent（可以伪装你的浏览器），你就要用urllib2.

### urllib支持设置编码的函数，urllib.urlencode,在模拟登陆的时候，经常要post编码之后的参数，所以要想不使用第三方库完成模拟登录，你就需要使用urllib。

### urllib一般和urllib2一起搭配使用。

# Python2中的urllib与urllib2的区别

如果你用过 Python 写一些网络应用，应该对[urllib](http://www.nowamagic.net/academy/tag/urllib)和urllib2比较熟悉。那么urllib和urllib2之间有什么区别呢？我们应该用urllib还是urllib2呢？

看到老外写的一篇《[Python: difference between urllib and urllib2](http://www.hacksparrow.com/python-difference-between-urllib-and-urllib2.html)》，这里简单翻译一下。

You might be intrigued by the existence of two separate URL modules in Python -urllib and urllib2. Even more intriguing: they are not alternatives for each other. So what is the difference between urllib and urllib2, and do we need them both?

你可能对于Python中两个独立存在的 urllib 和 urllib2 感到好奇。更有趣的是：它们并不是可以相互代替的。那么这两个模块间的区别是什么，并且这两个我们都需要吗？

**urllib and  urllib2** are both Python modules that do URL request related stuff but offer different functionalities. Their two most significant differences are listed below:

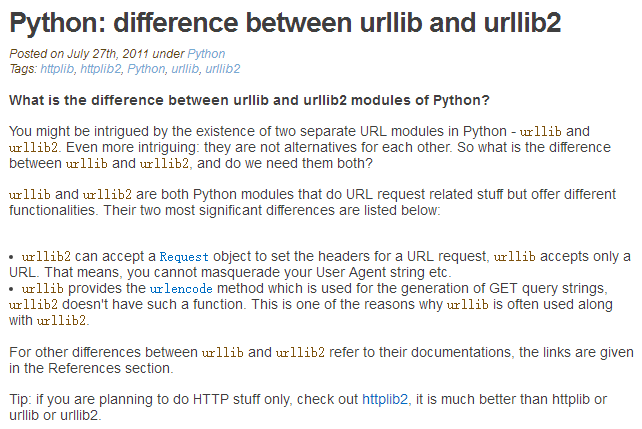
**urllib 和**[**urllib2**](http://www.nowamagic.net/academy/tag/urllib2)都是接受URL请求的相关模块，但是提供了不同的功能。两个最显著的不同如下：

* urllib2 can accept a Request object to set the headers for a URL request,urllib accepts only a URL. That means, you cannot masquerade your User Agent string etc. **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 provides the urlencode method which is used for the generation of GET query strings, urllib2 doesn't have such a function. This is one of the reasons why urllib is often used along with urllib2.  **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。**

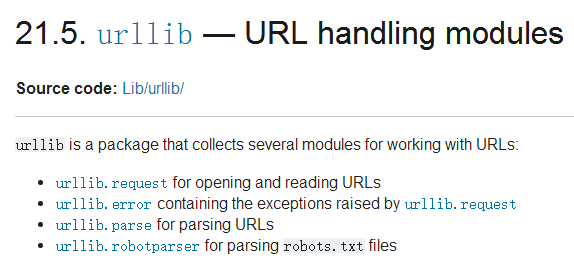
For other differences between urllib and urllib2 refer to their documentations, the links are given in the References section.

Tip: if you are planning to do HTTP stuff only, check out httplib2, it is much better than httplib or urllib or urllib2.

提示：如果你仅做HTTP相关的，看一下httplib2，比其他几个模块好用。



# urllib主要方法



**urllib- - - - - - URL处理模块**

源代码:Lib / urllib /

urllib是一个包,收集几个模块来处理网址:

