



## 爬虫基本原理

**Dwzb**

统计专业学生

关注他

874 人赞同了该文章

这篇文章的定位是，给有一些python基础，但是对爬虫一无所知的人写的。文中只会涉及到爬虫最核心的部分，完全避开莫名其妙的坑或概念，让读者觉得爬虫是一件非常简单的事情，而事实上爬虫确实是一件非常简单的事情（如果你不是以爬虫为工作的话）。

本文分为如下几个部分

- 引言
- 概念介绍
- HTML介绍
- 解析代码介绍
- chrome检查工具介绍

### 引言

简单理解网络爬虫就是自动抓取网页信息的代码，可以简单理解成代替繁琐的复制粘贴操作的手段。

首先必须声明，爬虫的对象必须是你已经看到的网页，比如你不能说你想找到知乎上哪个用户的关注人数最多，就希望通过写一个爬虫来帮你爬到答案。你必须明确地知道这个人，找到他的主页，然后才能用爬虫来抓取他页面上的信息。

下面我们用一个简单的例子来展示爬虫的工作流程。感觉多数教程第一篇都使用的是豆瓣top250，我们这里换一个，抓取CSDN首页的文章标题，链接在[这里](#)，页面样子是这样的

▲ 赞同 874



抓取标题完整代码如下

```
import requests # 导入网页请求库
from bs4 import BeautifulSoup # 导入网页解析库

# 传入URL
r = requests.get('https://www.csdn.net/')

# 解析URL
soup = BeautifulSoup(r.text, 'html.parser')
content_list = soup.find_all('div', attrs = {'class': 'title'})

for content in content_list:
    print(content.h2.a.text)
```

这样就会打印出所有标题，展示一部分如下

从扩线查询能力分析分布式图数据库Titan的设计改进点

Android 框架学习4: 一次读懂热门图片框架 Picasso 源码及流程

一条数据的HBase之旅，简明HBase入门教程-开篇

记录一次壮烈牺牲的阿里巴巴面试

JVM知识问答集锦

一图看懂 | 人工智能知识体系大全

深圳社招大厂面试分享

记录一次壮烈牺牲的阿里巴巴面试

创投日报：3月27日收录投融资项目31起

程序员如何进阶成为大神？

推荐 | 别让职业生涯规划毁掉你的成长

TensorFlow实现流行机器学习算法的教程汇总（2/3）

离职前需要注意的事情？

上述过程是一个最简单的完整爬虫流程，可以看出它的功能就是把那些标题复制粘贴到一起，免除了手动操作的繁琐。其实爬虫一般就是做这些事的，比如我们需要用链家的数据进行分析，看到链家的页面是这样的



我们想获取每个房子的标题、几室几厅、多少平米、朝向、装修、价格等等字段（即指标），就可以通过爬虫进行定位，自动化抓取这100页所有房子的这些字段信息。比如100页里有2000个房子，总共抓取10个字段，爬虫运行结束就可以得到一个2000行10列的excel表格。

注：如果还没有安装上面两个库的读者可以在命令行下分别运行下面两行命令完成安装

```
pip install requests
pip install beautifulsoup4
```

概念介绍

知道了爬虫是用来干什么的之后，我们来介绍一些最常见到的概念

1.URL

URL中文称为统一资源定位符，其实可以理解成网页的链接，比如上面的 <https://www.csdn.net/> 就是一个URL。

但是更广义的URL不只是我们常看到的网页资源链接，而是资源在网页中的定位标识。我们通常说的网页是一个资源，网页中加载的每一张图片也是一个资源，它们在互联网中也有唯一的定位URL。比如我们从CSDN网页上随便找一张图片



这个链接 <https://csdnimg.cn/feed/20180330/49f4cd810ad4606e3c45ed9edb16a8b8.jpg> 就是这个图片资源的定位符，将这个链接输入浏览器中就会显示出这张图片，所以说这张图片也对应一个URL。

不过知道这么回事就好，我们通常所说的传入URL指的就是把网页的链接传进去。上面代码中

```
r = requests.get('https://www.csdn.net/')
```

