

州的先生

Python 爬虫实战 入门教程

浅显易懂、注重实操、广受好评的Python爬虫教程

从HTTP请求到HTML解析，从静态抓取到动态采集

腾讯新闻、今日头条、智联招聘、QQ空间等多个案例

州的先生 - Python编程应用实战系列教程

微信公众号：州的先生

《Python 爬虫实战入门教程》

作者：州的先生

微信公众号：州的先生 博客：<http://zmister.com>

2018/3/24

目录

目录	2
第一章：工具准备	3
1.1、基础知识	3
1.2、开发环境、	3
1.3、第三方依赖库	3
1.4、第三方库安装：	3
第二章：从一个简单的 HTTP 请求开始	7
2.1、为什么从 HTTP 请求开始	7
2.2、基本的 HTTP 概念	9
2.3、用 Python 进行 HTTP 请求	10
第三章：简单的 HTML 解析——爬取腾讯新闻	12
3.1、爬取腾讯新闻	12
第四章：使用 Cookie 模拟登录——获取电子书下载链接	17
4.1、使用 Cookie 爬取看看都电子书下载链接	18
第五章：获取 JS 动态内容——爬取今日头条	24
5.1、如何处理 JS 生成的网页内容	24
5.2、爬取今日头条	25
第六章：提高爬虫效率——并发爬取智联招聘	31
6.1、分析 URL 和页面结构	31
第七章：使用 Selenium——以抓取 QQ 空间好友说说为例	36
7.1、Selenium 简介	36
7.2、在 Python 中使用 Selenium 获取 QQ 空间好友说说	36
7.3、代码简析	39
第八章：数据储存——MongoDB 与 MySQL	42
8.1、MySQL	42
8.2、MongoDB	47
第九章：下一步	50

第一章：工具准备

1.1、基础知识

使用 Python 编写爬虫，当然至少得了解 Python 基本的语法，了解以下几点即可：

- 基本数据结构
- 数据类型
- 控制流
- 函数的使用
- 模块的使用

不需要过多过深的 Python 知识，仅此而已。个人推荐《Python 简明教程》：http://www.kuqin.com/abyteofpython_cn/、Python 官方的《Python 教程》http://python.usyiyi.cn/translate/python_352/tutorial/index.html

如果需要 PDF 版 Python 入门资料，可以关注我的微信公众号：州的先生，回复关键字：python 入门资料

1.2、开发环境、

- 操作系统：Windows 7
- Python 版本：Python 3.4
- 代码编辑运行环境：个人推荐 PyCharm 社区版，当然，Python 自带的 IDLE 也行，Notepad++ 亦可，只要自己使用得习惯。

1.3、第三方依赖库

- Requests：一个方便、简洁、高效且人性化的 HTTP 请求库
- BeautifulSoup：HTML 解析库
- Pymongo：MongoDB 的 Python 封装模块
- Selenium：一个 Web 自动化测试框架，用于模拟登录和获取 JS 动态数据
- pytesseract：一个 OCR 识别模块，用于验证码识别
- Pillow：Python 图像处理模块

1.4、第三方库安装：

上面列出的第三方模块大多可以通过 `pip install ××` 的方式直接安装，部分模块安装方式不一样，下面一一演示：

1.4.1、requests

```
pip install requests
```

```
<SCRAP_~1> F:\Virtual Machines\scrap_book>pip install requests
Collecting requests
  Downloading requests-2.12.4-py2.py3-none-any.whl (576kB)
```

1.4.2、BeautifulSoup

```
pip install bs4
```

```
<SCRAP_~1> F:\Virtual Machines\scrap_book>pip install bs4
Collecting bs4
  Collecting beautifulsoup4 (from bs4)
    Downloading beautifulsoup4-4.5.3-py3-none-any.whl (85kB)
      48% |#####| 40kB 60kB/s eta 0:00:
      60% |#####| 51kB 64kB/s eta 0:
      72% |#####| 61kB 76kB/s eta
      84% |#####| 71kB 74kB/
      96% |#####| 81kB 8
     100% |#####| 92kB
    92kB/s
Installing collected packages: beautifulsoup4, bs4
Successfully installed beautifulsoup4-4.5.3 bs4-0.0.1
```

1.4.3、Pymongo

```
pip install pymongo
```

```
<SCRAP_~1> F:\Virtual Machines\scrap_book>pip install pymongo
Collecting pymongo
  Downloading pymongo-3.4.0-cp34-none-win32.whl (264kB)
      38% |#####| 102kB 6.2kB/s eta 0:00:2
      42% |#####| 112kB 6.8kB/s eta 0:00:
      46% |#####| 122kB 6.0kB/s eta 0:00:
      50% |#####| 133kB 5.9kB/s eta 0:
      54% |#####| 143kB 6.0kB/s eta 0
      58% |#####| 153kB 14kB/s eta 0
      62% |#####| 163kB 13kB/s eta
      65% |#####| 174kB 13kB/s eta
      69% |#####| 184kB 13kB/s e
      73% |#####| 194kB 12kB/s
      77% |#####| 204kB 12kB/s
      81% |#####| 215kB 12kB/
      85% |#####| 225kB 16k
      89% |#####| 235kB 18
      93% |#####| 245kB 1
      96% |#####| 256kB
     100% |#####| 266k
    B 24kB/s
Installing collected packages: pymongo
Successfully installed pymongo-3.4.0
```

1.4.4、Selenium

```
pip install selenium
```

```
<SCRAP_~1> F:\Virtual Machines\scrap_book>pip install selenium
Collecting selenium
  Downloading selenium-3.0.2-py2.py3-none-any.whl (915kB)
    35% |#####| 327kB 3.4MB/s eta 0:00:00
    36% |#####| 337kB 5.1MB/s eta 0:00:00
    38% |#####| 348kB 5.1MB/s eta 0:00:00
    39% |#####| 358kB 10.2MB/s eta 0:00:00
```

1.4.5、Pillow

1. 打开 <http://www.lfd.uci.edu/~gohlke/pythonlibs/>
2. 搜索找到“pillow”
3. 根据自己系统的版本选择对应的下载包

Pillow, a replacement for PIL, the Python Image Library, which provides image processing functionality and supports many file formats. Use `from PIL import Image` instead of `import Image`.

[Pillow-4.0.0-cp27-cp27m-win32.whl](#)
[Pillow-4.0.0-cp27-cp27m-win_amd64.whl](#)
[Pillow-4.0.0-cp34-cp34m-win32.whl](#)
[Pillow-4.0.0-cp34-cp34m-win_amd64.whl](#)
[Pillow-4.0.0-cp35-cp35m-win32.whl](#)
[Pillow-4.0.0-cp35-cp35m-win_amd64.whl](#)
[Pillow-4.0.0-cp36-cp36m-win32.whl](#)
[Pillow-4.0.0-cp36-cp36m-win_amd64.whl](#)
[Pillow-4.0.0-pp256-pypy_41-win32.whl](#)

4. 安装

```
pip install Pillow-4.0.0-cp34-cp34m-win32.whl
```

```
<SCRAP_~1> F:\Virtual Machines\scrap_book>pip install Pillow-4.0.0-cp34-cp34m-win32.whl
Processing f:\virtual machines\scrap_book\pillow-4.0.0-cp34-cp34m-win32.whl
Collecting olefile (from Pillow==4.0.0)
  Downloading olefile-0.43.zip (119kB)
    42% |#####| 51kB 26kB/s eta 0:00:03
    51% |#####| 61kB 24kB/s eta 0:00:00
    59% |#####| 71kB 28kB/s eta 0:00:00
    68% |#####| 81kB 20kB/s eta 0:00:00
    77% |#####| 92kB 14kB/s
    85% |#####| 102kB 14kB/s
    94% |#####| 112kB
    100% |#####| 122kB
122kB 14kB/s
Building wheels for collected packages: olefile
  Running setup.py bdist_wheel for olefile ... done
  Stored in directory: C:\Users\PC\AppData\Local\pip\Cache\wheels\3c\0e\53\f89b4f8e637b40ca1fe13e1f1f6fd7402c411504e25bbca8a5
Successfully built olefile
Installing collected packages: olefile, Pillow
Successfully installed Pillow-4.0.0 olefile-0.43

<SCRAP_~1> F:\Virtual Machines\scrap_book>
```

1.4.6、pytesseract

1. 安装 pytesseract

```
pip install pytesseract
```

```
<SCRAP_~1> F:\Virtual Machines\scrap_book>pip install pytesseract  
Collecting pytesseract  
Installing collected packages: pytesseract  
Successfully installed pytesseract-0.1.6
```

2. 安装 tesseract

下载并安装: <https://tesseract-ocr.googlecode.com/files/tesseract-ocr-setup-3.02.02.exe>

这样，我们的准备工作就基本完成，如果有另外的需求，在实战中再进行安装，接下来就可以实战 Python 爬虫了。

第二章：从一个简单的 HTTP 请求开始

2.1、为什么从 HTTP 请求开始

无论我们通过浏览器打开网站、访问网页，还是通过脚本对 URL 网址进行访问，本质上都是对 HTTP 服务器的请求，浏览器上所呈现的、控制台所显示的都是 HTTP 服务器对我们请求的响应。

以打开我的个人网站为例，我们在地址栏输入“zmister.com”，浏览器上呈现的是下图：



图 1 州的先生博客

我们在浏览器上按 F12 打开网页调试工具，选择“network”选项卡，可以看到我们对 zmister.com 的请求，以及 zmister.com 给我们的响应：

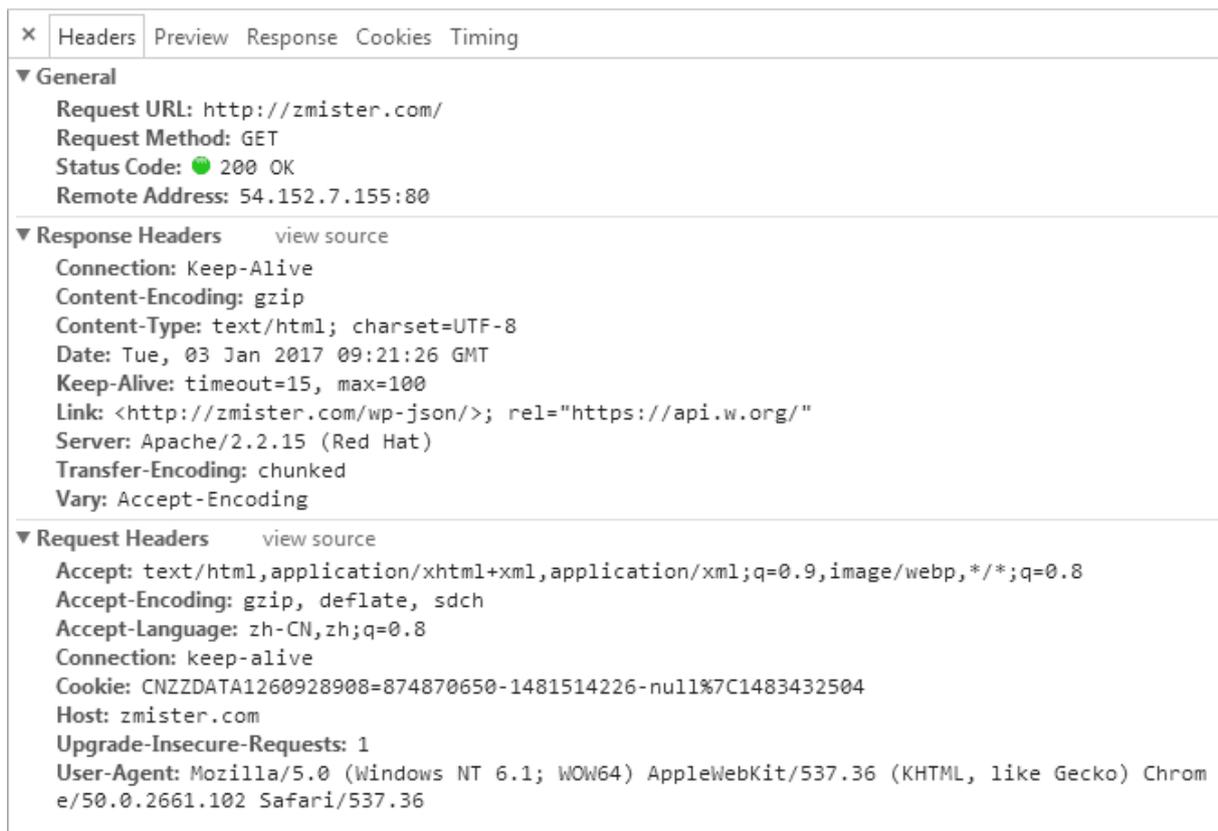
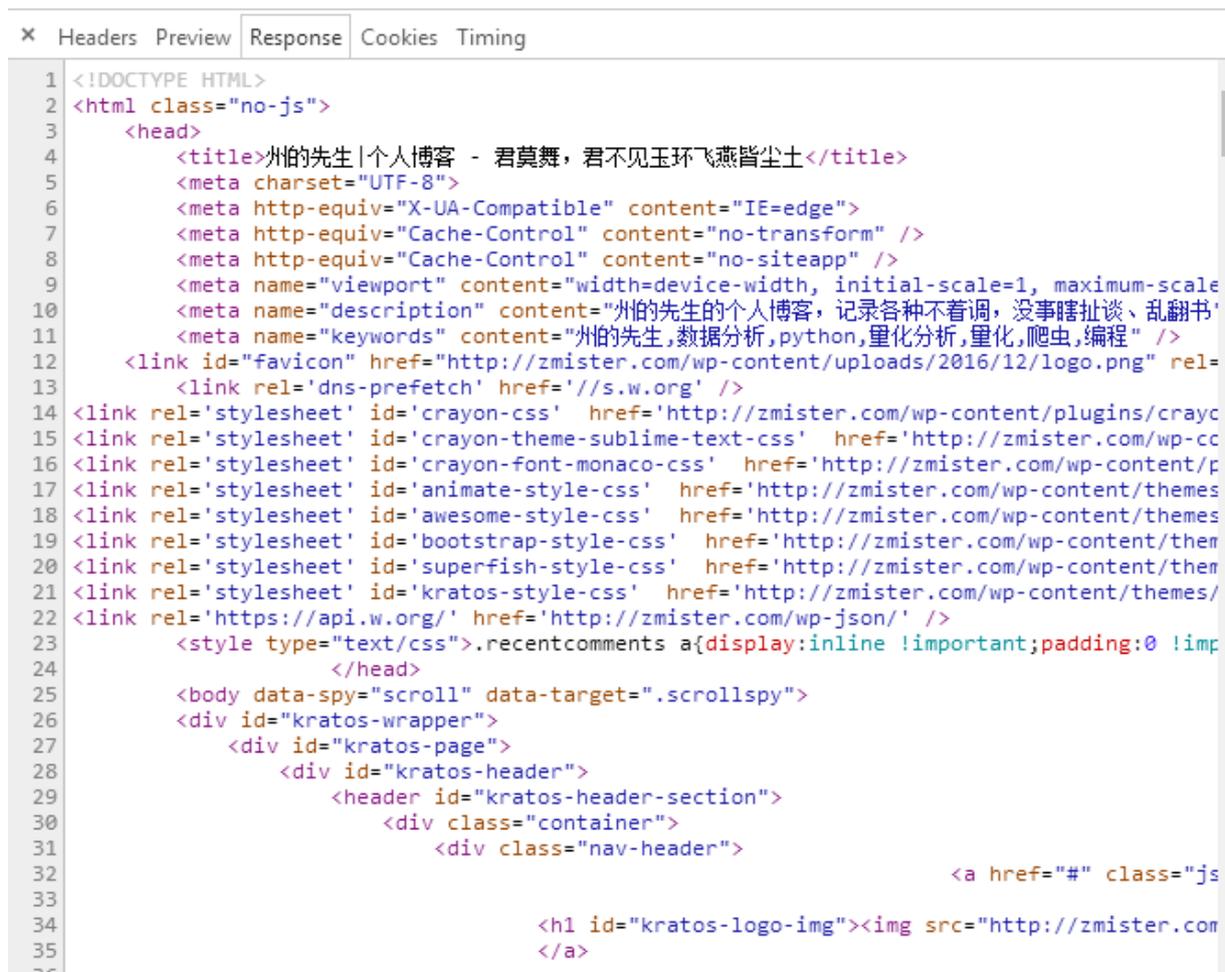