### urllib.request打开和浏览url中内容

### urllib.error包含从 urllib.request发生的错误或异常

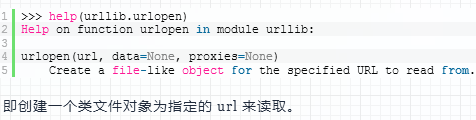
### urllib.parse解析url

### urllib.robotparser解析 robots.txt文件

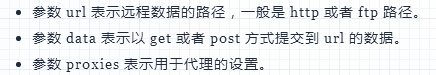
# urllib包中的urlopen方法

## 介绍urlopen方法

Python urllib 库提供了一个从指定的 URL 地址获取网页数据，然后对其进行分析处理，获取想要的数据。urllib 是 python 自带的一个抓取网页信息一个接口，他最主要的方法是[urlopen()](http://www.nowamagic.net/academy/tag/urlopen)，是基于 python 的 open() 方法的。‘’



详细点就是，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：



## urlopen返回类文件对象的方法

urlopen 返回一个类**文件对象**，它提供了如下方法：

### read() , readline() , readlines()，fileno()和close()： 这些方法的使用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地址。

## 示例：

import urllib

**ret = urllib.urlopen('http://www.baidu.com');**

print(ret.getcode());

print'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

print ret.geturl();

print'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

print ret.info();

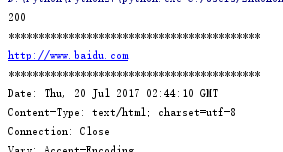
print'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

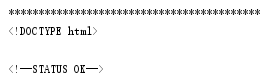
**for line in ret:**

**print line;**

**ret.close();**

**输出：**



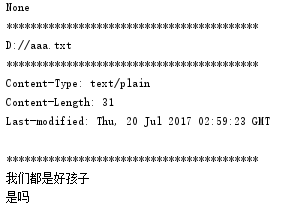


## URL支持的协议http或ftp等

**urllib.urlopen('网址')**

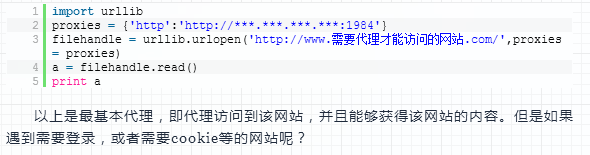
* 这里传入urlopen()的参数有特别说要求，要遵循一些网络协议，比如http,ftp,也就是说，在网址的开头必须要有http://这样的说明，如：urllib.urlopen('http://www.baidu.com')。
* 要么就是本地文件，本地文件需要使用file关键字，比如 urllib.urlopen('file:nowamagic.py')，注意，这里的hello.py是指的是当前的classpath所指定的内容，如果对hello.py这里有什么疑问那一定是python寻找classpath的顺序不是很清楚了，当然也可以直接写全部路径，urllib.urlopen('file:F:\pythontest\nowamagic.py')。
* 打开 ftp 文件也是可以的，写法 urllib.urlopen(url='ftp://用户名:密码@ftp地址/') 等。

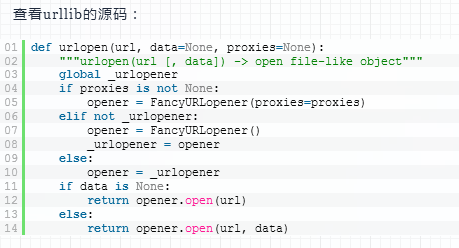
**import** urllibret = urllib.urlopen(**'file:D://aaa.txt'**);  
**print**(ret.getcode());  
**print'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'**;  
**print** ret.geturl();  
**print'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'**;  
**print** ret.info();  
**print'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'**;  
**print** ret.read();  
ret.close();



# 利用urllib包实现代理的功能

**urllib的**[**代理**](http://www.nowamagic.net/academy/tag/%E4%BB%A3%E7%90%86)**设置：**





由上面**urllib的urlopen的源码**，可以看出，还可以传入一个data参数，data参数也应该是一个字典，因为在使用浏览器向服务器发送数据的时候，我们发送的就是字典类型的数据。

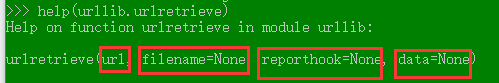
还有一点，就是代理支持是 **python 2.3** 以后加入的。

# urllib包中urlretrieve方法

## urlretrieve()函数的功能

**urlretrieve() 方法**直接将**远程数据**下载到本地。

## 利用help命令获取帮助



•**参数 finename** 指定了保存本地路径（如果参数未指定，urllib会生成一个临时文件保存数据。）

**•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时会触发该回调，我们可以利用这个回调函数来显示当前的下载进度**。

•**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(filename, headers)元组，filename 表示保存到本地的路径，header 表示服务器的响应头。