就是在将URL传入请求函数。

▲ 赞同 874



2.网页请求

说到网页请求，就有必要讲一下我们平常浏览网页时，信息交互的模式大概是什么样的。我们平常用浏览器浏览网页的时候，鼠标点了一个链接，比如你现在[点击这里](#)，其实浏览器帮你向这个网页发送了请求(request)，维护网页的服务器（可以理解为CSDN公司里的一台电脑，在维护这CSDN上的各个网页）收到了这个请求，判定这个请求是有效的，于是返回了一些响应信息(response)到浏览器，浏览器将这些信息进行渲染（可以理解成 处理成人能看懂的样子），就是你看到的网页的样子了。发送请求与接收请求的过程就和 发微信和收到回复的过程类似。

而现在我们要用代码来模拟鼠标点击的过程。上面的 requests.get 就是让代码帮你向这个网页发送了这个请求，如果请求被判定为有效，网页的服务器也会把信息传送给你，传送回来的这些信息就被赋值到变量 r 之中。所以这个变量 r 里就包含有我们想要的信息了，也包括那些我们想要提取的标题。

我们可以 print(r.text) 看一下里面有什么东西

```
In [2]: print(r.text)

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <script src="/js/tingyun-rum.js?1511847956" type="text/javascript"></script>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
  <meta name="apple-mobile-web-app-status-bar-style" content="black">
  <meta name="referrer" content="always">
  <meta name="msvalidate.01" content="3189512127C34C468C748E05852D45E4" />
  <title>CSDN- 专业IT技术社区</title>
  <link rel="canonical" href="https://www.csdn.net/">
  <link href="//csdnimg.cn/public/common/libs/jquery/jquery-1.9.1.min.js" type="text/javascript"></script>
  <script src="//csdnimg.cn/pubfooter/js/tracking-1.0.2.js" type="text/javascript" charset="utf-8"></script>
  <script src="//csdnimg.cn/rabbit/exposure-click/main_flow.js?v1.15.35"></script>
  <link rel="stylesheet" href="//csdnimg.cn/public/common/toolbar/content_toolbar_css/content_toolbar.css">
  <link rel="stylesheet" href="//csdnimg.cn/public/common/libs/bootstrap/css/bootstrap.min.css">
  <link rel="stylesheet" href="//csdnimg.cn/public/static/css/avatar.css">
```

