



使用requests网页爬取



朱映秋

CONTENTS

▶ **PART 1**

库安装和实例展示

▶ **PART 2**

GET请求

▶ **PART 3**

POST请求

▶ **PART 4**

响应



- 在抓取页面的过程中，我们需要模拟浏览器向服务器发出请求，所以会用到一些Python库来实现HTTP请求操作，这里我们使用的第三方库是requests。
- 相关文档
 - PyPI: <https://pypi.python.org/pypi/requests>



- requests库是一个原生的HTTP库，比urllib3库更为容易使用。requests库发送原生的HTTP 1.1请求，无需手动为URL添加查询字串，也不需要为POST数据进行表单编码。相对于urllib3库，requests库会自动处理HTTP连接的保持和重用，包括Keep-alive功能和连接池的管理，简化了对HTTP连接的管理过程。requests库包含的特性如下。

| 连接特性 | 连接特性 | 连接特性 |
|----------------|---------------------|----------|
| Keep-Alive&连接池 | 基本/摘要式的身份认证 | 文件分块上传 |
| 国际化域名和URL | 优雅的key/value Cookie | 流下载 |
| 带持久Cookie的会话 | 自动解压 | 连接超时 |
| 浏览器式的SSL认证 | Unicode响应体 | 分块请求 |
| 自动内容解码 | HTTP(S)代理支持 | 支持.netrc |



➤ 1. 生成请求

- requests库生成请求的代码非常便利，其使用的request方法的语法格式如下。
- request方法常用的参数及其说明如下。

`requests.request.method(url,**kwargs)`

| 参数 | 说明 |
|----------|---|
| method | 接收string。表示请求的类型，如“GET”、“HEAD”、“DELETE”等。无默认值 |
| url | 接收string。表示字符串形式的网址。无默认值 |
| **kwargs | 接收dict或其他Python中的类型的数据。依据具体需要及请求的类型可添加的参数，通常参数赋值为字典类型或为具体数据 |



➤ 2. 查看状态码与编码

- 需要注意的是，当requests库猜测错时，需要手动指定encoding编码，避免返回的网页内容解析出现乱码。
- 虽然手动指定编码可以解决问题，但对于处理多个网页的爬取过程中，每个网页可能采用不同的编码，手动指定编码会显得不够灵活，无法自适应对应爬取过程中不同网页的编码，这时可以使用chardet库来检测字符串或文件的编码，它提供了一种更便捷灵活的方式来自适应地识别编码。，chardet库是一个非常优秀的字符串／文件编码检测模块。
- chardet库使用detect方法检测给定字符串的编码，detect方法常用的参数及其说明如下。

| 参数 | 说明 |
|-----------------|----------------------------|
| byte_str | 接收string。表示需要检测编码的字符串。无默认值 |



➤ 3. 请求头与响应头处理

- requests库中对请求头的处理与urllib3库类似，也使用headers参数在GET请求中上传参数，参数形式为字典。使用headers属性即可查看服务器返回的响应头，通常响应头返回的结果会与上传的请求参数对应。

➤ 4. Timeout设置

- 为了避免程序在等待服务器响应时陷入无限等待的情况，可以使用requests库的timeout参数进行设置。在requests库中，可以通过在发起请求时设置timeout参数来指定等待服务器响应的超时时间，以秒为单位。如果超过了设定的超时时间，请求将会被中断，程序将不再等待响应并继续执行后续代码。

➤ 5. 生成完整HTTP请求

- 使用requests库的request方法向网站“<http://www.tipdm.com/tipdm/index.html>”发送一个完整的GET请求，该请求包含链接、请求头、响应头、超时时间和状态码，并且编码应正确设置。



- 在命令行界面中运行如下命令，即可完成Requests库的安装：pip install requests

```
In [1]: pip install requests
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (2.27.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from requests) (2.0.4)
Note: you may need to restart the kernel to use updated packages.
```




- wheel是Python的一种安装包，其后缀为.whl，在网速较差的情况下可以选择下载wheel文件再安装，然后直接用pip3命令加文件名安装即可。不过在这之前需要先安装wheel库，安装命令如下： `pip install wheel`
- 然后到PyPI上下载对应的wheel文件，打开
- <https://pypi.python.org/pypi/requests/2.22.0#downloads>，下载requests-2.22.0-py2.py3-none-any.whl到本地。
- 随后在命令行界面进入wheel文件目录，利用pip安装即可：
- `pip3 install requests-2.17.3-py2.py3-none-any.whl`

```
C:\Users\Administrator>pip3 install requests-2.17.3-py2.py3-none-any.whl
```



