



lining0806 / PythonSpiderNotes

Watch 79 Star 580 Fork 395

Code Issues 0 Pull requests 0 Projects 0 Insights

Python入门网络爬虫之精华版

python zhihu captcha wechat scrapy selenium cookie

26 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Find file Clone or download

lining0806 test		Latest commit a077e25 on Dec 25, 2016
Captcha1	test	6 months ago
NewsSpider	test	6 months ago
QunarSpider	test	6 months ago
Spider_Java	test	6 months ago
Spider_Python	test	6 months ago
WechatSearchProjects	test	6 months ago
ZhihuSpider	test	6 months ago
ReadMe.md	test	5 months ago

ReadMe.md

## Python入门网络爬虫之精华版

Python学习网络爬虫主要分3个大的版块：抓取，分析，存储

另外，比较常用的爬虫框架Scrapy，这里最后也详细介绍一下。

首先列举一下本人总结的相关文章，这些覆盖了入门网络爬虫需要的基本概念和技巧：[宁哥的小站-网络爬虫](#)

当我们在浏览器中输入一个url后回车，后台会发生什么？比如说你输入<http://www.lining0806.com/>，你就会看到宁哥的小站首页。

简单来说这段过程发生了以下四个步骤：

- 查找域名对应的IP地址。
- 向IP对应的服务器发送请求。
- 服务器响应请求，发回网页内容。
- 浏览器解析网页内容。

网络爬虫要做的，简单来说，就是实现浏览器的功能。通过指定url，直接返回给用户所需要的数据，而不需要一步步人工去操纵浏览器获取。

# 抓取

这一步，你要明确要得到的内容是什么？是HTML源码，还是Json格式的字符串等。

## 1. 最基本的抓取

抓取大多数情况属于get请求，即直接从对方服务器上获取数据。

首先，Python中自带urllib及urllib2这两个模块，基本上能满足一般的页面抓取。另外，[requests](#)也是非常有用的包，与此类似的，还有[httplib2](#)等等。

```
Requests:
    import requests
    response = requests.get(url)
    content = requests.get(url).content
    print "response headers:", response.headers
    print "content:", content
Urllib2:
    import urllib2
    response = urllib2.urlopen(url)
    content = urllib2.urlopen(url).read()
    print "response headers:", response.headers
    print "content:", content
HttpLib2:
    import httpLib2
    http = httpLib2.Http()
    response_headers, content = http.request(url, 'GET')
    print "response headers:", response_headers
    print "content:", content
```

此外，对于带有查询字段的url，get请求一般会将来请求的数据附在url之后，以?分割url和传输数据，多个参数用&连接。

```
data = {'data1': 'XXXXX', 'data2': 'XXXXX'}
Requests: data为dict, json
    import requests
    response = requests.get(url=url, params=data)
Urllib2: data为string
    import urllib, urllib2
    data = urllib.urlencode(data)
    full_url = url+'?' + data
    response = urllib2.urlopen(full_url)
```

相关参考：[网易新闻排行榜抓取回顾](#)

参考项目：[网络爬虫之最基本的爬虫：爬取网易新闻排行榜](#)

## 2. 对于登陆情况的处理

### 2.1 使用表单登陆

这种情况属于post请求，即先向服务器发送表单数据，服务器再将返回的cookie存入本地。

```
data = {'data1': 'XXXXX', 'data2': 'XXXXX'}
Requests: data为dict, json
    import requests
    response = requests.post(url=url, data=data)
Urllib2: data为string
    import urllib, urllib2
    data = urllib.urlencode(data)
    req = urllib2.Request(url=url, data=data)
    response = urllib2.urlopen(req)
```

### 2.2 使用cookie登陆

使用cookie登陆，服务器会认为你是一个已登陆的用户，所以就会返回给你一个已登陆的内容。因此，需要验证码的情况可以使用带验证码登陆的cookie解决。

```
import requests
requests_session = requests.session()
response = requests_session.post(url=url_login, data=data)
```

若存在验证码，此时采用response = requests\_session.post(url=url\_login, data=data)是不行的，做法应该如下：

```
response_captcha = requests_session.get(url=url_login, cookies=cookies)
response1 = requests.get(url_login) # 未登陆
response2 = requests_session.get(url_login) # 已登陆，因为之前拿到了Response Cookie!
response3 = requests_session.get(url_results) # 已登陆，因为之前拿到了Response Cookie!
```

相关参考：[网络爬虫-验证码登陆](#)

参考项目：[网络爬虫之用户名密码及验证码登陆：爬取知乎网站](#)

### 3. 对于反爬虫机制的处理

#### 3.1 使用代理

适用情况：限制IP地址情况，也可解决由于“频繁点击”而需要输入验证码登陆的情况。

这种情况最好的办法就是维护一个代理IP池，网上有很多免费的代理IP，良莠不齐，可以通过筛选找到能用的。对于“频繁点击”的情况，我们还可以通过限制爬虫访问网站的频率来避免被网站禁掉。

```
proxies = {'http': 'http://XX.XX.XX.XX:XXXX'}
Requests:
    import requests
    response = requests.get(url=url, proxies=proxies)
Urllib2:
    import urllib2
    proxy_support = urllib2.ProxyHandler(proxies)
    opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)
    urllib2.install_opener(opener) # 安装opener，此后调用urlopen()时都会使用安装过的opener对象
    response = urllib2.urlopen(url)
```

