使用urllib获取www资源

目标：学习使用urllib编写简单爬虫程序，获取webpage资源

Copyright reserved. hhhparty@163.com 本文限课堂使用。

1. 掌握单页面普通网页的获取方法

urllib.request.urlopen()方法应用

urllib.request中最常用的方法是urlopen(),它也是我们使用urllib获取普通网页的基本方法。
在应用之前，我们先看一下urllib的源代码，这是从事IT软件类技术工作要养成的职业习惯。
由于urllib是python3内置库，所以无需安装。源代码的路径可以在import urllib或import.request后，使用"**file**"属性查看。
从头部注释中可以了解urlopen方法需要传入一个字符串参数：页面的URL，然后它会打开这个URL，返回类文件对象的响应对象。

> def urlopen(url, data=None, timeout=socket._GLOBAL_DEFAULT_TIMEOUT,*, cafile=None, capath=None, cadefault=False, context=None)

查看上面urlopen方法原型，了解它的功能和调用方法。可以看到，url是必须给定的参数，其他参数可以默认。下面我们尝试使用urlopen打开百度网页。 我们使用了with...as...语句调用，这样会更有利于在不使用时正常关闭连接。返回的结果是HTTPResponse对象。调用这个对象的read()方法，可以访问具体的文件内容。

In [ ]:

```
"""使用urlopen()实现最简单的url访问
"""
import urllib.request

#可以使用语句查看摘要信息：print(urllib.request.__all__)
#可以使用语句查看urllib的本地位置：print(urllib.request.__file__)


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

with urllib.request.urlopen(url) as response:
    print(response)
    #print(response.read())
    #print(response.read().decode('utf-8'))
    print(response.read()[100:2000].decode('utf-8'))
```

urllib.request.urlretrieve()方法

urllib.request.urlretrieve()方法能够以另一种形式获取页面内容，它会将页面内容存为临时文件并获取response头。
可以查看urlretrieve方法的原型： `def urlretrieve(url, filename=None, reporthook=None, data=None)`

In [ ]:

```
"""使用urlretrieve()将页面内容存为临时文件，并获取response头
"""
import urllib.request

url = 'http://www.baidu.com'
localfile, headers = urllib.request.urlretrieve(url)
print(headers)
print('--'*10)
print(localfile)
```

理解HTTPResponse对象

HTTPResponse对象是一种类文件对象，除了可以文件的read()方法读取它的内容外，还有别的属性和方法。例如：r.code与r.status属性存放本次请求的响应码;r.headers属性存放响应头；r.url属性存放了发出响应的服务器URL；还可以尝试info()和geturl()方法。使用response的geturl()和info方法来验证请求与响应是否如我们希望的一样。有时会出现请求发往的服务器与应答服务器不是同一台主机的情况。

In [ ]:

```
"""理解HTTPResponse对象
"""
import urllib.request

url = 'http://www.baidu.com'
with urllib.request.urlopen(url) as r:
    print(r)
    print(r.code)
    print(r.status)
    print(r.headers)
    print(r.url)
    print(response.info())
    print(response.geturl())
```

2. 掌握使爬虫更像浏览器的方法

默认情况下，urllib发出的请求头大致如下所示：

> GET / HTTP/1.1
> Accept-Encoding: identity Host: 10.10.10.135
> User-Agent: Python-urllib/3.6
> Connection: close

大多数网站的服务器端会进行内容审查，检查客户端类型，一方面是为了满足多样化的需求；另一方面也可以限制一些网络爬虫程序的访问。
上面内容中的User-Agent就是一个内容审查重点，一般的浏览器发出的请求头如下所示：

> Accept:
> text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,/;q=0.8
> Accept-Encoding: gzip, deflate, br Accept-Language: zh-CN,zh;q=0.9 Cache-Control: max-age=0 Connection: keep-alive Cookie:
> BAIDUID=AFA97F911C6ADE2D2B06C704D9581140:FG=1;
> BIDUPSID=AFA97F911C6ADE2D2B06C704D9581140; PSTM=1537772835;
> BD_UPN=12314753; BDORZ=B490B5EBF6F3CD402E515D22BCDA1598; ispeed_lsm=2;
> MCITY=-131%3A; delPer=0; BD_CK_SAM=1; PSINO=2;
> H_PS_PSSID=26523_1434_21094_20929;
> H_PS_645EC=c9eazWKQBOzC0E25N41dsXkBqUjMnVbupNHxd9OHwW7BqVtt5h0N0eT2dFs;
> BD_HOME=0 DNT: 1 Host: www.baidu.com Referer: https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=python%20urllib%20request%20%E5%86%85%E5%AE%(https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=python%20urllib%20request%20%E5%86%85%E5%AE%
> Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
> AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36

服务器发现不是正常浏览器可以拒绝提供服务，例如访问www.z.cn 时，使用下面代码会报出 HTTP Error 503: Service Unavailable： ()

```
with urllib.request.urlopen(url) as response:
    print(response.status)
```

()

