Python 爬虫（计算机网络）

4974

如何入门 Python 爬虫？

6 条评论 分享

95 个回答

按投票排序

谢科，用python分布式地爬过豆瓣/Twitter Search

GuestChen、鼠熊、刘文佳 等人赞同

“入门”是良好的动机，但是可能作用缓慢。如果你手里或者脑子里有一个项目，那么实践起来你会被目标驱动，而不会像学习模块一样慢慢学习。

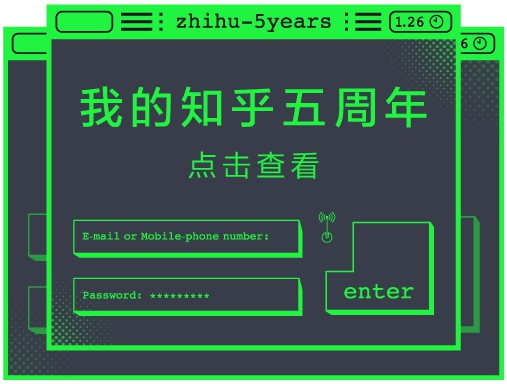
另外如果说知识体系里的每一个知识点是图里的点，依赖关系是边的话，那么这个图一定不是一个有向无环图。因为学习A的经验可以帮助你学习B。因此，你不需要学习怎么样“入门”，因为这样的“入门”点根本不存在！你需要学习的是怎么样做一个比较大的东西，在这个过程中，你会很快地学会需要学会的东西的。当然，你可以争论说需要先懂python，不然怎么学会python做爬虫呢？但是事实上，你完全可以在做这个爬虫的过程中学习python :D

看到前面很多答案都讲的“术”——用什么软件怎么爬，那我就讲讲“道”和“术”吧——爬虫怎么工作以及怎么在python实现。

先长话短说summarize一下：
你需要学习

1. 基本的爬虫工作原理
2. 基本的http抓取工具，scrapy
3. Bloom Filter: Bloom Filters by Example
4. 如果需要大规模网页抓取，你需要学习分布式爬虫的概念。其实没那么玄乎，你只要学会怎样维护一个所有集群机器能够有效分享的分布式队列就好。最简单的实现是python-rq: github.com/nvie/rq
5. rq和Scrapy的结合：darkrho/scrapy-redis · GitHub

关注问题 19754 人关注该问题



相关问题

换一换

有哪些「神奇」的数据获取方式？ 46 个回答

会一门脚本语言，学 bash 就显得浪费时间？ 38 个回答

Python 在网页爬虫、数据挖掘、机器学习和自然语言处理领域的应用情况如何？ 14 个回答

能利用爬虫技术做到哪些很酷很有趣很有用的事情？ 338 个回答

业余时间如何学数据分析？ 36 个回答

6. 后续处理，网页析取(grangier/python-goose · GitHub)，存储(Mongodb)

以下是短话长说：

4974

说说当初写的一个集群爬下整个豆瓣的经验吧。

1）首先你要明白爬虫怎样工作。
想象你是一只蜘蛛，现在你被放到了互联“网”上。那么，你需要把所有的网页都看一遍。怎么办呢？没问题呀，你就随便从某个地方开始，比如说人民日报的首页，这个叫initial pages，用$表示吧。

在人民日报的首页，你看到那个页面引向的各种链接。于是你很开心地从爬到了“国内新闻”那个页面。太好了，这样你就已经爬完了俩页面（首页和国内新闻）！暂且不用管爬下来的页面怎么处理的，你就想象你把这个页面完完整整抄成了个html放到了你身上。

突然你发现， 在国内新闻这个页面上，有一个链接链回“首页”。作为一只聪明的蜘蛛，你肯定知道你不用爬回去的吧，因为你已经看过了啊。所以，你需要用你的脑子，存下你已经看过的页面地址。这样，每次看到一个可能需要爬的新链接，你就先查查你脑子里是不是已经去过这个页面地址。如果去过，那就别去了。

好的，理论上如果所有的页面可以从initial page达到的话，那么可以证明你一定可以爬完所有的网页。

那么在python里怎么实现呢？
很简单

```
import Queue

initial_page = "http://www.renminribao.com"

url_queue = Queue.Queue()
seen = set()

seen.insert(initial_page)
url_queue.put(initial_page)

while(True): #一直进行直到海枯石烂
```

```
    if url_queue.size()>0:
        current_url = url_queue.get()     #拿出队例中第
        store(current_url)                #把这个url代
        for next_url in extract_u         urrent_url): #
            if next_url not in se
                seen.put(next_url
                url_queue.put(nex         )
    else:
        break
```

写得已经很伪代码了。

所有的爬虫的backbone都在这里，下面分析一下为什么爬虫事实上是个非常复杂的东西——搜索引擎公司通常有一整个团队来维护和开发。

2）效率
如果你直接加工一下上面的代码直接运行的话，你需要一整年才能爬下整个豆瓣的内容。更别说Google这样的搜索引擎需要爬下全网的内容了。

问题出在哪呢？需要爬的网页实在太多太多了，而上面的代码太慢太慢了。设想全网有N个网站，那么分析一下判重的复杂度就是N*log(N)，因为所有网页要遍历一次，而每次判重用set的话需要log(N)的复杂度。OK，OK，我知道python的set实现是hash——不过这样还是太慢了，至少内存使用效率不高。

通常的判重做法是怎样呢？**Bloom Filter**. 简单讲它仍然是一种hash的方法，但是它的特点是，它可以使用固定的内存（不随url的数量而增长）以O(1)的效率判定url是否已经在set中。可惜天下没有白吃的午餐，它的唯一问题在于，如果这个url不在set中，BF可以100%确定这个url没有看过。但是如果这个url在set中，它会告诉你：这个url应该已经出现过，不过我有2%的不确定性。注意这里的不确定性在你分配的内存足够大的时候，可以变得很小很少。一个简单的教程:Bloom Filters by Example

注意到这个特点，url如果被看过，那么可能以小概率重复看一看（没关系，多看看不会累死）。但是如果没被看过，一定会被看一下（这个很重要，不然我们就要漏掉一些网页了！）。[IMPORTANT: 此段有问题，请暂时略过]

好，现在已经接近处理判重最快的方法了。另外一个瓶颈——你只有一台机器。不管你的带宽有多大，只要你的机器下载网页的速度是瓶颈的话，那么你只有加快这个速度。用一台机子不够的话——用很多台吧！当然，我们假设每台机子都已经进了最大的效率——使用多线程（python用协程，多进程吧）。

3）集群化抓取
爬取豆瓣的时候，我总共用了100多台机器昼夜不停地运行了一个月。想象如果只用一台机子你就得运行100个月了…

那么，假设你现在有100台机器可以用，怎么用python实现一个分布式的爬取算法呢？

我们把这100台中的99台运算能力较小的机器叫作slave，另外一台较大的机器叫作master，那么回顾上面代码中的url_queue，如果我们能把这个queue放到这台master机器上，所有的slave都可以通过网络跟master联通，每当一个slave完成下载一个网页，就向master请求一个新的网页来抓取。而每次slave新抓到一个网页，就把这个网页上所有的链接送到master的queue里去。同样，bloom filter也放到master上，但是现在master只发送确定没有被访问过的url给slave。Bloom Filter放到master的内存里，而被访问过的url放到运行在master上的Redis里，这样保证所有操作都是O(1)。（至少平摊是O(1)，Redis的访问效率见:LINSERT – Redis ）