我们再看一下网页的源代码（如何看懂这个源码，以及这个源码怎么查看下一节HTML会详细讲到）





```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <script src="//js/lingyun-run.js?1511847956" type="text/javascript"></script>
6   <meta http-equiv="content-type" content="text/html; charset=utf-8">
7   <meta http-equiv="X-UA-Compatible" content="IE=Edge">
8   <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
9   <meta name="apple-mobile-web-app-status-bar-style" content="black">
10  <meta name="referrer" content="always">
11  <meta name="msvalidate.01" content="3189512127C34C468C748ED5852D45E4" />
12  <title>CSDN- 专业IT技术社区</title>
13  <link rel="canonical" href="https://www.csdn.net/">
14  <link href="//csdnimg.cn/public/favicon.ico" rel="SHORTCUT ICON">
15  <script src="//csdnimg.cn/public/common/libs/jquery/jquery-1.9.1.min.js" type="text/javascript"></script>
16  <script src="//csdnimg.cn/pub/footer/js/tracking-1.0.2.js" type="text/javascript" charset="utf-8"></script>
17  <script src="//csdnimg.cn/rabbit/exposure-click/main_flow.js?v1.15.35"></script>
18  <link rel="stylesheet" href="//csdnimg.cn/public/common/toolbar/content_toolbar_css/content_toolbar.css">
19  <link rel="stylesheet" href="//csdnimg.cn/public/common/libs/bootstrap/css/bootstrap.min.css">
20  <link rel="stylesheet" href="//csdnimg.cn/public/static/css/avatar.css">
21  <link href="//css/csdn_feed.css?1521452600" rel="stylesheet" />
22  <script src="//js/modernizr.js?1508677690" type="text/javascript"></script>
23  <style type="text/css">
24  .sidebar_fr > .box.hot:first-child {display:none!important;display:none}</style>
25  <script language="javascript" type="text/javascript" src="https://202.102.100.100/35ff706f457d11c141cdefcd58d6562b.js" charset="gb2312">
26  </script><script type="text/javascript">
27  HQUHuMEAYin(('.blog_ad_#layerd,iframe[src*="pos.baidu.com"],.append_mark_img');</script></head>
28  <body data-category="home" data-host_type="www">
29  <script id="toolbar_tpl-scriptId" prod="download" skin="black" src="//csdnimg.cn/public/common/toolbar/js/content_toolbar.js"
30  type="text/javascript" domain="http://blog.csdn.net"></script>
31  <div class="container clearfix">
32  <div id="nav" class="clearfix">
33  <div class="clearfix">
34  <div class="nav_com">
35  <ul>
36  <li class="active"><a href="/">推荐</a></li>
37  <li class=""><a href="/nav/newarticles">最新文章</a></li>
38  <li class=""><a href="/nav/watchers">关注</a></li>
39  <li class=""><a href="/nav/news">资讯</a></li>
40  <li class=""><a href="/nav/ai">人工智能</a></li>
41  <li class=""><a href="/nav/cloud">云计算/大数据</a></li>
42  <li class=""><a href="/nav/blockchain">区块链</a></li>
43  <li class=""><a href="/nav/db">数据库</a></li>
44  <li class=""><a href="/nav/career">程序人生</a></li>
45  <li class=""><a href="/nav/game">游戏开发</a></li>
46  <li class=""><a href="/nav/engineering">研发管理</a></li>
47  <li class=""><a href="/nav/web">前端</a></li>
48  <li class=""><a href="/nav/mobile">移动开发</a></li>
49  <li class=""><a href="/nav/iot">物联网</a></li>
50  <li class=""><a href="/nav/ops">运维</a></li>
51  <li class=""><a href="/nav/fund">计算机基础</a></li>
52  <li class=""><a href="/nav/lang">编程语言</a></li>
53  <li class=""><a href="/nav/arch">架构</a></li>
54  <li class=""><a href="/nav/avi">音视频开发</a></li>
55  <li class=""><a href="/nav/sec">安全</a></li>
56  <li class=""><a href="/nav/other">其他</a></li>
57  </ul>
58  </div>
59  </div>
60  </div>
61  <div class="fixed_content">
62  <div id="banner-ad-box">
63  <div class="banner-ad-box">
64  <div data-revive-zoneid="281" data-revive-id="8c38e720de1c90a6f6ff52f3f89c4d57"></div>
65  <div data-revive-zoneid="282" data-revive-id="8c38e720de1c90a6f6ff52f3f89c4d57"></div>
66  </div>
67  <!--头部banner广告 end-->
68  <main>
69  <div class="carousel">
70  <div class="carousel-left">
71  <div id="myCarousel" class="slide" data-ride="carousel">
72  <!-- Indicators -->
73  <ol class="carousel-indicators">
74  <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
75  <li data-target="#myCarousel" data-slide-to="1" class=""></li>
76  <li data-target="#myCarousel" data-slide-to="2" class=""></li>
77  <li data-target="#myCarousel" data-slide-to="3" class=""></li>
78  <li data-target="#myCarousel" data-slide-to="4" class=""></li>
79  </ol>
80  <div class="carousel-inner" role="listbox">
81  <div class="carousel-item csdn-tracking-statistics active" data-mod="popu_465" data-dsm="post">
82  <a href="https://blog.csdn.net/csdnnews/article/details/79777598" target="_blank">
83  
84  </a>
85  <a href="https://blog.csdn.net/csdnnews/article/details/79777598" target="_blank">
86  <div class="carousel-caption">
87  从编程语言进化史，看 Java、C、C++ 等语言的演变 </div>
88  </a>
89  <a href="https://blog.csdn.net/csdnnews/article/details/79777598" target="_blank" style="display:block;">
90  <div class="cover"></div>
91  </a>
92  </div>
93  <div class="carousel-item csdn-tracking-statistics " data-mod="popu_465" data-dsm="post">
94  <a href="https://bss.csdn.net/m/topic/blockchain2018/live" target="_blank">
95  
96  </a>
97  <a href="https://blog.csdn.net/dQCFKyQDXyM3F8rB0/article/details/79762534" target="_blank">
98  <div class="carousel-caption">
99  Google、高通都在研究的芯片架构，是他们对抗ARM的武器 </div>
100  </a>
101  <a href="https://blog.csdn.net/dQCFKyQDXyM3F8rB0/article/details/79762534" target="_blank" style="display:block;">
102  <div class="cover"></div>
103  </a>
104  </div>
105  <div class="carousel-item csdn-tracking-statistics " data-mod="popu_465" data-dsm="post">
106  <a href="http://bss.csdn.net/m/topic/blockchain2018/live" target="_blank">
107  
108  </a>
109  <a href="http://bss.csdn.net/m/topic/blockchain2018/live" target="_blank">
110  <div class="carousel-caption">
111  2018 区块链技术及应用峰会(BTA)·中国在北京盛大召开 </div>
112  </a>
113  <a href="http://bss.csdn.net/m/topic/blockchain2018/live" target="_blank" style="display:block;">
114  <div class="cover"></div>
115  </a>
116  </div>
117  </div>
118  </div>
119  </div>
```

