

```
scheme://host:port/path/?query-string=xxx#anchor
```

```
scheme://host:port/path/?query-string=xxx#anchor
```

数

数据采集和解析

通过上一个章节的讲解，我们已经了解到了开发一个爬虫需要做的工作以及一些常见的问题，下面我们给出一个爬虫开发相关技术的清单以及这些技术涉及到的标准库和第三方库，稍后我们会一一介绍这些内容。

1. 下载数据 - **urllib** / **requests** / **aiohttp**。
2. 解析数据 - **re** / **lxml** / **beautifulsoup4** / **pyquery**。
3. 缓存和持久化 - **pymysql** / **sqlalchemy** / **peewee** / **redis** / **pymongo**。
4. 生成数字签名 - **hashlib**。
5. 序列化和压缩 - **pickle** / **json** / **zlib**。
6. 调度器 - 多进程 (**multiprocessing**) / 多线程 (**threading**) 。

HTML页面

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home</title>
    <style type="text/css">
      /* 此处省略层叠样式表代码 */
    </style>
  </head>
  <body>
    <div class="wrapper">
      <header>
        <h1>Yoko's Kitchen</h1>
        <nav>
          <ul>
            <li><a href="" class="current">Home</a></li>
            <li><a href="">Classes</a></li>
            <li><a href="">Catering</a></li>
            <li><a href="">About</a></li>
            <li><a href="">Contact</a></li>
          </ul>
        </nav>
      </header>
      <section class="courses">
        <article>
          <figure>
            
            <figcaption>Bok Choi</figcaption>
          </figure>
          <hgroup>
            <h2>Japanese Vegetarian</h2>
            <h3>Five week course in London</h3>
          </hgroup>
          <p>A five week introduction to traditional Japanese vegetarian meals, teaching you a selection of rice and noodle dishes.</p>
        </article>
      </section>
    </div>
  </body>
</html>
```

```

</article>
<article>
  <figure>
    
    <figcaption>Teriyaki Sauce</figcaption>
  </figure>
  <hgroup>
    <h2>Sauces Masterclass</h2>
    <h3>One day workshop</h3>
  </hgroup>
  <p>An intensive one-day course looking at how to create the
most delicious sauces for use in a range of Japanese cookery.</p>
</article>
</section>
<aside>
  <section class="popular-recipes">
    <h2>Popular Recipes</h2>
    <a href="">Yakitori (grilled chicken)</a>
    <a href="">Tsukune (minced chicken patties)</a>
    <a href="">Okonomiyaki (savory pancakes)</a>
    <a href="">Mizutaki (chicken stew)</a>
  </section>
  <section class="contact-details">
    <h2>Contact</h2>
    <p>Yoko's Kitchen<br>
    27 Redchurch Street<br>
    Shoreditch<br>
    London E2 7DP</p>
  </section>
</aside>
<footer>
  &copy; 2011 Yoko's Kitchen
</footer>
</div>
<script>
  // 此处省略JavaScript代码
</script>
</body>
</html>

```

如果你对上面的代码并不感到陌生，那么你一定知道HTML页面通常由三部分构成，分别是用来承载内容的Tag（标签）、负责渲染页面的CSS（层叠样式表）以及控制交互式行为的JavaScript。通常，我们可以在浏览器的右键菜单中通过“查看网页源代码”的方式获取网页的代码并了解页面的结构；当然，我们也可以通过浏览器提供的开发人员工具来了解更多的信息。

使用requests获取页面

1. GET请求和POST请求。

```

python之使用request模块发送post和get请求
import requests
import json

#发送get请求并得到结果
# url = 'http://api.nnzhp.cn/api/user/stu_info?stu_name=小黑马' #请求接口
# req = requests.get(url)#发送请求
# print(req.text)#获取请求，得到的是json格式

```

```

# print(req.json())#获取请求，得到的是字典格式
# print(type(req.text))
# print(type(req.json()))

#发送post请求,注册接口
# url = 'http://api.nnzhp.cn/api/user/user_reg'
# data = {'username':'mpp0130','pwd':'Mp123456','cpwd':'Mp123456'}
# req = requests.post(url,data)#发送post请求，第一个参数是URL，第二个参数是请求数据
# print(req.json())