这时我们可以定制请求对象HTTPRequest，是指更像是浏览器发出的。

In [ ]:

```
"""定制request对象，使爬虫更像浏览器
"""
import urllib.request

url = 'https://www.amazon.cn/gp/goldbox/ref=cngwIter_DEAL_PER_DAY_title?pf_rd_p=1d2f8bbc-3f28-47e6-acb8-ce35d618c4e5&pf_rd_s=desktop-1&pf_rd_t=36701&pf_rd_i=desktop&pf_rd_m=A1AJ19PSB66TGU&pf_rd_r=8XXT48R1QRS14P1OW9M5&pf_rd_r=8XXT48R1QRS14P1OW9M5&pf_rd_p=1d2f8bbc-3f28-47e6-acb8-ce35d618c4e5'
'''使用下面代码会报出 HTTP Error 503: Service Unavailable
with urllib.request.urlopen(url) as response:
    print(response.status)

with urllib.request.urlopen(url) as response:
    print(response.status)
'''
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36'}
request = urllib.request.Request(url,headers=headers)
with urllib.request.urlopen(request) as response:
    print(response.status)
```

3. 掌握向服务器传递参数的方法

许多HTTP方法都可以用来向服务器提供数据，最常见的GET和POST方法都可以，但方式不同

使用GET方法向服务器提供数据

In [ ]:

```
"""使用GET方法，向百度服务器发送查询请求
"""
import urllib.request
import urllib.parse

querystr = {'wd':'北航'}
querystr_encode = urllib.parse.urlencode(querystr)
print(querystr_encode)
#https://www.baidu.com/s?wd=%E5%8C%97%E8%88%AA
url = 'http://www.baidu.com/s?' + querystr_encode
headers = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
           'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
          }
request = urllib.request.Request(url,headers=headers)
with urllib.request.urlopen(request) as response:
    print(response.status)
    print(response.headers)
    print(response.read().decode('utf-8'))
```

使用POST方法向服务器提供数据

In [ ]:

```
"""利用POST方法，向http://httpbin.org 提交
事前应在该网站进行设置，启动试用链接。
"""
import urllib.request
import urllib.parse

url = "http://httpbin.org/post"
payload = {'key1': 'value1', 'key2': 'value2'}
headers = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
           'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
          }
req = urllib.request.Request(url,data=urllib.parse.urlencode(payload).encode('utf-8'),headers=headers)
with urllib.request.urlopen(req) as r:
    print(r.read().decode('utf-8'))
```

In [ ]:

```
"""利用POST方法，向http://10.10.10.135/WebGoat/ 提交用户名和密码
"""
import urllib.request
import urllib.parse

url = 'http://10.10.10.135/dvwa/login.php'
cookie = 'PHPSESSID=898c1rsum58475qh3nros002n7; path=/'
headers = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
           'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
         'Cookie':cookie,
          }
authstr = {'username':'admin',
           'password':'admin',
           'Login':'Login',}
data = urllib.parse.urlencode(authstr).encode('utf-8')

request = urllib.request.Request(url,data=data,headers=headers)
with urllib.request.urlopen(request) as response:
    print(response.status)
    print(response.headers)
    cookie1 = response.headers['Set-Cookie']

url = 'http://10.10.10.135/dvwa/index.php'
headers = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
           'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
          'Cookie':cookie+';'+cookie1,
          }
print(headers)
request = urllib.request.Request(url,headers=headers)
with urllib.request.urlopen(request) as response:
    print(response.read())
```

4. 掌握设置超时访问限制和处理异常的方法

urllib.error处理异常,两个常用异常类：urllib.error.URLError和HTTPError

In [ ]:

```
""" 设置time-out"""
import socket
import urllib.request
# timeout in seconds
timeout = 3
socket.setdefaulttimeout(timeout)
# this call to urllib.request.urlopen now uses the default timeout
# we have set in the socket module
req = urllib.request.Request('http://www.python.org/')
a = urllib.request.urlopen(req).read()
print(a)
```

In [ ]:

```
"""使用urllib.error处理异常
URLError继承自OSError，是urllib的异常的基础类
HTTPError是验证HTTP response实例的一个异常类。

HTTP protocol errors是有效的response，有状态码、headers、body。

一个成熟的程序需要管理所有输出，不仅有希望见到的输出，还要有意料之外的异常。
logging的使用可以参考https://docs.python.org/3.5/howto/logging.html
"""

import urllib.request
import urllib.error
import urllib.parse
import logging

logging.basicConfig(format='%(asctime)s:%(levelname)s:%(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S',
                    filename='C:\\Users\\leo\Documents\\crawlerslesson1_crawler.log',
                    level=logging.DEBUG)
try:
    #url = 'http://www.baidu11.com'
    url = 'http://10.10.10.135/WebGoat/attack'
    headers = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8',
           'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
          }
    request = urllib.request.Request(url,headers=headers)
    with urllib.request.urlopen(request) as response:
        print(response.status)
        print(response.read().decode('utf-8'))

except urllib.error.HTTPError as e:
    import http.server
    #print(http.server.BaseHTTPRequestHandler.responses[e.code])
    logging.error('HTTPError code: %s and Messages: %s'% (str(e.code),http.server.BaseHTTPRequestHandler.responses[e.code]))
    logging.info('HTTPError headers: ' + str(e.headers))
    logging.info(e.read().decode('utf-8'))
    print('不好意思，服务器卡壳儿了，请稍后重试。')
except urllib.error.URLError as e:
    logging.error(e.reason)
    print('不好意思，服务器卡壳儿了，请稍后重试。')
```

5. 实例：从百度贴吧下载多页话题内容

先了解以下百度贴吧http://tieba.baidu.com/f? (http://tieba.baidu.com/f?) 我们定义几个函数：

- loadPage(url) 用于获取网页
- writePage(html,filename) 用于将已获得的网页存储为本地文件
- tiebaCrawler(url,beginpPage,endPage,keyword)用于调度，提供需要抓取的页面URLs
- main：程序主控模块，完成基本命令行交互接口