源代码和 r.text 其实是一模一样的东西。r.text 其实就是一个字符串，字符串中有我们刚刚抓取到的所有标题，我们只要通过字符串匹配方法（比如正则表达式）将他们提取出来就可以了。这样说是不是感觉爬虫非常简单呢？只要这样傻瓜操作

```
r = requests.get('https://www.csdn.net/')
```

再直接从 r.text 字符串中提取信息即可。其实爬虫就是这么简单。

▲ 赞同 874



但是解析是怎么回事呢，为什么刚刚不直接用正则而要用bs4呢？因为方便，但是正则也是完全可以的，只是相对麻烦一些、需要写更多的代码而已。

3.网页解析

网页解析其实就从网页服务器返回给我们的信息中提取我们想要数据的过程。其实使用正则表达式提取我们要的标题的过程也可以称为网页解析。

因为当前绝大多数网页源代码都是用HTML语言写的，而HTML语言时非常有规律性的，比如我们要的所有文章标题都具有相同结构，也就是说它周围的字符串都是非常类似的，这样我们才能批量获取。所以就有大佬专门封装了如何从HTML代码中提取特定文本的库，也就是我们平时说的网页解析库，如 bs4 lxml pyquery 等，其实把他们当成处理字符串的就可以了。

为了更清楚地了解如何对网页进行解析，我们需要先粗略掌握HTML代码的结构。

HTML介绍

引用维基百科中的一段话来介绍HTML

超文本标记语言（英语：HyperText Markup Language，简称：HTML）是一种用于创建网页的标准标记语言。HTML是一种基础技术，常与CSS、JavaScript一起被众多网站用于设计令人赏心悦目的网页、网页应用程序以及移动应用程序的用户界面[1]。网页浏览器可以读取HTML文件，并将其渲染成可视化网页。

为了让读者对HTML有更清楚的认识，我们来写一点简单的HTML代码。用文本编辑器（记事本也可以）创建一个名字为a.html的文件，在里面写下如下代码

```
<!DOCTYPE html>
<html>
<head>
<title>爬虫基本原理</title>
</head>
<body>
<h1>HTML介绍</h1>
<p>第一段</p>
<p>第二段</p>
</body>
</html>
```

保存，然后你双击这个文件，就会自动用浏览器打开，然后你就能看到下面这个样子的页面



你如果按照我的操作来做的话，你已经创建了一个简单的网页，现在你看到的所有网页都是这样设计的，只是比你的复杂一点而已，不信你去看看刚才截图下来的网页源代码图片。

接下来，我们来看一下HTML语言的特点。最重要的一点是，文本都是被标签(h1标签 p标签)夹在中间的，而这些标签都是特定的，有专门用途的。比如 <h1> 就表示一级标题，包在里面的文本自

