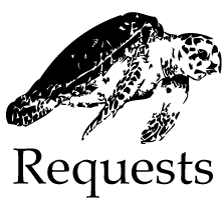
**多功能的 Requests**

作者: **莫烦**编辑: **莫烦 2017-12-30**

学习资料:

* [本节学习代码](https://github.com/MorvanZhou/easy-scraping-tutorial/blob/master/notebook/3-1-requests.ipynb)
* Requests 模块[英文官网](http://docs.python-requests.org/en/master/), [中文官网](http://cn.python-requests.org/zh_CN/latest/)

[之前](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/1-01-understand-website/) 我们通常使用 Python 的自带模块 urllib, 来提交网页请求. 这个模块能满足我们大部分的需求, 但是为了满足你日益膨胀的其他需求, 比如向网页发送信息, 上传图片等等, 我们还有一个伟大的 Python 外部模块 [requests](http://docs.python-requests.org/en/master/), 来有效的处理这些问题.

[](https://morvanzhou.github.io/static/results/scraping/3-1-1.png)

[**获取网页的方式**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/#%E8%8E%B7%E5%8F%96%E7%BD%91%E9%A1%B5%E7%9A%84%E6%96%B9%E5%BC%8F)

其实在加载网页的时候, 有几种类型, 而这几种类型就是你打开网页的关键. 最重要的类型 (method) 就是 get 和 post (当然还有[其他的](https://www.w3schools.com/tags/ref_httpmethods.asp), 比如 head, delete). 刚接触网页构架的朋友可能又会觉得有点懵逼了. 这些请求的方式到底有什么不同? 他们又有什么作用?

我们就来说两个重要的, get, post, 95% 的时间, 你都是在使用这两个来请求一个网页.

* post
  + 账号登录
  + 搜索内容
  + 上传图片
  + 上传文件
  + 往服务器传数据 等
* get
  + 正常打开网页
  + **不**往服务器传数据

这样看来, 很多网页使用 get 就可以了, 比如 [莫烦Python](https://morvanzhou.github.io/) 里的所有页面, 都是只是 get 发送请求. 而 post, 我们则是给服务器发送个性化请求, 比如将你的账号密码传给服务器, 让它给你返回一个含有你个人信息的 HTML.

从主动和被动的角度来说, post 中文是*发送*, 比较主动, 你**控制**了服务器返回的内容. 而 get 中文是*取得*, 是被动的, 你**没有**发送给服务器个性化的信息, 它**不会**根据你个性化的信息返回**不一样**的 HTML.

拿登录账号举个例子. 比如我想登录知乎账号(<https://www.zhihu.com/people/morvan/activities>, 在登录前, 我看到的页面是这样.

[](https://morvanzhou.github.io/static/results/scraping/3-1-2.png)

而如果我登录了(登录一般用 post), 我再次登录这个网页 <https://www.zhihu.com/people/morvan/activities>, 现在显示的 HTML 界面就和我上面那个不一样了, 这就是使用 post 给网页输入你的个性化信息后 (账号密码), 得到的服务器返回的个性化网页. 每个人的账号密码的到的页面都不一样.

[](https://morvanzhou.github.io/static/results/scraping/3-1-3.png)

而今天要说的 requests 模块就是干这些的. 它有着各种不同的请求方法, 而且用起来很方便.

[**安装 requests**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/#%E5%AE%89%E8%A3%85-requests)

[Requests](http://docs.python-requests.org/en/master/) 是一个 Python 的外部模块, 我们需要手动安装它. 简单的方法, 在你的 terminal 或者是 cmd, 使用 pip 安装就好了.

# python 2+

pip install requests

# python 3+

pip3 install requests

[官网](http://docs.python-requests.org/en/master/user/install/#install)上还提供了其他途径的安装.

[**requests get 请求**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/#requests-get-%E8%AF%B7%E6%B1%82)

有了 requests, 我们可以发送个中 method 的请求. 比如 get. 我们想模拟一下百度的搜索. 首先我们需要观看一下百度搜索的规律. 在百度搜索框中写上 “莫烦python” 我们发现它弹出了一串这么长的网址.