图 2 zmister.com 的请求与响应头



```
1 <!DOCTYPE HTML>
2 <html class="no-js">
3   <head>
4     <title>州的先生|个人博客 - 君莫舞，君不见玉环飞燕皆尘土</title>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta http-equiv="Cache-Control" content="no-transform" />
8     <meta http-equiv="Cache-Control" content="no-siteapp" />
9     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
10    <meta name="description" content="州的先生的个人博客，记录各种不着调，没事瞎扯谈、乱翻书" />
11    <meta name="keywords" content="州的先生,数据分析,python,量化分析,量化,爬虫,编程" />
12    <link id="favicon" href="http://zmister.com/wp-content/uploads/2016/12/logo.png" rel="
13      <link rel='dns-prefetch' href='//s.w.org' />
14    <link rel='stylesheet' id='crayon-css' href='http://zmister.com/wp-content/plugins/crayc
15    <link rel='stylesheet' id='crayon-theme-sublime-text-css' href='http://zmister.com/wp-cc
16    <link rel='stylesheet' id='crayon-font-monaco-css' href='http://zmister.com/wp-content/p
17    <link rel='stylesheet' id='animate-style-css' href='http://zmister.com/wp-content/themes
18    <link rel='stylesheet' id='awesome-style-css' href='http://zmister.com/wp-content/themes
19    <link rel='stylesheet' id='bootstrap-style-css' href='http://zmister.com/wp-content/ther
20    <link rel='stylesheet' id='superfish-style-css' href='http://zmister.com/wp-content/ther
21    <link rel='stylesheet' id='kratos-style-css' href='http://zmister.com/wp-content/themes/
22    <link rel='https://api.w.org/' href='http://zmister.com/wp-json/' />
23    <style type="text/css">.recentcomments a{display:inline !important;padding:0 !imp
24  </head>
25  <body data-spy="scroll" data-target=".scrollspy">
26    <div id="kratos-wrapper">
27      <div id="kratos-page">
28        <div id="kratos-header">
29          <header id="kratos-header-section">
30            <div class="container">
31              <div class="nav-header">
32                <a href="#" class="js
33
34              <h1 id="kratos-logo-img">>> import requests
>>> url = "http://zmister.com"
>>> data = requests.get(url)
>>> data.status_code
200
>>> data.content
b'<!DOCTYPE HTML>\r\n<html class="no-js">\r\n<head>\r\n\t\t\t<title>\xe5\xb7\x9e
\xe7\x9a\x84\xe5\x85\x88\xe7\x94\x9f/\xe4\xb8\xaa\xe4\xba\xba\xe5\x8d\x9a\xe5\xa
e\xa2 - \xe5\x90\x9b\xe8\x8e\xab\xe8\x88\xe9\xef\xbc\x8c\xe5\x90\x9b\xe4\xb8\x8d
\xe8\xa7\x81\xe7\x8e\x89\xe7\x8e\xaf\xe9\xa3\xe7\x87\x95\xe7\x9a\x86\xe5\xb0
\x98\xe5\x9c\x9f</title>\r\n\t\t\t<meta charset="UTF-8">\r\n\t\t\t<meta http-equiv="
X-UA-Compatible" content="IE=edge">\r\n\t\t\t\t\t<meta http-equiv="Cache-Control"
content="no-transform" /> \r\n\t\t\t\t\t<meta http-equiv="Cache-Control" content="
no-siteapp" /> \r\n\t\t\t\t\t<meta name="viewport" content="width=device-width, init
ial-scale=1, maximum-scale=1">\r\n\t\t\t\t\t<meta name="description" content="\xe5\xb7
\x9e\xe7\x9a\x84\xe5\x85\x88\xe7\x94\x9f/\xe7\x9a\x84\xe4\xb8\xaa\xe4\xba\xba\xe5
\x8d\x9a\xe5\xae\xa2\xef\xbc\x8c\xe8\xae\xb0\xe5\xbd\x95\xe5\x90\x84\xe7\xa7\x8d
\xe4\xb8\x8d\xe7\x9d\x80\xe8\xb0\x83\xef\xbc\x8c\xe6\xb2\xal\xe4\xba\x8b\xe7\x9e
\x8e\xe6\x89\xaf\xe8\xb0\x88\xe3\x80\x81\xe4\xb9\xb1\xe7\xbf\xbb\xe4\xb9\xa6" />
\r\n\t\t\t\t\t<meta name="keywords" content="\xe5\xb7\x9e\xe7\x9a\x84\xe5\x85\x88\xe7
\x94\x9f, \xe6\x95\xb0\xe6\x8d\xae\xe5\x88\x86\xe6\x9e\x90, python, \xe9\x87\x8f\xe5
\x8c\x96\xe5\x88\x86\xe6\x9e\x90, \xe9\x87\x8f\xe5\x8c\x96, \xe7\x88\xac\xe8\x99\x
ab, \xe7\xbc\x96\xe7\xa8\x8b" />\r\n\t\t\t\t\t<link id="favicon" href="http://zmister.com
/wp-content/uploads/2016/12/logo.png" rel="icon" type="image/x-icon" />\r\n\t\t\t\t\t<
link rel="dns-prefetch" href="//s.w.org" />\n<link rel="stylesheet" id="c
rayon-css" href="http://zmister.com/wp-content/plugins/crayon-syntax-highligh
ter/css/min/crayon.min.css?ver=2.7.2_beta" type="text/css" media="all" />
\n<link rel="stylesheet" id="crayon-theme-sublime-text-css" href="http://zm
ister.com/wp-content/plugins/crayon-syntax-highlighter/themes/sublime-text/subli
me-text.css?ver=2.7.2_beta" type="text/css" media="all" />\n<link rel="st
ylesheet" id="crayon-font-monaco-css" href="http://zmister.com/wp-content/p
lugins/crayon-syntax-highlighter/fonts/monaco.css?ver=2.7.2_beta" type="text/
css" media="all" />\n<link rel="stylesheet" id="animate-style-css" href=
"http://zmister.com/wp-content/themes/Kratos-master/css/animate.min.css?ver=3.5
```

对这些凌乱的 html 源码进行处理，就需要使用到 BeautifulSoup 模块了，下一章咱们继续。

□



第三章：简单的 HTML 解析——爬取腾讯新闻

上一章咱们使用 Python 实现了一个简单的 HTTP 请求。瞧着简单，爬虫就是模拟人打开一个个 URL 浏览一个个网页来爬取数据的，一个成功的 HTTP 请求，就是一个爬虫的基础。

接下来，咱们以一个实际的例子：爬取腾讯新闻，来介绍使用 BeautifulSoup 对 HTML 进行解析处理。

3.1、爬取腾讯新闻

3.1.1、寻找数据特征

腾讯新闻的网址 URL 为：<http://news.qq.com/> 我们打开网页看看：



我们需要爬取这个页面每一条新闻的标题，鼠标右击一条新闻的标题，选择“审查元素”，出现下图的窗口：



图片中红框的位置就是那一条新闻标题在 HTML 中的结构、位置和表现形式：

```
<a target="_blank" class="linkto" href="http://news.qq.com/a/20170104/001280.htm">北京 2016 年 39 天重污染 PM2.5 仍超国标一倍</a>
```

它上一级元素为：<em class="f14 l24">，再上一级元素为：<div class="text">

我们再看另一条新闻的标题，发现它的结构和之前我们分析的新闻标题的结构是一样的：

```
<div class="text">
<em class="f14 l24">
<a target="_blank" class="linkto" href="http://news.qq.com/a/20170104/000968.htm">台拟将
蔡英文贺岁春联“自自冉冉”一词列入辞典
</a>
</em>
</div>
```

通过这些信息，我们就可以确定新闻标题在 HTML 文档中的位置。

接下来，我们开始使用 Python 对腾讯新闻标题进行爬取

3.1.2、编写爬取代码

首先呈上完整的代码：

```
# coding:utf-8
# 引入相关模块
import requests
from bs4 import BeautifulSoup

url = "http://news.qq.com/"
# 请求腾讯新闻的 URL，获取其 text 文本
wbdata = requests.get(url).text
# 对获取到的文本进行解析
```

```
soup = BeautifulSoup(wbdata, 'lxml')
# 从解析文件中通过 select 选择器定位指定的元素, 返回一个列表
news_titles = soup.select("div.text > em.f14 > a.linkto")

# 对返回的列表进行遍历
for n in news_titles:
    # 提取出标题和链接信息
    title = n.get_text()
    link = n.get("href")
    data = {
        '标题': title,
        '链接': link
    }
    print(data)
```

运行程序, 获取到的部分结果为如下所示:

```
C:\Python34\python.exe H:/python/scrap/scrap_tech_book/scrap_tennetNews.py
{'标题': '2017 年首个断崖降级官员曝光 他犯了什么事?', '链接': 'http://news.qq.com/a/20170104/002209.htm'}
{'标题': '湄公河武装执法纪实: 队员吃饭都要死死盯着前方', '链接': 'http://news.qq.com/a/20170104/007148.htm'}
{'标题': '河南台前县出台彩礼“指导标准”: 总数不得高于 6 万', '链接': 'http://news.qq.com/a/20170104/001693.htm'}
{'标题': '摆射击摊获刑老太谈上诉原因: 怕预订的咸菜浪费了', '链接': 'http://news.qq.com/a/20170104/013261.htm'}
{'标题': '美国联军承认在叙伊有“误伤”: 打死近二百平民', '链接': 'http://news.qq.com/a/20170104/005283.htm'}
{'标题': '辽宁黑山企业人员退休先缴四百元: 未经省级审批', '链接': 'http://news.qq.com/a/20170104/006323.htm'}
{'标题': '罗布泊遇难者李中华的骨灰回家: 身份仍有 3 大谜团', '链接': 'http://news.qq.com/a/20170104/003139.htm'}
{'标题': '嫌楼上邻居吵, 这家人买“震楼神器”反击', '链接': 'http://news.qq.com/a/20170104/001582.htm'}
{'标题': '12 岁女孩疑似被弃 因妈妈买不起她的火车票?', '链接': 'http://news.qq.com/a/20170104/004622.htm'}
{'标题': '向美国商务部说“NO” 这家中国民企打赢美国官司', '链接': 'http://news.qq.com/a/
```

```
20170104/000322. htm'}  
{ '标题': '台拟将蔡英文贺岁春联“自自冉冉”一词列入辞典', '链接': 'http://news.qq.com/a/20170104/000968. htm'}  
{ '标题': '新华社：姚明资历经验尚浅 任篮协主席尚有障碍', '链接': 'http://news.qq.com/a/20170104/000330. htm'}  
{ '标题': '新京报：法院拍卖 BB 弹玩具枪，自己人也糊涂？', '链接': 'http://news.qq.com/a/20170104/001344. htm'}
```

