

飞扬研发第二次例会

四川大学飞扬俱乐部

胡宗尧 2023.11.19

Part0. 目录

..

:

Part1. 爬虫详解

爬虫简介

正则表达式

HTTP请求

JSON格式

HTML是什么

爬虫快速入门

学会使用F12(审查元素)

处理数据

爬虫伪装

Part1. 爬虫详解

- **简单概括：**自动从网络上收集信息的一种程序。
- **复杂点说：**就是一整套关于数据请求、处理、存储的程序。爬虫涉及到很多关于网络、前端的一些知识，非常杂。今天先讲其中一部分，之后也会分别详细介绍关于数据的采集、处理、存储这三方面的知识。

本课的目的也是，**快速入门**。撇去一切多余的代码，尽量让初学者保持学习的热情。

0. 爬虫简介

WHY PYTHON?

爬虫并不是Python独有的，那我们为什么选择python？
因为我们只学python啊。主要是因为下面几点：

- **开发效率高，代码简洁**，一行代码就可完成请求，100行可以完成一个复杂的爬虫任务；
- **爬虫对于代码执行效率要求不高**，网站IO才是最影响爬虫效率的。如一个网页请求可能需要100ms，数据处理10ms还是1ms影响不大；
- **非常多优秀的第三方库**，如requests, beautifulsoup, selenium等等；

爬虫是搜索引擎的一部分：

百度搜索的一部分工作，就是运行它自己的爬虫，从上千万的网站，采集到网页，然后存起来，等待你的搜索。

爬虫违法吗？

有关互联网法规正在逐步建立和完善中，就目前而言，如果抓取的数据仅供个人使用或者用于科研（机器学习、大数据）一般并无大碍；但如果数据抓取用于商业范畴，就要就事论事了，有可能属于违法也有可能并无大碍。

如何学好爬虫？

一句话：**多写代码，从小项目做起，别上来就一个劲地啃教程，消磨掉兴趣。**

1. 正则表达式

正则表达式是一种查找以及字符串替换操作。比如，我想要在编辑器里匹配所有的数字，怎么办？这就需要正则表达式了。

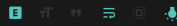
几乎所有的高级编程语言和代码编辑器都支持正则表达式。但是这玩意学习难度挺高。。。这里先不做介绍，留为回家作业自学（）

请在GitHub上找正则表达式（Regex）的教程，或者玩那种闯关模式的教学网站，比如 <https://regexlearn.com>

一些小工具：

- [Test RegEx](#)
- [I Hate Regex](#)
- [常用正则表达式](#)
- [正则表达式可视化工具](#)

