

# day02-课堂笔记

## day02-课堂笔记

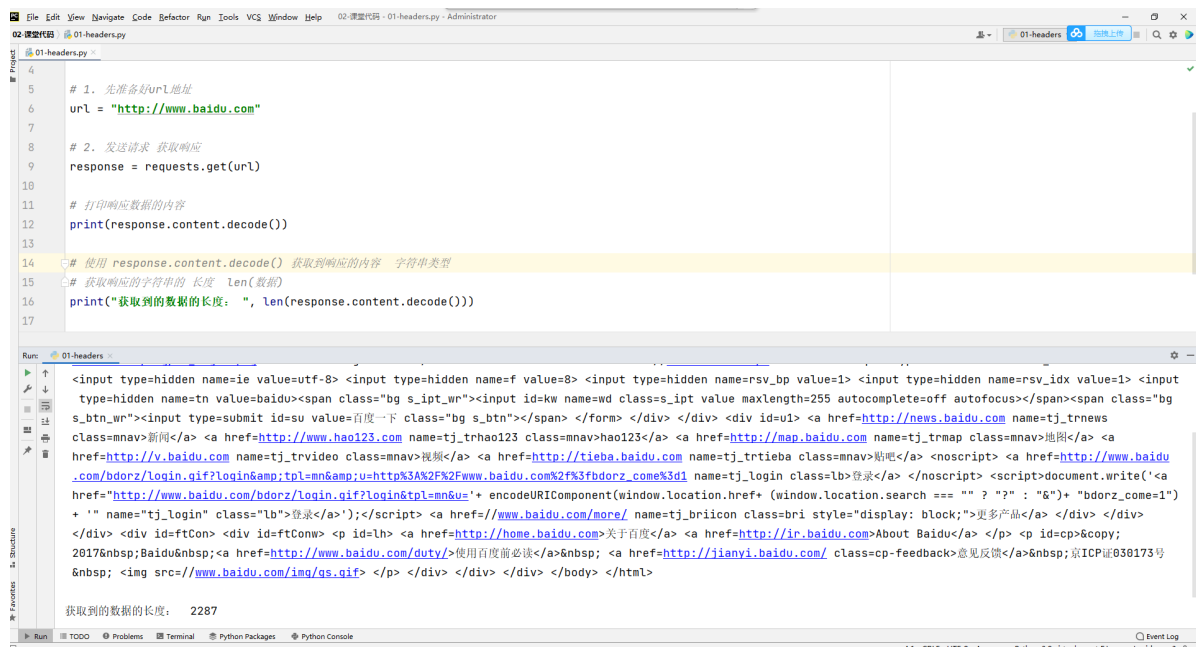
1. requests发送 带headers的请求
2. requests发送 带params的请求
  - 2.1 第一种方式
  - 2.2 第二种方式
3. requests发送post请求
4. requests使用代理IP

## 1. requests发送 带headers的请求

就是在使用requests模块发送请求的时候 携带 请求头信息。

去请求百度 拿到的数据内容

```
1 # 需求： 使用requests模块去请求baidu首页，拿到首页的响应内容 打印内容，以及去打印响应内容
  的数据的长度
2
3 import requests
4
5 # 1. 先准备好url地址
6 url = "http://www.baidu.com"
7
8 # 2. 发送请求 获取响应
9 response = requests.get(url)
10
11 # 打印响应数据的内容
12 print(response.content.decode())
13
14 # 使用 response.content.decode() 获取到响应的内容 字符串类型
15 # 获取响应的字符串的 长度 len(数据)
16 print("获取到的数据的长度：", len(response.content.decode()))
17
```



通过浏览器打开百度首页，查看网页源码，我们发现，使用浏览器拿到的百度首页的html的源码 要比在代码中获取到的百度首页的源码要多得多。

通过对比 代码中携带的请求头 和 浏览器中携带的请求头信息，发现，User-Agent 完全不一样的。

User-Agent： 用户代理，浏览器的身份标识（身份证）

程序中拿到不完整数据的原因： 在程序中使用requests模块发送请求的是，在请求头中携带的User-Agent并不是一个正常浏览器的User-Agent，百度服务器接收到我们的请求之后，是可以获取到User-Agent的，对比，发现请求是一个爬虫程序。