## urlopen()与urlretrieve()的区别

**urlopen()** 可以轻松获取远端 html 页面信息，然后通过 python 正则对所需要的数据进行分析，匹配出想要用的数据，在利用**urlretrieve()** 将数据下载到本地。对于访问受限或者对连接数有限制的远程 url 地址可以采用 proxies（代理的方式）连接，如果远程数据量过大，单线程下载太慢的话可以采用多线程下载，这个就是传说中的爬虫。

## 下载文件示例（显示下载进度）

**import** urllib  
**def** Schedule(a, b, c):  
 per = 100.0 \* a \* b / c  
 **if** per > 100:  
 per = 100  
 **print**(**'%.2f%%'** % per);  
url = **'http://rpic.douyucdn.cn/a1707/19/12/1710060\_170719124107.jpg'**local = url.split(**'/'**)[-1]  
local = **'D://beautifulgirl.jpg'**;  
**urllib.urlretrieve(url,local,Schedule)**

# URL编码与解码

**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。比如以 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' '

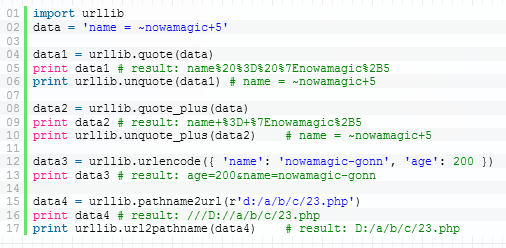
### urllib.unquote\_plus(string ) ：对字符串进行解码；

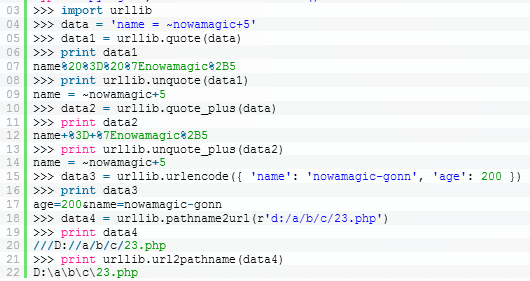
### urllib.urlencode(query[, doseq])：将dict或者包含两个元素的元组列表转换成url参数。例如 字典{'name': 'dark-bull', 'age': 200}将被转换为"name=dark-bull&age=200"

### 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；

示例：





# urllib.request 介绍

原地址：   
<https://docs.python.org/3.5/library/urllib.request.html#module-urllib.request>

**urllib.request — 为打开url提供的可扩展类库**

源代码:Lib / urllib / request.py

urllib.request模块定义了方法和类,帮助打开url(主要是HTTP)在一个复杂的世界——基本和摘要式身份验证,重定向,cookies等等。

## urllib.request.urlopen()

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

**打开网址URL,这可以是一个字符串或一个 Request对象。**

数据必须是一个字节对象指定额外的数据发送到服务器或 None。如果没有这样的数据是必要的，数据也可能是一个iterable对象而且在这种情况下必须在最开始时指定内容的长度。目前HTTP是唯一一个这样请求数据的，当数据参数被提供时，**HTTP请求将会执行POST请求而不是GET请求**。

数据应该是一个缓冲的在标准应用程序中以 x-www-form-urlencoded的格式。 urllib.parse.urlencode()函数接受一个映射或序列集合,并返回一个ASCII文本字符串的格式。它应该在被用作数据参数之前，被编码为字节。

**urllib.request 模块** 使用 HTTP/1.1协议，并且包括请求 Connection:close在HTTP请求头。

可选的第二个超时参数**timeout**，用于阻塞操作,比如连接请求(如果未指定,全球将使用默认超时设置)。这实际上只适用于HTTP、HTTPS和FTP连接。

如果**context**被指定，它必须是一个 ssl.SSLContext实例描述各种SSL选项。点击HTTPSConnection查看更多细节。

可选**cafile和capath**参数指定一组被HTTPS请求信任的CA证书。cafile应该指向一个文件包含CA证书的包,而capath应该指向一个散列的证书文件的目录。点击ssl.SSLContext.load\_verify\_locations()查看更多的信息。