[](https://morvanzhou.github.io/static/results/scraping/3-1-4.png)

但是仔细一看, 和 “莫烦Python” 有关的信息, 只有前面一小段 (“s?wd=莫烦python”), 其他的对我们来说都是无用的信息. 所以我们现在来尝试一下如果把后面的”无用” url 都去掉会怎样? Duang! 我们还是能搜到 “莫烦python”.

[](https://morvanzhou.github.io/static/results/scraping/3-1-5.png)

所以 “s?wd=莫烦python” 这就是我们搜索需要的关键信息. 我们就能用 get 来搭配一些自定义的搜索关键词来用 python 个性化搜索. 首先, 我们固定不动的网址部分是 “http://www.baidu.com/s”, ? 后面的东西都是一些参数 (parameters), 所以我们将这些 parameters 用 python 的字典代替, 然后传入 requests.get() 功能. 然后我们还能用 python (webbrowser模块) 打开一个你的默认浏览器, 观看你是否在百度的搜索页面.

**import** requests

**import** webbrowser

param = {"wd": "莫烦Python"} # 搜索的信息

r = requests.get('http://www.baidu.com/s', params=param)

print(r.url)

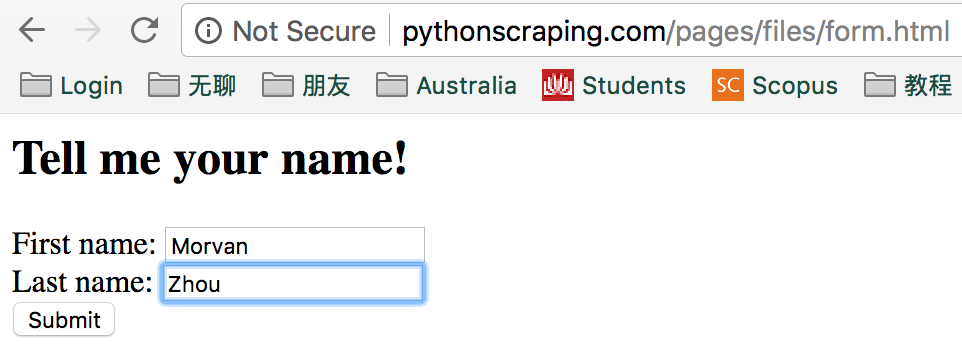
webbrowser.open(r.url)

# http://www.baidu.com/s?wd=%E8%8E%AB%E7%83%A6Python

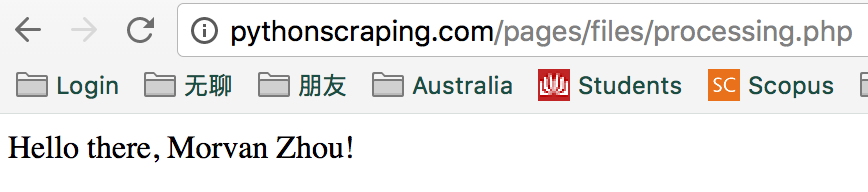
这时, python 会弹出一个浏览器界面, 然后你看到的, 就是 “莫烦Python” 的搜索结果了.

[**requests post 请求**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/#requests-post-%E8%AF%B7%E6%B1%82)

post 又怎么用呢? 我们举个小例子, 在这个[简单网页](http://pythonscraping.com/pages/files/form.html)中, 我们有一个提交信息的窗口, 如果我提交上去这个信息, 那边的服务器会更加这个提交的信息返回出另一个网页. 这就是网页怎么样使用你 post 过去的信息了.

[](https://morvanzhou.github.io/static/results/scraping/3-1-6.png)

比如我在这里填上自己的姓名, 当我点 “submit” 的时候, 这个姓名(Morvan, Zhou) 就会被提交给服务器, 然后它会根据提交的姓名返回这个网页.