如何去解决这个问题：

- 在发送请求之前，将requests模块默认使用的 User-Agent 改为一个 正常浏览器的User-Agent即可。

语法：

```
1 # 准备一个字典， 放的就是要替换的请求头的信息。 key: value
2 headers = {
3     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
4     AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36"
5 }
6 # 在发送请求的时候 使用准备好的User-Agent
7 # 在发送请求的时候，使用的就是我们自己修改好的一个正常的user-agent去请求服务器。
8 response = requests.get(url, headers=headers)
```

后续想要能够正确的拿到响应数据内容，基本上在每次发送请求的时候都需要携带一个正常的浏览器的User-Agent

```
1 # 需求： 使用requests模块去请求baidu首页，拿到首页的响应内容 打印内容，以及去打印响应内
  容的数据的长度
2
3 import requests
4
5 # 1. 先准备好url地址
6 url = "http://www.baidu.com"
7
8 # 准备一个请求头的字典
9 headers = {
10     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
11     AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36"
12 }
13
14 # 2. 发送请求 获取响应
15 # 发送请求的时候，携带上我们自己修改好的请求头中的User-Agent
16 response = requests.get(url, headers=headers)
17
18 # 打印响应数据的内容
19 print(response.content.decode())
```

```

19
20 # 使用 response.content.decode() 获取到响应的内容 字符串类型
21 # 获取响应的字符串的 长度 len(数据)
22 print("获取到的数据的长度: ", len(response.content.decode()))
23
24 # 获取当前这次请求 携带的 请求头信息
25 print(response.request.headers)

```

```

24 # 获取当前这次请求 携带的 请求头信息
25 print(response.request.headers)
26
Run: Q2-with_headers
});
</script>
<script src="https://dss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/js/s_super_index-855fcd82e.js"></script>
<script src="https://dss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/js/min_super-5685956f44.js"></script>

<script>
if(navigator.cookieEnabled){
    document.cookie="NOJS=;expires=Sat, 01 Jan 2000 00:00:00 GMT";
}
</script>

<script src="https://dss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/js/components/hotsearch-c445acece1.js"></script>
<script defer src="//h5p0static.baidu.com/cd37ed75a9387c5b.js"></script>

</body>

</html>
获取到的数据的长度: 308307
{'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive', 'Cookie': 'BAIDUID=A523D6D212C9FBC46FE8EEB8EF5C71D:F6=1; BIDUPSID=A523D6D212C9FBC4C7C19338C41C76DB; PSTM=1630385327; BD_LAST_QID=14230731513700132126'}
Process finished with exit code 0

```

## 2. requests发送 带params的请求

params参数的意思,

url完整格式: 协议://域名:port/xxxx/xxx?key=value&

这个地方所说的参数 url地址中, 格式以? 开始, key=value的格式, 就是参数, 查询字符串参数 (query\_string)

```

1 | https://www.baidu.com/s?wd=要搜索的关键字

```

参数可以分为

- 必传参数: 在请求的时候是必须要携带的参数, 如果不携带此参数或对程序的结果有影响、
- 非必传参数: “在请求的时候 可以传 也可以不传, 不会影响到我们程序的结果。

需求: 输入一个关键字, 使用爬虫实现百度搜索操作。

### 2.1 第一种方式

直接在url地址中携带参数

```

1 # 需求: 输入一个关键字, 使用爬虫实现百度搜索操作。
2
3 import requests
4

```

```

5 # 获取要去搜索的关键字
6 word = input("请输入要搜索的关键字: ")
7
8 # 准备一个请求头的字典
9 headers = {
10     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36"
11 }
12
13 # 1. 准备url地址
14 url = "https://www.baidu.com/s?wd={}".format(word)
15
16 # 2. 对url地址发送请求 获取到响应
17 response = requests.get(url, headers=headers)
18
19 print(response.content.decode())

```

## 2.2 第二种方式

要使用到requests模块中提供的一个参数 params

语法

```

1 # 准备一个参数的字典 key:value
2 # 因为参数的格式就是一种 key=value
3 params = {key: value}
4 params = {"wd": "要搜索的关键词", "":""}
5
6 # 在发送请求的时候 携带 字典即可
7 response = requests.get(url, headers={}, params=params)