**cadefault**参数被忽略。

**这个函数始终返回一个对象，像context（上下文） 管理者并提供这些方法**

geturl()——返回URL的资源检索,常常重定向之后使用

info()——返回页面的元信息,如标题，组成 email.message\_from\_string(的)实例(见快速参考HTTP头)

getcode()——返回响应的HTTP状态代码。

为HTTP和HTTPS url，这个函数返回的一个 **http.client.HTTPResponse**对象略有不同。除了上面的三种新方法中，这个message属性包含相同的信息像reason属性——由服务器返回的原因——而不是响应头,因为它在文档中指定 HTTPResponse。

FTP、文件和数据请求url和显式地处理 URLopener和 FancyURLopener类，这个函数返回一个 urllib.response.addinfourl对象。

**urllib.request.urlopen()会在 URLError中抛出协议错误。**

请注意,，可能返回None，这在没有处理程序处理请求(尽管全球默认安装 OpenerDirector并使用 UnknownHandler以确保这不会发生)时发生。

此外，如果检测到代理设置(例如,当一个 \*\_proxy环境变量如 http\_proxy已经被设定),，ProxyHandler默认安装并确保请求都通过代理来处理。

遗留的 urllib.urlopen从Python 2.6和更早已经被中断;；urllib.request.urlopen()对应于旧的 urllib2.urlopen。代理处理,是通过字典参数完成的 urllib.urlopen可以使用 ProxyHandler对象。

3.2版本的变化：cafile和capath被补充。

3.2版本的变化：如果可能的话，现在支持HTTPS虚拟主机(也就是说，如果 ssl.HAS\_SNI是真的)。

在新的3.2版本：数据可以是一个iterable对象。

3.3版本的变化：cadefault被补充。

3.4.3版本的变化：context被补充。

## urllib.request.install\_opener(opener)

**安装一个 OpenerDirector实例作为全球默认的opener** 。安装一个opener 必要的,如果你想让urlopen使用这个opener ;否则,简单地调用 OpenerDirector.open()而不是 urlopen()。这样代码不会检查一个真实的 OpenerDirector并且任何类的适当的接口都可以运作。

## urllib.request.build\_opener([handler, …])

返回一个顺序的链的处理程序 OpenerDirector的实例。处理程序可以是BaseHandler的实例,或者 BaseHandler的子类(在这种情况下,必须调用没有参数的构造函数)。下面这些类的实例将提前处理程序,除非处理程序包含它们，或者它们子类的实例：ProxyHandler(如果检测到代理设置), UnknownHandler, HTTPHandler, HTTPDefaultErrorHandler, HTTPRedirectHandler, FTPHandler, FileHandler, HTTPErrorProcessor.

如果Python安装SSL支持(即如果 ssl模块可以被导入)， HTTPSHandler也将被添加。

一个 BaseHandler子类也可以通过改变它的 handler\_order属性来修改它再处理程序列表中的位置。

## urllib.request.pathname2url(path)

**将路径名转换成路径**，从本地语法形式的路径中使用一个URL的路径组成部分。这不会产生一个完整的URL。它将返回引用 quote()函数的值。

## urllib.request.url2pathname(path)

**将路径组件转换为本地路径的语法**。这个不接受一个完整的URL。这个函数使用 unquote()解码的通路。

## urllib.request.getproxies()

这个helper函数返回一个日程表dictionary 去代理服务器的URL映射。扫描指定的环境变量 \_proxy大小写不敏感的方法,对所有的[**操作系统**](http://lib.csdn.net/base/operatingsystem)，当它不能找到它，从Mac OS X的Mac OSX系统配置和Windows系统注册表中寻找代理信息。如果两个大写和小写环境变量存在(或不一样)，小写优先。