[](https://morvanzhou.github.io/static/results/scraping/3-1-7.png)

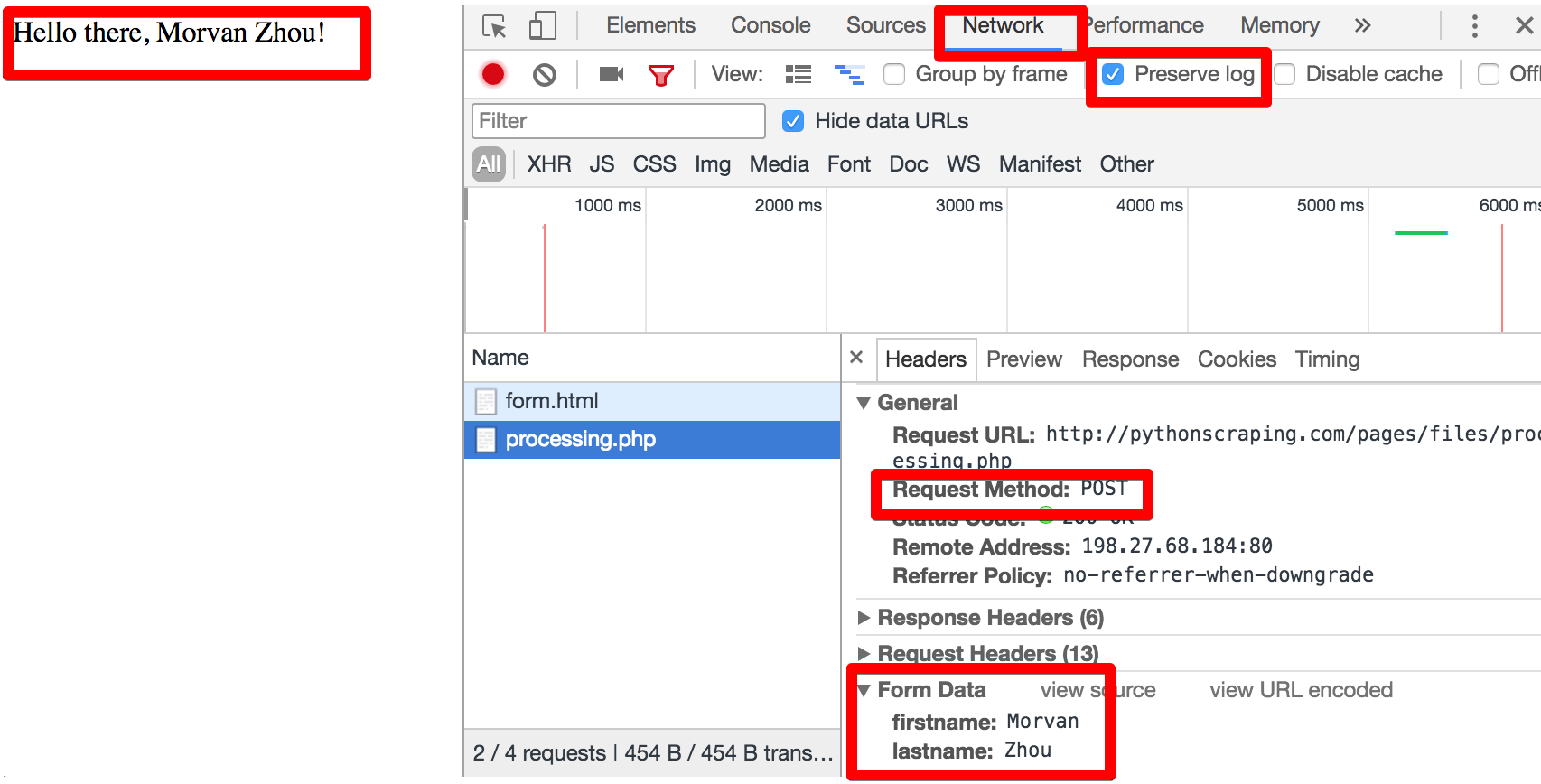
这样咋看起来好像和上面讲的 get 百度搜索没区别呀? 都是提交一些信息, 返回一个界面. 但是, **重点来了**. 你看看网址栏. 你 post 上去的个人信息, 有没有显示在 url 里? 你愿意将你的私密信息显示在 url 里吗? 你 post 过去的信息是交给服务器内部处理的. 不是用来显示在网址上的.

