3.7爬虫应用

2019年2月20日 21:08

```
from urllib import request
resp=request.urlopen("http://www,baidu.com")
print(resp.read())


request.urlretrieve("http://www,baidu.com","baidu.html")
#保存到文件


from urllib import parse
data={'name':'爬虫基础','greet':'hello world','age':100}
qs=parse.urlencode(data)
print(qs)
#例 将汉字进行编码
url='http://www.baidu.com/s'
params={"wd":"刘德华"}
qs=parse.urlencode(params)
url=url+"?"+qs
resp=request.urlopen(url)
print(resp.read())


#将列表进行解码
params={"wd":"刘德华"}
qs=parse.urlencode(params)
print(qs)
result=parse.parse_qs(qs)
print(result)


from urllib import parse
url='----------'
result=parse.urlpaese(url)
print(result)
#输出关于scheme netloc path params query fragment 的网址分解
print('scheme:':result.scheme)
#等等

result=parse.urlsplit(url)
print(result)
#唯一区别无params


from urllib import request
url='-----------'
#resp=request.urlopen(url)
#print(resp.read())
headers={'User-Agent':'Mozilla/5.0 (Windows NT 10.0；Win64；x64） AppleWebKit/537.36 (KHTML，like Gecko)chrome/62.0.3202.94 Safari/537.36'}
#referer中复制 然后'Referer':'----'
data={'first':'true','pn':1,'kd':'python'}
req=request.Request(url,headers=headers,data=parse.urlencode(data).encode('utf-8'),method='POST')
#request.urlopen(req)
resp=request.urlopen(req)
print(resp.read().decode('utf-8'))
#申请头



#网页neihanshequ.com/bar/1/


from urllib import request
url='http://httpbin.org/ip'
resp=request.urlopen(url)
print(resp.read())
#获取本身的ip

#ProxyHandler处理器 即换ip地址
#西刺 http://www.xicidaili.com/
```

#快代 http://www.kuaidaili.com/

#获得修改之后的ip

1.代理的原理：在请求目的url之前，先请求代理服务器，然后让代理服务器去请求目的url，代理服务器拿到目的 网站的数据后，再转发给代码
2.http://httpbin.org 这个网站可以方便查看http请求的一些参数
3.在代码中使用代理
    使用 ‘urllib.request.ProxyHandler'传入一个字典形式的代理。其key依赖于代理服务器能接收的类型。一般情况是http或者https。值是’ ip:port'
    使用上一步创建的handler以及request.build_opener创建一个opener对象
    使用上一步创建的opener调用open函数发起请求
    如下代码：
    from urllib import request
    url='http://httpbin.org/ip'
    handler=request.ProxtHandler({"http":"218.66.161.88:31769"})opener=request.build_opener(handler)
    #req=request.Request("http://httpbin.org/ip")
    #resp=opener.open(req)
    resp=opener.open(url)
    print(resp.read())

cookie是存储在浏览器的登陆账号密码
NAME 名字
VALUE 值
Expires 过期时间
Path 作用路径
Domain 作用域名 即子页面也可应用
SECURE 是否只在https协议下起作用
在浏览器抓包中是存储在profile-Response Headers中
Set-cookie：---

模拟cookie登陆
```
from urllib import request
url=""
headers={
    'User-Agent':-----
    "Cookie":----
}
req=request.Request
resp=request.urlopen(req)
with open('baocundewenjian.html'.'w',encoding='utf-8') as fp:
    fp.write(resp,read().decode('utf-8))
```

http.cookiejar模块
主要模块类有CookieJar FileCookieJar MozillaCookieJar LWPCookieJar
```
from urllib import request
from urllib import parse
from http.cookiejar import CookieJar
#登陆 创建对象 使用模块创建对象 登陆
cookiejar=CookieJar()
handler=request.HTTPCookieProcessor(cookiejar)
opener=request.build_opener(handler)
headers={
    'User-Agent':
}
data={
    'email':
    'password':
}
login_url=""
req=request.Request(login_url,data=parse.urlencode(data).encode('utf-8'),handers=headers)
opener.open(req)
#访问个人主页
dapeng_url="-----"
#获取个人主页的页面的时候 不要新建一个opener
#而应该使用之前那个opener 因为之前的opener已经包含了
#登陆所需要的cookie信息
req=request.Request(dapeng_url,headers=headers)
resp=opener.open(req)
with open('-----.html','w',encoding='utf-8') as fp:
    fp.write(resp.read().decode('utf-8'))
```

对以上代码进行封装
```
from urllib import request
```

```
from urllib import parse
from http.cookiejar import CookieJar
headers={
    'User-Agent':
}

def get_opener():
    cookiejar=CookieJar()
    handler=request.HTTPCookieProcessor(cookiejar)
    opener=request.build_opener(handler)
    return opener

def login_renren(opener):
    data={
        'email':
        'password':
    }
    login_url=""
    req=request.Request(login_url,data=parse.urlencode(data).encode('utf-8'),handers=headers)
    opener.open(req)

def visit_profile(opener):
    dapeng_url="-----"
    req=request.Request(dapeng_url,headers=headers)
    resp=opener.open(req)
    with open('-----.html','w',encoding='utf-8') as fp:
        fp.write(resp.read().decode('utf-8'))

if __name__ == '__main__':
    opener = get_opener()
    login_renren(opener)
    visit_profile(opener)
```