请注意，如果环境变量 REQUEST\_METHOD已被设置,这通常表明你在CGI脚本运行环境,此时环境变量 HTTP\_PROXY(大写 \_PROXY)将被忽略。这是因为该变量可以被客户端使用注射“代理:”HTTP头。如果你需要使用一个HTTP代理在CGI环境中,要么使用 ProxyHandler明确,或者确保变量名小写(或至少是 \_proxy后缀)。

### 类 urllib.request.Request(url, data=None, headers={}, origin\_req\_host=None, unverifiable=False, method=None)

这个类是一个抽象的URL请求。

url应该是一个字符串包含一个有效的url。

数据必须是一个字节对象指定额外的数据发送到服务器或 None。如果没有这样的数据是必要的，数据也可能是一个iterable对象而且在这种情况下必须在最开始时指定内容的长度。目前HTTP是唯一一个这样请求数据的，当数据参数被提供时，HTTP请求将会执行POST请求而不是GET请求。

数据应该是一个缓冲的在标准应用程序中以 x-www-form-urlencoded的格式。 urllib.parse.urlencode()函数接受一个映射或序列集合,并返回一个ASCII文本字符串的格式。它应该在被用作数据参数之前，被编码为字节。

**headers 应该是一个字典，如果 add\_header()被称为与每个键和值作为参数。这通常是用来“恶搞” User-Agent头的值，因为使用一个浏览器识别本身——一些常见HTTP服务器只允许请求来自浏览器而不是脚本。**例如，Mozilla Firefox可能识别本身 “Mozilla/5.0 (X11; U; [**Linux**](http://lib.csdn.net/base/linux) i686) Gecko/20071127 Firefox/2.0.0.11”。而 **urllib默认的用户代理字符串 是”Python-urllib/2.6”在Python 2.6（）**。

一个Content-Type header的例子 用数据论证将发送一个字典 {“Content-Type”:”application/x-www-form-urlencoded”}。

最后两个参数只是正确处理第三方HTTP cookie：

origin\_req\_host应该请求原始的主机交易，就像定义的RFC 2965。它默认为 http.cookiejar.request\_host(self)。这是原始请求的主机名或IP地址，由用户发起。例如。如果请求是一个图像在HTML文档中，这应该是请求的请求主机包含图像的页面。

无法核实的表明是否应该请求是无法核实的，这由RFC 2965定义。它默认为 False。一个无法核实的请求的URL的用户没有允许的选择。例如，如果请求是一个图像在一个HTML文档,和用户没有选择通过图像的自动抓取，这应该是正确的。

这个方法应该是一个字符串,表示将使用(如HTTP请求方法。 ‘HEAD’)。如果提供，其值是存储在 method属性和使用 get\_method()。通过设置子类可能表明一个默认的方法 method类本身的属性。

3.3版本的变化:：Request.method参数是添加到请求类。

3.4版本的变化：默认的 Request.method可能会显示在类级别。

### 类urllib.request.OpenerDirector

OpenerDirector类打开url并通过 BaseHandler连接在一起。它管理处理程序的连接,和恢复错误。

### 类 urllib.request.BaseHandler

这是对于所有已注册的处理程序的基类

### 类 urllib.request.HTTPRedirectHandler

一个类来处理重定向

### 类urllib.request.HTTPCookieProcessor(cookiejar=None)

一个类来处理HTTP cookie。

### 类 urllib.request.ProxyHandler(proxies=None)

导致请求通过一个代理。如果代理是给定的，它必须是一个字典的代理协议名称映射到url。默认值是从环境变量的列表 \_proxy中读取代理。如果没有代理设置环境变量，那么在Windows环境中代理设置了从注册表部分的网络设置，在Mac OS X环境代理信息检索的OS X系统配置框架。

禁用一个代理传递一个空的字典。

no\_proxy环境变量可以被用来指定主机不能通过代理；如果设置,它应该是一个以逗号分隔的主机名后缀。可选 ：port附加为例 cern.ch,ncsa.uiuc.edu,some.host:8080.

请注意HTTP\_PROXY如果一个变量将被忽略 REQUEST\_METHOD设置;参见文档 getproxies().

