# 了解网页结构

作者: **莫烦**编辑: **莫烦 2017-12-29**

学习资料:

* [本节学习代码](https://github.com/MorvanZhou/easy-scraping-tutorial/blob/master/notebook/1-1-urllib.ipynb)
* 本节使用的爬虫[测试网页](https://morvanzhou.github.io/static/scraping/basic-structure.html)

学习爬虫, 首先要懂的是网页. 支撑起各种光鲜亮丽的网页的不是别的, 全都是一些代码. 这种代码我们称之为 [HTML](https://baike.baidu.com/item/HTML/97049?fr=aladdin), HTML 是一种浏览器(Chrome, Safari, IE, Firefox等)看得懂的语言, 浏览器能将这种语言转换成我们用肉眼看到的网页. 所以 HTML 里面必定存在着很多规律, 我们的爬虫就能按照这样的规律来爬取你需要的信息.

其实除了 HTML, 一同构建多彩/多功能网页的组件还有 [CSS](https://baike.baidu.com/item/CSS/5457) 和 [JavaScript](https://baike.baidu.com/item/javascript). 但是这个简单的爬虫教程, 大部分时间会将会使用 HTML. CSS 和 JavaScript 会在后期简单介绍一下. 因为爬网页的时候多多少少还是要和 CSS JavaScript 打交道的.

[](https://morvanzhou.github.io/static/results/scraping/1-1-00.jpg)

虽然[莫烦Python](https://morvanzhou.github.io/)主打的是机器学习的教程. 但是这个爬虫教程适用于任何想学爬虫的朋友们. 从机器学习的角度看, 机器学习中的大量数据, 也是可以从这些网页中来, 使用爬虫来爬取各种网页上面的信息, 然后再放入各种机器学习的方法, 这样的应用途径正在越来越多被采用. 所以如果你的数据也是分散在各个网页中, 爬虫是你减少人力劳动的必修课.

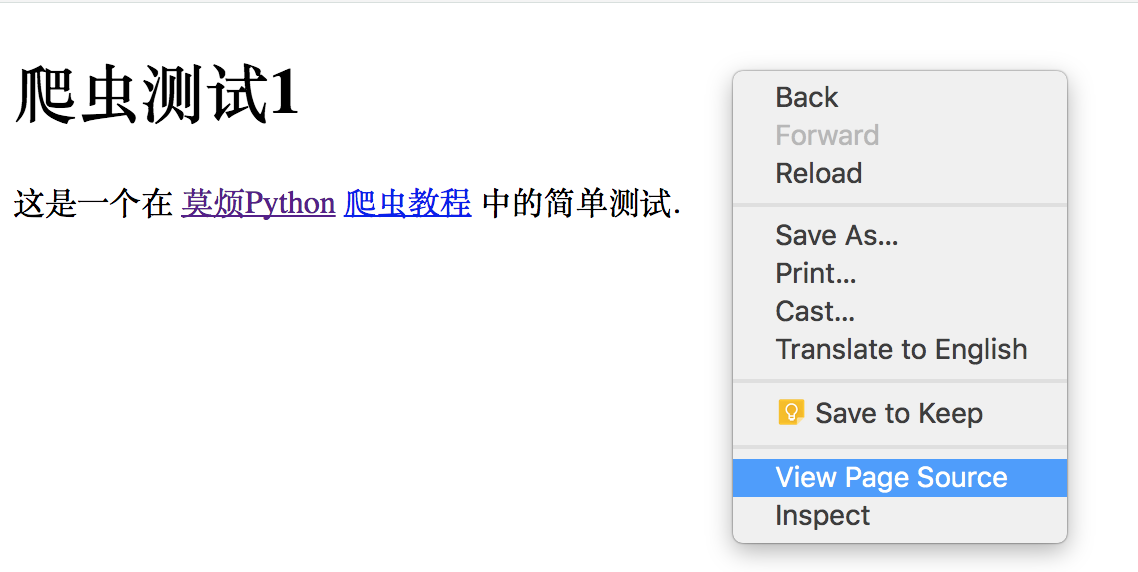
## [网页基本组成部分](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/1-01-understand-website/#%E7%BD%91%E9%A1%B5%E5%9F%BA%E6%9C%AC%E7%BB%84%E6%88%90%E9%83%A8%E5%88%86)

在真正进入爬虫之前, 我们先来做一下热身运动, 弄明白网页的基础, HTML 有哪些组成部分, 是怎么样运作的. 如果你已经非常熟悉网页的构造了, 欢迎直接跳过这一节, 进入下面的学习.

我制作了一个[非常简易的网页](https://morvanzhou.github.io/static/scraping/basic-structure.html), 给大家呈现以下最骨感的 HTML 结构. 如果你点开它, 呈现在你眼前的, 就是下面这张图的上半部分. 而下半部分就是我们网页背后的 HTML code.

[](https://morvanzhou.github.io/static/results/scraping/1-1-1.png)

想问我是如何看到 HTML 的 source code 的? 其实很简单, 在你的浏览器中 (我用的是 [Google Chrome](https://www.google.com.au/chrome/browser/desktop/)), 显示网页的地方, 点击鼠标右键, 大多数浏览器都会有类似这样一个选项 “View Page Source”. 点击它就能看到页面的源码了.