▲ 赞同 874



然会被放大显示；而 `<p>` 标签则表示段落。

再看上面的源代码截图，`head meta script title div li` 每一个都是标签，层层嵌套。我们完全不需要知道总共有哪些种标签，也不需要知道这些标签都是用来干什么的，我们只要找到我们要的信息包含在什么标签里就行了。比如使用正则表达式就直接用 `<p>(.*?)</p>` 就可以把里面的内容提取出来了。

但是事实好像没有那么简单，看上面的截图标签怎么是这样的 `<nav id="nav" class="clearfix">`？其实这是一个 `<nav>` 标签，后面的 `id class` 是这个标签的属性。

为什么要给标签设置属性呢？我们先考虑这样一个问题：我们看到的网页千差万别，文字的颜色字体等都不一样，这是怎么设置的呢？答案是使用css样式。

css语句类似这样

```
h1 {
    color: white;
    text-align: center;
}
p {
    font-family: verdana;
    font-size: 20px;
}
```

即设置对应标签的颜色、字体、大小、居中等。而当有的段落使用这个字体，有的段落使用那个字体怎么办呢？css这样设置

```
p.test1 {
    font-size: 20px;
}
p.test2 {
    font-size: 15px;
}
```

在HTML代码中则这样写

```
<p class="test1">20px大小的字</p>
<p class="test2">15px大小的字</p>
```

所以不同属性就是为了区分相同标签用的，这相当于给标签进行了分类，在统一设计样式上更方便，同时对于我们根据属性定位我们想要内容的位置其实也是更方便了。这里要说明一下，`class id` 这两个属性比较特殊，用的也最多，所以各自弄了一个快捷键来表示，`class` 用 `.`，`id` 用 `#`。

做爬虫不需要了解刚刚编写的css代码内容放在哪里之类的问题，也不需要了解css代码设置了什么，我们只会和HTML打交道，所以只要理解HTML中属性的作用就可以了。

如果想要更进一步了解HTML和CSS，可以到w3school网站学习。

现在你就已经具备了解析网页需要的全部HTML知识了。我们一般就是根据标签名配合属性值来定位我们想要资源的位置的，其他的都不用管。这时，我们再来看爬虫的解析代码

## 解析代码介绍

把前面的代码再粘贴一遍如下

```
import requests # 导入网页请求库
from bs4 import BeautifulSoup # 导入网页解析库
```

▲ 赞同 874



```
# 传入URL
r = requests.get('https://www.csdn.net/')

# 解析URL
soup = BeautifulSoup(r.text, 'html.parser')
content_list = soup.find_all('div', attrs = {'class': 'title'})

for content in content_list:
    print(content.h2.a.text)
```

解释一下上面代码的过程

- 第一步，把 r.text 这个网页源代码字符串传到 BeautifulSoup 函数中（第二个参数固定用法先不用管），得到的 soup 对象，可以理解成这一步是为了让这个字符串可以用一些更简单的方法，比如它可以调用 find\_all 方法，寻找class属性是title的div标签，这样的标签有非常多，返回了一个list。
- 第二步，对这个list进行循环，每一个元素内部寻找h2标签里的a标签，提取a标签中的文本，即我们要的标题。