```

```

1 # 需求: 输入一个关键字, 使用爬虫实现百度搜索操作。
2
3 import requests
4
5 # 获取要去搜索的关键字
6 word = input("请输入要搜索的关键字: ")
7
8 # 准备一个请求头的字典
9 headers = {
10     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36"
11 }
12
13 # 1. 准备url地址
14 url = "https://www.baidu.com/s"
15
16 # 发送请求的时候 是需要携带参数, 所以先准备好我们要携带的参数
17 params = {
18     "wd": word
19 }
20
21 # 2. 对url地址发送请求 获取到响应
22 response = requests.get(url, headers=headers, params=params)
23
24 print(response.content.decode())

```

### 3. requests发送post请求

浏览器请求服务器的时候，请求方式常见的方式有四种：

- get 在地址栏中看到的url地址，发送的请求都是get请求， 查询数据（从服务器获取数据）
- post： 在浏览器地址栏中是不能看到post请求的url地址。提交数据（新增数据）
- put： 修改，发送请求的时候也是需要向服务器提交数据。
- delete： 删除数据。

post请求，前面学到的使用requests模块发送的请求都是get请求，

post请求的应用场景：

- 登录
- 注册
- 上传

get和post之间的区别：

- get请求的url地址在浏览器地址栏中是可以看到的，并且携带的数据也是可以直接看到的，
- post请求的url地址是不能在浏览器地址栏中看到的，并且携带的数据也是不能直接看到的，想要去看post的url地址以及携带的数据要使用 浏览器的开发者工具（就是在浏览器中右键 检查）中等Network 选项中是可以找到的。
- post请求相对于get请求 要安全。
- post请求在向服务器发送数据的时候，数据的大小是不受限制的，而get请求是有限制。

如何在代码中去发送post请求

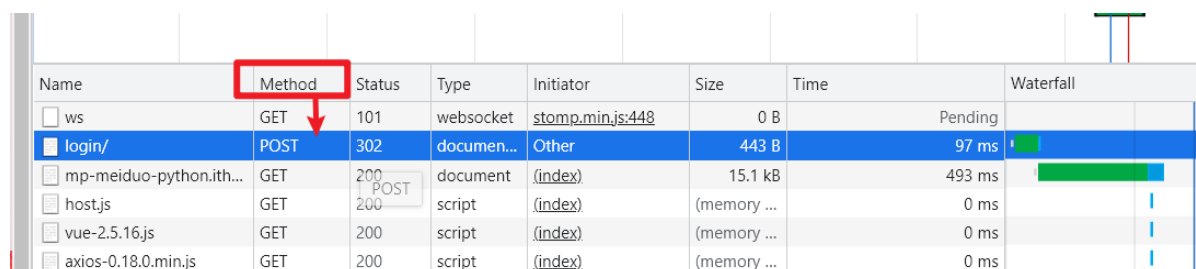
语法：

```
1 requests.post(url, data={})
2
3 data参数：
4  指的就是请求体数据，
5  在发送http请求的时候，请求报文格式， 请求行 请求头 空行 请求体
6  发送get请求的时候是没有请求体，发送post请求的时候是有请求体的，请求体中就是我们向服务器要发送的数据。
```