这正是我们所需要的。虽然代码很简单，但还是做一点点讲解，方便刚刚接触的同学。

3.1.3、代码解析

```
# coding:utf-8
```

首先，我们定义了文件的编码形式为 UTF-8，以避免一些编码错误导致中文乱码。

```
import requests  
from bs4 import BeautifulSoup
```

然后，我们引入了相关的模块，`requests` 用于 HTTP 请求，`BeautifulSoup` 用于解析 HTML 响应。

```
url = "http://news.qq.com/"
```

设置一个变量 `url`，值为腾讯新闻的 URL

```
wbdata = requests.get(url).text
```

使用 `requests.get()` 对 URL 发起 GET 方式的 HTTP 请求，并使用 `text()` 方法获取响应的文本内容，最后将其赋值给变量 `wbdata`。

```
soup = BeautifulSoup(wbdata, 'lxml')
```

使用 `BeautifulSoup` 对响应文本 `wbdata` 进行解析处理，这里使用的是 `lxml` 库，如果没有安装，可以使用 Python 自带的 `html.parser`，效果也是一样的。

```
news_titles = soup.select("div.text > em.f14 > a.linkto")
```

在解析后的文本中，使用 `select` 选择器，在文本中选择指定的元素，通常我们还会使用 `find()` 和 `findall()` 方法来进行元素选择。这一步返回的为一个列表，列表内的元素为匹配的元素的 HTML 源码。

```
for n in news_titles:
    # 提取出标题和链接信息
    title = n.get_text()
    link = n.get("href")
    data = {'标题':title, '链接':link}
    print(data)
```

对结果列表进行遍历，再从遍历的元素中提取出数据，`get("href")`表示获取属性名为“href”的属性值，`get_text()`表示获取标签的文本信息。

这样，一个简单的腾讯新闻爬虫就完成了，如果对 `requests` 模块和 `BeautifulSoup` 模块有更加深的学习欲望，可以查看它们的官方文档：

requests 官方文档（中文）：http://docs.python-requests.org/zh_CN/latest/

BeautifulSoup 文档（中文）：

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/index.zh.html>

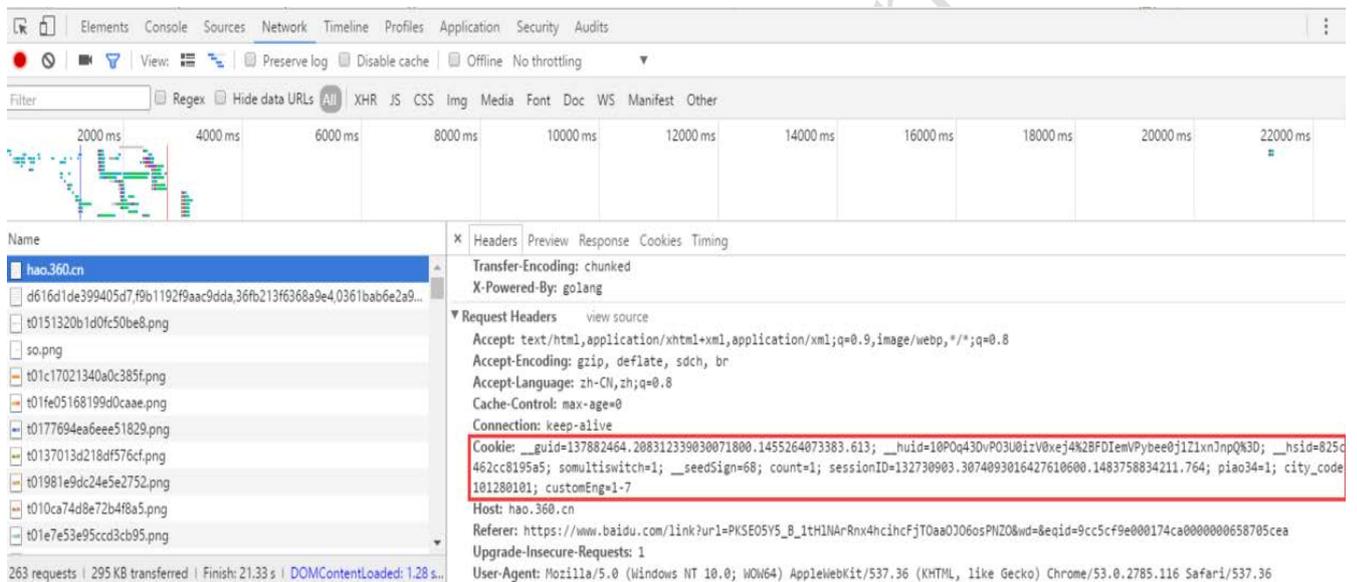
□

第四章：使用 Cookie 模拟登录——获取电子书下载链接

在实际情况中，很多网站的内容都是需要登录之后才能看到，如此我们就需要进行模拟登录，使用登录后的状态进行爬取。这里就需要使用到 Cookie。

现在大多数的网站都是使用 Cookie 跟踪用户的登录状态，一旦网站验证了登录信息，就会将登录信息保存在浏览器的 cookie 中。网站会把这个 cookie 作为验证的凭据，在浏览网站的页面是返回给服务器。

因为 cookie 是保存在本地的，自然 cookie 就可以进行篡改和伪造，暂且不表，我们先来看看 Cookie 长什么样子。打开网页调试工具，随便打开一个网页，在“network”选项卡，打开一个链接，在 headers 里面：



我们复制出来看看：

```
__guid=137882464.208312339030071800.1455264073383.613;
__huid=10P0q43DvP03U0izV0xej4%2BFDIemVpybee0j1Z1xnJnpQ%3D;
__hsid=825c7462cc8195a5;
somultiswitch=1;
__seedSign=68;
count=1; sessionId=132730903.3074093016427610600.1483758834211.764;
piao34=1;
city_code=101280101;
customEng=1-7
```

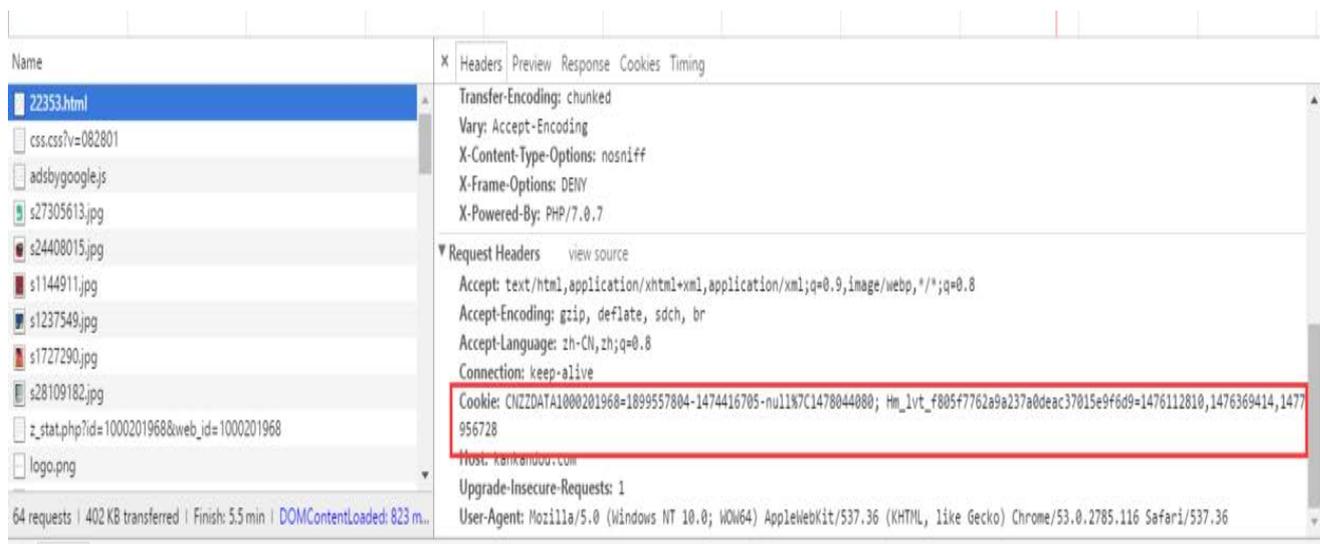
Cookie 由一个个键值对组成。接下来，我们以看看豆 (<http://kankandou.com>) 的一本书籍的详情页为例，讲解一下 Cookie 的使用。

4.1、使用 Cookie 爬取看看都电子书下载链接

看看豆是一个电子书下载网站，自己 Kindle 上的书多是从这上面找寻到的。示例网址为：<https://kankandou.com/book/view/22353.html> 正常情况下，未登录用户是看不到下载链接的，比如这样：

The screenshot shows a book page for '宇宙是猫咪酣睡的梦' (The Universe is the Cat Sleeping Dream). The page includes a navigation bar with categories like '小说', '文学', '历史', etc. The book information section shows the title, author '余思', and tags '中国 随笔 中国小说'. There is a button '收藏本书' and an advertisement for '新户0首存, 0成本炒黄金'. Below the book information, there is a '书籍导购' section with a button '去亚马逊购买《宇宙是猫咪酣睡的梦》'. The '书籍介绍' section is partially visible at the bottom.

隐藏了书籍的下载链接。其头信息如下：



我们再看看登录之后的页面：



下载链接已经显示出来了，我们看看头信息的 Cookie 部分


```
<h2 class="title">书籍导购</h2>
<div class="object">
<div class="object-files">
<div class="amazon"><a href="http://www.amazon.cn/gp/search?ie=UTF8&camp=536&creative=3200&index=books&keywords=宇宙是猫咪酣睡的梦&linkCode=ur2&tag=kkd8-23" target="_blank" title="如果您喜欢《宇宙是猫咪酣睡的梦》这本书，请去亚马逊购买。">去亚马逊购买《宇宙是猫咪酣睡的梦》</a></div>
</div>
</div>
```

如同我们在未登录状态使用浏览器访问这个网址一样，只显示了亚马逊的购买链接，而没有电子格式的下链接。我们尝试使用以下登录之后的 Cookie：
使用 Cookie 有两种方式：

1、直接将 Cookie 写在 header 头部

完整代码如下：

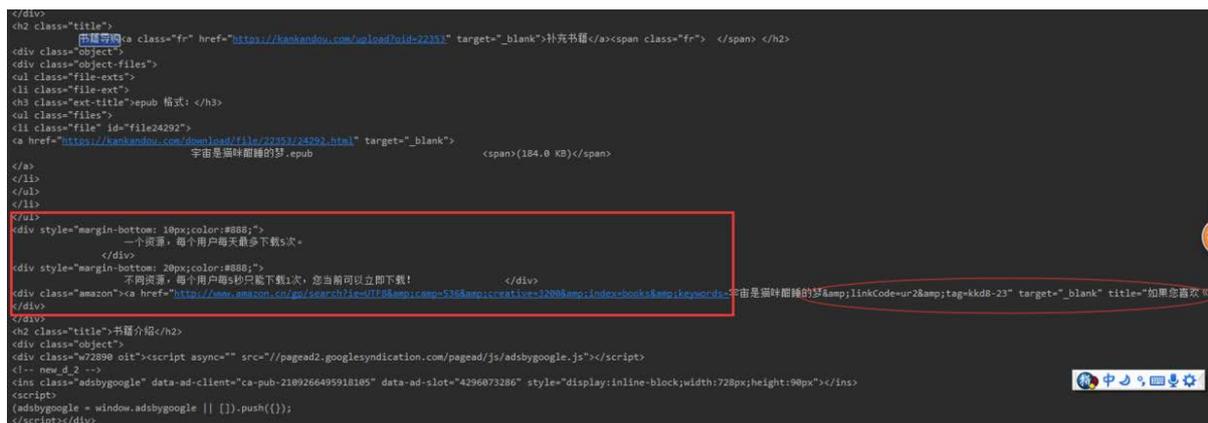
```
# coding:utf-8
import requests
from bs4 import BeautifulSoup

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)
```

上述代码返回了对页面（<https://kankandou.com/book/view/22353.html>）的响应，我们搜索响应的代码，



红色椭圆的部分与未带 Cookie 访问是返回的 HTML 一致，为亚马逊的购买链接，红色矩形部分则为电子书的下载链接，这是在请求中使用的 Cookie 才出现的对比实际网页中的模样，与用网页登录查看的显示页面是一致的。功能完成。接下来看看第二种方式