#入参是json
# url = 'http://api.nnzhp.cn/api/user/add_stu'
# data = {'name':'mapeiwei','grade':'Mp123456','phone':15601301234}
# req = requests.post(url,json=data)
# print(req.json())

#添加header
# url = 'http://api.nnzhp.cn/api/user/all_stu'
# header = {'Referer':'http://api.nnzhp.cn/'}
# res = requests.get(url,headers=header)
# print(res.json())

# 添加cookie
# url = 'http://api.nnzhp.cn/api/user/gold_add'
# data = {'stu_id':231,'gold':123}
# cookie = {'niuanyang':'7e4c46e5790ca7d5165eb32d0a895ab1'}
# req = requests.post(url,data,cookies=cookie)
# print(req.json())

#上传文件
# url = 'http://api.nnzhp.cn/api/file/file_upload'
# f = open(r'E:\besttest\te\python-mpp\day7\练习\11.jpg','rb')
# r = requests.post(url,files={'file':f})
# users_dic = r.json()
# print(users_dic)

# 下载文件
# url =
'http://www.besttest.cn/data/upload/201710/f_36b1c59ecf3b8ff5b0acaf2ea42bafef0.jpg'
# r = requests.get(url)
# print(r.status_code)#获取请求的状态码
# print(r.content)#获取返回结果的二进制格式
# fw = open('mpp.jpg','wb')
# fw.write(r.content)
# fw.close()

#把浏览器页面下载到本地    保存网页，可以理解为简单的爬虫工具
url='http://www.nnzhp.cn/archives/630'
r = requests.get(url)
f = open('nnzhp.html','wb')
f.write(r.content)
f.close()

```

2. URL参数和请求头。

一、发送请求

```
r = requests.get('https://api.github.com/events') # GET请求
r = requests.post('http://httpbin.org/post', data = {'key': 'value'}) # POST
请求
r = requests.put('http://httpbin.org/put', data = {'key': 'value'}) # PUT请求
r = requests.delete('http://httpbin.org/delete') # DELETE请求
r = requests.head('http://httpbin.org/get') # HEAD请求
r = requests.options('http://httpbin.org/get') # OPTIONS请求
type(r)
```

1

2

3

4

5

6

7

requests.models.Response

1

二、传递URL参数

URL传递参数的形式为: `httpbin.org/get?key=val`。但是手动的构造很麻烦,这是可以使用 `params` 参数来方便的构造带参数URL。

```
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.get("http://httpbin.org/get", params=payload)
print(r.url)
```

`http://httpbin.org/get?key1=value1&key2=value2`

同一个key可以有多个value

```
payload = {'key1': 'value1', 'key2': ['value2', 'value3']}
r = requests.get('http://httpbin.org/get', params=payload)
print(r.url)
```

`http://httpbin.org/get?key1=value1&key2=value2&key2=value3`

三、定制headers

只需要将一个dict传递给headers参数便可以定制headers