考虑如何用python实现：
在各台slave上装好scrapy，那么各台机子就变成了一台有抓取能力的slave，在master上装好Redis和rq用作分布式队列。

代码于是写成

```
#slave.py

current_url = request_from_master()
to_send = []
for next_url in extract_urls(current_url):
    to_send.append(next_url)

store(current_url);
send_to_master(to_send)
```

```
#master.py
distributed_queue = DistributedQueue()
bf = BloomFilter()

initial_pages = "www.renmingribao

while(True):
    if request == 'GET':
        if distributed_queue.size()>0:
            send(distributed_queue.get())
        else:
            break
    elif request == 'POST':
        bf.put(request.url)
```


好的，其实你能想到，有人已经给你写好了你需要的：darkrho/scrapy-redis · GitHub

4）展望及后处理
虽然上面用很多“简单”，但是真正要实现一个商业规模可用的爬虫并不是一件容易的事。上面的代码用来爬一个整体的网站几乎没有太大的问题。

但是如果附加上你需要这些后续处理，比如

1. 有效地存储（数据库应该怎样安排）
2. 有效地判重（这里指网页判重，咱可不想把人民日报和抄袭它的大民日报都爬一遍）
3. 有效地信息抽取（比如怎么样抽取出网页上所有的地址抽取出来，“朝阳区奋进路中华道”），搜索引擎通常不需要存储所有的信息，比如图片我存来干嘛...
4. 及时更新（预测这个网页多久会更新一次）

如你所想，这里每一个点都可以供很多研究者十数年的研究。
虽然如此，
“路漫漫其修远兮,吾将上下而求索”。

所以，不要问怎么入门，直接上路就好了：）

562

知乎用户

曹卓文、陈昂、阿奔 等人赞同

4974

————————最重要的话写在前面————————

0、**新手/喜欢练习/欢迎交流/邀请/我是看着这个问题下面的答案学习的**

1、带着一个目的来学爬虫。#我的目的：…所以我来写这个回答了。

2、**不要怂就是干**！系统学习固然好，直接一个项目出来效果更加简单粗暴！（不过自己现在的水平写出来都是流水一般的面向过程的代码，代码的重复部分太多，正在回过头去学习面向对象编程，学习类和方法的使用。不过**我还是坚定地认为入门的时候应该直接简单粗暴地实践一个项目**）

3、**哪里不会搜哪里！哪里报错改哪里！**相信我你遇到的99%的问题都能从网上找到相似的问题，**你需要做的就是写代码！搜问题！调BUG！你搜不到解决办法的情况下，80%的情况是你搜索的姿势不对，另外20%可能需要你自己动动脑子，换个思路去做。**

举个印象最深的例子。

我在统计知乎回答voters的具体情况的时候（后面会介绍）发现知乎的数据是这样发送的。

zhihu.com/answer/152197...

什么鬼（摔）。

等到我辛辛苦苦用正则把里面的信息提出来的时候发现我得到的数据是这样的…

	A	B	C	D	E
1	用户	赞同	感谢	提问	回答
2	smallE wang\	0	0	0	0
3	\u5239\u843d\u661f\u5149\	0	1	1	5
4	\u5b9b\u9675\u4eba\	0	0	0	0
5	\u9093\u7eea\u6d9b\	4	1	1	2
6	4e11\u5c31\u8be5\u591a\u8b	0	0	1	0
7	\u5b59\u5fd7\u9e4f\	81	18	86	167
8	\u80e1\u6c49\u4e09\	0	0	1	0
9	\u674e\u5609\u6b23\	12	4	4	6
10	Mata Zeng\	164	19	14	8
11	\u674e\u6587\u660a\	0	0	0	0
12	\u666f\u745f\	2	0	1	5
13	\u7f57\u4f73\u5f6c\	51	7	1	12
14	\u674e\u6728\u6728\	1	0	0	2
15	\u66f2\u5c0f\u82b1\	62	12	0	6
16	\u7389\u77f3\u5934\	0	0	6	6
17	'a7a\u660e\u516e\u6eaf\u6d4	253	38	3	17

我的内心是崩溃的……

问题很明显是编码问题……用户那一列全部是unicode编码……转成中文就好了嘛……
我刚开始也是这么想的…当我尝试了各种encode和decode…以后整个人都不好了。
大概是这样的。我用Shell演示一下…应该能看懂。

但是我的字符串是自动获取的啊，怎么可能挨着用 u' '赋值……
于是我开始了漫长的搜索之路……在看了无数篇重复度高于百分之80的关于编码的文章后，在我都快要放弃的时候…看到了这个…水木社区-源于清华的高知社群　你能理解我当时内心的酸爽吗…

大概就是这样。
所以**我遇到的问题已经很奇葩了依然还是有解决办法的嘛。**
Windows下面编码各种混乱，系统编码，编程时编辑器的编码，抓取网页以后网页的编码，Python2的编码，Python3的编码……新人真的很容易搞昏头。
例子不多言了。后面还有很多。

————————正文1：我的爬虫入门，不谈学习，只聊项目（代码已贴）————————
前面说到学爬虫需要一个目标。那我的目标是什么呢？听我慢慢讲。
前些日子我回答了一个问题高考后暑假应该做什么事？ - 生活

我回答这个问题的时候呢关注人数也就才刚刚过百，我的赞数也涨的很慢…
可是突然有一天呢，我发现突然就出现了一个300赞的回答…
当时那个问题的关注似乎还不到300。我不得其解…但是我看了看那个回答的赞同和答主的主页。
大概是这样的：

然后我隐隐觉得…可能会是刷赞？当然我们不能恶意地去揣测别人，要拿数据说话，毕竟知乎现在的三零真实用户还是蛮多的，不一定都是水军的小号。

于是我一个从来没有学过爬虫的人就开始学爬虫了…然而并不一帆风顺。首先是知乎显示“等人赞同”的方式做了修改，参见如何评价知乎新的「某某等人赞同」显示方[illegible] - 如何评价 X。
其次我刚开始的时候不会维持登陆…每次抓[illegible]的数据里都有很多的“知乎用户”（也就是未登录状态下抓[illegible]成功）。
为了行文的连贯我跳过中间学习时做的几个小爬虫…直接放我做成功的结果吧。

选取的样本回答依次为：
@段晓晨高考后暑假应该做什么事？ - 段晓晨的回答
@EdgeRunner高考后暑假应该做什么事？ - EdgeRunner 的回答
@孔鲤高考后暑假应该做什么事？ - 孔鲤的回答
@Emily L能利用爬虫技术做到哪些很酷很有趣很有用的事情？ - Emily L 的回答
@chenqin为什么 2015 年初，上海有卫计委官员呼吁大家生二胎？ - chenqin 的回答
感兴趣的可以下载数据
getvoters.xls_免费高速下载
getvoters2.xls_免费高速下载
getvoters3.xls_免费高速下载
getvoters4.xls_免费高速下载
getvoters5.xls_免费高速下载
结论就是……没有结论。
话说我回答的那个三零用户比例也好高啊……我真的没有刷赞我保证！（话说我的赞里面要是有水军的话我会很伤心的……我一直以为是我写的好他们才赞我的QAQ）