完————————————————————————————————

cookie保存到本地

```
from urllib import request
from http.cookiejar import MozillaCookieJar
cookiejar=MozillaCookieJar('cookie.txt')
handler=request.HTTPCookieProcessor(cookiejar)
opener=request.build_opener(handler)
resp=opener.open('http://www.baidu.com/')
cookiejar.save()
```

打开浏览器http://httpbin.org/cookies/set?course=spider
会建立一个名为course 内容为spider 的cookie 浏览器关闭后失效
也就是说代码内建立的cookie也会同等失效

```
from urllib import request
from http.cookiejar import MozillaCookieJar
cookiejar=MozillaCookieJar('cookie.txt')
handler=request.HTTPCookieProcessor(cookiejar)
opener=request.build_opener(handler)
resp=opener.open('http://httpbin.org/cookies/set?course=abc')
cookiejar.save(ignore_discard=True)
```

运行后具体页将会保存在cookie.txt中

```
from urllib import request
from http.cookiejar import MozillaCookieJar
cookiejar=MozillaCookieJar('cookie.txt')
cookiejar.load(ignore_discard=True)
handler=request.HTTPCookieProcessor(cookiejar)
opener=request.build_opener(handler)
resp=opener.open('http://httpbin.org/cookies')
for cookie in cookiejar:
    print(cookie)
#将会打印cookie
```

requests库

```
import requests
response=requests.get('http://www.baidu.com/')
#print(type(response.txt))
#未传入指定解码方式 可能会解码错误 需要手动解码
```

```
#print(response.txt)
#会发现类型是str 并且解码失败
print(type(response.content))
#直接从网络上抓取的数据所以是bytes类型
print(response.content.decode('utf-8'))
#即解码成功
print(response.url)
#查看完整url地址
print(response.encoding)
#查看相应头部字符编码
print(response.status_code)
#查看响应码

import requests
params={
    'wd':'中国'
}
#以字典的形式传入
headers={
    'User-Agent':'---'
}
response=requests.get('http://www.baidu.com/s',params=params,headers=headers
)
with open('baidu.html','w',encoding='utf-8') as fp:
    fp.write(response.content.decode('utf-8')
print(response.url)


#多线程爬虫
#进阶知识点
import threading
VALUE=0
gLock=threading.Lock()#上锁                    ↓就不会出错
def add_value():
    global VALUE #调入并修改全局变量
    gLock.acquire()
    for x in range(1000);#改为1000000时输出数据会发生错误
        VALUE +=1
    gLock.release()
    print('value:%d'%VALUE)
def main ():
    for x in range(2):
        t=threading.Thread(target=add_value)
        t.start()
if __name__ == '__main__':
    main()

Lock法
import threading
import rendom #随机
import time   #时间
gmoney=1000
gTotalTimes=10
gTimes=0
gLock=threading.Lock()#上锁
class Producer(threading.Thread:
    def run(self):
        global gMoney
        global gTimes
        while True:
            money=random.randint(100,1000)
            gLock.acquire()
            if gTimes>=gTotalTimes:
                gLock.release()#记得先解锁
                break#停止生产
            qMoney += money
            print('%s生产了%元钱，剩余%d元钱'%(threading.current_thread(),money,qMoney))
            gTimes+=1
            gLock.release()
            time.sleep(0.5)
class Consumer(threading.Thread:
    def run(self):
        global gMoney
        while True:
```

```
            money=random.randint(100,1000)
            gLock.acquire()
            if gMoney >= money:
                gMoney-=money
                print('%消费者消费了%d元钱，剩余%d元钱'%(threading.current_thread(),money,gMoney))
            else:
                if gTimes>=gTotalTimes:
                    gLock.release()
                    break
                print('%s消费者准备消费%d元钱，剩余%d元钱，不足！'%(threading.current_thread(),money,gMoney))
                gLock.release()#必须解锁
                time.sleep（0.5）
def main():
    for x in range(3):
        t=Consumer(name='消费者线程%d'%x)
        t.start()
    for x in range(5):
        t=Producer(name='生产者线程%d'%x)
        t.start()
if __name__ == '__main__':
    main()
```

Condition版模式

与Lock不同的是 L运行时一直在不断地加锁解锁而C是一次性的从而减少了cpu资源的占用

1、acquire： 上锁。

2．release： 解锁。

3．wait： 将当前线程处于等待状态，并且会释放锁。可以被其他线猩使用notify和notify-an函数唤。被唤醒后会继续等待上锁，上锁后继续执行下面的代码。