2、使用 requests 插入 Cookie

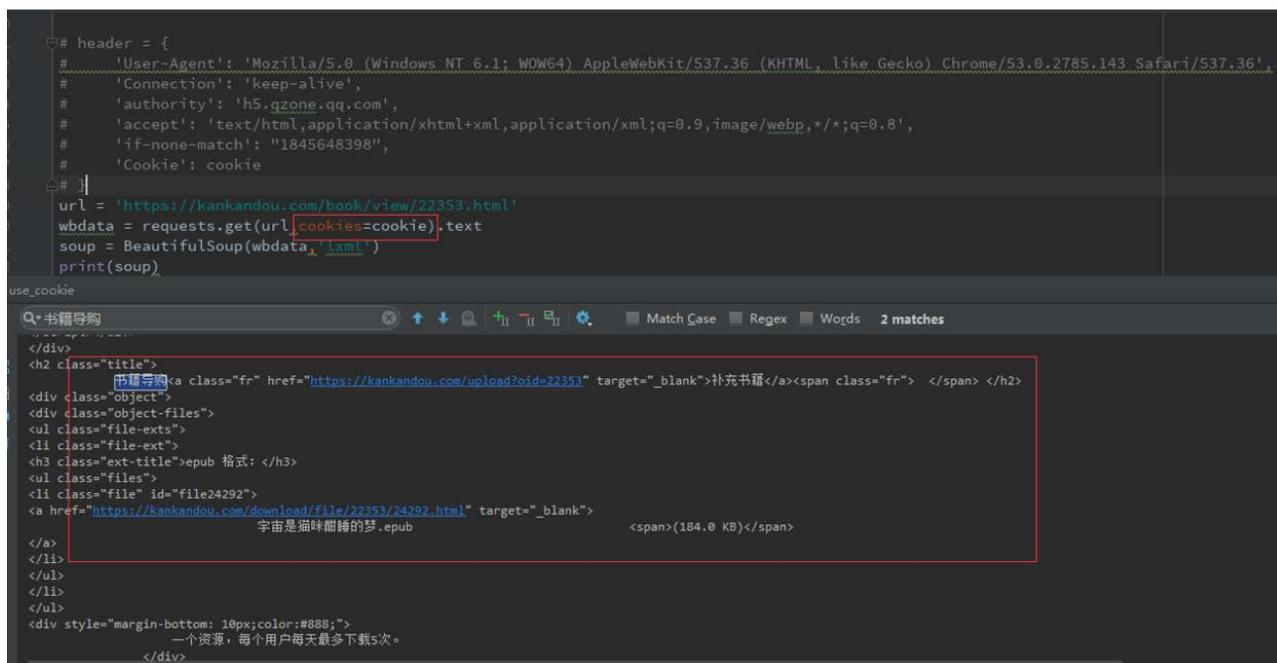
完整代码如下：

```
# coding:utf-8
import requests
from bs4 import BeautifulSoup

cookie = {
    "cission": "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)
```

如此获取到的也是登录后显示的 HTML：



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到的网页和资源了。这里只是简单介绍了对 Cookie 的使用，关于 Cookie 如何获取，手动复制是一种办法，通过代码获取，需要使用到 Selenium，接下来的章节会讲解，这里暂且不表。

第五章：获取 JS 动态内容—爬取今日头条

5.1、如何处理 JS 生成的网页内容

之前我们爬取的网页，多是 HTML 静态生成的内容，直接从 HTML 源码中就能找到看到的数据和内容，然而并不是所有的网页都是这样的。

有一些网站的内容由前端的 JS 动态生成，由于呈现在网页上的内容是由 JS 生成而来，我们能够在浏览器上看得见，但是在 HTML 源码中却发现不了。比如今日头条：浏览器呈现的网页是这样的：



查看源码，却是这样的：

```
60 <div riot-tag="login"></div>
61 <div id="toast"></div>
62
63
64 <div class="y-box container">
65   <div class="y-left index-channel">
66     <div riot-tag="wchannel"></div>
67   </div>
68   <div class="y-left index-content">
69
70     <div id="_carousel" riot-tag="carouselBox" style="height: 300px; margin-bottom: 16px;"></div>
71
72
73
74     <div riot-tag="subchannel" id="subchannel"></div>
75     <div riot-tag="feedBox"></div>
76   </div>
77   <div class="y-right index-modules">
78     <div id="module-place"></div>
79     <div class="module-inner" id="module-inner">
80       <div riot-tag="wsearch" style="margin-bottom: 16px;"></div>
81       <div riot-tag="tbanneradd"></div>
82       <div style="height: 366px; overflow: hidden;">
83         <div id="m-hotNews">
84           <div riot-tag="hotNews"></div>
85           <div id="adindexhover" style="margin-top: -10px; padding: 0 20px;">
86             <iframe src="about://blank" onload="var adWrap=this;setTimeout(function(){if(!adWrap.loaded){adWrap.src='//www.toutiao.com/api/pc/adsafe/'+ '#group=ir
scrolling= "no"}</iframe>
87 </div>
88
89 </div>
90 </div>
91 <div style="margin-bottom:16px; padding: 20px; background-color: #f4f5f6;">
92 <iframe src="about://blank" onload="var adWrap=this;setTimeout(function(){if(!adWrap.loaded){adWrap.src='//www.toutiao.com/api/pc/adsafe/'+ '#group=ir
</iframe>
93 </div>
94 </div>
95
96 <div riot-tag="adTab"></div>
97 <div riot-tag="whotvideo"></div>
98 <div style="margin-bottom:16px; padding: 20px; background-color: #f4f5f6;">
99 <iframe src="about://blank" onload="var adWrap=this;setTimeout(function(){if(!adWrap.loaded){adWrap.src='//www.toutiao.com/api/pc/adsafe/'+ '#group=index_
</iframe>
100 </div>
101
102 <div riot-tag="whotpicture"></div>
103 <div class="friendLink module" ga_event="click_friend_link">
104 <div class="module-head link-head">友情链接</div>
```

网页的新闻在 HTML 源码中一条都找不到，全是由 JS 动态生成加载。遇到这种情况，我们应该如何对网页进行爬取呢？有两种方法：

- 1、从网页响应中找到 JS 脚本返回的 JSON 数据；
- 2、使用 Selenium 对网页进行模拟访问

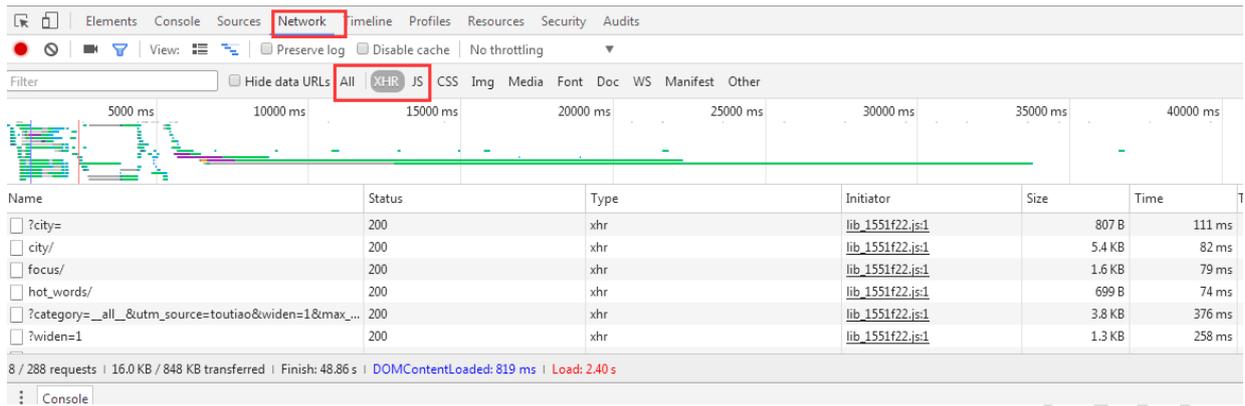
在此只对第一种方法作介绍，关于 Selenium 的使用，后面有专门的一篇。

5.2、爬取今日头条

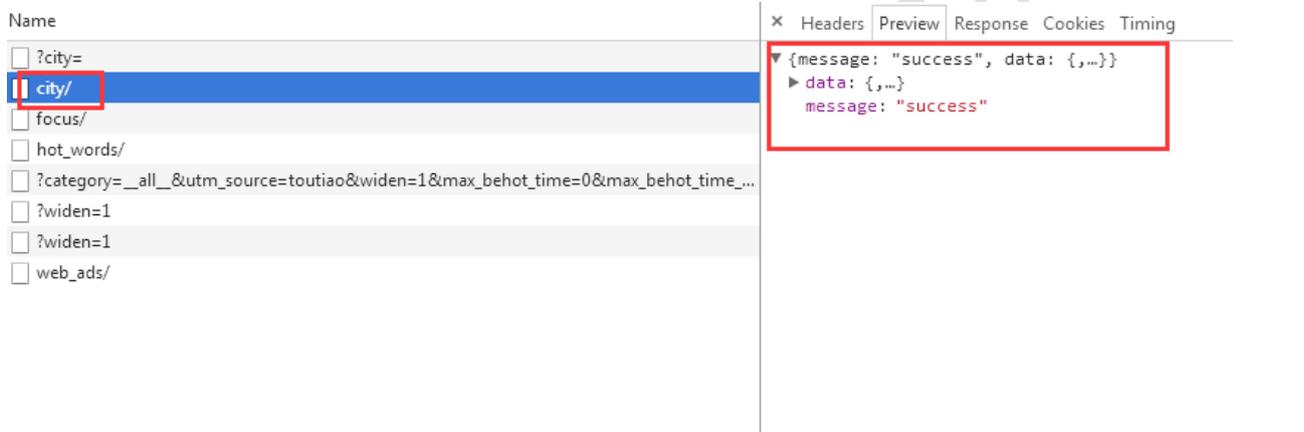
即使网页内容是由 JS 动态生成加载的，JS 也需要对某个接口进行调用，并根据接口返回的 JSON 数据再进行加载和渲染。所以我们可以找到 JS 调用的数据接口，从数据接口中找到网页中最后呈现的数据。就以今日头条为例来演示：

5.2.1、从找到 JS 请求的数据接口

在浏览器页面按 F12 键打开网页调试工具：



选择“网络”选项卡后，发现有很多响应，我们筛选一下，只看 XHR 响应。（XHR 是 Ajax 中的概念，表示 XMLHttpRequest）然后我们发现少了很多链接，随便点开一个看看：我们选择 city，预览中有一串 json 数据：