懂了这些, 我们就来看使用 python 和 requests 怎么做 post 这个操作吧.

首先我们调出浏览器的 inspect (右键点击 inspect, 中文是检查还是什么来着). 然后发现我们填入姓名的地方原来是在一个 <form> 里面.

[](https://morvanzhou.github.io/static/results/scraping/3-1-8.png)

这个 <form> 里面有一些 <input> 个 tag, 我们仔细看到 <input> 里面的这个值 name="firstname" 和 name="lastname", 这两个就是我们要 post 提交上去的关键信息了. 我们填好姓名, 为了记录点击 “submit” 后, 浏览器究竟发生了什么翻天覆地的变化, 我们在 inspect 窗口, 选择 Network, 勾选 Preserve log, 再点击 “submit”, 你就能看到服务器返回给你定制化后的页面时, 你使用的方法和数据.

[](https://morvanzhou.github.io/static/results/scraping/3-1-9.png)

这些数据包括了:

* Request URL (post 要用的 URL)
* Request Method (post)
* Form Data (post 去的信息)

有了这些记录, 我们就能开始写 Python 来模拟这一次提交 post 了. 根据 'firstname' 和 'lastname', 也就是上图里面的 Form data, 组织成一个 python 字典. 让后把这个字典传入 requests.post(), 注意, 这里的 post 里面的 url, 不是我们填表时的 url (http://pythonscraping.com/pages/files/form.html), 而是要把 Form 信息提交去的那个网页, 也就是上图中查看到的 Request URL (http://pythonscraping.com/files/processing.php).

data = {'firstname': '莫烦', 'lastname': '周'}

r = requests.post('http://pythonscraping.com/files/processing.php', data=data)

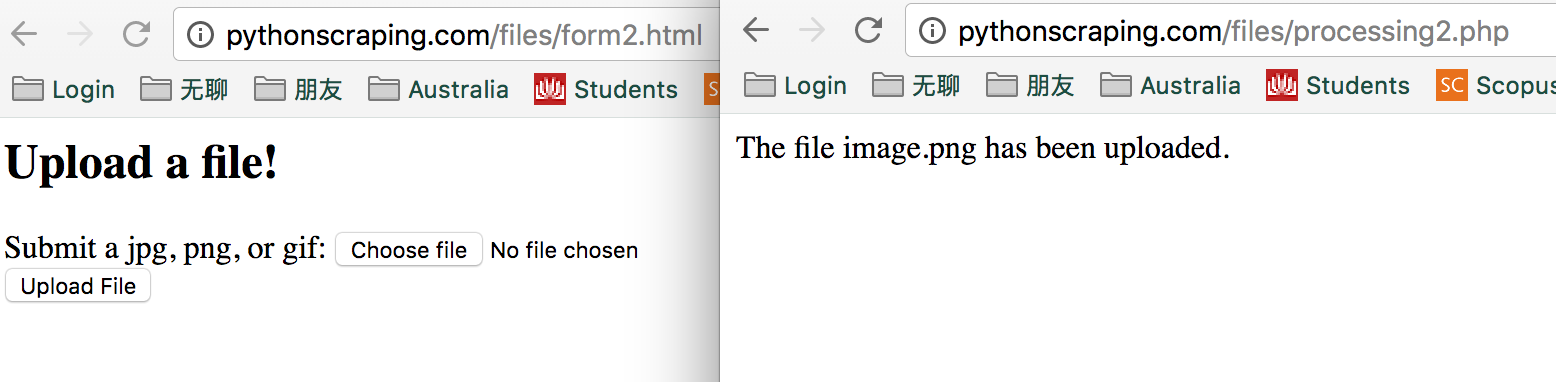
print(r.text)