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-type" content="text/html; charset=gb2312" />
5   <title>Sina Visitor System</title>
6 </head>
7 <body>
8   <span id="message"></span>
9   <script type="text/javascript" src="/js/visitor/mini_original.js?v=20161116"></script>
10  <script type="text/javascript" />
11
12 </script>
13 </body>
14 </html>
```

 <(.*)(.*)>.*<\1>|<(.*?)>|

示例

网址 (URL)	<code>`[a-zA-z]+://[^\\s]*`</code>
Email地址	<code>`\\w+([-+.]\\w+)*@[\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*`</code>
QQ号码	<code>`[1-9]\\d{4,}`</code>
HTML标记(包含内容或自闭合)	<code>`<(.*)(.*)>.*<\\/\\1> <(.*?) \\/>`</code>
密码(由数字/大写字母/小写字母/标点符号组成, 四种都必有, 8位以上)	<code>`(?:^.{8,}\$)(?=.*\\d)(?=.*\\W+)(?=.*[A-Z])(?=.*[a-z])(?!.*\\n).*`\$`</code>
中文及全角标点符号(字符)	<code>`[\\u3000-\\u301e\\ufe10-\\ufe19\\ufe30-\\ufe44\\ufe50-\\ufe6b\\uff01-\\uffee]`</code>
正整数	<code>`[0-9]*[1-9][0-9]*`</code>
负整数	<code>`-[0-9]*[1-9][0-9]*`</code>

2. HTTP请求

要搞明白如何爬虫，先得知道什么是http请求。

拓展阅读：从输入URL到页面加载的过程？如何由一道题完善自己的前端知识体系！

当你在浏览器中输入一个网址或点击一个链接时，你实际上发起了一个**HTTP请求**，由以下几个部分组成：

- **HTTP方法**：`GET`（获取资源）、`POST`（提交数据）
- **URL（统一资源定位符）**：<http://www.fyscu.com>
- **HTTP标头**：它包含了一些额外的信息，用于描述请求的细节。下面会详细解释。

HTTP响应：

- **状态码**：你经常能看到浏览器显示`404 not found`、但是其实还有`502 bad gateway`、`403 forbidden`、`200`（成功）、和`500`（服务器错误）等等，其实都是服务器告诉我们的浏览器的。
- **HTTP标头**：它包含了一些额外的信息，用于描述响应的细节。
- **响应正文**：服务器返回的实际内容，可以是HTML页面、JSON数据等。

HTTP标头

常见的HTTP标头包括：

- **User-Agent**：标识发送请求的用户代理（通常是浏览器）。
- **Content-Type**：指定请求或响应正文的媒体类型，例如text/html或application/json。
- **Content-Length**：指定请求或响应正文的长度。
- **Cookie**：用于在客户端和服务器之间传递状态信息。重点说一下这个cookie哈，它不是曲奇饼干，而是一段存储在用户计算机上的小型文本，往往只有几Kb，所以才塞得到这个header里面。那几kb够干啥的？比如说，你登录了一个网站，你也不想刷新以后你的登录状态就丢失了吧？那么cookie就是帮助你保存登录状态的。比如服务器在发送我登录成功的信息时，还在响应头里放入了一段cookie（独一无二的），那浏览器就可以保存它，下次请求的时候就带上它，那么服务器就能认得，原来这就是刚才那家伙，已经登录了。那脑洞大开一下，如果你在请求的时候，用别人登录以后的cookie会怎么样？那当然是你冒名顶替了别人啊，你可以带着这串cookie，获得对他账户的访问权限，相当于“盗号”了。
- **Cache-Control**：指定请求或响应的缓存行为。
- **Authorization**：用于在请求中发送身份验证凭据。

3. JSON格式

如果我想给服务器发送信息，这个信息应该是什么格式的？这样看起来很合理吧：

姓名：小明
学号：2022123456789
性别：男

唉？`key-value`的形式，这不就是python字典吗？那我们用字典把它写出来：

```
{  
    "姓名": "小明",  
    "学号": 2022123456789,  
    "性别": "男"  
}
```

基本上所有JSON都可以转变成Python字典。传说当年有JSON和XML之争，最后JSON胜在简洁、轻量、易读、能够方便地转化成字典（便于计算机读取），于是JSON成了现在最流行的数据交换格式。

与python字典一样，JSON也可以嵌套。
因此能够胜任很多复杂的数据：

```
[  
    {  
        "姓名": "小明",  
        "学号": 2022123456789,  
        "性别": "男"  
    },  
    {  
        "姓名": "小红",  
        "学号": 2023123456789,  
        "性别": [  
            "女",  
            "男"  
        ]  
    },  
    {  
        "姓名": "小军",  
        "学号": 2024123456789,  
        "性别": "男"  
    }  
]
```

4. HTML是什么

比如我们输入了 <https://www.fyscu.com>，向飞扬官网的服务器发送了一个请求(request)，那么服务器就会给我们一个响应(response)，这个响应就是一个HTML页面，如果你点击「查看网页源代码」，就能看到HTML大概长这样：

```
<html>
<head>
  <title>Title</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="http://www.xxx.com/xxx.css">
  <script src="http://www.xxx.com/xxx.js">
  </script>
</head>
<body>
  <p class="title">
    <b>bold text</b>
    normal text
  </p>
</body>
</html>
```

HTML标签由很多个节点 (Tag) 组成。比如

`<head>`、`<body>`、`<p>`、``。这些节点之间的关系有父子关系、兄弟关系。

- 父子关系：子节点被包括在父节点中。比如body内的所有标签，都是body的子节点。
- 兄弟关系：两个节点位于同一层级，比如我们的所有的p标签。他们的直接父节点都是body。

所以，节点之间就像树一样 (DOM树)，再复杂的页面也是标签套标签形成的。每个标签都可以有自己的一些属性。

比如：`class`、`href`、`src`、`id`。这些属性决定了他们的特点。

5. 爬虫快速入门

我们需要安装三个模块，后面会介绍它们的用途：