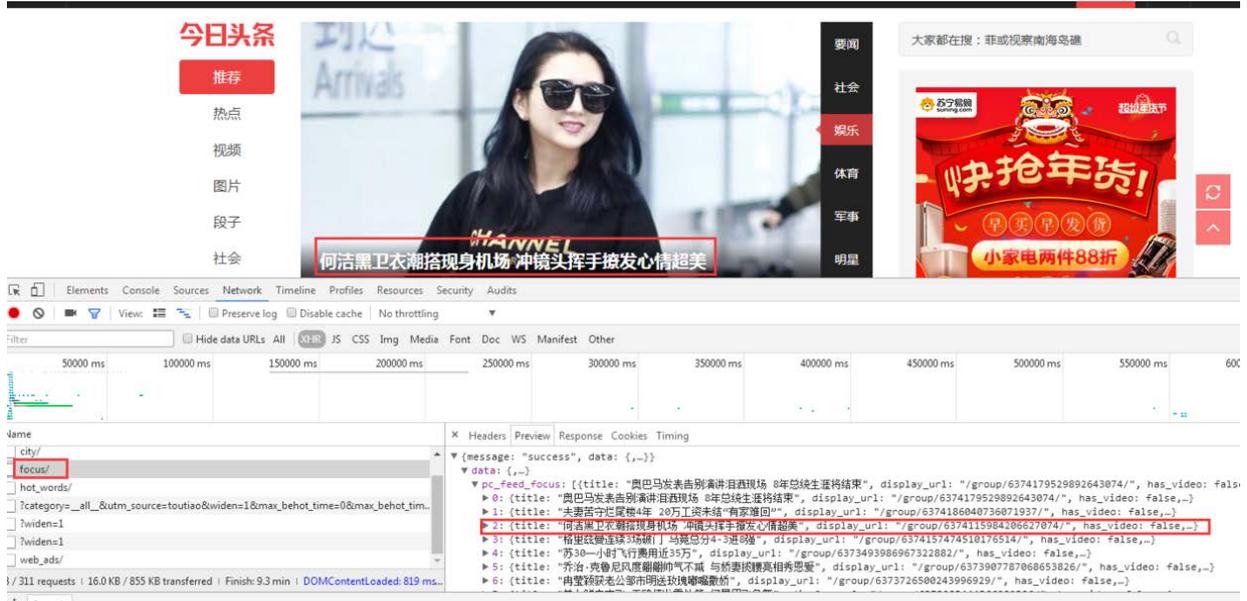


我们再点开看看：

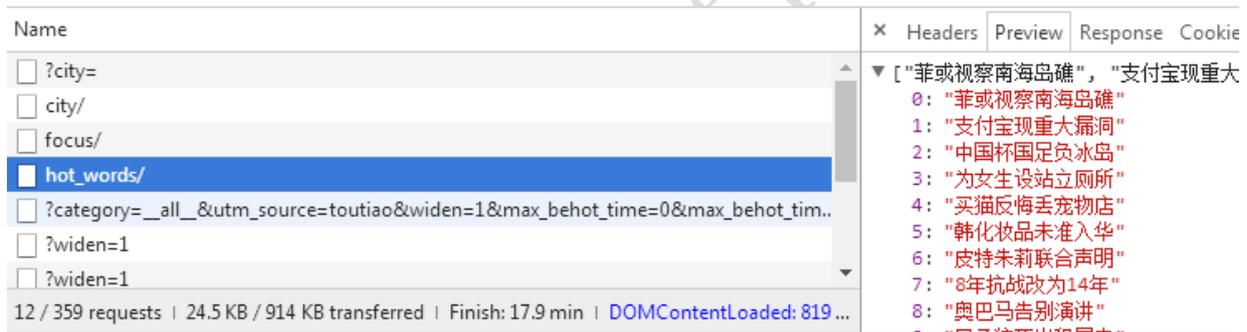


原来全都是城市的列表，应该是加载地区新闻之用的。现在大概了解了怎么找 JS 请求的接口的吧？

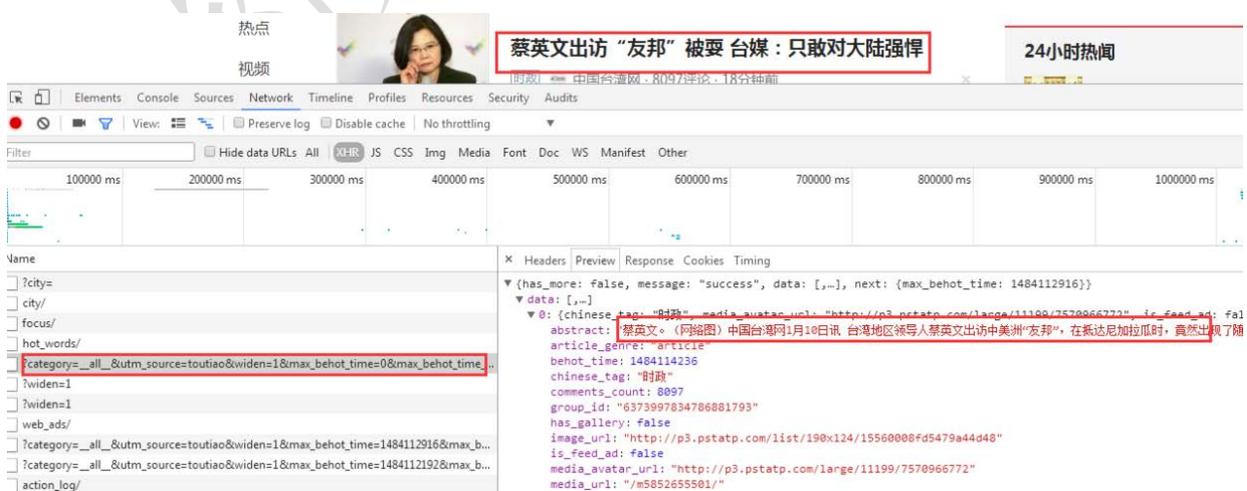
但是刚刚我们并没有发现想要的新闻，再找找看：有一个 focus，我们点开看看：



与首页的图片新闻呈现的数据是一样的，那么数据应该就在这里面了。看看其他的链接：



这应该是热搜关键词，再看看下一个链接：




```
news = data['data']['pc_feed_focus']

for n in news:
    title = n['title']
    img_url = n['image_url']
    url = n['media_url']
    print(url, title, img_url)
```

返回出来的结果如下：

```
scrap.js
C:\Python34\python.exe H:/python/scrap/scrap_tech_book/scrap_js.py
http://toutiao.com/m5784742177 奥巴马发表告别演讲泪洒现场 8年总统生涯将结束 http://p3.pstatp.com/origin/150e00025cf3a0e34c3b
http://toutiao.com/m5739097906 夫妻苦守烂尾楼4年 20万工资未结“有家难回” http://p3.pstatp.com/origin/150d00025ba4d91b90b2
http://toutiao.com/m6159077688 何洁黑衣潮搭现身机场 冲镜头挥手撒发心情超美 http://p1.pstatp.com/large/135400115133193691f7
http://toutiao.com/m6159306637 格里兹曼连续3场破门 马竞总分4-3进8强 http://p2.pstatp.com/large/1353001155987d1639a8
http://toutiao.com/m4166980049 苏30一小时飞行费用近35万 http://p3.pstatp.com/origin/13540011551f8b7026c5
http://toutiao.com/m6159077688 乔治·克鲁尼风度翩翩帅气不减 与娇妻挽腰亮相秀恩爱 http://p9.pstatp.com/large/1354001104ede74c65f7
http://toutiao.com/m6159077688 冉莹颖获老公邹市明送玫瑰嘟嘴撒娇 http://p1.pstatp.com/large/13540010c3b0cacbc8cf
http://toutiao.com/m6159077688 美女鲜肉齐飞 王鸥撩发露灿笑 何炅眉飞色舞 http://p3.pstatp.com/large/135400102adf8dccbdb9
http://toutiao.com/m6159077688 宁泽涛人气爆棚机场戴口罩任粉丝拍照 狂收情书 http://p3.pstatp.com/large/123300189de26ec38fc2
http://toutiao.com/m5738017030 林宥嘉、丁文琪结婚 唱着失恋情歌的少年终成幸福新郎 http://p1.pstatp.com/large/150e0000c6a6ac426bed

Process finished with exit code 0
```

照例，稍微讲解一下代码，代码分为四部分：

第一部分：引入相关的库

```
# coding: utf-8
import requests
import json
```

第二部分：对数据接口进行 http 请求

```
url = 'http://www.toutiao.com/api/pc/focus/'
wbdata = requests.get(url).text
```

第三部分：对 HTTP 响应的数据 JSON 化，并索引到新闻数据的位置

```
data = json.loads(wbdata)
news = data['data']['pc_feed_focus']
```

第四部分：对索引出来的 JSON 数据进行遍历和提取

```
for n in news:
    title = n['title']
```

```
img_url = n['image_url']  
url = n['media_url']  
print(url, title, img_url)
```

如此，就完成了从 JS 网页中爬取数据。

□

州的先生Python教程

第六章：提高爬虫效率—并发爬取智联招聘

之前文章中所介绍的爬虫都是对单个 URL 进行解析和爬取，url 数量少不费时，但是如果我们需要爬取的网页 url 有成千上万或者更多，那怎么办？使用 for 循环对所有的 url 进行遍历访问？

嗯，想法很好，但是如果 url 过多，爬取完所有的数据会不会太过于耗时了？对此我们可以使用并发来对 URL 进行访问以爬取数据。一般而言，在单机上我们使用三种并发方式：

- 多线程(threading)
- 多进程(multiprocessing)
- 协程(gevent)

对于以上三种方法的具体概念解释和说明，各位可以自行网上搜索了解，相信会比我解释得清楚，所以在此就不对它们进行解释说明了。

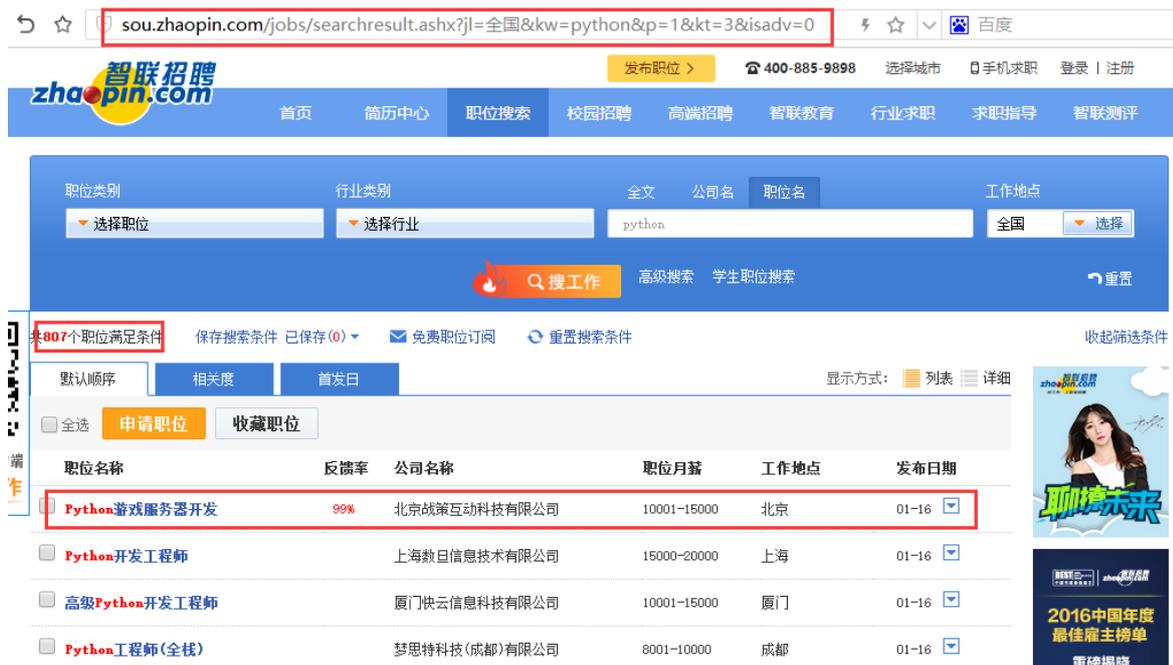
本系列文章有两个重点，一个是实战，一个是入门，既为实战，理论性的东西就描述得比较少；既为入门，所讲述的都是简单易懂易操作的东西，高深的技术还请入门之后自行探索，那样也会成长得更快。

那么下面，开始并发爬取的实战入门，以多进程为例，并发爬取智联招聘的招聘信息。

6.1、分析 URL 和页面结构

6.1.1、搜索全国范围内职位名称包含“Python”的职位招聘

我们不分职业类别、不分行业类别，工作地点选为全国，职位名称为“Python”，对招聘信息进行搜索，结果如下图：



我们注意图中三个红框的信息：

1. 搜索结果的 url 结构；（构造 url 地址进行 for 循环遍历）
2. 搜索结果的条数；（判断 url 的数量）
3. 采集的信息的主体；（解析数据）

通过筛选 url 参数，我们确定了需要爬取的基本 URL 为：

```
http://sou.zhaopin.com/jobs/searchresult.ashx?jl=全国&kw=python&kt=3&p=2
```

其中：

```
http://sou.zhaopin.com/jobs/searchresult.ashx
```

为请求地址和目录

- jl:工作地点参数
- kw:搜索的关键字
- kt:以职位名称搜索
- p:页数

我们可以发现，除了页数会变化之外，其余的参数值都是固定的值。我们来确定一下搜索结果的总页数。因为网页上有提示一共有多少个职位满足条件，我们拿总职位数除以单页显示的职位数量即可知道搜索结果的页数。

```
# coding:utf-8
```

```
import requests
from bs4 import BeautifulSoup
import re

url = 'http://sou.zhaopin.com/jobs/searchresult.ashx?jl=全国&kw=python&p=1&kt=3'
wdata = requests.get(url).content
soup = BeautifulSoup(wdata, 'lxml')

items = soup.select("div#newlist_list_content_table > table")
count = len(items) - 1
# 每页职位信息数量
print(count)

job_count = re.findall(r"共<em>(.*?)</em>个职位满足条件", str(soup))[0]
# 搜索结果页数
pages = (int(job_count) // count) + 1
print(pages)
```

结果返回每页 60 条职位信息，一共有 14 页。那么我们的待爬取的 url 地址就有 14 个，url 地址中参数 p 的值分别从 1 到 14，这么多的 url，使用 for 循环也可以很快完成，但在此我们使用多进程进行演示。