# Hello there, 莫烦 周!

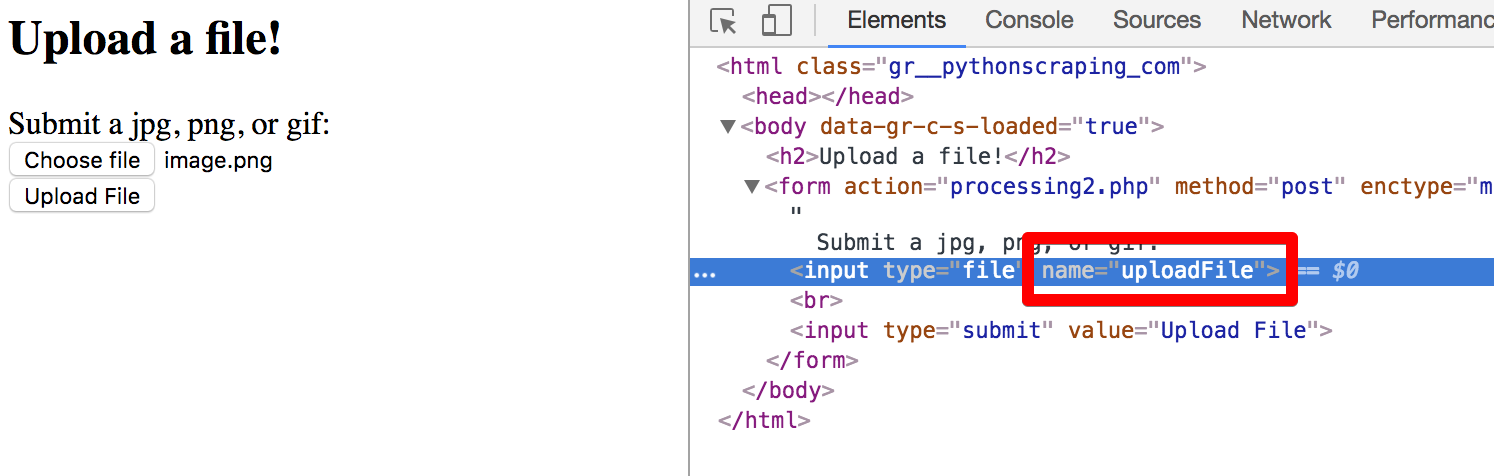
通过这个练习, 我们对 HTML 中的 Form 有了理解, 学会了怎么样使用 python 来提交 Form, 登录上提交后的页面.

[**上传图片**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/#%E4%B8%8A%E4%BC%A0%E5%9B%BE%E7%89%87)

传照片也是 post 的一种, 我们得将本地的照片文件传送到服务器. 我们使用这个[网页](http://pythonscraping.com/files/form2.html)来模拟一次传照片的过程.

[](https://morvanzhou.github.io/static/results/scraping/3-1-10.png)

如果你留意观察 url, 你会发现, 传送完照片以后的 url 有变动. 我们使用同样的步骤再次检查, 发现, “choose file” 按键链接的 <input> 是一个叫 uploadFile 的名字. 我们将这个名字记下, 放入 python 的字典当一个 “key”.

[](https://morvanzhou.github.io/static/results/scraping/3-1-11.png)

接着在字典中, 使用 open 打开一个图片文件, 当做要上传的文件. 把这个字典放入你的 post 里面的 files 参数. 就能上传你的图片了, 网页会返回一个页面, 将你的图片名显示在上面.