```
pip install requests beautifulsoup4 lxml
```

安装好以后，运行下面这个程序，来爬取微博热搜：

```
import requests
response = requests.get(
    'https://s.weibo.com/top/summary?cate=realtimehot')
print(response.text)
print(response.status_code) # 200
```

requests.get() 函数就对微博热搜的URL进行了一次get请求，注意到我们这里没有携带任何的headers和cookies，response.text 就是一个字符串，整个网页的内容。

真的有这么简单吗？我们来看看输出了什么：

输出：

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type"
      content="text/html; charset=gb2312"/>
    <title>Sina Visitor System</title>
  </head>
  <body>
    <span id="message"></span>
    <script type="text/javascript"
      src="/js/visitor/mini_original.js?v=20161116">
    </script>
    <script type="text/javascript">
      一堆JS函数
    </script>
  </body>
</html>
```

怎么一个热搜都没有？这合理吗？为什么我用浏览器就可以？

这是因为，你第一次用浏览器访问这个页面的时候，服务器发现你没有cookie，所以也返回给浏览器这个东西，但是这里面的JS代码让浏览器以游客的身份登录了微博，并获得了一个cookie，作为身份的一个标识。浏览器有cookie了，下次访问的时候就畅通无阻了。

我们用python爬取了这个url以后，当然没法运行JS代码，那怎么办？

唉，就像我之前提到的，我们可以「窃取」浏览器的cookie啊，那我们不就处于游客的登录状态了吗？

于是把cookies加到请求中去：

关于如何获取浏览器的cookie...这个马上就说...

```
import requests

cookies = {
    'SUB': '_2AkMSD9vLf8NxqwFRmfoUymnraIhywwvEieKkUyoQJRMxHRl-yj9kqkJYtRB60Y_1JJbLkxq_V-5FsAoYEr7y0zTco-_r',
    'SUBP': '0033WrSXqPxfM72-Ws9jqgMF55529P9D9W5JoYoKxZJ1WiP3cS6-NBbL',
}

response = requests.get(
    'https://s.weibo.com/top/summary?cate=realtimehot', cookies=cookies)
print(response.text)
print(response.status_code) # 200
```

指哪打哪

虽然我们能看到微博热搜了，但是是网页的源代码，看起来非常丑，也不是我们想要的数据格式。

那如何在这么长的字符串中匹配到这些微博热搜呢？

曾经，你只能用正则表达式匹配，写起来异常繁琐，可读性为0。但是现在有了beautifulsoup4库，敲代码的幸福感直线上升。不过很多教程都把bs4的门槛拉的很高，这不应该啊，人家本来发明出来就是为了简化爬虫的难度的。。。

我的经验是，过一遍前端知识再来看这个，会好很多。

CSS选择器

其实，定位/匹配一个或者一堆网页元素，也是CSS解决的痛点之一。那CSS是如何解决的呢？

```
<p id="text_id" class="big bold">一段文字</p>
```

比如，匹配这个元素的，可以是：

- `p`` 标签选择器
- `#text_id`` ID选择器
- `.big`` 类选择器（选择了一个类）
- `.bold`` 类选择器（选择了一个类）
- `.big.bold`` 类选择器（同时选择两个类）

HTML里面不会有两个相同的ID，identification嘛，像身份证一样唯一的。

一个选择器可以匹配很多元素，在CSS里面，这段表示了所有`class="bold"`的元素都加粗变红色：

一段文字

```
.bold {  
  font-weight: bold;  
  color: #f92472;  
}
```

后代选择器

后代选择器的语法是用空格分隔的多个选择器组合，它的作用是在 A 选择器的后代元素中找到 B 选择器所指的元素。

```
<div class="page">
  <div class="article">
    <h1>我是标题部分</h1>
    <p>黑化肥发灰，灰化肥发黑</p>
    <p>黑化肥发灰会挥发；灰化肥挥发会发黑</p>
    <p>黑化肥挥发发灰会花飞；灰化肥挥发发黑会飞花</p>
  </div>
  <p class="footer">我是footer脚注</p>
</div>
```

- `.page p`：表示在`.page`选定的区域里去找p标签，就找到了所有p标签。

假如我们只想查找`class="article"`元素里的`p`标签呢，就再套一层：

- `.page .article p`：`.page`里面的`.article`里面的p标签。

那如果我只想选择儿子，不要这些子子孙孙呢？

子元素选择器：`.page > p`就行了。

讲了这么多，就是为了给这段代码做铺垫：

```
### 接上面的代码
from bs4 import BeautifulSoup

soup = BeautifulSoup(response.text, "lxml")
print(soup.select(".td-02 a"))
```

解释一下：

- BeautifulSoup就是从bs4（刚刚下载的python库）中的一个解析函数
- lxml（也是刚刚下的）就相当于告诉BeautifulSoup如何解析：**按照html的语法解析。**
- 解析完了返回一个特殊的格式，存到soup里面
- `soup.select()` 就是选择器语法，我选择了所有含有`class="td-02"`的函数，又继续选择了所有`a`标签。