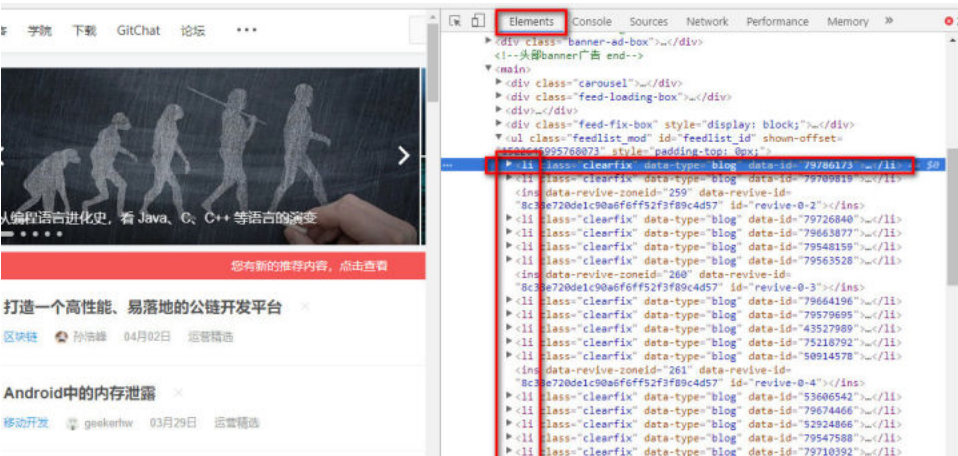
可以看到上面的代码非常简洁，思路清晰，读者可以自己想一想如果要用正则表达式如何匹配这些标签，会发现代码繁琐很多，虽然它也有更快的优势。

那么我们是知道要寻找什么样属性的div标签，为什么要找h2 a标签而不是其他的呢？这就要去分析网页的源代码了。而这个过程也非常简单。

chrome检查工具介绍

我们现在用谷歌浏览器打开CSDN这个网站，找一个空白的位置右键-查看网页源代码，这时就会打开一个新的页面这个页面就是这个网站的HTML源代码了，我们可以通过这个页面来看我们要的信息在哪里，但是感觉非常不方便，因为有太多无用的信息做干扰，我们无法快速掌控网页的结构。所以我们可以用另一种方式查看源代码。

用谷歌浏览器打开CSDN这个网站，找一个空白的位置右键-检查，就会弹出一个框，如下图所示



(如果没有看到这个界面，注意要切换到Element中)

这个页面最大的好处是通过折叠来让人更快探索出网页的结构。

其中的那些代码就是HTML代码，该页面的一个个标题就存在这一个个li里面。点击li前面的三角就可以展开具体的代码内容，如下图所示

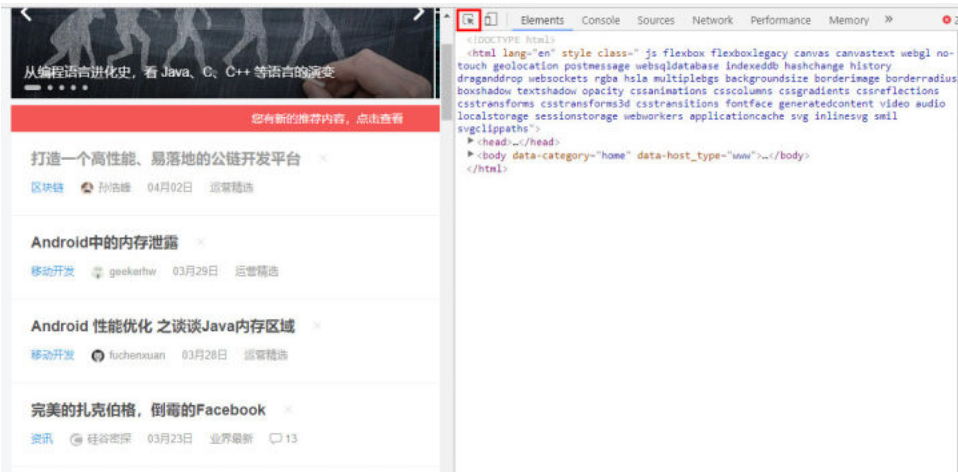




可以看到文章的标题（打造一个高性能、易落地的公链开发平台）就在这个源代码之中，也就是说在我们刚刚获得的 `r.text` 字符串之中。而我们代码定位路径也一目了然了，因为每个 `li` 里面都会有一个 `<div class="title">` 而每一个 `div` 里面都会有一个 `h2` 里面有一个 `a`，`a` 中包含我们要的标题名称。所以我们就用 `find_all` 找到所有这样的 `div` 标签，存储为一个 `list`，再对 `list` 进行循环，对每一个元素提取 `h2` `a` 再提取标签中的内容。