file = {'uploadFile': open('./image.png', 'rb')}

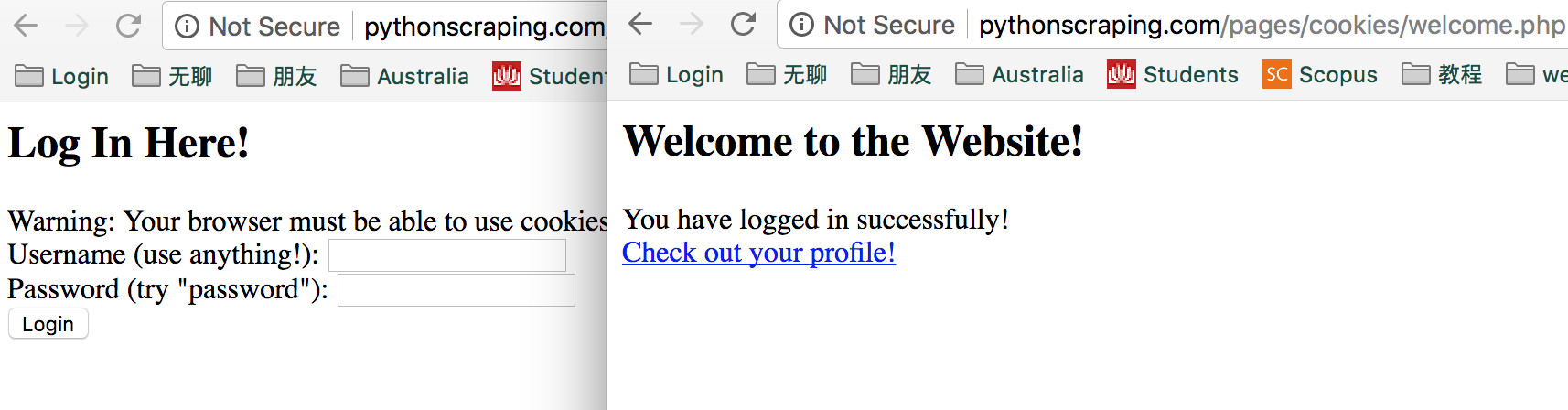
r = requests.post('http://pythonscraping.com/files/processing2.php', files=file)

print(r.text)

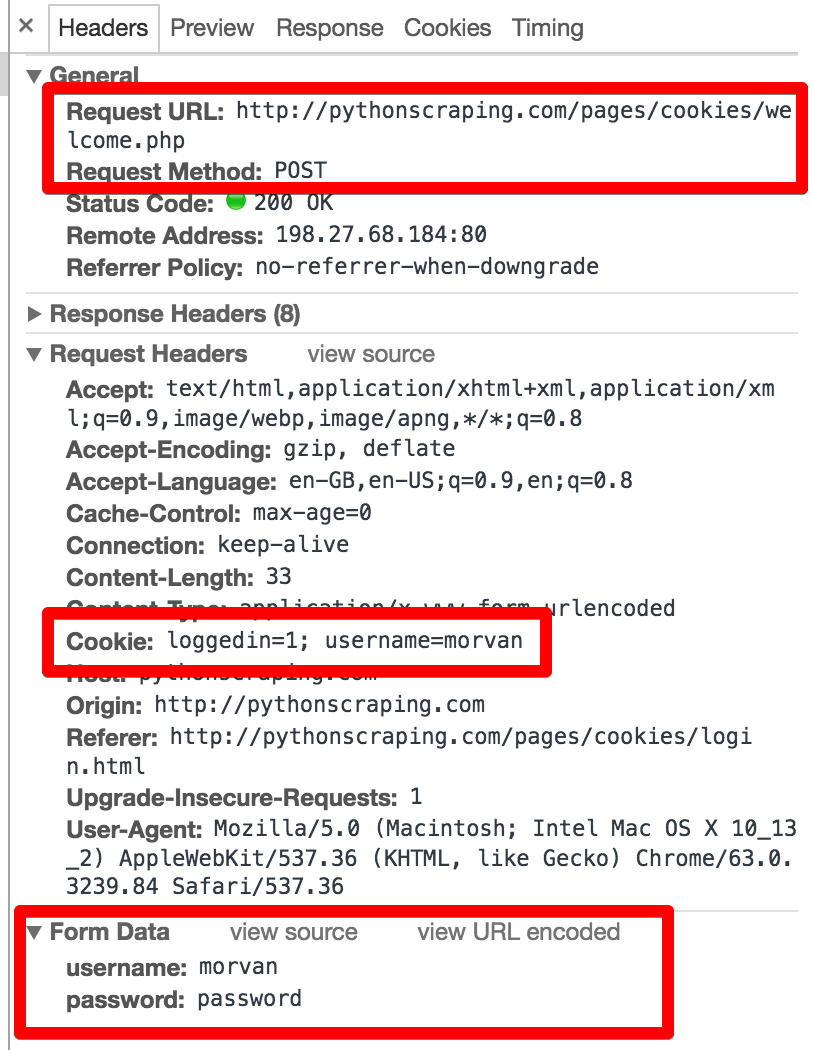
# The file image.png has been uploaded.