输出结果是一个List，里面都是我们想要的链接、热搜标题啥的：






```
[<a href="/weibo?q=%23%E6%97%B6%E6%97%B6%E6%94%BE%E5%BF%83%E4%B8%8D%E4%B8%8B%E7%9A%84%E7%89%B5%E6%8C%82%23&Refer=new_time" target="_blank">
时时放心不下的牵挂</a>,
...,
...,
<a href="/weibo?q=%23LPL%E9%80%89%E6%89%8B%E7%BA%A2%E6%AF%AF%E8%A5%BF%E8%A3%85%E6%9D%80%23&t=31&band_rank=50&Refer=top" target="_blank">
LPL选手红毯西装杀</a>]
```


那`td-02`，`a`这些是如何找到的呢？

就是**网页调试模式**的功劳了。(大部分浏览器都是F12，审查元素)







6. 学会使用F12(审查元素)

 搜索微博



登录注册

微博热搜

-  热搜榜
-  要闻榜
-  文娱榜
-  体育榜
-  游戏榜
-  好友搜


序号	关键词	
	总书记在华北大地留下温暖人心的足迹	热
1	张翰回应油腻 1350780	热
2	渐冻人蔡磊激动试用人工喉 1345480	暖
3	朝着构建亚太命运共同体不断迈进 1342153	
	荣耀100满分上镜	官宣
4	竞燃之夜 1341488	新
5	凌晨5点车灯一开照出2只大熊猫 1247351	暖
6	联合国总部降半旗 1220541	热
7	两位唐氏孩子妈妈用20年互相治愈 990905	暖
8	何猷君 小家庭永远高于大家庭 综艺 924497	新
9	孙颖莎什么时候出现 521067	新
10	快乐小赵 去世 368057	
11	缅北电诈家族曾邀请国内明星网红庆生 367595	新
12	孙菲菲公开暴力事件最新进展 364600	热
13	YG回应与BLACKPINK续约 363974	新
14	1020缅北到底发生了什么 361797	新
15	竞燃之夜红毯显眼包是IQOO12 360520	
16	Jennie50亿韩元购入高档别墅 358152	

后退
前进
重新加载

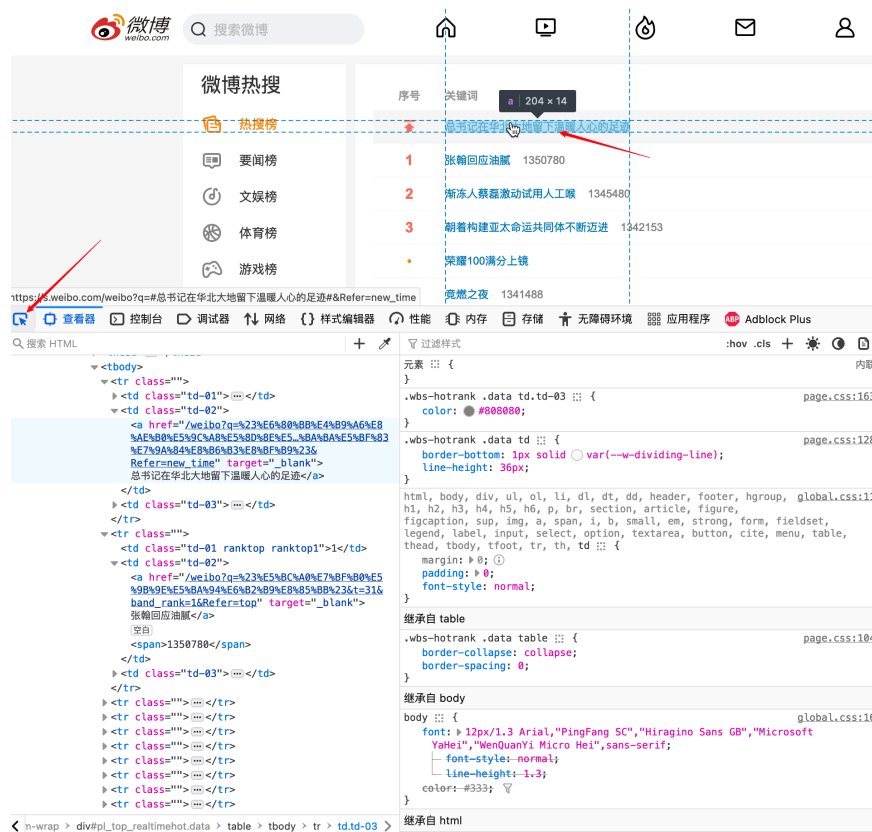
将页面加入书签...
另存页面为...
保存页面到 Pocket
发送页面到设备
全选