当然我们也可以 `find_all` 最外面的 `li` 标签，再一层层往里找，都是一样的。只要找到定位信息的唯一标识（标签或者属性）就可以了。

虽然在这里看源代码可以折叠一些没用的代码，但是其实还有一些更好用的工具来辅助我们找到我们要的信息在网页源码中的位置。比如下面这个鼠标符号。



在所有代码都折叠起来的情况下，点击这个鼠标，之后再去点击网页中的元素，浏览器就会自动帮你把你点击的元素选中出来，其实你鼠标悬在一个元素上面的时候，就已经帮你定位了，如下图所示



## 总结

当我们要爬一个网页的时候，只需要如下流程

- 导入两个库，一个用于请求，一个用于网页解析
- 请求网页，获得源代码
- 初始化 `soup` 对象，使其可以调用更简单易用的方法



- 用浏览器打开网页，右键-检查，使用那个鼠标定位你要找的资源的位置
- 分析那个位置的源代码，找到合适的用于定位的标签及属性
- 编写解析代码，获得想要的资源

现在，对于一些没有丝毫反爬措施的网站我们都可以游刃有余了。至于抓取多个字段的数据如何组织在一起、抓取多页（URL有规律的情况下）的代码如何设计，就不是爬虫知识范畴了，这是用python基础知识就可以解决的。下一系列文章就主要讲这一部分。接下来给几个当前可以练手的网站

- CSDN首页除了标题之外的其他信息，比如作者名、发布的时间等，可以存储到一个以字典为元素的list里
- 豆瓣top250的信息，会翻页的话250个电影的信息可以直接一个循环抓取下来，其实找到规律构造出10个URL即可
- stackoverflow 简书 伯乐在线 等等网站都可以拿来练手

如果使用BeautifulSoup的定位的过程中遇到困难，可以直接到网上搜教程，也可以等我们这个专题后面更新的BeautifulSoup详细介绍。

如果你去抓取其他网站，最好先看一下 `r.text` 是不是和网站源代码一模一样，如果不是，说明你对方服务器没有把真正的信息给你，说明他可能看出你是爬虫了（进行网页请求的时候，浏览器和 `requests.get` 都相当于带着一堆资格证去敲门，对方会检查你这些资格证，浏览器的资格证一般是可以通过的，而代码的资格证就可能不合格，因为代码的资格证可能有一些比较固定的特点，对方服务器预先设定好，资格证是这样的请求一律拒绝，因为他们一定是爬虫，这就是反爬虫机制），这时就需要懂一些反爬措施才能获得真正的信息，反爬方法的学习是一个积累的过程，我们后面再讲。读者如果遇到一些反爬机制，可以到网上查这个网站的爬虫，估计都能查到一些博客讲如何破解，甚至直接贴出代码。

在这篇的基础上抓取多页以及代码设计的改进看下面这三篇续篇

[爬虫代码改进（一）](#)

[爬虫代码改进（二）](#)

[爬虫代码改进（三）](#)

## 专栏信息

专栏主页: [python编程](#)

专栏目录: [目录](#)

爬虫目录: [爬虫系列目录](#)

版本说明: [软件及包版本说明](#)

编辑于 2018-07-30

「希望对你有帮助！」

赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

[Python](#) [爬虫 \(计算机网络\)](#) [python爬虫](#)



文章被以下专栏收录



**python编程**

分享python有趣用法和爬虫

关注专栏



**Python程序员**

公众号：Python爱好者社区（python\_shequ） 欢迎投稿！

关注专栏

推荐阅读



**如何让Python爬虫一天抓取100万张网页**

猿人学


65 条评论

切换为时间排序

写下你的评论...