[**登录**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/#%E7%99%BB%E5%BD%95)

用 post 还有一个重要的, 就是模拟登录. 再登录的时候发生了什么事情呢? 我们使用这个[简单的登录网页](http://pythonscraping.com/pages/cookies/login.html)进行说明.

[](https://morvanzhou.github.io/static/results/scraping/3-1-12.png)

通过之前提到的方法, 我们观察一下浏览器给出的记录. 三个重要的方面都被我圈出来了.

[](https://morvanzhou.github.io/static/results/scraping/3-1-13.png)

我们总结一下, 为了这次登录账号, 我们的浏览器做了什么.

1. 使用 post 方法登录了第一个红框的 url
2. post 的时候, 使用了 Form data 中的用户名和密码
3. **生成了一些 cookies**

第三点我们是从来没有提到过的. cookie, 听起来很熟呀! 每当游览器出现问题的时候, 网上的解决方法是不是都有什么清除 cookie 之类的, 那 cookie 实际上是什么呢? [这里](https://baike.baidu.com/item/cookie/1119?fr=aladdin)给出了和全面的介绍.

简单来说, 因为打开网页时, 每一个页面都是不连续的, 没有关联的, cookies 就是用来衔接一个页面和另一个页面的关系. 比如说当我登录以后, 浏览器为了保存我的登录信息, 将这些信息存放在了 cookie 中. 然后我访问第二个页面的时候, 保存的 cookie 被调用, 服务器知道我之前做了什么, 浏览了些什么. 像你在网上看到的广告, 为什么都可能是你感兴趣的商品? 你登录淘宝, 给你推荐的为什么都和你买过的类似? 都是 cookies 的功劳, 让服务器知道你的个性化需求.

所以大部分时候, 每次你登录, 你就会有一个 cookies, 里面会提到你已经是登录状态了. 所以 cookie 在这时候很重要. cookies 的传递也特别重要, 比如我用 requests.post + payload 的用户信息发给网页, 返回的 r 里面会有生成的 cookies 信息. 接着我请求去登录后的页面时, 使用 request.get, 并将之前的 cookies 传入到 get 请求. 这样就能已登录的名义访问 get 的页面了.

payload = {'username': 'Morvan', 'password': 'password'}

r = requests.post('http://pythonscraping.com/pages/cookies/welcome.php', data=payload)