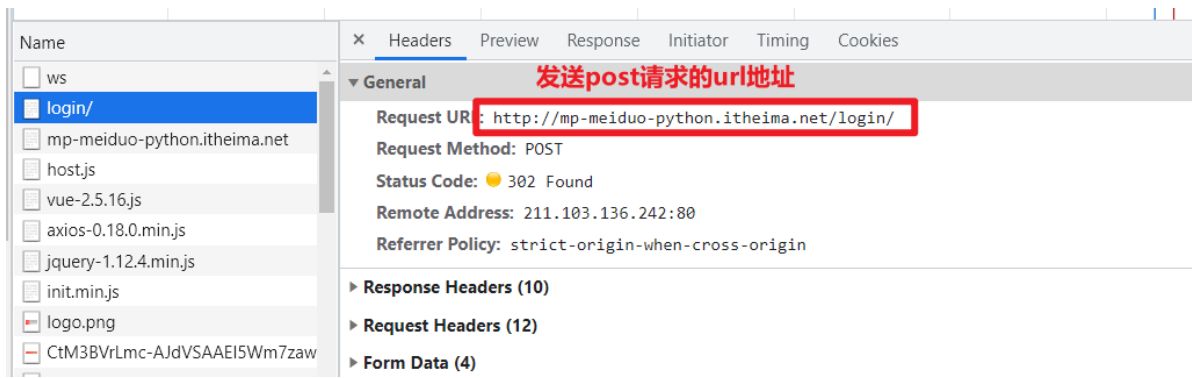
寻找 post请求的url地址，

在浏览器中先打开登录页面，鼠标右键 检查 network 将用户名和密码输入到输入框中，去点击一下登录按钮，

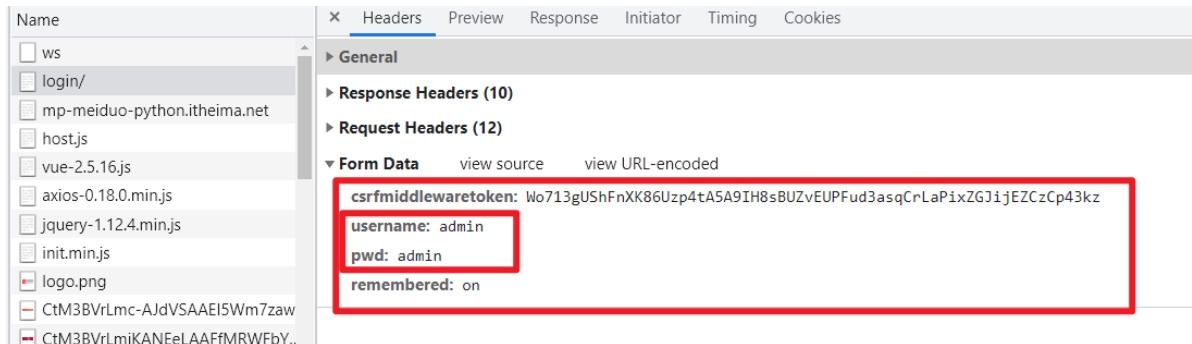
在network中就会有许多的url地址显示出来，就找到 method 是post的即可。



Name	Method	Status	Type	Initiator	Size	Time	Waterfall
ws	GET	101	websocket	stomp.min.js:448	0 B	Pending	
login/	POST	302	document...	Other	443 B	97 ms	
mp-meiduo-python.ith...	GET	200	document	(index)	15.1 kB	493 ms	
host.js	GET	200	script	(index)	(memory ...)	0 ms	
vue-2.5.16.js	GET	200	script	(index)	(memory ...)	0 ms	
axios-0.18.0.min.js	GET	200	script	(index)	(memory ...)	0 ms	



请求体数据就是 form data



在上述过程中 我们已经找到了 登录的url地址以及携带的请求体数据，可以在代码中去实现。

```
1 import requests
2
3 # 1. 准备url地址， post的url地址
4 url = "http://mp-meiduo-python.itheima.net/login/"
5
6 # 准备请求体字典（登录，先将用户名和密码准备好）
7 data = {
8     "csrfmiddlewaretoken":
9     "Wo713gUShFnXK86Uzp4tA5A9IH8sBUZvEUPFud3asqCrLaPixZGJijEZCzCp43kz",
10     "username": "admin",
11     "pwd": "admin",
12     "remembered": "on"
13 }
14
15 # 2. 发送请求， 登录 发送的请求是一个post请求
16 response = requests.post(url, data=data)
17
18 # 打印响应状态码
19 print(response.status_code)
20
21 # 如何去验证到底有没有登录成功呢？ 根据 响应的内容来进行判断，
22 # 如果打印出来的响应内容 中市首页的内容，表示登录成功了,如果 获取到的页面的内容是 显示的
23 # 用户名或者密码错误,表示登录不成功
24 print(response.content.decode())
```