[](https://morvanzhou.github.io/static/results/scraping/1-1-2.png)

在 HTML 中, 基本上所有的实体内容, 都会有个 tag 来框住它. 而这个被 tag 住的内容, 就可以被展示成不同的形式, 或有不同的功能. 主体的 tag 分成两部分, header 和 body. 在 header 中, 存放这一些网页的网页的元信息, 比如说 title, 这些信息是不会被显示到你看到的网页中的. 这些信息大多数时候是给浏览器看, 或者是给搜索引擎的爬虫看.

<head>

<meta charset="UTF-8">

<title>Scraping tutorial 1 | 莫烦Python</title>

<link rel="icon" href="https://morvanzhou.github.io/static/img/description/tab\_icon.png">

</head>

HTML 的第二大块是 body, 这个部分才是你看到的网页信息. 网页中的 heading, 视频, 图片和文字等都存放在这里. 这里的 <h1></h1> tag 就是主标题, 我们看到呈现出来的效果就是大一号的文字. <p></p> 里面的文字就是一个段落. <a></a>里面都是一些链接. 所以很多情况, 东西都是放在这些 tag 中的.

<body>

<h1>爬虫测试1</h1>

<p>

这是一个在 <a href="https://morvanzhou.github.io/">莫烦Python</a>

<a href="https://morvanzhou.github.io/tutorials/scraping">爬虫教程</a> 中的简单测试.

</p>

</body>

爬虫想要做的就是根据这些 tag 来找到合适的信息.

## [用 Python 登录网页](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/1-01-understand-website/#%E7%94%A8-Python-%E7%99%BB%E5%BD%95%E7%BD%91%E9%A1%B5)

好了, 对网页结构和 HTML 有了一些基本认识以后, 我们就能用 Python 来爬取这个[网页](https://morvanzhou.github.io/static/scraping/basic-structure.html)的一些基本信息. 首先要做的, 是使用 Python 来登录这个网页, 并打印出这个网页 HTML 的 source code. 注意, 因为网页中存在中文, 为了正常显示中文, read() 完以后, 我们要对读出来的文字进行转换, decode() 成可以正常显示中文的形式.

**from** urllib.request **import** urlopen

# if has Chinese, apply decode()

html = urlopen(

"https://morvanzhou.github.io/static/scraping/basic-structure.html"

).read().decode('utf-8')

print(html)

print 出来就是下面这样啦. 这就证明了我们能够成功读取这个网页的所有信息了. 但我们还没有对网页的信息进行汇总和利用. 我们发现, 想要提取一些形式的信息, 合理的利用 tag 的名字十分重要.

<!DOCTYPE html>

<html lang="cn">

<head>

<meta charset="UTF-8">

<title>Scraping tutorial 1 | 莫烦Python</title>

<link rel="icon" href="https://morvanzhou.github.io/static/img/description/tab\_icon.png">

</head>

<body>

<h1>爬虫测试1</h1>

<p>

这是一个在 <a href="https://morvanzhou.github.io/">莫烦Python</a>

<a href="https://morvanzhou.github.io/tutorials/scraping">爬虫教程</a> 中的简单测试.

</p>

</body>

</html>

## [匹配网页内容](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/1-01-understand-website/#%E5%8C%B9%E9%85%8D%E7%BD%91%E9%A1%B5%E5%86%85%E5%AE%B9)

所以这里我们使用 Python 的正则表达式 RegEx 进行匹配文字, 筛选信息的工作. 我有一个很不错的[正则表达式的教程](https://morvanzhou.github.io/tutorials/python-basic/basic/13-10-regular-expression/), 如果是初级的网页匹配, 我们使用正则完全就可以了, 高级一点或者比较繁琐的匹配, 我还是推荐使用 [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/). 不急不急, 我知道你想偷懒, 我之后马上就会教 beautiful soup 了. 但是现在我们还是使用正则来做几个简单的例子, 让你熟悉一下套路.

如果我们想用代码找到这个网页的 title, 我们就能这样写. 选好要使用的 tag 名称 <title>. 使用正则匹配.

**import** re

res = re.findall(r"<title>(.+?)</title>", html)

print("\nPage title is: ", res[0])

# Page title is: Scraping tutorial 1 | 莫烦Python

如果想要找到中间的那个段落 <p>, 我们使用下面方法, 因为这个段落在 HTML 中还夹杂着 tab, new line, 所以我们给一个 flags=re.DOTALL 来对这些 tab, new line 不敏感.

res = re.findall(r"<p>(.\*?)</p>", html, flags=re.DOTALL) # re.DOTALL if multi line

print("\nPage paragraph is: ", res[0])

# Page paragraph is:

# 这是一个在 <a href="https://morvanzhou.github.io/">莫烦Python</a>

# <a href="https://morvanzhou.github.io/tutorials/scraping">爬虫教程</a> 中的简单测试.

最后一个练习是找一找所有的链接, 这个比较有用, 有时候你想找到网页里的链接, 然后下载一些内容到电脑里, 就靠这样的途径了.

res = re.findall(r'href="(.\*?)"', html)

print("\nAll links: ", res)

# All links:

['https://morvanzhou.github.io/static/img/description/tab\_icon.png',

'https://morvanzhou.github.io/',

'https://morvanzhou.github.io/tutorials/scraping']

[下次](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/2-01-beautifulsoup-basic/)我们就来看看为了图方面, 我们如何使用 BeautifulSoup.