```
url = 'https://api.github.com/some/endpoint'
headers = {'user-agent': 'my-app/0.0.1'}
r = requests.get(url, headers=headers)
```

URL的组成和说明

URL是uniform Resource locator的简写,中文意思是统一资源定位符

`scheme://host:port/path/?query-string=xxx#anchor`

scheme: 代表的是访问的协议, 一般为http或者https以及ftp等

host: 主机名, 域名, 比如: `www.baidu.com`

port: 端口号, 当你访问一个网站时, 浏览器默认使用80端口

path: 查找路径, 比如: `https://mp.csdn.net/mdeditor/103399059`, 后面的 `/mdeditor/103399059` 就是查找路径

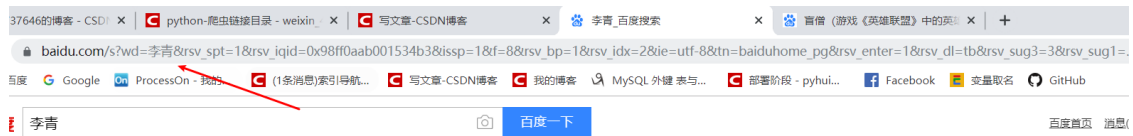
query-string: 查询字符串。比如: `www.baidu.com/s?wd=python`, 后面 `wd=python` 就是查询的字符串

anchor: 锚点, 一般都是前端做页面定位的

再浏览器中请求的url, 浏览器会对url进行一个编码。除了英文字符, 数字和部分符号外, 其他全部使用百分号%+十六进制码值进行编码

比如:

复制下来的url路径是这样的:



粘贴到文本是这样: <https://baike.baidu.com/item/%E7%9B%B2%E5%83%A7/6153769?fr=aladdin>

说明了浏览器对我们中文字符进行了编码

常用的请求方法

在http协议中,定义了八种请求方法。常用的是get和post

get请求: 一般情况下,只从服务器获取数据下来,并不会对服务器资料产生任何影响

post请求: 向服务器发送数据(登录)、上传文件等。会对服务器资源产生影响

注:以上是一般情况下会遵循使用原则,但是有的网站和服务器回了做反爬虫机制,不会按照常理出牌。

请求头常见参数

在http协议中,向服务器发送一个请求,数据分为三部分,第一个是把数据放在url中,第二个是把数据放在body中(post请求中),第三个是把数据放在head中。

user-Agent: 浏览器名称。请求网页的时候,服务器通过这个参数知道这个请求是由那个浏览器发送的

Referer: 表明当前这个请求是从那个url过来的。一般用在反爬虫技术,如果不是从指定页面过来,那么就不做对应的响应

Cookie: http协议是无状态的,也就是同一个人发送了两次请求,服务器没有能力知道这两个请求是否来自同一个人,一次这个时候使用cookie来做标识。一般都如果想要登录后才能访问的网站,那么就需要发送cookie信息了

常见响应状态码

200: 请求正常,服务器正常的返回数据

301: 永久重定向

302: 临时重定向

400: 请求的url在服务器上找不到

403: 服务器拒绝访问,权限不够

500: 服务器内部错误。可能是服务器出现了bug

3. 复杂的POST请求(文件上传)。

进入python安装根目录下面的Scripts目录执行pip install requests命令,等待安装成功之后,执行import requests命令来查看是否安装成功。

```
C:\Python27\Scripts>python
Python 2.7.14rc1 (v2.7.14rc1:c707893, Aug 27 2017, 00:09:00) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import requests
>>>
```

post方法参数

`requests.post(url,data=data,header=header,files=files)`

1

- data设置body数据

- header设置请求头

- files设置上传的文件

示例代码

该WebService接口使用的是multipart/form-data协议