## 4. requests使用代理IP

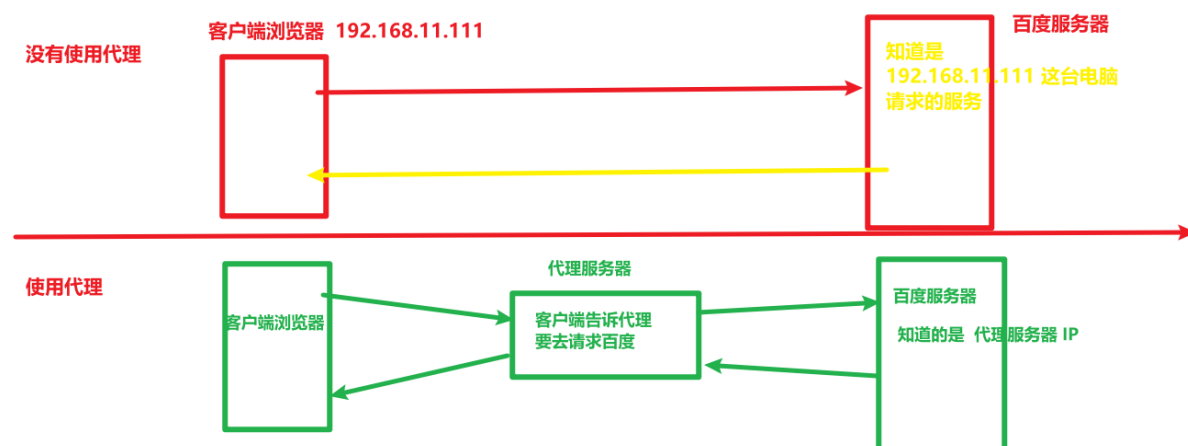
场景：现在使用的是 机房 这台电脑，每一台电脑都有一个IP地址，原因就是在发送请求的时候，会携带着当前这台电脑的IP地址去访问服务器。

对方服务器是可以获取到我们的电脑的IP地址的，

使用爬虫程序去请求服务器的时候，速度是特别快的，但是每一次请求的时候都是使用的同一个IP地址，如果在一秒中之内，对百度服务器请求了

1000次，1000次请求使用的都是同一个IP。百度服务器就会将 IP 封掉。

现在服务器已经将IP封掉，但是现在我又想去访问百度。这个时候我就可以换一个IP，使用代理IP。



使用代理IP的目的：

- 防止被封IP
- 防止被追究责任

在爬虫代码中如何去使用代理IP，语法：

```
1 # 先去准备一个代理IP的字典
2 proxy = {
3     "协议": "协议://代理IP:port"
4 }
5
6 # 在发送请求的时候 就可以 将代理IP的字典 传递给 proxies参数，就可以实现在发送请求的时候使用代理IP。
7 response = requests.get(url, proxies=proxy)
```

使用一个代理IP

```
1 import requests
2
3
4 # 1. 准备url地址
5 url = "http://www.baidu.com"
6
7 # 准备一个代理IP的字典
8 proxy = {
9     "http": "http://106.45.104.87:3256"
10 }
```

```
11
12 # 2. 发送请求获取响应
13 response = requests.get(url, proxies=proxy)
14
15 # 打印响应状态码
16 print(response.status_code)
```

使用代理IP池，每次请求随机使用一个IP

```
1 import requests, random
2
3
4 # 1. 准备url地址
5 url = "http://www.baidu.com"
6
7 # 准备一个代理IP的字典
8 proxy_list = [
9     {"http": "http://106.45.104.87:3256"},
10    {"http": "http://106.45.104.88:3256"},
11    {"http": "http://106.45.104.89:3256"},
12    {"http": "http://106.45.104.90:3256"},
13    {"http": "http://106.45.104.91:3256"},
14    {"http": "http://106.45.104.93:3256"},
15    {"http": "http://106.45.104.66:3256"},
16    {"http": "http://106.45.104.77:3256"},
17    {"http": "http://106.45.104.44:3256"},
18    {"http": "http://106.45.104.11:3256"}
19 ]
20
21 # 在上面已经准备好了一个代理IP池，每一次发送请求的时候 随机的取一个IP来使用
22 proxy = random.choice(proxy_list)
23
24 print(f'本次请求使用的代理IP是: {proxy}')
25
26 # 2. 发送请求获取响应
27 response = requests.get(url, proxies=proxy)
28
29 # 打印响应状态码
30 print(response.status_code)
```