到这里第一个项目就结束了…
这个我暂时不贴代码…代码不完善…还得有点小修改。两天内放上来。

——来贴代码——
loveQt/Zhihu_voters · GitHub
使用前请填写config.ini文件，cookie不
4974
依然不完善。是这样的，知乎在获取“等人赞同”的时候有一个很畸形的地方在于……答案的id很畸形。
比如我现在这个答案。
zhihu.com/question/2089...
当我点击“等人赞同”的时候。抓包得到请求地址。我用的是Firefox的Firebug

这个地址是这样的：
zhihu.com/answer/152992...
如果你继续往下拉，知乎会自动加载更多用户，你会得到形如这样的地址：
zhihu.com/answer/152992...
分析这个地址的构成就会发现
/answer/这里应该是这个回答的唯一id，而这个id显然与上面的/question/20899988/answer/49749466是不一样的，我抓了好多个回答，结论应该是没有规律的，知乎应该是给每个问题一个id，每个问题下面的回答一个id，但是只有点赞的时候这个回答才会得到它关于voters的id……

所以我没办法实现完全的自动化…你如果想爬指定的回答，似乎得先手动抓包了 QAQ
抓包的示意如上，打开网络面板，点击“等人赞同”，找到地址中的数字就可以了。

如果你会抓包了请接着看下去。

代码的下载地址在上面的github。Python版本为2.7，希望你们会用pip安装依赖的库。
简单说几个方面。

1、知乎的登陆。我模仿了 @egrcc 和 @rsdream 的项目，使用了requests.session。

```
def login():
    cf = ConfigParser.ConfigParser()
    cf.read("config.ini")
    cookies = cf._sections['cookies']

    email = cf.get("info", "email")
    password = cf.get("info", "password")
    cookies = dict(cookies)
    global s
    s = requests.session()
    login_data = {"email": email, "password": passwor
    header = {
    'User-Agent': "Mozilla/5.0 (X11; Ubuntu; Linux x8
    'Host': "www.zhihu.com",
    'Referer': "http://www.zhihu.com/",
    'X-Requested-With': "XMLHttpRequest"
        }
    r = s.post(Login_url, data=login_data, headers=he
```

在实现了登陆之后，只要使用s.get(url)得到的页面都是登陆成功的状态。
//**注意，不登陆或者登陆不成功的情况下也可以爬取数据。点赞、感谢、提问、回答的数值都能正常获取，但是会出现部分用户无法获取名称和用户地址，显示为“知乎用户”**

2、获取数据
假如我们获取到了单页数据，那么使用正则可以很简单地获取到想要的数据，具体参见代码。
我们需要做的，其实是获取那些需要爬取的URL。
通过上面对于网址的分析我们可以发现，网址的组成为domain/answer/ans_id/voters_profile?total=xxx&offset=xx&...
后面那堆乱码不重要，重要的是total和offset，每次会展示出10个用户的数据，所以我们只需要获取到点赞的总数total，就可以知道需要循环多少步（total/10），注意从是0开始，不然会漏掉前十个数据。
而这也就是我在zhihu-vote.py 中的做法。其余的部分就没有什么难度了，入门的同学应该都可以看懂。

3、改进

我们在zhihu-vote.py 中通过构造地址的方法来，通过循环实现对所有voters_profile的遍历。但是如果不了解json的知识的话，我们可以发现其实每个页面都是json格式的。

4974

其中最关键的地方在于**next**。我们会发现**其实每个页面中都包含了下一页的地址！**这样我们是不是可以让爬虫每爬一页自己找到一个地址，然后自己去爬下一页呢？

可是这样做有一个问题，如何控制循环呢？

假如我们去看最后一个页面的话，会发现是这样的。

注意这里的next值为空。

而我们知道（不知道的你现在知道了），**空字符串在作为条件判断时相当于False**

所以我写了zhihu-voteV2.py

其中核心的改动是

```
Vote_url = Zhihu + 'answer/' + ans_id +'/voters_profi
h = s.get(Vote_url)
html = h.content.encode('utf-8')
target = json.loads(html)
while target['paging']['next']:
```

```
    Vote_url = 'http://www.zhihu.com'+target['paging'
```

这样就实现了程序每次爬取页面时从页面中获取地址，而不是人为构造地址循环。下面是原来的做法。

```
for num in range (0,page_num):
    Vote_url = Zhihu + 'answer/'  _id +'/voters_p
```

讲实话我不知道这两种写法哪种好，但我还是蛮高兴自己发现了第二种做法。
于是我做了一个运行时间的测试…提车了，下一步需要做什么？ - 车海沉浮高永强的回答 16K的赞
运行结果如下：

构造地址的办法用时451秒，第二种办法用时251秒。
……我不知道为什么方法二会比方法一快，可能是网速吧……QAQ。有了解的前辈还望告知原因…

到这里也就结束了。最后的结果是写入excel的，有知友说让我去学习csv，已经在看了，不过这次还是用的让人又爱又恨的excel。

按照惯例写To-dos：

- 完善Github的文档说明
- 想办法看能不能自动获取那个蛋疼的ans_id就不用每次都手动抓包了，selenium？我不会用啊TAT
- 在点赞的页面我们只能得到用户的4个数据，也就是赞同、感谢、提问、回答，有些时候我们或许想知道他的关注人数和被关注人数…然而那个得到用户的页面中去爬取了。不过想通过用户URL得到用户的具体数据是有现成的轮子的……egrcc/zhihu-python · GitHub （PY2） @egrcc 和

7sDream/zhihu-py3 · GitHub （PY3） @7sDream 我想办法看怎么把我现在获取答案点赞用户信息的方法pull给他们…直接调用他们的User类就ok了~

- 抓特别多的数据时考虑多线程和gzip…下来就要学这个了…会的话我就用了…还记得我去爬知乎赞数最高的答案…一个答案爬了30分钟…

——更新完毕，大家学习愉快，共同进步——

——**Windows 平台Py2编码问题畸形但有效解法**——

在..\Python27\Lib\site-packages\下新建**sitecustomize.py**添加代码

```
import sys
sys.setdefaultencoding("utf-8")
```