```
import requests
```

```
import hashlib
```

```
import time
```

```
url = "http://test"
```

```
clientId = "1111"
```

```
clientKey = "2222"
```

```
timestamp = (str)(int(round(time.time()*1000)))
```

```

clientSecret = hashlib.sha256(clientId.encode("utf-8") +
clientKey.encode("utf-8") + timestamp.encode("utf-8"))
header =
{'clientId':clientId,'timestamp':timestamp,'clientSecret':clientSecret}
files = {'apk':open('D:\\test.apk','rb')}
data = {'enctype':'multipart/form-data','name':'wang'}
reponse = requests.post(url,data=data,header=header,files=files)
text = reponse.text
print (text)

```

4. 操作Cookie。

python爬虫使用Cookie的两种方法



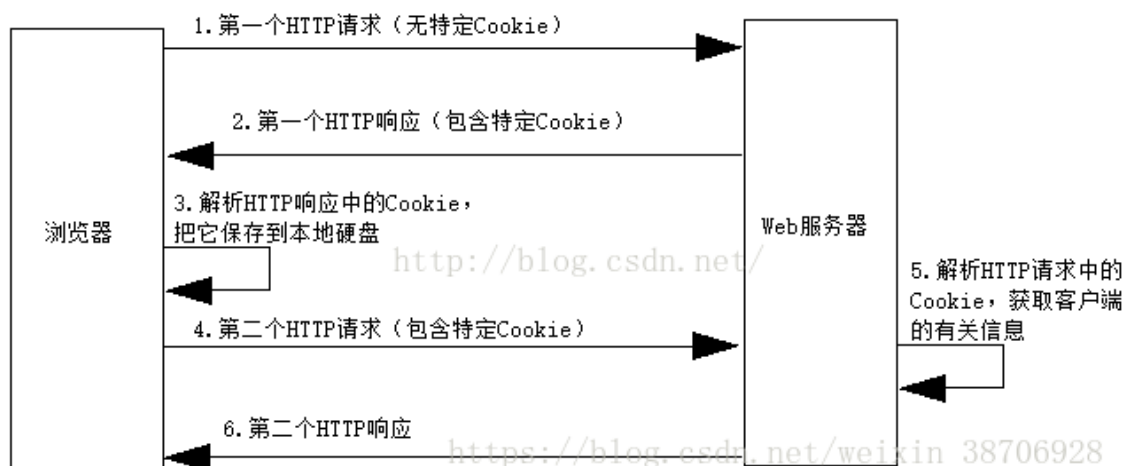
但是当我们登录以后，下载链接会显示出来，这样在爬虫的时候，可以把下载链接解析出来使用。



登录前后网页Headers-Request Headers显示的Cookie不同。下图为浏览器和Web服务器之间的交互，也显示了Cookie的信息。

Cookie的引文原意是“点心”，它是在客户端访问Web服务器时，服务器在客户端硬盘上存放的信息，好像是服务器发送给客户的“点心”。服务器可以根据Cookie来跟踪客户状态，这对于需要区别客户的场合（如电子商务）特别有用。

当客户端首次请求访问服务器时，服务器先在客户端存放包含该客户的相关信息的Cookie，以后客户端每次请求访问服务器时，都会在HTTP请求数据中包含Cookie，服务器解析HTTP请求中的Cookie，就能由此获得关于客户的相关信息。



因此我们使用登录以后的Cookie。自己编辑Cookie访问Web服务器。网络爬虫中Cookie的两种使用方式，代码中网页已经不存在，但是可以自己在其他网页中尝试。我们使用的是登录之后的Cookie。

1、直接将Cookie写在header头部

```
# coding:utf-8
import requests
from bs4 import BeautifulSoup
cookie = '''cisession=19dfd70a27ec0eecf1fe3fc2e48b7f91c7c83c60;CNZZDATA1000201968=1815846425-1478580135-https%253A%252F%252Fwww.baidu.com%252F%7C1483922031;Hm_lvt_f805f7762a9a237a0deac37015e9f6d9=1482722012,1483926313;Hm_lpvt_f805f7762a9a237a0deac37015e9f6d9=1483926368'''
header = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36',
    'Connection': 'keep-alive',
    'accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8'
    ,
    'Cookie': cookie}
url = 'https://kankandou.com/book/view/22353.html'
wbdata = requests.get(url,headers=header).text
soup = BeautifulSoup(wbdata,'lxml')
print(soup)
```

2、使用requests插入Cookie

```
# coding:utf-8
import requests
from bs4 import BeautifulSoup
cookie = {
    "cisession": "19dfd70a27ec0eecf1fe3fc2e48b7f91c7c83c60",
    "CNZZDATA100020196": "1815846425-1478580135-https%253A%252F%252Fwww.baidu.com%252F%7C1483922031",
    "Hm_lvt_f805f7762a9a237a0deac37015e9f6d9": "1482722012,1483926313",
    "Hm_lpvt_f805f7762a9a237a0deac37015e9f6d9": "1483926368"
}
url = 'https://kankandou.com/book/view/22353.html'
wbdata = requests.get(url,cookies=cookie).text
soup = BeautifulSoup(wbdata,'lxml')
print(soup)
```

这样我们就轻松的使用Cookie获取到了需要登录验证后才能浏览的网页和资源了。上面的Cookie是从自己浏览的网页中复制粘贴得到的。

5. 设置代理服务器。