截图

查看页面源代码
检查无障碍环境属性
检查

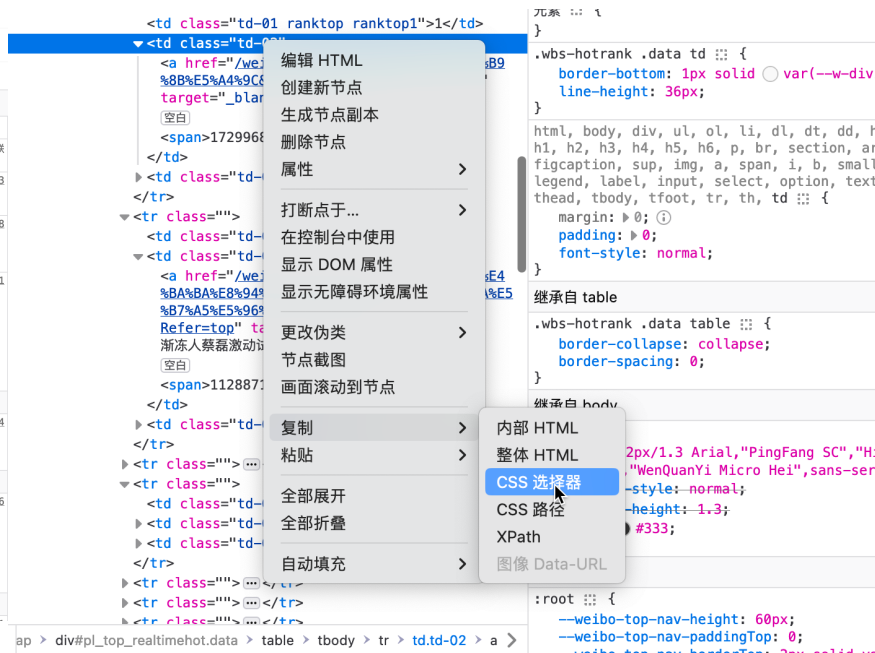
 Flagfox

6.1. 定位元素，复制CSS选择器




可以看到我们想要的元素，有个共同特点就是
`class="td-02"` 是吧？a标签在它里面，自然而然就想
到了这个选择器：






- `td-02 a`，完美取出所有匹配的元素！




6.2. 网络与资源嗅探


 微博
weibo.com


Q 搜索微博





登录注册


微博热搜


 热搜榜

 要闻榜













 文娱榜


 体育榜

 游戏榜

 好友搜

序号	关键词	
	时时放心不下的牵挂	热
1	竞燃之夜 1729968	新
2	斯冻人蔡磊激动试用人工眼 1128871	暖
3	朝着构建亚太命运共同体不断迈进 1005380	
4	何猷君 小家庭永远高于大家庭 综艺 930630	新
5	凌晨5点车灯一开照出2只大熊猫 699709	暖
6	联合国总部降半旗 690333	热

 查看器  控制台  调试器 ** 网络**  样式编辑器  性能  内存  存储  无障碍环境  应用程序  Adblock Plus 

 过滤 URL

状态	方...	域名	文件	发起者	类...	传输	大小	消息头	Cookie	请求	响应	耗时	安全性
302	GET	login.sina.com.cn	visitor?a=crossdomain&s=...	visitor:141 (d...	ht...	10.57 kB	55.73 kB	过滤消息头					
200	GET	s.weibo.com	summary?cate=realtimehot	document	ht...	9.96 kB	55.73 kB	GET https://s.weibo.com/top/summary?cate=realtimehot					

状态 200 ?