————————正文2：学习路上顺便写的项目————————

在学习路上写了许多类似test的小小项目，就不赘述了。下面贴出来三个还算有结果的。

1、抓取知乎话题下面的问题，分析容易得赞的问题

具体描述在 第一次在知乎获得 1000 以上的赞是什么体验？ - 段晓晨的回答 写过了。

代码在知乎将如何应对「狗日的知乎」计划？ - 段晓晨的回答里面有。需要用到7sDream/zhihu-py3 · GitHub

2、写完1中项目以后。我爬取了爬虫话题分类下面的所有回答。结果爬虫话题所有问题_20150531.xls_免费高速下载

然后我从其中挑选了“关注量/回答量”较大的问题（也就是有人关注但有效回答较少）写了以下两个回答，大家可以看看。

如何使用 python 抓取 雪球网页？ - 段晓晨的回答

如何用Python抓取写一个抓取新浪财经网指定企业年报的脚本？ - 段晓晨的回答

————————结语：谈谈学习————————

至此我能说的就说完了。

鼓起勇气来回答这个问题，不知道自己有没有资格。毕竟自己也就才学了一周多一点。自认为还谈不上入门……因为不会的实在太多。

系统学习爬虫的思路别人讲的肯定比我好。我的经验在开头已经说过了……**不要怂就是干！哪里不会搜哪里！哪里报错改哪里！**

如果一定要再补充写什么，贴上我之前回复知友的评论吧。

首先要带着一个目的去学，这个目的不能太复杂，不能一上来就搞那种需要模拟登陆，需要js动态实现的网站，那样你会在登陆那儿卡很久，又在js实现那儿卡很久，很容易挫伤学习积极性。比如我最初的目的就是爬知乎。知乎登陆/不登陆数据会有差别，比如抓不到某些人的数据，返回“知乎用户”这种。
有了目的，你需要一些基础知识。html是什么，标签是什么，浏览器和服务器之间通信（比如抓包）。爬虫的原理就是要把网页的源码整个下载下来，然后去里面寻找我们需要的信息。
所以首先你得能获取正确的网址，然后通过配置你的程序（Headers伪装浏览器，代理防止封ip等）来成功访问网页并获取源码。……诸如此类的基础知识，其实特别简单。你可以去找一些爬百度贴吧，爬煎蛋，爬糗事百科的例子，很容易就会上手。
有了源码你需要去里面寻找东西，比较简单的有正则表达式，更方便的有BeautifulSoup。对json解析有json。等等。
最后你可能需要一些模块化的思想。比如我在写爬知乎问题的时候，写了一些代码来让它把输出的结果自动保存到excel里…那我是不是可以把写入excel这个行为单独抽出来，定义为一个方法。以后每次遇到需要excel的地方我就拿过来改一下就能用。同样的思路，登陆过程，post数据的过程，解析数据的过程，是不是都可以自己慢慢积累为模块。就好像你有了很多乐高积木，以后再做的时候就不需要做重复的事情，直接搭积木就好~

最后感谢一下在我学习过程中参考过的别人的回答和博客。太多了无法一一列举。再次感谢。

编程是最容易获得的超能力。你还在等什么？

编辑于 2015-06-04　85 条评论　感谢　分享　收藏
• 没有帮助 • 举报 • 作者保留权利

282

Greysign，Simple & Stupid

Jervions、知乎用户、知乎用户 等人赞同

从爬虫必要的几个基本需求来讲：
1.抓取
py的urllib不一定去用，但是要学，如果你还没用过的话。
比较好的替代品有requests等第三方更人性化、成熟的库，如果pyer不了解各种库，那就白学了。
抓取最基本就是拉网页回来。
如果深入做下去，你会发现要面对不同的网页要求，比如有认

证的，不同文件格式、编码处理，各种奇怪的url合规化处理、重复抓取问题、cookies跟随问题、多线程多进程抓取、多节点抓取、抓取调度、资源压缩等一系列问题。
所以第一步就是拉网页回来，慢慢你会发现各种各样的问题待你优化。

4974

2.存储
抓回来一般会用一定策略存下来，而不是直接分析，个人觉得更好的架构应该是把分析和抓取分离，更加松散，每个环节出了问题能够隔离另外一个环节可能出现的问题，好排查也好更新发布。
那么存文件系统、SQLorNOSQL数据库、内存数据库，如何去存就是这个环节的重点。
你可以选择存文件系统开始，然后以一定规则命名。
3.分析
对网页进行文本分析，提取链接也好，提取正文也好，总之看你的需求，但是一定要做的就是分析链接了。
可以用你认为最快最优的办法，比如正则表达式。
然后将分析后的结果应用与其他环节：）
4.展示
要是你做了一堆事情，一点展示输出都没有，如何展现价值？
所以找到好的展示组件，去show出肌肉也是关键。
如果你为了做个站去写爬虫，抑或你要分析某个东西的数据，都不要忘了这个环节，更好地把结果展示出来给别人感受。

397

黑板客，python爱好者

知乎用户、知乎用户、xin tong 等人赞同

个人觉得：
新手学习python爬取网页先用下面4个库就够了：（第4个是实在搞不定用的，当然某些特殊情况它也可能搞不定）

1. 打开网页，下载文件：urllib
2. 解析网页：BeautifulSoup　，熟悉JQuery的可以用Pyquery　（感谢 @李林蔚 的建议）
3. 使用Requests　来提交各种类型的请求，支持重定向，cookies等。
4. 使用Selenium　，模拟浏览器提交类似用户的操作，处理js动态产生的网页

这几个库有它们各自的功能。配合起来就可以完成爬取各种网页并分析的功能。具体的用法可以查他们的官网手册(上面有链接)。

4974

做事情是要有驱动的，如果你没什么特别想抓取的，新手学习可以从这个爬虫闯关网站　开始
，目前更新到第五关，闯过前四关，你应该就掌握了这些库的基本操作。

实在闯不过去，再到这里　看题解吧，第四关会用到并行编程。（串行编程完成第四关会很费时间哦），第四，五关只出了题，还没发布题解。。。

学完这些基础，再去学习scrapy这个强大的爬虫框架会更顺些。这里有它的中文介绍　。

感谢大家支持，改了改链接的格式，好看多了，呵呵

编辑于 2015-10-09　50 条评论　感谢　分享　收藏
• 没有帮助 • 举报 • 作者保留权利

462

静觅，静静寻觅生活的美好

李红、姚曳其实是只熊、FireFireFire 等人赞同

啦啦啦，我就是来送福利得！在这里分享自己写的抓取淘宝MM照片的爬虫
原文：Python爬虫实战四之抓取淘宝MM照片
其实还有好多，大家可以看 Python爬虫学习系列教程
福利啊福利，本次为大家带来的项目是抓取淘宝MM照片并保存起来，大家有没有很激动呢？
本篇目标
1.抓取淘宝MM的姓名，头像，年龄
2.抓取每一个MM的资料简介以及写真图片
3.把每一个MM的写真图片按照文件夹保存到本地
4.熟悉文件保存的过程
1.URL的格式
在这里我们用到的URL是 mm.taobao.com/json/requ...　，
问号前面是基地址，后面的参数page是代表第几页，可以随意更换地址。点击开之后，会发现有一些淘宝MM的简介，并附有超链接链接到个人详情页面。
我们需要抓取本页面的头像地址，MM姓名，MM年龄，MM居住地，以及MM的个人详情页面地址。

2.抓取简要信息

相信大家经过上几次的实战，对抓取和提取页面的地址已经非常熟悉了，这里没有什么难度了，我们首先抓取本页面的MM详情页面地址，姓名，年龄等等的信息抓取来，直接贴代码如下


```
__author__ = 'CQC'
# -*- coding:utf-8 -*-

import urllib
import urllib2
import re

class Spider:

    def __init__(self):
        self.siteURL = 'http://mm.taobao.com/json/req

    def getPage(self,pageIndex):
        url = self.siteURL + "?page=" + str(pageIndex
        print url
        request = urllib2.Request(url)
        response = urllib2.urlopen(request)
        return response.read().decode('gbk')

    def getContents(self,pageIndex):
        page = self.getPage(pageIndex)
        pattern = re.compile('<div class="list-item".
        items = re.findall(pattern,page)
        for item in items:
            print item[0],item[1],item[2],item[3],ite

spider = Spider()
spider.getContents(1)
```

运行结果如下

2.文件写入简介

在这里，我们有写入图片和写入文本两种方式

1）写入图片

```
#传入图片地址，文件名，保存单张图片
def saveImg(self,imageURL,fileName):
     u = urllib.urlopen(imageURL)
     data = u.read()
     f = open(fileName, 'wb')
     f.write(data)
     f.close()
```