```
urllib 方式
def user_proxy(proxy_addr, url):
    import urllib.request
    proxy = urllib.request.ProxyHandler({'http': proxy_addr})
    opener = urllib.request.build_opener(proxy, urllib.request.HTTPHandler)
    urllib.request.install_opener(opener)
```

```

data = urllib.request.urlopen(url).read().decode('utf-8')
return data
proxy_addr = "114.82.109.134:8118"
data = user_proxy(proxy_addr, "https://www.baidu.com")
print(data)
print(len(data))

```

requests 方式

免费代理的网站: <http://www.xicidaili.com/nn/>

代码部分:

```

import requests
proxy='124.243.226.18:8888'
#如果代理需要验证, 只需要在前面加上用户名密码, 如下所示
# proxy='username:password@124.243.226.18:8888'
proxies={
    'http': 'http://' + proxy,
    'https': 'https://' + proxy,
}
try:
    response=requests.get('http://httpbin.org/get',proxies=proxies)
    print(response.text)
except requests.exceptions.ConnectionError as e:
    print("Error",e.args)

```

输出:

```

{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Connection": "close",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.20.0"
  },
  "origin": "124.243.226.18",
  "url": "http://httpbin.org/get"
}

```

基于 selenium的代理设置:

```

from selenium import webdriver
proxy='124.243.226.18:8888'
option=webdriver.ChromeOptions()
option.add_argument('--proxy-server=http://' + proxy)
driver = webdriver.Chrome(options=option)
driver.get('http://httpbin.org/get')

```

一个代理 ip 池的例子

将西刺免费代理IP入 MySQL 数据库, 然后每次用的时候随机选取, 如果不可用的话, 再次随机选取, 直到选到为止。

说明: 关于requests的详细用法可以参考它的[官方文档](#)。

页面解析

几种解析方式的比较

解析方式	对应的模块	速度	使用难度	备注
正则表达式解析	re	快	困难	常用正则表达式 在线正则表达式测试
XPath解析	lxml	快	一般	需要安装C语言依赖库 唯一支持XML的解析器
CSS选择器解析	bs4 / pyquery	不确定	简单	

说明：BeautifulSoup可选的解析器包括：Python标准库（html.parser）、lxml的HTML解析器、lxml的XML解析器和html5lib。

使用正则表达式解析页面

如果你对正则表达式没有任何的概念，那么推荐先阅读[《正则表达式30分钟入门教程》](#)，然后再阅读我们之前讲解在Python中如何使用正则表达式一文。

XPath解析和lxml

XPath是在XML文档中查找信息的一种语法，它使用路径表达式来选取XML文档中的节点或者节点集。这里所说的XPath节点包括元素、属性、文本、命名空间、处理指令、注释、根节点等。

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

对于上面的XML文件，我们可以用如下所示的XPath语法获取文档中的节点。

路径表达式	结果
bookstore	选取 bookstore 元素的所有子节点。
/bookstore	选取根元素 bookstore。注释：假如路径起始于正斜杠(/)，则此路径始终代表到某元素的绝对路径！
bookstore/book	选取属于 bookstore 的子元素的所有 book 元素。
//book	选取所有 book 子元素，而不管它们在文档中的位置。
bookstore//book	选择属于 bookstore 元素的后代的所有 book 元素，而不管它们位于 bookstore 之下的什么位置。
//@lang	选取名为 lang 的所有属性。

在使用XPath语法时，还可以使用XPath中的谓词。

路径表达式	结果
/bookstore/book[1]	选取属于 bookstore 子元素的第一个 book 元素。
/bookstore/book[last()]	选取属于 bookstore 子元素的最后一个 book 元素。
/bookstore/book[last()-1]	选取属于 bookstore 子元素的倒数第二个 book 元素。
/bookstore/book[position()<3]	选取最前面的两个属于 bookstore 元素的子元素的 book 元素。
//title[@lang]	选取所有拥有名为 lang 的属性的 title 元素。
//title[@lang='eng']	选取所有 title 元素，且这些元素拥有值为 eng 的 lang 属性。