6.1.2、在爬虫中使用多进程

照例先上代码：

```
# coding:utf-8

import requests
from bs4 import BeautifulSoup
from multiprocessing import Pool

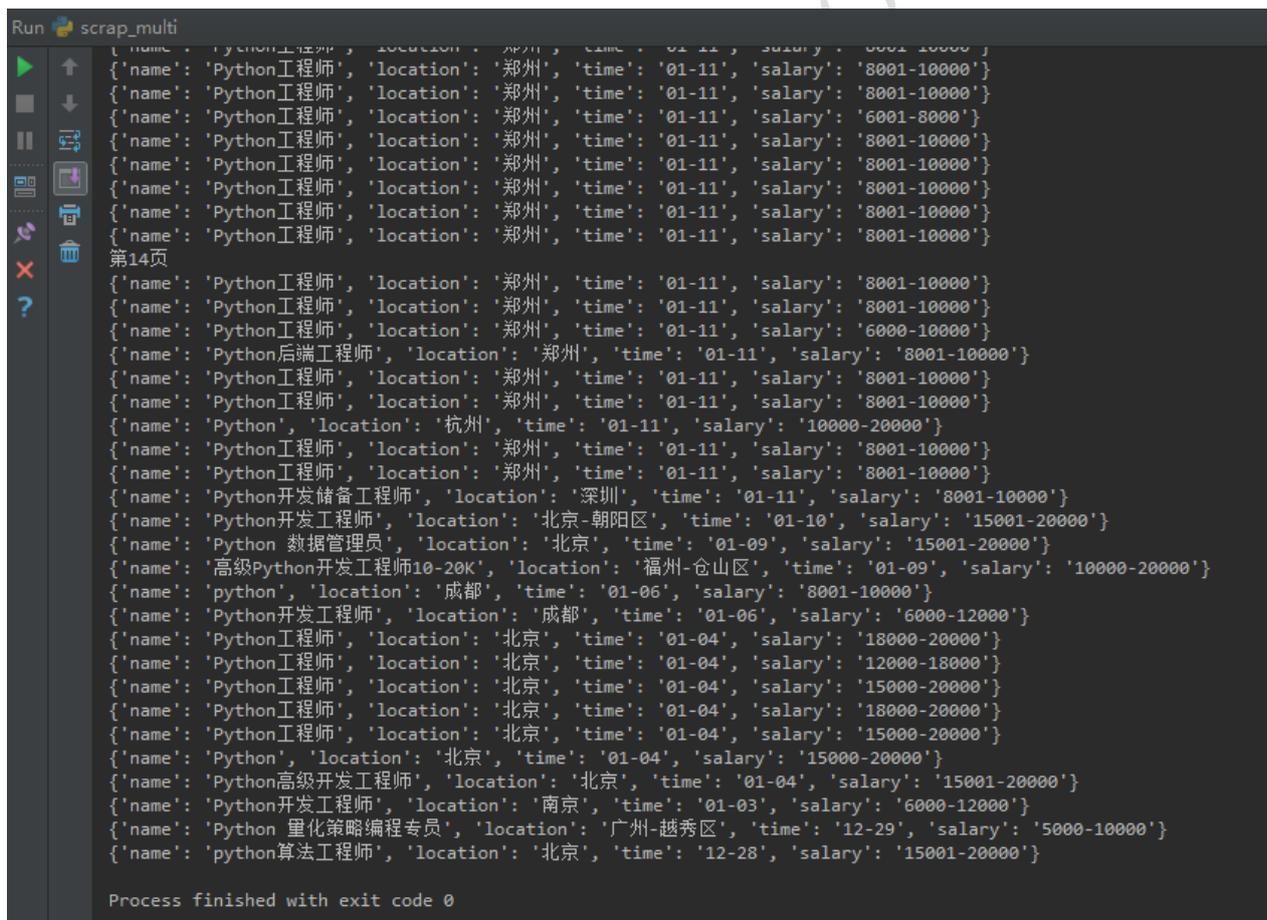
def get_zhaopin(page):
    url = 'http://sou.zhaopin.com/jobs/searchresult.ashx?jl=全国&kw=python&p={0}&kt=3'.format(page)
    print("第{0}页".format(page))
    wdata = requests.get(url).content
    soup = BeautifulSoup(wdata, 'lxml')

    job_name = soup.select("table.newlist > tr > td.zwmc > div > a")
    salarys = soup.select("table.newlist > tr > td.zwyx")
    locations = soup.select("table.newlist > tr > td.gzdd")
    times = soup.select("table.newlist > tr > td.gxsj > span")
```

```
for name, salary, location, time in zip(job_name, salaries, locations, times):
    data = {
        'name': name.get_text(),
        'salary': salary.get_text(),
        'location': location.get_text(),
        'time': time.get_text(),
    }
    print(data)

if __name__ == '__main__':
    pool = Pool(processes=2)
    pool.map_async(get_zhaopin, range(1, pages+1))
    pool.close()
    pool.join()
```

运行程序结果如下：



```
Run scrapy_multi
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '6001-8000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
第14页
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '6000-10000'}
{'name': 'Python后端工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python', 'location': '杭州', 'time': '01-11', 'salary': '10000-20000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python工程师', 'location': '郑州', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python开发储备工程师', 'location': '深圳', 'time': '01-11', 'salary': '8001-10000'}
{'name': 'Python开发工程师', 'location': '北京-朝阳区', 'time': '01-10', 'salary': '15001-20000'}
{'name': 'Python 数据管理员', 'location': '北京', 'time': '01-09', 'salary': '15001-20000'}
{'name': '高级Python开发工程师10-20K', 'location': '福州-仓山区', 'time': '01-09', 'salary': '10000-20000'}
{'name': 'python', 'location': '成都', 'time': '01-06', 'salary': '8001-10000'}
{'name': 'Python开发工程师', 'location': '成都', 'time': '01-06', 'salary': '6000-12000'}
{'name': 'Python工程师', 'location': '北京', 'time': '01-04', 'salary': '18000-20000'}
{'name': 'Python工程师', 'location': '北京', 'time': '01-04', 'salary': '12000-18000'}
{'name': 'Python工程师', 'location': '北京', 'time': '01-04', 'salary': '15000-20000'}
{'name': 'Python工程师', 'location': '北京', 'time': '01-04', 'salary': '18000-20000'}
{'name': 'Python工程师', 'location': '北京', 'time': '01-04', 'salary': '15000-20000'}
{'name': 'Python', 'location': '北京', 'time': '01-04', 'salary': '15000-20000'}
{'name': 'Python高级开发工程师', 'location': '北京', 'time': '01-04', 'salary': '15001-20000'}
{'name': 'Python开发工程师', 'location': '南京', 'time': '01-03', 'salary': '6000-12000'}
{'name': 'Python 量化策略编程专员', 'location': '广州-越秀区', 'time': '12-29', 'salary': '5000-10000'}
{'name': 'python算法工程师', 'location': '北京', 'time': '12-28', 'salary': '15001-20000'}

Process finished with exit code 0
```

因为除了使用了多进程之外，其他的代码与之前文章介绍的方法大同小异，所以在此只介绍一下多进程的核心代码：

```
from multiprocessing import Pool
```

`multiprocessing` 是 Python 自带的一个多进程模块，在此我们使用其 `Pool` 方法。

```
if __name__ == '__main__':  
    pool = Pool (processes=2)  
    pool.map_async (get_zhaopin, range (1, pages+1))  
    pool.close()  
    pool.join()
```

1. 实例化一个进程池，设置进程为 2；
2. 调用进程池的 `map_async()` 方法，接收一个函数(爬虫函数)和一个列表(url 列表)

如此，在爬虫中使用多进程进行并发爬取就搞定了，更多高级、复杂强大的方法，还请各位参考其他文档资料。

□

第七章：使用 Selenium——以抓取 QQ 空间好友说说为例

前面我们接触到的，都是使用 `requests+BeautifulSoup` 组合对静态网页进行请求和数据解析，若是 JS 生成的内容，也介绍了通过寻找 API 借口来获取数据。但是有的时候，网页数据由 JS 生成，API 借口又死活找不着或者是 API 借口地址随机变换，时间不等人。那就只能使用 Selenium 了。

7.1、Selenium 简介

Selenium 是一个用于 Web 应用的功能自动化测试工具，Selenium 直接运行在浏览器中，就像真正的用户在操作一样。由于这个性质，Selenium 也是一个强大的网络数据采集工具，其可以让浏览器自动加载页面，获取需要的数据，甚至页面截图，或者是判断网站上某些动作是否发生。

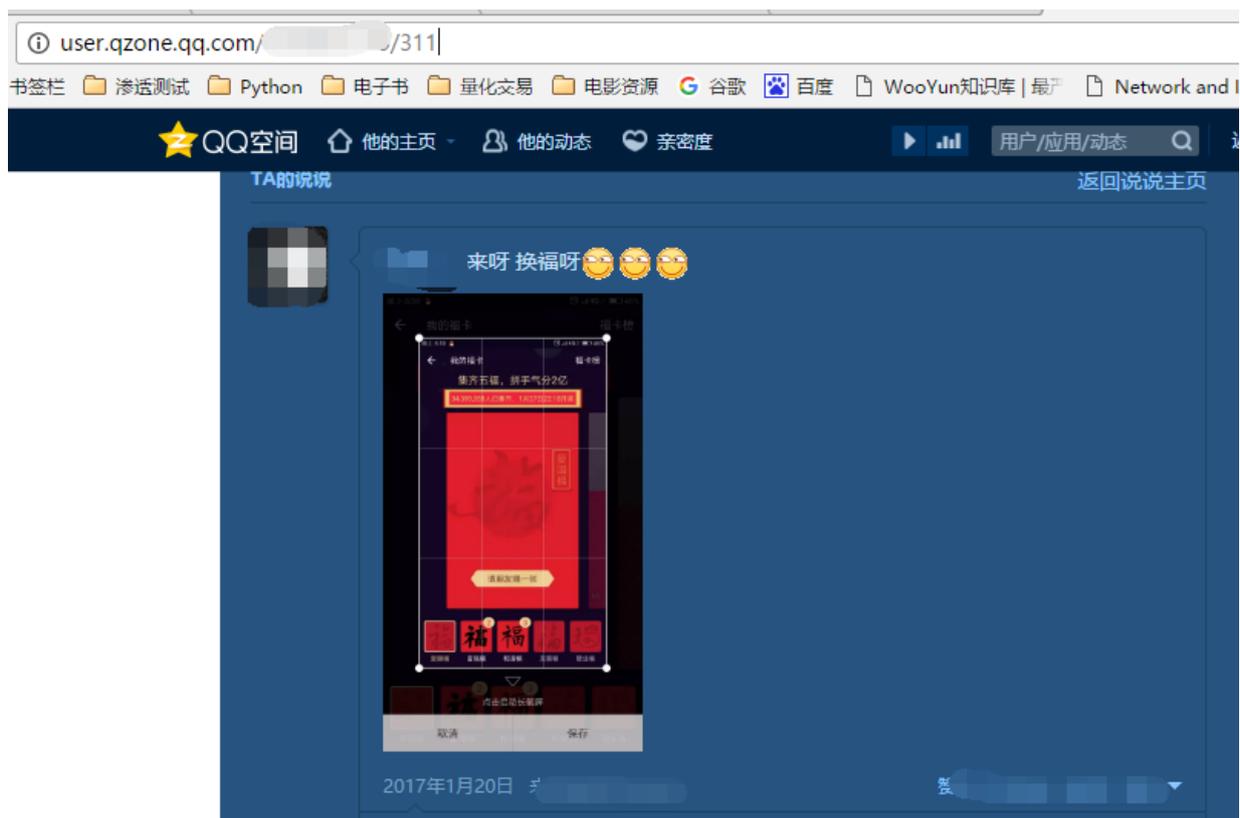
Selenium 自己不带浏览器，需要配合第三方浏览器来使用。支持的浏览器有 Chrome、Firefox、IE、Phantomjs 等。如果使用 Chrome、FireFox 或 IE，我们可以看得到一个浏览器的窗口被打开、打开网站、然后执行代码中的操作。但是，让程序在后台中运行更符合我们爬虫的气质，所以自己多使用 Phantomjs 作为浏览器载体，本篇文章也以 Phantomjs 作介绍。

Phantomjs 是一个“无头”浏览器，也就是没有界面的浏览器，但是功能与普通的浏览器无异。

7.2、在 Python 中使用 Selenium 获取 QQ 空间好友说说

之前使用 pip 安装好了 selenium，直接在代码中 import 即可。下面我们以一个实际的例子——获取一个 QQ 空间好友的说说信息，来简单讲解一下 Selenium+Phantomjs 的使用。

我们需要爬取的页面时这样的：



QQ 空间好友说说的链接为: <http://user.qzone.qq.com/{好友 QQ 号}/311>, 我们抓取他发的说说的时间和内容。

依旧先上代码:

```
from bs4 import BeautifulSoup
from selenium import webdriver
import time

#使用 selenium
driver = webdriver.PhantomJS(executable_path="D:\\phantomjs.exe")
driver.maximize_window()
#登录 QQ 空间
def get_shuoshuo(qq):
    driver.get('http://user.qzone.qq.com/{}/311'.format(qq))
    time.sleep(5)
    try:
        driver.find_element_by_id('login_div')
        a = True
    except:
        a = False
    if a == True:
        driver.switch_to.frame('login_frame')
        driver.find_element_by_id('switcher_plogin').click()
```

```
driver.find_element_by_id('u').clear()#选择用户名框
driver.find_element_by_id('u').send_keys('QQ号')
driver.find_element_by_id('p').clear()
driver.find_element_by_id('p').send_keys('QQ密码')
driver.find_element_by_id('login_button').click()
time.sleep(3)
driver.implicitly_wait(3)
try:
    driver.find_element_by_id('QM_OwnerInfo_Icon')
    b = True
except:
    b = False
if b == True:
    driver.switch_to.frame('app_canvas_frame')
    content = driver.find_elements_by_css_selector('.content')
    stime = driver.find_elements_by_css_selector('.c_tx.c_tx3.goDetail')
    for con, sti in zip(content, stime):
        data = {
            'time': sti.text,
            'shuos': con.text
        }
        print(data)
    pages = driver.page_source
    soup = BeautifulSoup(pages, 'lxml')

cookie = driver.get_cookies()
cookie_dict = []
for c in cookie:
    ck = "{0}={1};".format(c['name'], c['value'])
    cookie_dict.append(ck)
i = ''
for c in cookie_dict:
    i += c
print('Cookies:', i)
print("=====完成=====")

driver.close()
driver.quit()

if __name__ == '__main__':
    get_shuoshuo('好友QQ号')
```