2）写入文本

```
def saveBrief(self,content,name):
    fileName = name + "/" + name + ".txt"
    f = open(fileName,"w+")
    print u"正在偷偷保存她的个人信息为",fileName
    f.write(content.encode('utf-8'))
```

3）创建新目录

```
#创建新目录
def mkdir(self,path):
    path = path.strip()
    # 判断路径是否存在
    # 存在     True
    # 不存在   False
    isExists=os.path.exists(path)
    # 判断结果
    if not isExists:
        # 如果不存在则创建目录
        # 创建目录操作函数
        os.makedirs(path)
        return True
    else:
        # 如果目录存在则不创建，并提示目录已存在
        return False
```

3.代码完善

主要的知识点已经在前面都涉及到了，如果大家前面的章节都已经看了，完成这个爬虫不在话下，具体的详情在此不再赘述，直接帖代码啦。

spider.py

```
__author__ = 'CQC'
# -*- coding:utf-8 -*-

import urllib
import urllib2
import re
import tool
import os
```

```
#抓取MM
class Spider:

    #页面初始化
    def __init__(self):
        self.siteURL = 'http://mm                ao.com/json/req
        self.tool = tool.Tool()

    #获取索引页面的内容
    def getPage(self,pageIndex):
        url = self.siteURL + "?page=" + str(pageIndex
        request = urllib2.Request(url)
        response = urllib2.urlopen(request)
        return response.read().decode('gbk')

    #获取索引界面所有MM的信息，list格式
    def getContents(self,pageIndex):
        page = self.getPage(pageIndex)
        pattern = re.compile('<div class="list-item".
        items = re.findall(pattern,page)
        contents = []
        for item in items:
            contents.append([item[0],item[1],item[2],
        return contents

    #获取MM个人详情页面
    def getDetailPage(self,infoURL):
        response = urllib2.urlopen(infoURL)
        return response.read().decode('gbk')

    #获取个人文字简介
    def getBrief(self,page):
        pattern = re.compile('<div class="mm-aixiu-co
        result = re.search(pattern,page)
        return self.tool.replace(result.group(1))

    #获取页面所有图片
    def getAllImg(self,page):
        pattern = re.compile('<div class="mm-aixiu-co
        #个人信息页面所有代码
        content = re.search(pattern,page)
        #从代码中提取图片
        patternImg = re.compile('<img.*?src="(.*?)"',
        images = re.findall(patternImg,content.group(
        return images


    #保存多张写真图片
    def saveImgs(self,images,name):
```

4974

```
        number = 1
        print u"发现",name,u"共有",len(images),u"张照片
        for imageURL in images:
            splitPath = imageURL.split('.')
            fTail = splitPath.pop()
            if len(fTail) > 3:
                fTail = "jpg"
            fileName = name + "/" + str(number) + "."
            self.saveImg(imageURL,fileName)
            number += 1

    # 保存头像
    def saveIcon(self,iconURL,name):
        splitPath = iconURL.split('.')
        fTail = splitPath.pop()
        fileName = name + "/icon." + fTail
        self.saveImg(iconURL,fileName)

    #保存个人简介
    def saveBrief(self,content,name):
        fileName = name + "/" + name + ".txt"
        f = open(fileName,"w+")
        print u"正在偷偷保存她的个人信息为",fileName
        f.write(content.encode('utf-8'))


    #传入图片地址，文件名，保存单张图片
    def saveImg(self,imageURL,fileName):
         u = urllib.urlopen(imageURL)
         data = u.read()
         f = open(fileName, 'wb')
         f.write(data)
         print u"正在悄悄保存她的一张图片为",fileName
         f.close()

    #创建新目录
    def mkdir(self,path):
        path = path.strip()
        # 判断路径是否存在
        # 存在     True
        # 不存在   False
        isExists=os.path.exists(path)
        # 判断结果
        if not isExists:
            # 如果不存在则创建目录
            print u"偷偷新建了名字叫做",path,u'的文件夹'
            # 创建目录操作函数
            os.makedirs(path)
            return True
        else:
```

```
            # 如果目录存在则不创建，并提示目录已存在
            print u"名为",path,'的文件夹已经创建成功'
            return False

    #将一页淘宝MM的信息保存起来
    def savePageInfo(self,pageIndex):
        #获取第一页淘宝MM列表
        contents = self.getContents(pageIndex)
        for item in contents:
            #item[0]个人详情URL,item[1]头像URL,item[2]
            print u"发现一位模特,名字叫",item[2],u"芳龄
            print u"正在偷偷地保存",item[2],"的信息"
            print u"又意外地发现她的个人地址是",item[0]
            #个人详情页面的URL
            detailURL = item[0]
            #得到个人详情页面代码
            detailPage = self.getDetailPage(detailURL
            #获取个人简介
            brief = self.getBrief(detailPage)
            #获取所有图片列表
            images = self.getAllImg(detailPage)
            self.mkdir(item[2])
            #保存个人简介
            self.saveBrief(brief,item[2])
            #保存头像
            self.saveIcon(item[1],item[2])
            #保存图片
            self.saveImgs(images,item[2])

    #传入起止页码，获取MM图片
    def savePagesInfo(self,start,end):
        for i in range(start,end+1):
            print u"正在偷偷寻找第",i,u"个地方，看看MM们
            self.savePageInfo(i)


#传入起止页码即可，在此传入了2,10,表示抓取第2到10页的MM
spider = Spider()
spider.savePagesInfo(2,10)
```

tool.py

```
__author__ = 'CQC'
#-*- coding:utf-8 -*-
import re

#处理页面标签类
class Tool:
    #去除img标签,1-7位空格, 
    removeImg = re.compile('<img.*?>| {1,7}| ')
```

```
    #删除超链接标签
    removeAddr = re.compile('<a.*?>|</a>')
    #把换行的标签换为\n
    replaceLine = re.compile('<tr      />|</div>|</p>'
    #将表格制表<td>替换为\t
    replaceTD= re.compile('<td>')
    #将换行符或双换行符替换为\n
    replaceBR = re.compile('<br><br>|<br>')
    #将其余标签剔除
    removeExtraTag = re.compile('<.*?>')
    #将多行空行删除
    removeNoneLine = re.compile('\n+')
    def replace(self,x):
        x = re.sub(self.removeImg,"",x)
        x = re.sub(self.removeAddr,"",x)
        x = re.sub(self.replaceLine,"\n",x)
        x = re.sub(self.replaceTD,"\t",x)
        x = re.sub(self.replaceBR,"\n",x)
        x = re.sub(self.removeExtraTag,"",x)
        x = re.sub(self.removeNoneLine,"\n",x)
        #strip()将前后多余内容删除
        return x.strip()
```


以上两个文件就是所有的代码内容，运行一下试试看，那叫一个酸爽啊

看看文件夹里面有什么变化

4974

不知不觉，海量的MM图片已经进入了你的电脑，还不快快去试试看！！

4974

代码均为本人所敲，写的不好，大神勿喷，写来方便自己，同时分享给大家参考！希望大家支持！
送完福利走人！！
推荐大家直接看原文：Python爬虫实战四之抓取淘宝MM照片