/bookstore/book[price>35.00]	选取 bookstore 元素的所有 book 元素，且其中的 price 元素的值须大于 35.00。
/bookstore/book[price>35.00]/title	选取 bookstore 元素中的 book 元素的所有 title 元素，且其中的 price 元素的值须大于 35.00。

XPath还支持通配符用法，如下所示。

路径表达式	结果
/bookstore/*	选取 bookstore 元素的所有子元素。
//*	选取文档中的所有元素。
//title[@*]	选取所有带有属性的 title 元素。

如果要选取多个节点，可以使用如下所示的方法。

路径表达式	结果
//book/title //book/price	选取 book 元素的所有 title 和 price 元素。
//title //price	选取文档中的所有 title 和 price 元素。
/bookstore/book/title //price	选取属于 bookstore 元素的 book 元素的所有 title 元素，以及文档中所有的 price 元素。

说明：上面的例子来自于菜鸟教程网站上[XPath教程](#)，有兴趣的读者可以自行阅读原文。

当然，如果不理解或者不太熟悉XPath语法，可以在Chrome浏览器中按照如下所示的方法查看元素的XPath语法。

豆瓣 读书 电影 音乐 同城 小组 阅读 FM 时间 市集 更多


豆瓣电影

搜索电影、电视剧、综艺、影人

影讯&购票 选电影 电视剧 排行榜 分类 影评 2017年度榜单 2017观影报告


span | 517.16×30 50




肖申克的救赎 The Shawshank Redemption (1994)



导演: 弗兰克·德拉邦特
编剧: 弗兰克·德拉邦特 / 斯蒂芬·金
主演: 蒂姆·罗宾斯 / 摩根·弗里曼 / 鲍勃·冈顿 / 威廉姆·赛德勒 / 克兰西·布朗 / 更多...
类型: 剧情 / 犯罪
制片国家/地区: 美国

豆瓣评分

9.6  1041463人评价

5星  82.9%
4星  15.1%
3星  1.8%

在吗
优酷
搜捕
受

Elements Console Sources Network Performance Memory Application Security Audits

div id=wrapper
div id=content
div class=top250
div id=dale_movie_subject_top_icon ad-status=loaded
h1
span property=v:itemreviewed 肖申克的救赎 The Shawshank
span class=year (1994)
div class=grid-16-8 clearfix
div id=footer
script type=text/javascript src=https://img3.doubanio.com/

html.ua-mac.ua-webkit body div#wrapper div#content h1 span

Edit text
Edit as HTML
Delete element
Copy
Hide element
Break on
Expand recursively
Collapse children
Cut element
Copy element
Paste element
Copy outerHTML
Copy selector
Copy XPath

BeautifulSoup的使用

BeautifulSoup是一个可以从HTML或XML文件中提取数据的Python库。它能够通过你喜欢的转换器实现惯用的文档导航、查找、修改文档的方式。

1. 遍历文档树

- 获取标签
- 获取标签属性
- 获取标签内容
- 获取子(孙)节点
- 获取父节点/祖先节点
- 获取兄弟节点

2. 搜索树节点

- find / find_all
- select_one / select

说明: 更多内容可以参考BeautifulSoup的[官方文档](#)。

PyQuery的使用

pyquery相当于jQuery的Python实现, 可以用于解析HTML网页。

实例 - 获取知乎发现上的问题链接

```
from urllib.parse import urljoin

import re
import requests
```

```

from bs4 import BeautifulSoup

def main():
    headers = {'user-agent': 'Baiduspider'}
    proxies = {
        'http': 'http://122.114.31.177:808'
    }
    base_url = 'https://www.zhihu.com/'
    seed_url = urljoin(base_url, 'explore')
    resp = requests.get(seed_url,
                        headers=headers,
                        proxies=proxies)
    soup = BeautifulSoup(resp.text, 'lxml')
    href_regex = re.compile(r'^/question')
    link_set = set()
    for a_tag in soup.find_all('a', {'href': href_regex}):
        if 'href' in a_tag.attrs:
            href = a_tag.attrs['href']
            full_url = urljoin(base_url, href)
            link_set.add(full_url)
    print('Total %d question pages found.' % len(link_set))

if __name__ == '__main__':
    main()

```