版本 HTTP/2

传输 9.96 kB (大小 55.73 kB)

Referrer 策略 strict-origin-when-cross-origin

请求优先级 Highest

DNS 解析 系统

▼ 响应头 (274 字节)

content-encoding: gzip

content-type: text/html; charset=UTF-8

date: Tue, 14 Nov 2023 11:37:09 GMT

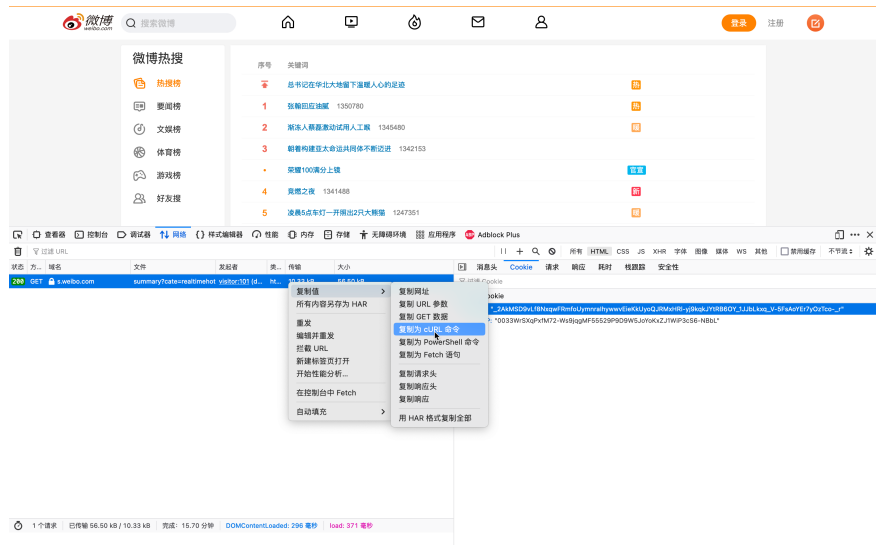
lb: 123.125.107.11

server: Weibo API Gateway

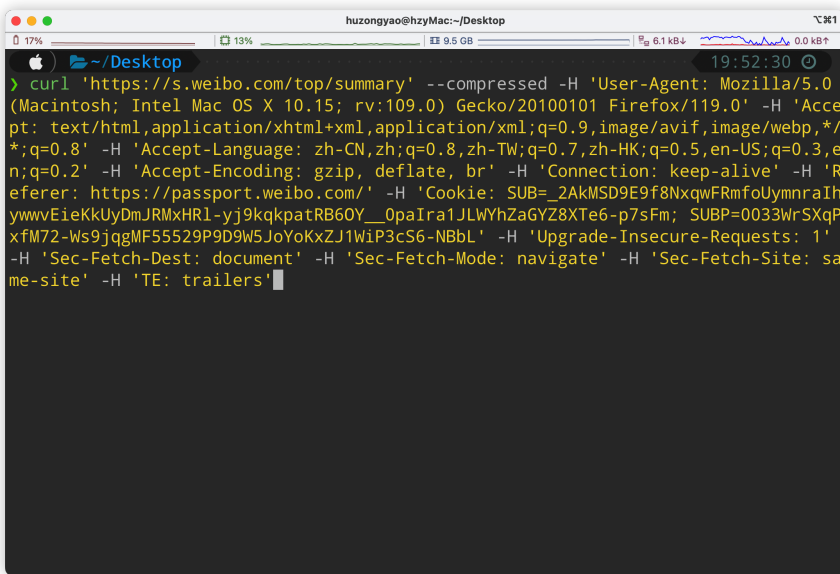
ssl_node: mweibo-sslv6-004.tc.intra.weibo.cn

CURL

• 复制Curl命令

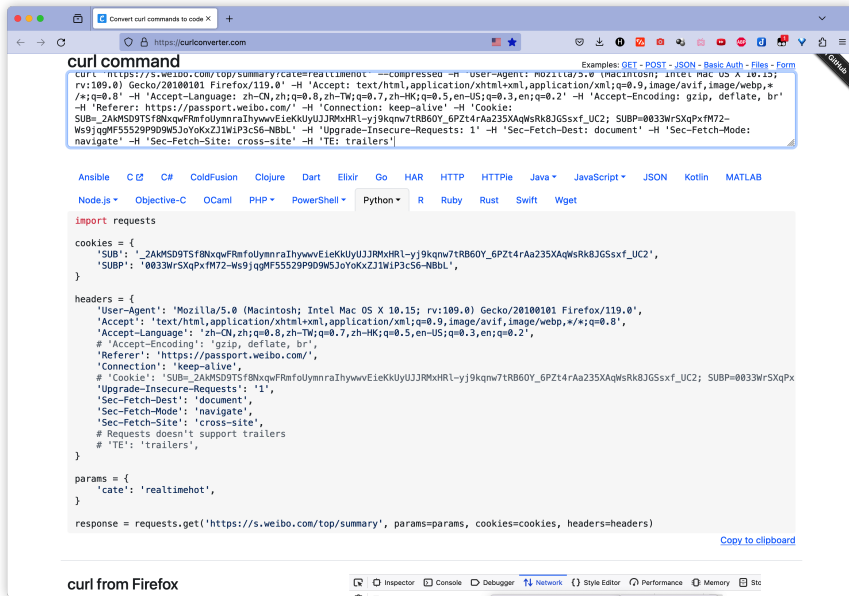


在Linux课上应该会学`curl`命令，命令行里常用的请求工具，但是由于过于丑陋，所以诞生了一些替代品，这里不多说。主要是看看这个小工具：<https://www.curlconverter.com>