还有更多：Python爬虫学习系列教程
希望对大家有帮助哈，如果支持的话就点击下网站里的广告支持一下嘿嘿。

编辑于 2015-02-24　45 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

91

鬼谷子，活着就是折腾

王家大少、Bry an、知乎用户 等人赞同

Scrapy爬虫轻松抓取网站数据　，以bbs为例详细介绍了抓取过程。
Scrapy爬虫抓取动态网站　，以典型的比价网站为例，如：搜狗购物　，一淘网　，抓取给定商品的价格及其来源网站。

编辑于 2014-12-25　6 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

1010

匿名用户

林中风扬、李永、周定安 等人赞同

呃本来只是想给题主一个传送，因为本身也是一个Python爱好者。

简单介绍一下我的那个入门教程，其实根本算不上教程，基本上算是一个学习的笔记，很多内容都是从网上整理然后自己实践得到的结果。

如果说深入学习爬虫，还是建议那本《自己动手写网络爬虫》，是我的启蒙教程，语法是Java的，但是思路是相通的。

Python爬虫的学习，最主要的是多摸索实验（哪个不是这样）。先从最简单的例子做起，比如爬取网站主页，爬取百度图片，然后正则，巴拉巴拉。

我的学习笔记可以作为一个参考的索引，里面很多东西没有深入探讨，因为毕竟当时我也只是一个小菜（现在也差不多）。

给初学者一个入门的途径，接下来的路还是要自己走^_^

至于匿名、个人习惯潜水。

继续匿了。

以前写过一个爬虫入门的系列，传送：专栏：Python爬虫入门教程

一共12篇：

[Python]网络爬虫（一）：抓取网页的含义和URL基本构成

[Python]网络爬虫（二）：利用urllib2通过指定的URL抓取网页内容

[Python]网络爬虫（三）：异常的处理和HTTP状态码的分类

[Python]网络爬虫（四）：Opener与Handler的介绍和实例应用

[Python]网络爬虫（五）：urllib2的使用细节与抓站技巧

[Python]网络爬虫（六）：一个简单的百度贴吧的小爬虫

[Python]网络爬虫（七）：Python中的正则表达式教程

[Python]网络爬虫（八）：糗事百科的网络爬虫（v0.2）源码及解析

[Python]网络爬虫（九）：百度贴吧的网络爬虫（v0.4）源码及解析

4974

[Python]网络爬虫（十）：一个爬虫的诞生全过程（以山东大学绩点运算为例）

[Python]网络爬虫（11）：亮剑！爬虫框架小抓抓Scrapy闪亮登场！

[Python]网络爬虫（12）：爬虫框架Scrapy的第一个爬虫示例入门教程

比较入门，不过多接触一些小demo没有坏处哈

编辑于 2014-07-26　84 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

655

egrcc

陈小白、青蓝、黄瓜匠 等人赞同

刚做完一个跟python爬虫相关的项目，跑来回答一下这个问题。作为一个曾经从此问题获益的人，也来说说自己的经验，希望对想学习python爬虫的人有所帮助。

题主既然问的是如何入门，我想一定是助学者，而且我觉得想学python的有很大一部分不是计算机相关专业的（比如我）。记得我当初想入门学python，学爬虫，最困惑的就是一大堆名词听都没听说过。我觉得对初学者而言，不应该一上来就提分布式，多线程，因为这些名词对于一个未入门的人来说很有可能是陌生的，而这些东西在初期学爬虫的时候是不需要用到的，只有当项目上了一定规模，需要提升性能的时候才会用到。而且我建议初学者不要去学什么框架。很多人建议去学scrapy，初学就去接触这些框架很容易学的云里雾里。就好比很多人一上来就建议刚接触web开发的人去学ruby on rails，rails虽好，但rails太“智能”了，它帮你做了太多的事情，以至于你甚至会觉得我都没怎么做，这东西是怎么出来的。这样就会学的很困惑。框架是给有基础的人提升开发速度用的，初学者还是老老实实从基础学起。

爬虫不外乎是为了获取网络上的信息，要取得信息，你就得给给服务器发请求，然后服务器把信息发给你，这一步一般较为

简单。服务器发给你的一般是一个html文件，拿到文件后，你可能会觉得这是什么乱七八糟的东西，怎么都看不懂。我觉得对于一个非计算机系的人来说，想要做点东西出来，最大的困难是缺的知识太多了，html看不懂，htt 也不懂。所以这时候你会发现你需要去学一点html，去w3cschool看一看教程，你并不需要懂很多，弄懂各种标签的，能看懂html文件里的内容就行。

拿到html文件，接下来你要做的就是提取信息，准确地说，是你感兴趣的信息。你需要两样东西，一样用来快速定位你要获取的信息在html源文件中的位置，让你知道要提取什么；另一样用来提取信息。第一样可以选firefox或chrome，都自带开发者功能，第二样可以用Beautiful Soup。所以你需要花一点时间了解开发者功能怎么用，以及Beautiful Soup这个库的用法。会发请求，会提取信息，这些就够了，赶紧做点东西，找点成就感，这样才能有动力继续做下去。比如可以做下载某些网站的图片，把知乎上的答案抓到本地。有点成就感之后就有动力更深入学习了。

在做的过程中，你可能会遇到一些困难，比如你拿到的html跟浏览器看到的html不一样，你上网找一找就会发现，你需要了解一个叫http请求头的东西。找一点资料看看，知道http请求头是怎么回事就行了，不需要完全弄懂http的原理，解决当下的问题才是最重要的，有什么不懂的稍后再补。学了点东西后，你就知道这怎么解决这个问题了，只需在发请求的时候加一个参数，带上http请求头即可，这叫做模拟浏览器的行为。把这个问题解决后，抓取大多数网站都没有问题了。成就感又提升了一点。

有时候你又会发现，有一些网站需要登录才能取得一些信息。找一找资料，你就会接触到“模拟登录”，“post请求”等名词。这时候，你又需要去学习一些http的知识，了解“get”，“post”是怎么回事，以及如何发post请求。为了方便处理http的相关东西，你最好学习一下requests这个库。学习之后，参照网上的代码，我相信模拟登录的问题也解决了。比如，你就可以模拟登录知乎，然后抓取知乎的首页看看，是不是跟你用浏览器中看到的一样？

继续深入，你就会发现这些也不够了，有些信息我需要点一下“更多”按钮才会加载，如何获取这些信息呢？这时候你就需要分析在点“更多”按钮的时候浏览器做了什么，然后去模拟浏览器的

行为。如何分析呢？我一般用firebug，看看点击更多按钮的时候，浏览器做了什么，浏览器一般会发一个post请求，会带上一些参数，你需要知道的就是要带上哪些参数，发请求给谁。这一步可能会有点困难，可以慢慢体会怎么样，一旦越过了这这道坎，你就几乎可以取得任何你想要的信息了。

假设你已经成功了，得到了服务器传回来的数据，你可能会发现，这又跟html不一样了。这是一个叫json的东西，传回来的数据都在里面了，接下来就是要提取数据了。所以你又要去了解json是什么，如何用python处理json。