- 为了验证库是否已经安装成功，可以在命令行模式测试一下：import requests

```
1 import requests
```

- 如果没有出现报错提示，就证明已经成功安装了requests。



```
1 import requests
```

```
1 res = requests.get('https://baidu.com/')
2 print(type(res))
```

```
<class 'requests.models.Response'>
```

```
1 print(res.status_code)
```

```
200
```

```
1 print(type(res.text))
2 print(res.text)
```

```
<class 'str'>
<!DOCTYPE html>
<!--STATUS OK--><html> <head><meta http-equiv=content-type content=text/html;charset=utf-8>
<meta content=always name=referrer><link rel=stylesheet type=text/css href=http://sl.bdstatic
%ä°|ä, ä, i¼ ä½ ä° ±ç ¥é </title></head> <body link=#0000cc> <div id=wrapper> <d
s_form> <div class=s_form_wrapper> <div id=lg> <img hidefocus=true src=//www.baidu.com/img/b
d=form name=f action=//www.baidu.com/s class=fm> <input type=hidden name=bdorz_come value=1>
type=hidden name=f value=8> <input type=hidden name=rsv_bp value=1> <input type=hidden name=
e=baidu><span class="bg s_ipt_wr"><input id=kwd name=wd class=s_ipt value maxlength=255 autoc
tn_wr"><input type=submit id=submit value=ç %ä°|ä, ä, class="bg s_btn"></span> </form> </div
om name=tj_trnews class=mnav>æ ° é »</a> <a href=http://www.hao123.com name=tj_trhao123 cl
```

```
1 print(res.cookies)
```

```
<RequestsCookieJar[<Cookie BDORZ=27315 for .baidu.com/>]>
```

➤ get()方法得到一个Response对象，输出：

- Response的类型
- 状态码
- 响应体的类型
- 响应体的内容
- Cookies
- 返回类型：requests.models.Response
- 响应体的类型：字符串str
- Cookies的类型：RequestsCookieJar



- 使用get()方法成功实现一个GET请求，更方便之处在于其他的请求类型依然可以用一句话来完成，示例如下：

```
[108]: 1 res = requests.post('http://httpbin.org/post')
      2 res = requests.put('http://httpbin.org/put')
      3 res = requests.delete('http://httpbin.org/delete')
```

- 分别用post()、put()、delete()等方法实现了POST、PUT、DELETE等请求。

CONTENTS

▶ **PART 1**

库安装和实例展示

▶ **PART 2**

GET请求

▶ **PART 3**

POST请求

▶ **PART 4**

响应



- HTTP中最常见的请求之一就是GET请求。
- 构建一个最简单的GET请求，请求的链接为<http://httpbin.org/get>，该网站会判断如果客户端发起的是GET请求的话，它返回相应的响应信息：

```
1 res = requests.get('http://httpbin.org/get')
2 print(res.text)

{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.30.0",
    "X-Amzn-Trace-Id": "Root=1-6458e05e-61e1786b77d9b42f08a2e398"
  },
  "origin": "183.136.238.108",
  "url": "http://httpbin.org/get"
}
```

```
1 res = requests.get('http://httpbin.org/post')|
2 print(res.text)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

- 可以发现，我们成功发起了GET请求，返回结果中包含请求头、URL、IP等信息。



- 对于GET请求，如果要附加额外的信息，一般怎样添加呢？
- 比如现在想添加两个参数，其中name是germey，age是22。要构造这个请求链接，可以利用params这个参数：

```
1 data = {  
2     'name': 'germey',  
3     'age': 22  
4 }  
5 res = requests.get('http://httpbin.org/get', params=data)  
6 print(res.text)  
  
{  
  "args": {  
    "age": "22",  
    "name": "germey"  
  },  
  "headers": {  
    "Accept": "*/*",  
    "Accept-Encoding": "gzip, deflate",  
    "Host": "httpbin.org",  
    "User-Agent": "python-requests/2.30.0",  
    "X-Amzn-Trace-Id": "Root=1-6458e0a5-3cef584e1b671a0e71464265"  
  },  
  "origin": "183.136.238.108",  
  "url": "http://httpbin.org/get?name=germey&age=22"  
}
```

- 通过运行结果可以判断，请求的链接自动被构造成了：
- <http://httpbin.org/get?age=22&name=germey>



- 网页的返回类型实际上是str类型，但是它很特殊，是JSON格式的。
- 如果想直接解析返回结果，得到一个字典格式，可以直接调用json()方法。