复制curl非常有用，因为一个curl就里面包含了你的header、cookie这些，是一个完整的http请求，而如果直接用request去请求是没有这些的。

获取了curl以后把它粘贴到这个网站里面，
就能获得完全一样效果的python代码：



The screenshot shows the curlconverter.com website. At the top, there's a search bar with the URL 'https://s.weibo.com/top/summary?cate=realtimehot'. Below it, a 'curl command' is displayed in a text area. The command is a complex curl request for a Weibo API endpoint. Below the curl command, there are tabs for different programming languages: Ansible, C#, C#, ColdFusion, Clojure, Dart, Elixir, Go, HAR, HTTP, HTTPie, Java, JavaScript, JSON, Kotlin, MATLAB, Node.js, Objective-C, OCaml, PHP, PowerShell, Python, R, Ruby, Rust, Swift, and Wget. The 'Python' tab is selected. Below the tabs, the converted Python code is shown. It uses the 'requests' library to make a GET request to the same URL. The code includes a 'cookies' dictionary, a 'headers' dictionary, and a 'params' dictionary. The 'headers' dictionary contains various headers like 'User-Agent', 'Accept', 'Accept-Language', 'Accept-Encoding', 'Referer', 'Connection', 'Cookie', 'Upgrade-Insecure-Requests', 'Sec-Fetch-Dest', 'Sec-Fetch-Mode', 'Sec-Fetch-Site', and 'TE'. The 'cookies' dictionary contains a 'SUBP' cookie. The 'params' dictionary contains a 'cate' parameter. The final line of code is 'response = requests.get('https://s.weibo.com/top/summary', params=params, cookies=cookies, headers=headers)'. At the bottom right of the code area, there is a 'Copy to clipboard' button. Below the code area, there is a 'curl from Firefox' section with a 'Copy to clipboard' button.

```
curl -https://s.weibo.com/top/summary?cate=realtimehot --compressed -H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15; rv:109.0) Gecko/20100101 Firefox/119.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8' -H 'Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2' -H 'Accept-Encoding: gzip, deflate, br' -H 'Referer: https://passport.weibo.com/' -H 'Connection: keep-alive' -H 'Cookie: SUBP=_2AKMSD9T5f8NqwfRnfoUymnraIhywvEieKkUyUJ3R9xHRL-yj9Kqmw7tRB60Y_6PZt4rAa235XAqWsrK8JG5sxf_UC2; SUBP=0033Wf5XqPxTW72-Ws9JqgMF5529P9D9W5JoYoKxZJ1WlP3c56-NBbl' -H 'Upgrade-Insecure-Requests: 1' -H 'Sec-Fetch-Dest: document' -H 'Sec-Fetch-Mode: navigate' -H 'Sec-Fetch-Site: cross-site' -H 'TE: trailers'
```

```
import requests

cookies = {
    'SUBP': '_2AKMSD9T5f8NqwfRnfoUymnraIhywvEieKkUyUJ3R9xHRL-yj9Kqmw7tRB60Y_6PZt4rAa235XAqWsrK8JG5sxf_UC2',
    'SUBP': '0033Wf5XqPxTW72-Ws9JqgMF5529P9D9W5JoYoKxZJ1WlP3c56-NBbl',
}

headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15; rv:109.0) Gecko/20100101 Firefox/119.0',
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8',
    'Accept-Language': 'zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2',
    'Accept-Encoding': 'gzip, deflate, br',
    'Referer': 'https://passport.weibo.com/',
    'Connection': 'keep-alive',
    'Cookie': 'SUBP=_2AKMSD9T5f8NqwfRnfoUymnraIhywvEieKkUyUJ3R9xHRL-yj9Kqmw7tRB60Y_6PZt4rAa235XAqWsrK8JG5sxf_UC2; SUBP=0033Wf5XqPxTW72-Ws9JqgMF5529P9D9W5JoYoKxZJ1WlP3c56-NBbl',
    'Upgrade-Insecure-Requests': '1',
    'Sec-Fetch-Dest': 'document',
    'Sec-Fetch-Mode': 'navigate',
    'Sec-Fetch-Site': 'cross-site',
    # Requests doesn't support trailers
    # 'TE': 'trailers',
}

params = {
    'cate': 'realtimehot',
}

response = requests.get('https://s.weibo.com/top/summary', params=params, cookies=cookies, headers=headers)
```