好了，基本上做完这些，爬虫就算入了门。接下来为了提高性能，扩大规模，再去搞多线程，分布式什么的吧，这也是我接下来需要努力和学习的方向。

那么，学了这些后，我做了点什么？

zhihu-python：egrcc/zhihu-python · GitHub ，一个获取知乎上各种信息的工具，你可以获取任何你想要的问题，答案，用户，收藏夹信息，并且可以方便地将答案备份导出为 txt 或 markdown 文件。获取某个问题下的全部回答，或者备份某大V的全部回答都变得很简单。当然，功能不只有这些，详情请见项目地址：egrcc/zhihu-python · GitHub 。在这个项目中遇到的跟爬虫相关的所有问题几乎都在上面了，当然也遇到了一些其他的问题，都通过网上的资料和他人的帮助解决了。这个项目不到一千行，虽然不是很大，跟一些大牛的项目没法比，代码写的可能也不是很好，请大家多多指教，也可以帮助完善和改进该项目。

以上就是我作为一个初学者的学习经历。题主提这个问题也有一段时间了，可能都已经学会了吧，我就全当做学完后的总结吧。希望对今后想学习的人有所帮助。

最后说一些感想：
1. 不要急于求成，编程虽然不难，但也没有那么简单，不要想着速成，特别是对于计算机基础不是很好的人。
2. 学习的过程中可能会遇到很多困难（上面可能没有提到），或许会有很多你没有接触的东西冒出来，善用google，一个个问题地解决，缺什么补什么。
3. 对于初学者来讲，最重要的不是去学各种吊炸天的框架，追最新的技术。技术，框架是学不完的，永远都会层出不穷，最

重要的是把基础学好。很多时候你有一个问题解决不了，都是你某些方面的知识缺了。慢慢来，不要急，随着学习的深入，再回过头来看以前的问题，会有豁然开朗的感觉。

4. 一定要动手做，找点成就感，对你继续 4974 去有很大的促进作用。不然的话，遇到点困难很容易就放弃了。

编辑于 2014-12-17　54 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

flyer，关注计算机技术和逻辑思维

袁力、知乎用户、知乎用户 等人赞同

73

1. 翻下 搜索引擎技术基础 (豆瓣) 中百度爬虫的一个基本架构，了解爬虫的构成

2. 通过 Python 下的 Scrapy | An open source web scraping framework for Python 框架快速完成简单的爬虫 (可参考我去年对 scrapy 的一个简单封装 flyer103/autospider · GitHub)

3. 之后有不同的方向：

- 研究 **headless browser** 技术，自动处理页面中的 js 请求等。可参考我之前的总结的两种实现 github.com 的页面 和效率比较高的 PhantomJS: Headless WebKit with JavaScript API (看文档就会了，一般还需要与 Squid 结合使用。若想与 Python 结合，可参考我在 stackoverflow 上的提问 Is there a way to use PhantomJS in Python?)
- 研究分布式爬虫的实现，主要还是根据 1) 中那本书中提到的架构思想 (个人打算明年一月份时实现一个类似的)

编辑于 2013-11-23　3 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

7sDream，二次元普通居民/软件爱好者

青年包、仔仔、文盲怪 等人赞同

12

我只是来广告我的项目的……

应该能算个知乎的小爬虫吧

简单说明：

备份某问题所有答案：

```
import zhihu

question = zhihu.Question('http://www.zhihu.com/quest
for answer in question.answers:
    answer.save()
```

4974

会在当前目录下新建以问题标题命名的文件夹，并将所有html文件保存到该文件夹。
save函数默认目录为当前目录下以问题标题开头的目录，默认文件名为问题标题加上答题者昵称，有相同昵称的情况下自动加上序号。
备份某用户所有答案：

```
import zhihu

author = zhihu.Author('http://www.zhihu.com/people/7s
for answer in author.answers:
    # print(answer.question.title)
    answer.save(path=author.name)
```

会在当前目录下新建以作者昵称命名的文件夹，并将所有html文件保存到该文件夹。
备份某收藏夹所有答案：

```
import zhihu

collection = zhihu.Collection('http://www.zhihu.com/c
for answer in collection.answers:
    # print(answer.question.title)
    answer.save(path=collection.name)
```

会在当前目录下新建以收藏夹名称命名的文件夹，并将所有html文件保存到该文件夹。

项目地址：github.com/7sDream/zhih...

写的很简单……用到requests和bs4，带点cookies管理什么的……
变量名，函数名啥的取得都还行，代码也感觉还挺好懂的，入门看看应该不错~~~嘿咻

编辑于 2015-02-27　1 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

57

张瑞，长期价值是个筐

根根、quietcool wu、徐鑫源 等人赞同

1.学习这几个函数

urllib2.urlopen
re.compile

4974

一个是从指定url读去内容
另一个是正则表达式获取特定内容

2.弄懂如下代码

```
#coding: utf8
import urllib2
import re
import StringIO
import gzip

ua = { #'User-Agent':'Mozilla/5.0 (compatible;
Googlebot/2.1; +Googlebot - Webmaster Tools Help )',
'User-Agent':'Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.11 (KHTML, like Gecko)
Chrome/23.0.1271.64 Safari/537.11',
'Connection': 'close',
'Accept-Language': 'zh-CN,zh;q=0.8',
'Accept-Encoding': 'gzip,deflate,sdch',
'Accept': 'text/html,application/xhtml+xml,application
/xml;q=0.9,*/*;q=0.8',
'Accept-Charset': 'GBK,utf-8;q=0.7,*;q=0.3',
'Cache-Control': 'max-age=0',
}

def get_html(url_address):
'''open url and read it'''
req_http = urllib2.Request(url_address,headers=ua)
#req_http = urllib2.Request(url_address)
html = urllib2.urlopen(req_http).read()
return html

def controller():
'''make url list and download page'''

urlt = 'matrix67.com/blog/page/... '
reget = re.compile('(<div class=\"post-wrapper.*?)<p
```

```
class=\"pagination\">',re.DOTALL)
fp=open("org.txt","w+")
for i in range (1,131):
html_c = get_html(urlt% (i))
print urlt%(i)
html_c =
gzip.GzipFile(fileobj=StringIO.StringIO(html_c)).read()
res = reget.findall(html_c)
for x in res :
fp.write(x)
fp.write("\n\n\n")

fp.close();
return

controller()
```


好了，你可以去改变世界了

知乎用户，lining0806.com/

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fireling的数据空间　，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, 'GE
    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=    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=
```

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
response1 = requests.get(url_login) # 未登陆
response2 = requests_session.get(url_login) # 已登陆，
response3 = requests_session.get(url_results) # 已登陆
```

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, urll
```

```
    urllib2.install_opener(opener) # 安装opener，此后调
    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
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):
    while True:
        retryTimes = 20
    while retryTimes>0:
        try:
            return session.post(*arg)
        except:
            print '.',
            retryTimes -= 1
```

或者

```
def multi_open(opener, *arg):
    while True:
        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。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：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，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

编辑于 2015-12-17 7 条评论 感谢 分享 收藏 • 没有帮助 • 举报 • 作者保留权利

4974

31

林少维，程序员，熟悉Python，Linux，Web开发，乐...

laoweng、王雷、知乎用户 等人赞同

首先，我们先来看看，如果是人正常的行为，是如何获取网页内容的。
(1)打开浏览器，输入URL，打开源网页
(2)选取我们想要的内容，包括标题，作者，摘要，正文等信息
(3)存储到硬盘中