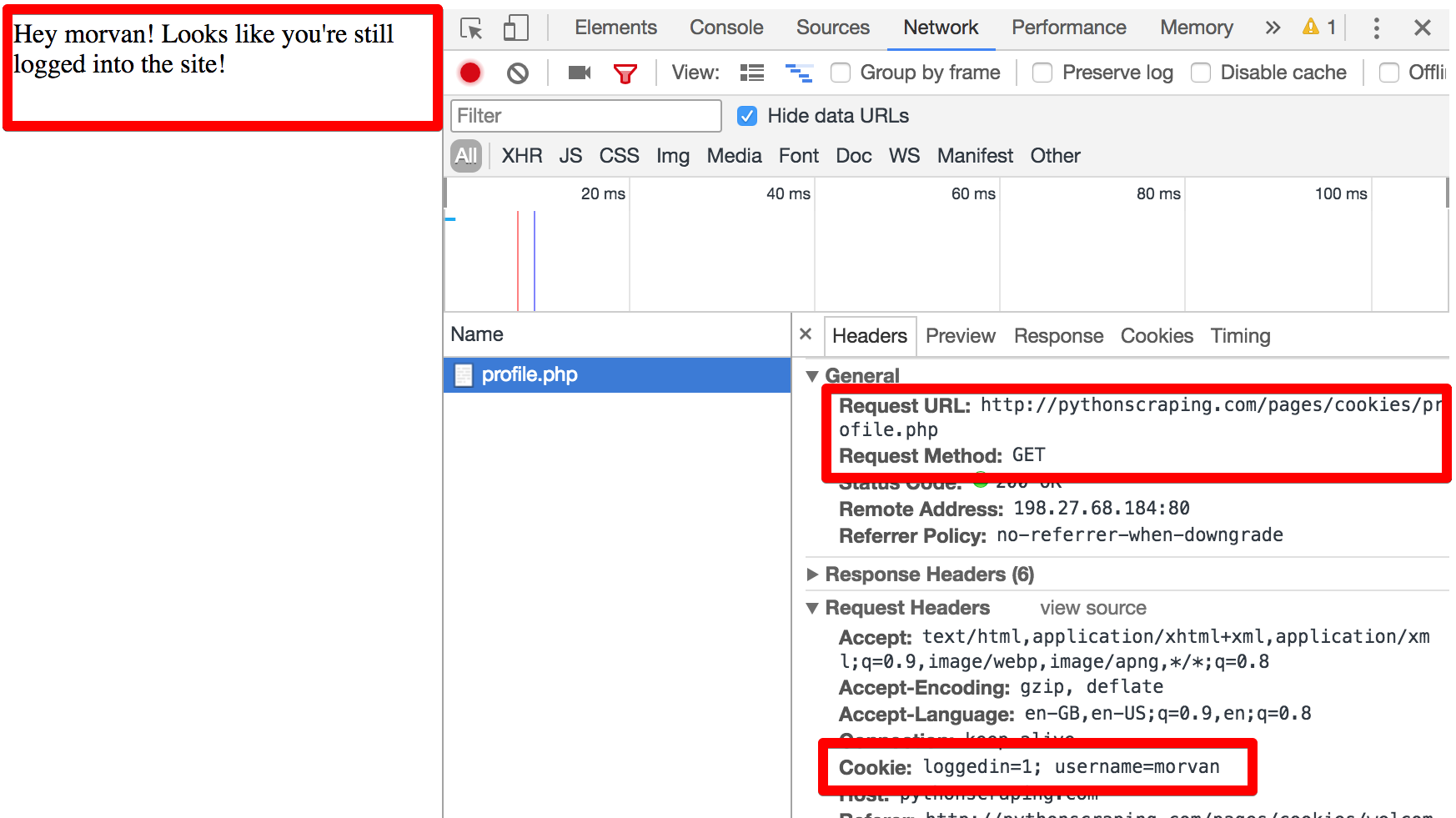
print(r.cookies.get\_dict())

# {'username': 'Morvan', 'loggedin': '1'}

r = requests.get('http://pythonscraping.com/pages/cookies/profile.php', cookies=r.cookies)

print(r.text)

# Hey Morvan! Looks like you're still logged into the site!

[](https://morvanzhou.github.io/static/results/scraping/3-1-14.png)

[**使用 Session 登录**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/#%E4%BD%BF%E7%94%A8-Session-%E7%99%BB%E5%BD%95)

不过每次都要传递 cookies 是很麻烦的, 好在 requests 有个很 handy 的功能, 那就是 Session. 在一次会话中, 我们的 cookies 信息都是相连通的, 它自动帮我们传递这些 cookies 信息. 这时我感叹, 程序员真会偷懒~ 哈哈.

同样是执行上面的登录操作, 下面就是使用 session 的版本. 创建完一个 session 过后, 我们直接只用 session 来 post 和 get. 而且这次 get 的时候, 我们并没有传入 cookies. 但是实际上 session 内部就已经有了之前的 cookies 了.

session = requests.Session()

payload = {'username': 'Morvan', 'password': 'password'}

r = session.post('http://pythonscraping.com/pages/cookies/welcome.php', data=payload)

print(r.cookies.get\_dict())

# {'username': 'Morvan', 'loggedin': '1'}

r = session.get("http://pythonscraping.com/pages/cookies/profile.php")

print(r.text)

# Hey Morvan! Looks like you're still logged into the site!

这就是我们这次的教学, 想了解更多 requests 使用的朋友看到[这里](http://docs.python-requests.org/en/master/). [下一次](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-02-download/)我们就来看看如何用不同的方式来下载网页内容.