获取到的数据截图如下：

```
C:\Python34\python.exe J:/python/scrap/scrap_tech_book/use_selenium.py
{'time': '2017年1月20日', 'shuos': '来呀 呀'}
{'time': '2017年1月19日', 'shuos': '挤 明显了。。。'}
{'time': '2017年1月3日', 'shuos': '好好。。。'}
{'time': '2017年1月1日', 'shuos': '一个人'}
{'time': '2016年12月23日', 'shuos': '恍惚。。。\n【传销的'}
{'time': '2016年11月28日', 'shuos': '各种 醒来才10点。。。'}
{'time': '2016年11月26日', 'shuos': '我凭本事贷的款，为什么要还'}
{'time': '2016年11月13日', 'shuos': '昏昏欲睡'}
{'time': '2016年11月4日', 'shuos': ' 不作不死'}
{'time': '2016年10月30日', 'shuos': '在。。。'}
{'time': '2016年10月27日', 'shuos': ''}
{'time': '2016年10月13日', 'shuos': '神奇的 20。。。'}
{'time': '2016年8月10日', 'shuos': '长夜难眠。。。'}
{'time': '2016年8月3日', 'shuos': '昨晚 觉得自己过节。'}
{'time': '2016年7月16日', 'shuos': '中'}
{'time': '2016年7月5日', 'shuos': '深夜无眠。。。'}

```

接下来我们通过讲解代码，稍微了解一下 Selenium 的使用

7.3、代码简析

1.照例，导入需要使用的模块：

```
from bs4 import BeautifulSoup
from selenium import webdriver
import time
```

2.使用 Selenium 的 webdriver 实例化一个浏览器对象，在这里使用 Phantomjs：

```
driver = webdriver.PhantomJS(executable_path="D:\\phantomjs.exe")
```

3.设置 Phantomjs 窗口最大化：

```
driver.maximize_window()
```

4.主函数部分

使用 get()方法打开待抓取的 URL：

```
driver.get('http://user.qqzone.qq.com/{}/311'.format(qq))
```

等待 5 秒后，判断页面是否需要登录，通过查找页面是否有相应的 DIV 的 id 来判断：

```
try:
    driver.find_element_by_id('login_div')
```

```
a = True
except:
    a = False
```

如果页面存在登录的 DIV，则模拟登录：

```
driver.switch_to.frame('login_frame') #切换到登录 ifram
driver.find_element_by_id('switcher_plogin').click()
driver.find_element_by_id('u').clear()#选择用户名框
driver.find_element_by_id('u').send_keys('QQ 号')
driver.find_element_by_id('p').clear()#选择密码框
driver.find_element_by_id('p').send_keys('QQ 密码')
driver.find_element_by_id('login_button').click()#点击登录按钮
time.sleep(3)
```

接着，判断好友空间是否设置了权限，通过判断是否存在元素 ID: QM_OwnerInfo_Icon

```
try:
    driver.find_element_by_id('QM_OwnerInfo_Icon')
    b = True
except:
    b = False
```

如果有权限能够访问到说说页面，那么定位元素和数据，并解析：

```
if b == True:
    driver.switch_to.frame('app_canvas_frame')
    content = driver.find_elements_by_css_selector('.content')
    stime = driver.find_elements_by_css_selector('.c_tx.c_tx3.goDetail')
    for con, sti in zip(content, stime):
        data = {
            # 'qq':qq,
            'time':sti.text,
            'shuos':con.text
        }
    print(data)
```

除了在 Selenium 中解析数据，我们还可以将当前页面保存为源码，再使用 BeautifulSoup 来解析：

```
pages = driver.page_source
soup = BeautifulSoup(pages, 'lxml')
```

最后，我们尝试一下获取 Cookie，使用 get_cookies():

```
cookie = driver.get_cookies()
cookie_dict = []
for c in cookie:
    ck = "{0}={1};".format(c['name'], c['value'])
    cookie_dict.append(ck)
i = ''
for c in cookie_dict:
    i += c
print('Cookies:', i)
```

另外，再介绍两个 Selenium 的常用方法：
保存屏幕截图：

```
driver.save_screenshot('保存的文件路径及文件名')
```

执行 JS 脚本：

```
driver.execute_script("JS 代码")
```

对于 Selenium 更加详细的操作和使用，推荐一本书《selenium webdriver(python)第三版》
网上可以搜索到；

需要电子书和 Phantomjs 的也可以关注微信公众号：州的先生，回复关键字：01sp

第八章：数据储存——MongoDB 与 MySQL

前面的文章里写的爬虫都是直接把爬取的数据打印出来。在实际的应用中，当然不能这么做，我们需要将数据存储起来。存储数据的方式有很多中，比如存储在文本文件中，或者是存储在数据库中。

为了使用数据的便捷性，我们选择将数据存储数据库中。

数据库主流的两种类型为：SQL（关系型数据库）和 NoSQL（非关系型数据库），我们在此选用使用比较广泛的 MySQL 和 MongoDB 作为讲解

8.1、MySQL

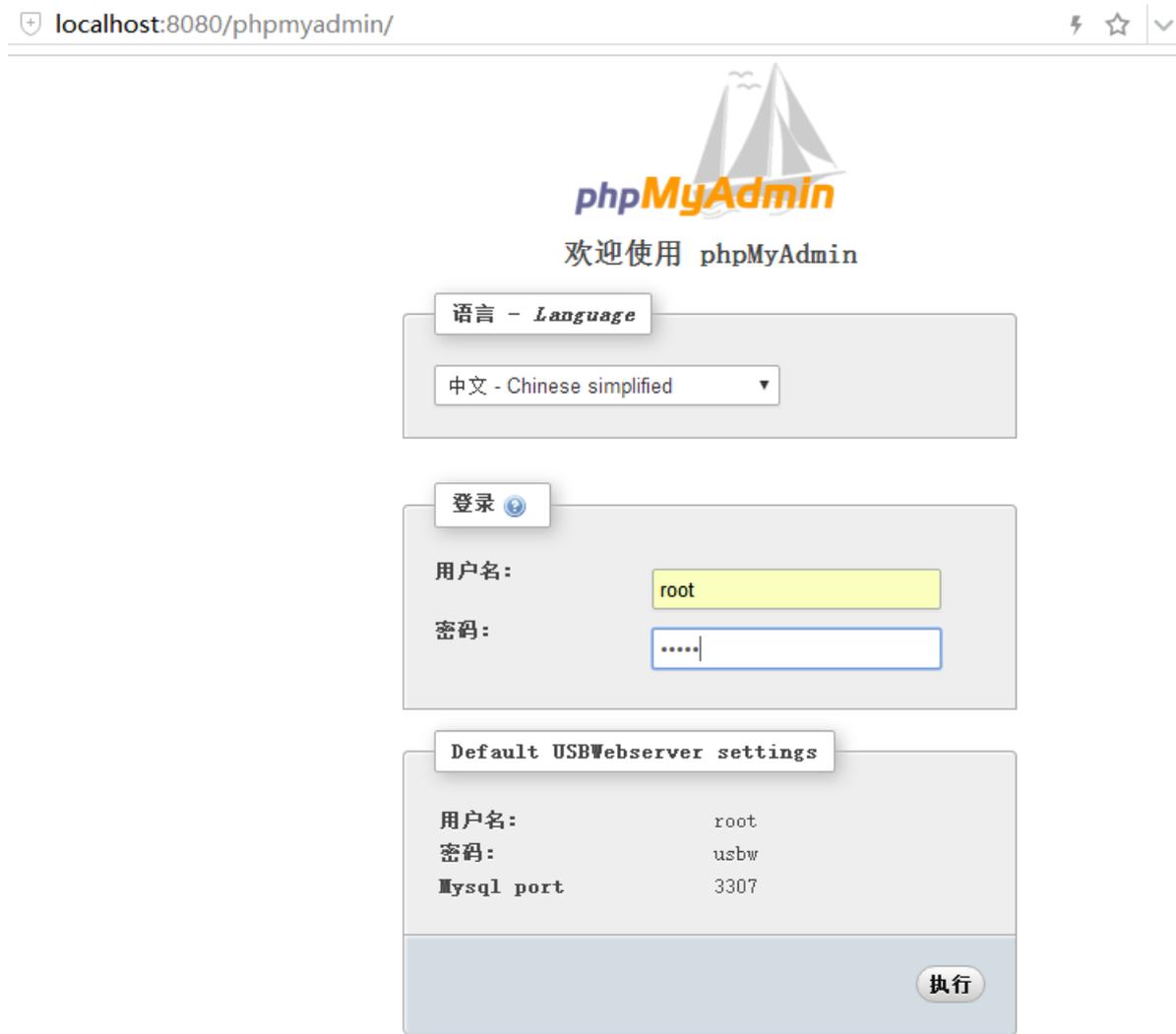
8.1.1、安装 MySQL

由于 MySQL 的安装文件比较大且配置稍微繁琐，个人推荐在普通环境下使用集成包，比如 USBWebserver。

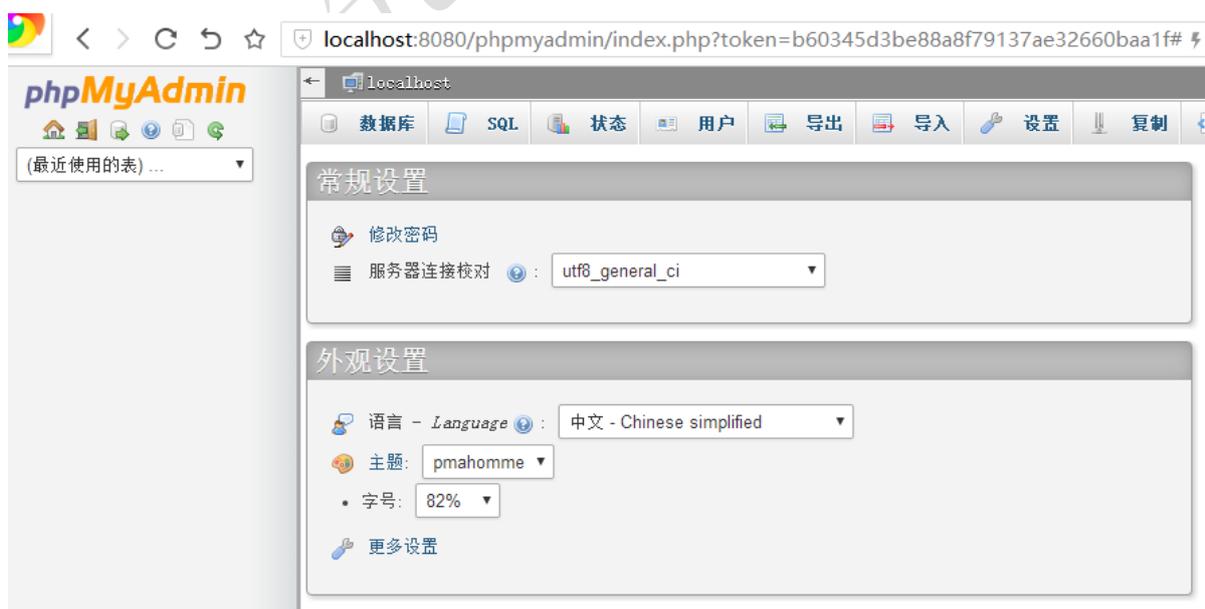
USBWebserver 其实是一款傻瓜式本地电脑快速架设 PHP 网站环境的工具，它最大特色是纯绿色便携，可直接放在 U 盘里随处运行它集成了 Apache (httpd)、PHP、MySQL 以及 PHPMyAdmin 等组件，而我们使用它的 MySQL 即可。



打开程序之后，看到 Mysql 运行成功，就可以打开 PHPMyAdmin



点击执行，进入控制页面



就可以在里面创建数据库，创建数据表了。更多介绍及下载：

<http://www.iplaysoft.com/usbwebserver.html>

3、安装 pymysql

在 Python 中使用 MySQL，有两种方式，使用 ORM（对象关系映射）框架和数据库模块，在此我们使用数据库模块 pymysql(Python3)。

安装 pymysql:

```
pip install pymysql
```

4、在爬虫程序中使用 mysql

我们以之前爬取今日头条的例子来扩展：之前的代码是这样的：

```
# coding: utf-8
import requests
import json

url = 'http://www.toutiao.com/api/pc/focus/'
wbdata = requests.get(url).text

data = json.loads(wbdata)
news = data['data']['pc_feed_focus']

for n in news:
    title = n['title']
    img_url = n['image_url']
    url = n['media_url']
    print(url, title, img_url)
```

在最后，我们直接使用 `print` 将数据打印了出来。现在我们使用 `pymysql` 将数据存储到 `MySQL` 中。(创建数据库 `toutiao`, 创建数据表 `data`)