这个代码当然是能跑的，这里的headers看起来很复杂，其实可以不用管它，重要的是cookie。

用了浏览器的cookie，我们就不是未登录状态了，相当于借用了浏览器的游客身份。

```
headers = {
    'Cookie': 'your-cookie',
}

response = requests.get(
    'https://s.weibo.com/top/summary?cate=realtimehot',
    headers=headers)

cookies = {
    'key1': 'value1',
    'key2': 'value2',
}

response = requests.get(
    'https://s.weibo.com/top/summary?cate=realtimehot',
    cookies=cookies)
```

cookie本身就是header的一部分。

7. 处理数据

观察其中一个a标签，发现，链接在**href属性**里面，标题在`<a>`中间。

```
<a href="/weibo?q=%23E6%97%B6%E6%97%B6%E6%94%BE%E5%BF%83%E4%B8%8D%E4%B8%8B%E7%9A%84%E7%89%B5%E6%8C%82%23&Refer=new_time" target="_blank">时时放心不下的牵挂</a>
```

`Beautifulsoup` 提供了很多方法来获取这些数据：

```
>>> soup = BeautifulSoup('<a href="/weibo?q=%23E6%97%B6%E6%97%B6%E6%94%BE%E5%BF%83%E4%B8%8D%E4%B8%8B%E7%9A%84%E7%89%B5%E6%8C%82%23&Refer=new_time" target="_blank">\n时时放心不下的牵挂</a>', "lxml")
>>> print(soup.prettify())
<html>
  <body>
    <a href="/weibo?q=%23E6%97%B6%E6%97%B6%E6%94%BE%E5%BF%83%E4%B8%8D%E4%B8%8B%E7%9A%84%E7%89%B5%E6%8C%82%23&Refer=new_time" target="_blank">
      时时放心不下的牵挂
    </a>
  </body>
</html>
```

```
>>> tag = soup.a
>>> tag
<a href="/weibo?q=%23E6%97%B6%E6%97%B6%E6%94%BE%E5%BF%83%E4%B8%8D%E4%B8%8B%E7%9A%84%E7%89%B5%E6%8C%82%23&Refer=new_time" target="_blank">时时放心不下的牵挂</a>
>>> type(tag)
<class 'bs4.element.Tag'>
>>> tag.string
'时时放心不下的牵挂'
>>> tag.attrs
{'href': '/weibo?q=%23E6%97%B6%E6%97%B6%E6%94%BE%E5%BF%83%E4%B8%8D%E4%B8%8B%E7%9A%84%E7%89%B5%E6%8C%82%23&Refer=new_time', 'target': '_blank'}
>>> tag.attrs['href']
'/weibo?q=%23E6%97%B6%E6%97%B6%E6%94%BE%E5%BF%83%E4%B8%8D%E4%B8%8B%E7%9A%84%E7%89%B5%E6%8C%82%23&Refer=new_time'
>>>
```


8. 爬虫伪装

网站可以通过IP和header确认你的身份。一旦发现你这个人不停地访问我的网站，什么100毫秒就刷新一次，完全可以把你这个IP封掉，于是**IP池**、**伪造请求头**这些技巧就诞生了。

- IP池的目的就是**允许我都用不同的IP**轮流去请求，服务器就会认为这个爬虫是很多人，就不会封我。再说了，封了一个IP，我不是还有一堆吗？当然，IP池很贵，一般的爬虫都不会用这个。
- 伪造请求头：这个更容易一些，目的就是模仿不同的人的请求（请求头里有你的浏览器、Referer等关键信息，如果每次都用同一个请求头，请求几千次，就容易露馅）`my-fake-useragent`是一个很好的库。

```
$ pip install my-fake-useragent
```

```
from my_fake_useragent import UserAgent
import requests
```

```
ua = UserAgent(family='chrome')
res = ua.random()
url = "https://www.baidu.com"
headers = {"User-Agent": res}
response = requests.get(url=url, headers=headers)
print(response.status_code) # 打印状态码
print(response.request.headers) # 打印自己的请求头
```