精选评论 (1)



Dwzb (作者)

09-18

我测试了一下发现CSDN网站改了，所以之前的代码不可用，目前CSDN首页已不适合初学者入门，我还没时间换一个例子。我看文中输出结果都以图片形式呈现了，读者应该可以借此了解基本爬虫的逻辑，这也是本文最想传达的部分。如果想要实操可以换一个网站，比如古诗文网 [古诗文大全](#) [古诗文网](#)

👍 1

评论 (65)




张灵老师

2018-04-05

讲的很清晰，感谢分享

👍 13




袜子君

2018-04-06

我实现了一下，可是只抓取了20条标题，可是首页的文章标题很多，想问下答主是什么原因:-|

👍 4



Dwzb (作者) 回复 袜子君

2018-04-06

刚进网页的时候就只有20条，你鼠标往下滚动的时候，它又加载了新的数据，这是一个动态加载问题，这个我以后的文章会专门说。现在的话你可以百度一下Python爬虫动态加载应该会有专门的文章。主体思想是看检查-network中的xhr，里面有一个文件是用json格式存储的数据，你鼠标下拉会发现xhr里又多出了类似新的文件，你可以遍历这些文件，从json中抓数据

👍 10



袜子君 回复 Dwzb (作者)

2018-04-07

哦哦，懂了！非常感谢~期待您后续更多优质的文章💖💖

赞同 874



👍 3

展开其他 1 条回复



US8.5

2018-04-07

不错 确实适合初学者 也可以让中学教师作为教案来用

👍 1



安世界

2018-04-10

写的真是简单易懂，非常适合小白（我）！期待更多的好文章！

👍 赞



MayD21

2018-04-16

感谢

👍 赞



胡刚鱼

2018-04-16

太基本了 以至于对有基础的没有任何收获

👍 赞



Dwzb (作者) 回复 胡刚鱼

2018-04-16

那是肯定的，这篇文章本来就是写给一点都不懂的人看的

👍 7



我家有个小茄子

2018-04-25

用的正则式爬取的网页，有一个问题就是只显示文字，看不到图片，不知道什么原因。

👍 赞



Dwzb (作者) 回复 我家有个小茄子

2018-04-25

什么意思 在哪里看不到图片

👍 赞



我家有个小茄子 回复 Dwzb (作者)

2018-04-26

就是抓取的网页有图片，但是抓下来看不到...

👍 赞

展开其他 3 条回复



谢天霸

2018-05-10

emm其实我觉得最难的是反爬.....

👍 赞



Dwzb (作者) 回复 谢天霸

2018-05-10

可能吧 没有接触非常非常深入 不知道最难的是什么 😊

👍 赞



谢天霸 回复 Dwzb (作者)

2018-05-10

我是被新浪给搞怕了2333

👍 赞

展开其他 1 条回复



NULL

2018-05-12

感谢分享，想请教下如果想把知乎收藏爬下来，而且希望文字和图片排版尽可能和网上效果一样的话，应该用什么库或者么方法呢？

👍 赞



Dwzb (作者) 回复 NULL

2018-05-12

▲ 赞同 874



这应该是要设计HTML和css代码的，要去学一下。另外如果要在web上展示，可以考虑一些web框架，比如flask

👍 赞



NULL 回复 Dwzb (作者)

2018-05-12

谢谢，只是想把自己的收藏备份下来，离线看看，方便查找。

👍 赞

展开其他 1 条回复



KaTrin

2018-05-13

赞。干货

👍 赞



Eric才不衰

2018-05-17

非常感谢

👍 赞



赖福子

2018-05-22

基本原理差不多，第一次用爬虫是用的matlab，urlread和正则表达式爬过几千个翼型数据，也是唯一一次。但是复杂的没深入过，比如有的要登录啊，一个网页进另一个网页啊这些不知道咋整

👍 赞



别气

2018-05-22

很感谢！

👍 赞



wili星星

2018-05-23

文章很棒

👍 赞



犹豫先生

2018-06-06

3Q3Q人生第一条虫虫成功。。。

👍 1



Admin张

2018-06-07

很适合新手学习

👍 赞



hyu jj

2018-06-26

这？这算采集吧，没见到自动爬的

👍 赞



Dwzb (作者) 回复 hyu jj

2018-06-26

这是爬虫的基本步骤，例子数据较少，但方法可以应用到大规模的爬取中

👍 赞



喵喵王

2018-06-30

感谢分享

👍 赞



空城梦

2018-07-01

感谢，很详尽

👍 赞