4、notify： 通知某个正在等待的线程，默认是第1个等待的

线程

5、notify-an： 通知所有正在等待的线程。 notify和notify-an不会放锁。并且需要在release之前调用。

```
import threading
import rendom #随机
import time  #时间
gMoney=1000
gTotalTimes=10
gTimes=0
gConditon=threading.Condition()#上锁
class Producer(threading.Thread):
    def run(self):
        global gMoney
        global gTimes
        while True:
            money=random.randint(100,1000)
            gConditon.acquire()
            if gTimes>=gTotalTimes:
                gConditon.release()
                break
            qMoney += money
            print('%s生产了%元钱，剩余%d元钱'%(threading.current_thread(),money,qMoney))
            gTimes+=1
            gConditon.notify_all()
            gConditon.release()
            time.sleep(0.5)
class Consumer(threading.Thread):
    def run(self):
        global gMoney
        while True:
            money=random.randint(100,1000)
            gConditon.acquire()
            while gMoney < money:
                if gTimes >=gTotalTimes:
                    gCondition.release()
                    return #注意
                ptiny('%s准备消费%d元钱，剩余%d元钱，不足！'%(threading.current_thread(),money,gMoney))
                gCondition.wait()
            gMoney-=money
            print('%s消费者消费了%d元钱，剩余%d元钱'%(threading.current_thread(),money,gMoney))
            gLock.release()#必须解锁
            time.sleep（0.5）
def main():
    for x in range(3):
        t=Consumer(name='消费者线程%d'%x)
        t.start()
```

```
    for x in range(5):
        t=Producer(name='生产者线程%d'%x)
        t.start()
if __name__ == '__main__':
    main()
```

Queue线程安全队列

1．初始化 Queue(maxsize)：创建一个先进先出的队列。
2．qsize()：返回队列的大小。
3．empty()：判断队列否为空。
4．full()：判断队列是否满了。
5．get()：从队列中取最后一个数据。先进先出。
6．put()：将一个数据放到队列中。

```
import requests
from lxml import etree
from urllib import request
import os
import re
def parse_page(url):
    headers={
    'User-Adgent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWekit/537.36 (KHTML, like Geoko) Chrome/62.0.3202.94 Safari/537.36'
    }
    response = resquests.get(url,headers=headers)
    text=reponse.text
    html=etree.HTML(text)
    imgs=html.xpath("//div[@class='page-content text-center']//img[@class!='gif'")
    for img in imgs:
        #print(etree.tostring(img)) 会发现能找都所有的标签了
        img_url=img.get('data-original')#data-original里是关于图片的链接
        #print(img_url)完成了提取的图片链接
        alt=img.get('alt')
        alt=re.sub(r'[\?? \.。\!！]','',alt)
        #'abc.txt'以os模块提取后缀名的文件格式
        #request.urlretrieve(img_url,'images/')
        suffix=os.path.splitext(img_url)[1]#os模块提取文件格式的后缀名
        #print(suffix)
        filename=alt+suffix
        #print(filename)
        request.urlretrieve(img_url,'images/'+filename)
def main():
    for x in range(1,101):
        url='http://www.doutula.com/photo/list/?page=%d'%x
        parse_page(url)
        #break 中断第一次爬取的数据
if __name__ == '__main__':
    main()
```

————完毕————

#接下来是以多线程方式去爬取图片

```python
import requests
from lxml import etree
from urllib import request
import os
import re
from queue import Queue
import threading
class Procuder(threading.Thread):
    headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36'
}
    def __init__(self, page_queue, img_queue, *args, **kwargs):
        super(Procuder, self).__init__(*args, **kwargs)
        self.page_queue = page_queue
        self.img_queue = img_queue
    def run(self):
        while True:
            if self.page_queue.empty():
                break
            url = self.page_queue.get()
            self.parse_page(url)
    def parse_page(self,url):
        response = requests.get(url, headers=self.headers)
        text = response.text
        html = etree.HTML(text)
        imgs = html.xpath("//div[@class='page-content text-center']//img[@class!='gif']")
        for img in imgs:
            img_url = img.get('data-original')
            alt = img.get('alt')
            alt = re.sub(r'[\?？\.。!！\*]', ' ', alt)
            suffix = os.path.splitext(img_url)[1]
            filename = alt + suffix
            self.img_queue.put((img_url, filename))
class Consumer(threading.Thread):
    def __init__(self, page__queue, img_queue, *args, **kwargs):
        super(Consumer, self).__init__(*args, **kwargs)
        self.page_queue = page__queue
        self.img_queue = img_queue
    def run(self):
        while True:
            if self.img_queue.empty() and self.page_queue.empty():
                break
            img_url,filename = self.img_queue.get()
            request.urlretrieve(img_url,'images/' + filename)
            print(filename + "下载完成！")
def main():
    page_queue = Queue(100)
    img_queue = Queue(1000)
    for x in range(1, 101):
        url = 'http://www.doutula.com/photo/list/?page=%d' % x
        page_queue.put(url)
    for x in range(5):
        t = Procuder(page_queue,img_queue)
        t.start()
    for x in range(5):
        t = Consumer(page_queue,img_queue)
if __name__ == '__main__':
    main()
```