#### 3.2 时间设置

适用情况：限制频率情况。

Requests，Urllib2都可以使用time库的sleep()函数：

```
import time
time.sleep(1)
```

#### 3.3 伪装成浏览器，或者反“反盗链”

有些网站会检查你是不是真的浏览器访问，还是机器自动访问的。这种情况，加上User-Agent，表明你是浏览器访问即可。有时还会检查是否带Referer信息还会检查你的Referer是否合法，一般再加上Referer。

```
headers = {'User-Agent': 'XXXXX'} # 伪装成浏览器访问，适用于拒绝爬虫的网站
headers = {'Referer': 'XXXXX'}
headers = {'User-Agent': 'XXXXX', 'Referer': 'XXXXX'}
Requests:
    response = requests.get(url=url, headers=headers)
Urllib2:
    import urllib, urllib2
    req = urllib2.Request(url=url, headers=headers)
    response = urllib2.urlopen(req)
```

## 4. 对于断线重连

不多说。

```
def multi_session(session, *arg):
    retryTimes = 20
    while retryTimes>0:
        try:
            return session.post(*arg)
        except:
            print '.',
            retryTimes -= 1
```

或者

```
def multi_open(opener, *arg):
    retryTimes = 20
    while retryTimes>0:
        try:
            return opener.open(*arg)
        except:
            print '.',
            retryTimes -= 1
```

这样我们就可以使用multi\_session或multi\_open对爬虫抓取的session或opener进行保持。

## 5. 多进程抓取

这里针对[华尔街见闻](#)进行并行抓取的实验对比：[Python多进程抓取](#) 与 [Java单线程和多线程抓取](#)

相关参考：[关于Python和Java的多进程多线程计算方法对比](#)

## 6. 对于Ajax请求的处理

对于“加载更多”情况，使用Ajax来传输很多数据。

它的工作原理是：从网页的url加载网页的源代码之后，会在浏览器里执行JavaScript程序。这些程序会加载更多的内容，“填充”到网页里。这就是为什么如果你直接去爬网页本身的url，你会找不到页面的实际内容。

这里，若使用Google Chrome分析“请求”对应的链接(方法：右键→审查元素→Network→清空，点击“加载更多”，出现对应的GET链接寻找Type为text/html的，点击，查看get参数或者复制Request URL)，循环过程。

- 如果“请求”之前有页面，依据上一步的网址进行分析推导第1页。以此类推，抓取抓Ajax地址的数据。
- 对返回的json格式数据(str)进行正则匹配。json格式数据中，需从'\uxxxx'形式的unicode\_escape编码转换成u'\uxxxx'的unicode编码。

## 7. 自动化测试工具Selenium

Selenium是一款自动化测试工具。它能实现操纵浏览器，包括字符填充、鼠标点击、获取元素、页面切换等一系列操作。总之，凡是浏览器能做的事，Selenium都能够做到。

这里列出在给定城市列表后，使用selenium来动态抓取[去哪儿网](#)的票价信息的代码。

参考项目：[网络爬虫之Selenium使用代理登陆：爬取去哪儿网站](#)

## 8. 验证码识别

对于网站有验证码的情况，我们有三种办法：

- 使用代理，更新IP。

- 使用cookie登陆。
- 验证码识别。

使用代理和使用cookie登陆之前已经讲过，下面讲一下验证码识别。

可以利用开源的Tesseract-OCR系统进行验证码图片的下载及识别，将识别的字符传到爬虫系统进行模拟登陆。当然也可以将验证码图片上传到打码平台上进行识别。如果不成功，可以再次更新验证码识别，直到成功为止。

参考项目：[验证码识别项目第一版：Captcha1](#)

爬取有两个需要注意的问题：

- 如何监控一系列网站的更新情况，也就是说，如何进行增量式爬取？
- 对于海量数据，如何实现分布式爬取？

## 分析

---

抓取之后就是对抓取的内容进行分析，你需要什么内容，就从中提炼出相关的内容来。

常见的分析工具有[正则表达式](#)，[BeautifulSoup](#)，[lxml](#)等等。

## 存储

---

分析出我们需要的内容之后，接下来就是存储了。

我们可以选择存入文本文件，也可以选择存入[MySQL](#)或[MongoDB](#)数据库等。

存储有两个需要注意的问题：

- 如何进行网页去重？
- 内容以什么形式存储？

## Scrapy

---

Scrapy是一个基于Twisted的开源的Python爬虫框架，在工业中应用非常广泛。

相关内容可以参考[基于Scrapy网络爬虫的搭建](#)，同时给出这篇文章介绍的[微信搜索](#)爬取的项目代码，给大家作为学习参考。

参考项目：[使用Scrapy或Requests递归抓取微信搜索结果](#)