修改的代码如下：

```
# coding: utf-8

import requests
import json
import pymysql

conn = pymysql.connect(host='localhost', port=3307, user='root', password='usbw', db='toutiao', charset='utf8')
cursor = conn.cursor()
```

```
url = 'http://www.toutiao.com/api/pc/focus/'
wbdata = requests.get(url).text

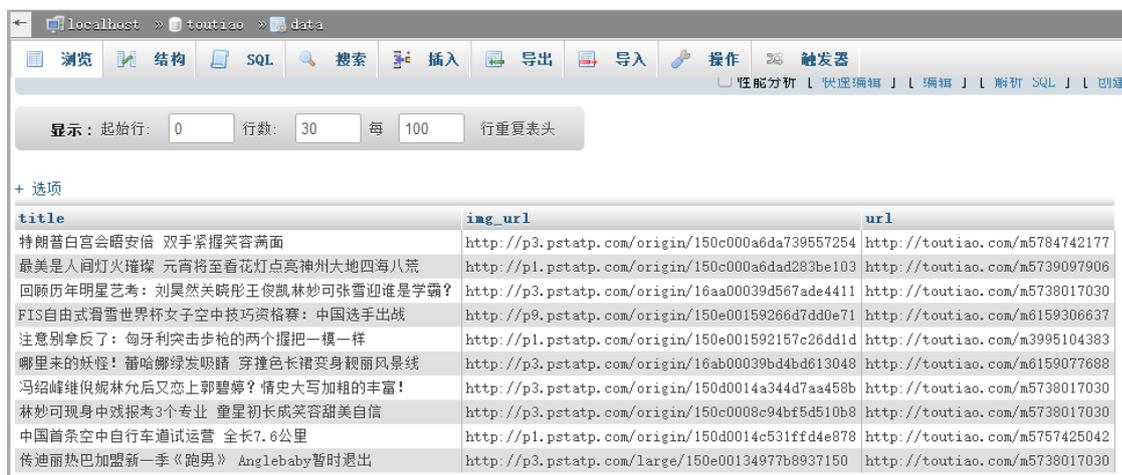
data = json.loads(wbdata)
news = data['data']['pc_feed_focus']
for n in news:
    title = n['title']
    img_url = n['image_url']
    url = n['media_url']
    print(url, title, img_url)
    cursor.execute("INSERT INTO data(title, img_url, url)VALUES(' {0} ', ' {1} ', ' {2} ');".format(
        title, img_url, url))
    conn.commit()

cursor.close()
conn.close()
```

最后，数据库中就已经存储了数据：

```
data = json.loads(wbdata)
news = data['data']['pc_feed_focus']
for n in news:
    title = n['title']
    img_url = n['image_url']
    url = n['media_url']
    print(url, title, img_url)
    cursor.execute("INSERT INTO data(title, img_url, url)VALUES(' {0} ', ' {1} ', ' {2} ');".format(title, img_url, url))
    conn.commit()
```

```
mysql
C:\Python34\python.exe I:/python/scrap/scrap_tech_book/use_mysql.py
http://toutiao.com/m5784742177 特朗普白宫会晤安倍 双手紧握笑容满面 http://p3.pstatp.com/origin/150c000a6da739557254
http://toutiao.com/m5739097906 最美是人间灯火璀璨 元宵将至看花灯点亮神州大地四海八荒 http://p1.pstatp.com/origin/150c000a6dad283be103
http://toutiao.com/m5738017030 回顾历年明星艺考：刘昊然关晓彤王俊凯林妙可张雪迎谁是学霸？ http://p3.pstatp.com/origin/16aa00039d567ade4411
http://toutiao.com/m6159306637 FIS自由式滑雪世界杯女子空中技巧资格赛：中国选手出战 http://p9.pstatp.com/origin/150e00159266d7dd0e71
http://toutiao.com/m3995104383 注意别拿反了：匈牙利突击步枪的两个握把一模一样 http://p1.pstatp.com/origin/150e001592157c26ddd1d
http://toutiao.com/m6159077688 哪里来的妖怪！蕾哈娜绿发吸睛 穿搭色长裙变身靓丽风景线 http://p3.pstatp.com/origin/16ab00039bd4bd613048
http://toutiao.com/m5738017030 冯绍峰继倪妮林允后又恋上郭碧婷？情史大写加粗的丰富！ http://p3.pstatp.com/origin/150d0014a344d7ae458b
http://toutiao.com/m5738017030 林妙可现身中戏报考3个专业 童星初长成笑容甜美自信 http://p3.pstatp.com/origin/150c0008c94bf5d510b8
http://toutiao.com/m5757425042 中国首条空中自行车道试运营 全长7.6公里 http://p1.pstatp.com/origin/150d0014c531ff8de878
http://toutiao.com/m5738017030 传迪丽热巴加盟新一季《跑男》 Angelababy暂时退出 http://p3.pstatp.com/large/150e00134977b8937150
```



title	img_url	url
特朗普白宫会晤安倍 双手紧握笑容满面	http://p3.pstatp.com/origin/150c000a6da739557254	http://toutiao.com/m5784742177
最美是人间灯火璀璨 元宵将至看花灯点高神州大地四海八荒	http://p1.pstatp.com/origin/150c000a6dad283be103	http://toutiao.com/m5739097906
回顾历年明星艺考：刘昊然关晓彤王俊凯林妙可张雪迎谁是学霸？	http://p3.pstatp.com/origin/16aa00039d567ade4411	http://toutiao.com/m5738017030
FIS自由式滑雪世界杯女子空中技巧资格赛：中国选手出战	http://p9.pstatp.com/origin/150e00159266d7dd0e71	http://toutiao.com/m6159306637
注意别拿反了：匈牙利突击步枪的两个握把一模一样	http://p1.pstatp.com/origin/150e001592157c26dd1d	http://toutiao.com/m3995104383
哪里来的妖怪！蕾哈娜绿发吸睛 穿搭色长裙变身靓丽风景线	http://p3.pstatp.com/origin/16ab00039bd4bd613048	http://toutiao.com/m6159077688
冯绍峰绯闻妮妮允后又恋上郭碧婷？情史大写加粗的丰富！	http://p3.pstatp.com/origin/150d0014a344d7aa458b	http://toutiao.com/m5738017030
林妙可现身中戏报考3个专业 童星初长成笑容甜美自信	http://p3.pstatp.com/origin/150c0008c94bf5d510b8	http://toutiao.com/m5738017030
中国首条空中自行车道试运营 全长7.6公里	http://p1.pstatp.com/origin/150d0014c531ffd4e878	http://toutiao.com/m5757425042
传迪丽热巴加盟新一季《跑男》 Angelababy暂时退出	http://p3.pstatp.com/large/150e00134977b8937150	http://toutiao.com/m5738017030

与之前的代码相比，有以下不同：

引入 `pymysql` 模块：

```
import pymysql
```

建立一个 `mysql` 的连接：

```
conn = pymysql.connect(host='localhost', port=3307, user='root', password='usbw', db='toutiao', charset='utf8')
```

创建一个游标 `cursor`：

```
cursor = conn.cursor()
```

执行一个 `SQL` 语句：

```
cursor.execute("INSERT INTO data(title, img_url, url) VALUES (' {0} ', ' {1} ', ' {2} ');".format(title, img_url, url))
```

提交执行（因为对数据进行和修改，如果只是 `select`，则不需要）：

```
conn.commit()
```

最后，关闭连接：

```
cursor.close()  
conn.close()
```

嗯，将数据保存在 `MySQL` 就完成了，更多的 `MySQL` 和 `PyMySQL` 的用法，还请看文档
下面看看 `MongoDB`

8.2、MongoDB

1、下载并安装 MongoDB:

<https://www.mongodb.com/download-center>

2、运行 mongod:

进入安装好之后的 mongo 目录的 bin 目录，打开命令行窗口，输入“mongod --dbpath=数据存放路径”

```
C:\Program Files (x86)\MongoDB\Server\3.2\bin>mongod --dbpath=e:\data
2017-02-11T13:05:06.378+0800 I CONTROL [main]
2017-02-11T13:05:06.378+0800 W CONTROL [main] 32-bit servers don't have journaling enabled by default.
ernal if you want durability.
2017-02-11T13:05:06.378+0800 I CONTROL [main]
2017-02-11T13:05:06.400+0800 I CONTROL [initandlisten] MongoDB starting : pid=17332 port=27017 dbpath=
st=yangjian-PC
2017-02-11T13:05:06.400+0800 I CONTROL [initandlisten] targetMinOS: Windows Vista/Windows Server 2008
2017-02-11T13:05:06.400+0800 I CONTROL [initandlisten] db version v3.2.3
2017-02-11T13:05:06.400+0800 I CONTROL [initandlisten] git version: b326ba837cf6f49d65c2f85e1b70f6f31e
2017-02-11T13:05:06.401+0800 I CONTROL [initandlisten] allocator: tcmalloc
2017-02-11T13:05:06.401+0800 I CONTROL [initandlisten] modules: none
2017-02-11T13:05:06.401+0800 I CONTROL [initandlisten] build environment:
2017-02-11T13:05:06.401+0800 I CONTROL [initandlisten] distarch: i386
2017-02-11T13:05:06.401+0800 I CONTROL [initandlisten] target_arch: i386
2017-02-11T13:05:06.401+0800 I CONTROL [initandlisten] options: { storage: { dbPath: "e:\data" } }
```

3、安装 pymongo:

```
pip install pymongo
```

5、使用 MongoDB 和 PyMongo

依然是扩展爬取今日头条的例子，先上代码：

```
# coding: utf-8

import requests
import json
import pymongo

conn = pymongo.MongoClient(host='localhost', port=27017)
toutiao = conn['toutiao']
newsdata = toutiao['news']

url = 'http://www.toutiao.com/api/pc/focus/'
wbdata = requests.get(url).text

data = json.loads(wbdata)
```

```

news = data['data']['pc_feed_focus']
for n in news:
    title = n['title']
    img_url = n['image_url']
    url = n['media_url']
    data = {
        'title':title,
        'img_url':img_url,
        'url':url
    }
    newsdata.insert_one(data)
for i in newsdata.find():
    print(i)

```

存储数据到 MongoDB 并读取出来

```

for n in news:
    title = n['title']
    img_url = n['image_url']
    url = n['media_url']
    # print(url, title, img_url)
    data = {
        'title':title,
        'img_url':img_url,
        'url':url
    }
    newsdata.insert_one(data)
for i in newsdata.find():
    print(i)

```

```

python.exe I:/python/scraps/scraps_tech_book/use_mongo.py
p://toutiao.com/6578474217', 'title': '特朗普白宫会晤安倍 双手紧握笑容满面', 'img_url': 'http://p3.pstatp.com/origin/150c000a6da739557254', '_id': ObjectId('589ea249e4f31
p://toutiao.com/65738097906', 'title': '最美是人间灯火璀璨 元宵将至看花灯点亮神州大地四海八荒', 'img_url': 'http://p1.pstatp.com/origin/150c000a6dad283be102', '_id': Objec
p://toutiao.com/65738017030', 'title': '回顾历年明星艺考: 刘昊然关晓彤王俊凯林妙可张雪迎谁是学霸?', 'img_url': 'http://p3.pstatp.com/origin/16aa00039d567ade4411', '_id':
p://toutiao.com/66159206637', 'title': 'FIS自由式滑雪世界杯女子空中技巧资格赛: 中国选手出战', 'img_url': 'http://p9.pstatp.com/origin/150e00159266d7dd0e71', '_id': ObjectI
p://toutiao.com/63995104382', 'title': '注意别拿反了: 匈牙利突击步枪的两个握把一模一样', 'img_url': 'http://p1.pstatp.com/origin/150e001592157c26add', '_id': ObjectId('58
p://toutiao.com/66159077688', 'title': '哪里来的妖怪! 鹿晗娜绿发吸睛 穿搭色长裙变身靓丽风景线', 'img_url': 'http://p3.pstatp.com/origin/16a00039bd4b4613048', '_id': Objec
p://toutiao.com/65738017030', 'title': '冯绍峰继倪妮林允后又恋上郭碧婷? 情史大写的丰富!', 'img_url': 'http://p3.pstatp.com/origin/150d0014a344d7eae58b', '_id': Object
p://toutiao.com/65738017030', 'title': '林妙可现身中戏报考3个专业 童星初长成笑容甜美自信', 'img_url': 'http://p3.pstatp.com/origin/150c0008e94bf5d510b8', '_id': ObjectId('
p://toutiao.com/65757425042', 'title': '中国首条空中自行车道试运营 全长7.6公里', 'img_url': 'http://p1.pstatp.com/origin/150d0014c531ffd4e878', '_id': ObjectId('589ea24ae4
p://toutiao.com/65738017030', 'title': '传迪丽热巴加盟新一季《跑男》 Anglebaby暂时退出', 'img_url': 'http://p3.pstatp.com/large/150e00134977b8927150', '_id': ObjectId('588

```

Pymongo 相关的代码为:

引入模块

```
import pymongo
```

连接到 Mongo

```
conn = pymongo.MongoClient(host='localhost', port=27017)
```

选择或创建数据库

```
toutiao = conn['toutiao']
```

选择或创建数据集

```
newsdata = toutiao['news']
```

插入一行数据:

```
newsdata.insert_one(data)
```

查询数据

```
newsdata.find()
```

如此，简单地对数据进行数据库存储就完成了。

第九章：下一步

完成了上面 8 章的学习，基本上就已经了解了使用 Python 编写网络爬虫进行数据采集的方法。

在实际的网络数据采集中，可能面对的网站部署了非常多非常复杂的反爬虫手段来限制爬虫的爬取行为，所以大家可以更加深入地了解如何使用代理 IP 池来避免频繁采集下的 IP 被封，了解如何使用 PyQt 来绕过一些网站的高等级登录限制（Selenium 操纵的 webdriver 会被识别出来），了解如何破解复杂的验证码形式，了解如何有效地对 URL 进行队列处理，了解如何部署分布式的爬虫，等等，这些都是深入学习爬虫所必须经过的路、踩下的坑。

祝大家学习顺利，工作顺利！