### 类 urllib.request.HTTPPasswordMgr

保持一个[**数据库**](http://lib.csdn.net/base/mysql) (realm, uri) -> (user, password)映射。

### 类 urllib.request.HTTPPasswordMgrWithDefaultRealm

保持一个数据库 (realm, uri) -> (user, password)映射。一个领域 None被认为是一个全方位领域,如果没有其他搜索领域

### 类 urllib.request.HTTPPasswordMgrWithPriorAuth

一个变体 HTTPPasswordMgrWithDefaultRealm还有一个数据库 uri -> is\_authenticated的映射。可以使用BasicAuth处理程序来确定当发送身份验证凭证立即而不是等待 401响应。

### 类 urllib.request.AbstractBasicAuthHandler(password\_mgr=None)

这是mixin类,帮助与HTTP身份验证,远程主机和代理。果有password\_mgr,应该是兼容 HTTPPasswordMgr的。请参阅部分 HTTPPasswordMgr对象必须支持的接口信息。如果passwd\_mgr还提供了 is\_authenticated和 update\_authenticated方法(见 HTTPPasswordMgrWithPriorAuth对象),然后处理程序将使用 is\_authenticated结果对于一个给定的URI来决定是否发送请求的身份验证凭证。如果 is\_authenticated返回 TrueURI,凭证发送。如果 is\_authenticated是 False凭证不发送,然后如果 401收到响应请求发送身份验证凭证。如果身份验证成功, update\_authenticated被称为设置 is\_authenticated TrueURI,这样后续请求的URI或任何super-URIs将自动包括身份验证凭证。

在新的3.5版本:添加 is\_authenticated支持。

### 类 urllib.request.HTTPBasicAuthHandler(password\_mgr=None)

与远程主机处理身份验证。如果有password\_mgr，应该是兼容HTTPPasswordMgr的。请参阅部分 HTTPPasswordMgr对象必须支持的接口信息。HTTPBasicAuthHandler将提高 ValueError当面对一个错误的身份验证方案。

### 类 urllib.request.ProxyBasicAuthHandler(password\_mgr=None)

处理与代理身份的验证。如果有password\_mgr,应该是兼容 HTTPPasswordMgr的。请参阅部分 HTTPPasswordMgr对象必须支持的接口信息。

### 类urllib.request.AbstractDigestAuthHandler(password\_mgr=None)

这是mixin类,帮助与HTTP身份验证,远程主机和代理。password\_mgr,如果有,应该是兼容的 HTTPPasswordMgr;请参阅部分 HTTPPasswordMgr对象必须支持的接口信息

### 类urllib.request.HTTPDigestAuthHandler(password\_mgr=None)

与远程主机处理身份验证。如果有password\_mgr，应该是兼容 HTTPPasswordMgr的；请参阅部分 HTTPPasswordMgr对象必须支持的接口信息。摘要式身份验证处理程序和基本身份验证处理器都是补充说,摘要式身份验证总是尝试第一次。如果主机返回一个40 x再次回应,它发送到基本身份验证处理程序来处理。这个处理程序方法将提高 ValueError当面对除了消化或基本身份验证方案。

3.3版本的变化:提高 ValueError不支持的身份验证方案。

### 类urllib.request.ProxyDigestAuthHandler(password\_mgr=None)

处理与代理身份验证。如果有password\_mgr，应该是兼容 HTTPPasswordMgr的；请参阅部分 HTTPPasswordMgr对象必须支持的接口信息

### 类 urllib.request.HTTPHandler

一个类来处理HTTP url

### 类 urllib.request.HTTPSHandler(debuglevel=0, context=None, check\_hostname=None)

一个类来处理开放的HTTPS url。在context 文和check\_hostname有相同的意义 http.client.HTTPSConnection.

3.2版本的变化:context and check\_hostname被补充。

### 类 urllib.request.FileHandler

打开本地文件。

### 类 urllib.request.DataHandler

开放数据的url。

### 类 urllib.request.FTPHandler

开放的FTP url。

### 类 urllib.request.CacheFTPHandler

打开FTP url,保持打开的FTP连接缓存来减少延迟。

### 类 urllib.request.UnknownHandler

全方位类处理未知的url。

### 类 urllib.request.HTTPErrorProcessor

HTTP错误响应过程。