上面的三个过程，映射到技术层面上，其实就是：网络请求，抓取结构化数据，数据存储。

我们使用Python写一个简单的程序，实现上面的简单抓取功能。

```
import urllib2, re, cookielib
def httpCrawler(url):
    '''
    @summary: 网页抓取
    '''
    content = httpRequest(url)
    title = parseHtml(content)
    saveData(title)

def httpRequest(url):
    '''
    @summary: 网络请求
    '''
    try:
        ret = None
        SockFile = None
        request = urllib2.Request(url)
        request.add_header('User-Agent', 'Mozilla/4.0
        request.add_header('Pragma', 'no-cache')
        opener = urllib2.build_opener()
        SockFile = opener.open(request)
        ret = SockFile.read()
    finally:
        if SockFile:
            SockFile.close()

    return ret

def parseHtml(html):
    '''
    @summary: 抓取结构化数据
    '''
    content = None
    pattern = '<title>([^<]*?)</title>'
    temp = re.findall(pattern, html)
    if temp:
        content = temp[0]

    return content

def saveData(data):
    '''
    @summary: 数据存储
    '''
    f = open('test', 'wb')
    f.write(data)
    f.close()
```

```
if __name__ == '__main__':
    url = 'http://www.baidu.com'
    httpCrawler(url)
```

4974

看着是不是很简单，是的，其实上面就是实现了一个简单的抓取，也是入门爬虫最基础的抓取代码，实际工作就是基于这种原理，再扩展出各种需求，各种监控和反监控，各种资源调度，各种服务以及管理，可以移步到这里：Python简单爬虫引出分布式爬虫

编辑于 2014-03-16　2 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

20

框框，只做前端的全栈工程师。

fish nettle、fanye、匿名用户 等人赞同

Cola，一朋友做的分布式爬虫，可以多台电脑一起爬，自带微博爬虫范例，还是比较简单的。

github.com/chineking/co...

发布于 2013-08-30　5 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

10

纪路，程序员

aBcs、阿满子、Tika Zhang 等人赞同

python可能就是因为有scrapy才成为最好学习的爬虫工具，不过在做爬虫之前有一点需要明确，爬虫要做到多大规模，因为这涉及到服务器的数量和存储的问题。其实说道服务器数量并不是说爬虫很需要计算量，而是大多数网站都有ip限制，比如同一个ip在一个小时之内访问不能超过1万次，或者更少，就算只用1核的电脑一个小时也肯定超过这个数了，所以爬虫之所以叫爬虫，就是要小但多～，每一台机器都有一个ip，用来突破ip限制（一定要确定每台机器都要有自己的公网ip，不能靠端口映射得到的ip，这样的ip还是会被限制，所以自己搭建爬虫集群变得几乎不可能，这个时候我们要用“云计算”）。二可能就是存储的问题了，当你MySQL一个表存了超过1亿行之后，别想MySQL还能正常工作，MongoDB也救不了你，通常的做法是分表分库，数据库集群等等。

如果小规模玩耍的话，那基本的html要会，因为要解析这些标签，需要程序员自己先看的懂。很多方法scrapy都封装了，除了使用现成的方法，还要试着理解其实现的原理。最后我觉得

xml和json要了解，因为无论如何你肯定不想把html原样存储，html不仅体积巨大，冗余信息多，而且难以解读，python对xml和json支持都非常好，尤其是json与python的字典对应，还与mongodb的doc对应，所以将有效的内容转化成JSON然后存进mongodb是一个不错的流程（我特别喜欢这种方式，在这种便利面前，MySQL简直不堪一视～）。当然爬虫只是数据业务的起步，如何有效的玩弄数据，挖掘数据，才是体现数据价值的地方～

4974

发布于 2014-07-05　1 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

17

郭麒，互联网广告行业老兵 php、python程序员

Guokr1991、王田、张华正 等人赞同

谢邀！

爬虫系统基本的结构：

1.页面下载程序；

最简单的工具就是urllib、urllib2。这两个工具可以实现基本的下载功能，如果进阶想要异步可以使用多线程，如果想效率更高采用非阻塞方案tornado和curl可以实现非阻塞的下载。

2.链接发掘程序；

要想在页面中找到新链接需要对页面解析和对url排重，正则和DOM都可以实现这个功能，看自己熟悉哪一种。

正则感觉速度较快一些，DOM相对较慢并且复杂一点，如果只是为了要url正则可以解决，如果还想要页面中其他的结构或者内容DOM比较方便。

url的排重两小可以用memcache或者redis，量大就要用到bloomfilter。

3.网页存储等部分组成；

抓的少怎么存都行，抓的多并且要方便读取那就要好好设计了，用哈希分布存储在RDBMS上或者直接存在HBase上都要看你的数据量和具体需求。

发布于 2013-04-08　3 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

21

张尧，在校小朋友

知乎用户、袅残烟、北蔡小生 等人赞同

福利预警！

运行本脚本，请保持周围安全。

我试着爬了推女郎的所有图片。

4974

戳这http://blog.csdn.net/zycode277/article/details/46474105
2.7和3.4版本的都有。

然后又试着做了一个URP系统评教，将来打算部署到微信公众号上去~
我是天工大的，其他学校的有没有帮忙测试的？
python模拟登录URP教务系统评教
还挺好用的，给很多同学用了。

编辑于 2015-08-10　9 条评论　感谢　分享　收藏
• 没有帮助 • 举报 • 作者保留权利

1

老夏，互联网数据交易平台finndy.com

海龟龟 赞同

这是网友分享的，之前都下载过，没事的时候再会去看看，有需求的朋友可以去下载

4974

发布于 2015-10-28　1 条评论　感谢　分享　收藏
• 没有帮助 • 举报 • 作者保留权利

31

高野良，弹弹钢琴/写写代码/做做策划

知乎用户、徐鼬、布鲁克斯 等人赞同

其实对于python我也几乎是0起步——之前写代码还是小学时候的visual basic，，，后来完全忘掉，大学也不是相关专业，工作后因为对数据分析很感兴趣，所以业余时间自学python。一开始就是接触爬虫，学习效果还不错~在这里和大家分享~

注：本答案采用脑图形式呈现，知乎好像预览图都会压缩，，点开后可以看清晰大图。由于如有问题欢迎指出！本答案将会不断迭代更新！


4974

4974

答案最后奉上我学爬虫的终极秘诀：

，
，
，
，
，
，
，
，
，
，
，
，
，
，
，
，
，
，
，
，
没错！就是抄！

开两个sublime的窗口（重点推荐sublir 4974 小巧轻便，主题好看，相比较下其他IDE太臃肿了，适合做[illegible]时候再用）左边是网上找到的代码，右边是自己抄写用

然后开始手打抄代码，不要复制！不要复制!不要复制！边抄边看这个代码自己有没有见过，知不知道基本的用法，如果不会，马上网上查文档看，搞清楚了，继续抄。。
直到抄到你发现都可以背出来了，就出师了！

ps：对于初学者，不断抄的过程还能提高输入速度。很多人在没有学代码之前打字都是中文思路，打英文很慢，这个需要多练习。我也是在学习码代码以后才发现自己的盲打能力慢慢上去，之前输个密码还要看键盘。

编辑于 2016-01-24　19 条评论　感谢　分享　收藏
• 没有帮助 • 举报 • 作者保留权利

6

神手，get a real life

徐鼬、王雷、flower sun 等人赞同

我的博客里有很详细的叙述和源码,python3.4实现。
欢迎交流 网络资源搜索爬虫(python 3.4.1实现)

编辑于 2014-10-15　1 条评论　感谢　分享　收藏
• 没有帮助 • 举报 • 作者保留权利

更多

我来回答这个问题

写回答…

我要回答