```
1 res = requests.get('http://httpbin.org/get')
2 print(type(res.text))
```

```
<class 'str'>
```

```
1 res.text
```

```
'{\n  "args": {}, \n  "headers": {\n    "Accept": "*/*", \n    "Accept-Encoding": "gzip, deflate", \n    "Host": "httpbin.org", \n    "User-Agent": "python-requests/2.30.0", \n    "X-Amzn-Trace-Id": "Root=1-6458f064-1dbalf46186221d95fab504d"\n  }, \n  "origin": "183.136.238.108", \n  "url": "http://httpbin.org/get"\n}\n'
```

```
1 print(type(res.json()))
2 print(res.json())
```

```
<class 'dict'>
```

```
{'args': {}, 'headers': {'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.30.0', 'X-Amzn-Trace-Id': 'Root=1-6458f064-1dbalf46186221d95fab504d'}, 'origin': '183.136.238.108', 'url': 'http://httpbin.org/get'}
```

- 可以发现，调用json()方法，就可以将返回结果是JSON格式的字符串转化为字典。



- 请求普通的网页，应该能获得相应的内容。
- 下面以“豆瓣读书”页面为例来看一下：

```
1 res = requests.get('https://book.douban.com/')  
2 print(res.status_code)
```

418

```
1 print(res.text)  
2 print(len(res.text))
```

0

- 可以发现，其网站禁止直接爬取。





- 增加headers信息，参数headers是一个字典，叫请求头，可在构造请求时通过headers参数直接构造。
- 添加请求头最常用的用法就是通过修改User-Agent（用户代理）来伪装浏览器。

```
1 headers = {  
2     'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36'  
3 }  
4 res = requests.get('https://book.douban.com/', headers=headers)  
5 print(res.status_code)  
6 print(type(res.text))
```

```
200  
<class 'str'>
```


```
1 res.text  
  
'\n\n<!DOCTYPE html>\n\n<html lang="zh-CN" class="ua-windows ua-webkit book-new-nav">\n\n  <head>\n\n    <meta charset="utf-8">\n\n    <meta name="google-site-verification" content="ok0wCgT20tBBgo9_zat2iAcimtN4Ftf5ccsh092Xeyw" />\n\n    <meta http-equiv="Pragma" content="no-cache">\n\n    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">\n\n    <meta http-equiv="Expires" content="Sun, 6 Mar 2005 01:00:00 GMT">\n\n    <meta http-equiv="mobile-agent" content="format=xhtml; url=https://m.douban.com/book/">\n\n    <meta name="keywords" content="豆瓣读书, 新书速递, 畅销书, 书评, 书单"/>\n\n    <meta name="description" content="记录你读过的、想读和正在读的书，顺便打分，添加标签及个人附注，写评论。根据你的口味，推荐适合的书给你。" />\n\n    <meta name="verify-v1" content="EYARGSAVd5U+06FeTmx08Mj28Fc/hM/9PqMfrlMo8YA=">\n\n    <meta property="wb:webmaster" content="7c86191e898cd20d">\n\n    <meta property="qc:admins" content="1520412177364752166375">\n\n\n  <title>\n\n    豆瓣读书\n\n</title>\n\n  <link rel="shortcut icon" href="https://img1.doubanio.com/favicon.ico" type="image/x-icon">\n\n  <script src="https://img1.doubanio.com/f/book/0495cb173e298c28593766009c7b0a953246c5b5/js/book/lib/jquery/jquery.js"></script>\n\n  <script>\n\n    var Douban = {}, Book = {}  
  
    var Do=function() {Do.actions.push([].slice.call(arguments));Do.readv=function() {Do.actions.push([].slice.call(arguments))}.Do.add=
```

- 此时便可以正常爬取网页了，r.text可返回响应的内容。



- User Agent中文名为用户代理，简称 UA，它是一个特殊字符串头，使得服务器能够识别客户使用的操作系统及版本、CPU 类型、浏览器及版本、浏览器渲染引擎、浏览器语言、浏览器插件等。获取自己浏览器的User Agent，地址栏中输入：about:version

 Chrome | chrome://version

Google Chrome: 114.0.5735.110 (正式版本) (64 位) (cohort: Stable) 
修订版本: 1c828682b85bbc70230a48f5e345489ec447373e-refs/branch-heads/5735_90@{#13}
操作系统: Windows 10 Version 21H2 (Build 19044.2006)
JavaScript: V8 11.4.183.19



- 图片、音频、视频这些文件本质上都是由二进制码组成的，由于有特定的保存格式和对应的解析方式，我们才可以看到这些形形色色的多媒体。所以，想要抓取它们，就要拿到它们的二进制码。下面以一个图片为例：





```
1 print(r.content)
```

[illegible]

- 这里打印了Response对象的两个属性，一个是text，另一个是content。
- 其中前面是r.text的结果，后面b' 之后是r.content的结果。
- 可以注意到，前者出现了乱码，后者结果前带有一个b，这代表是bytes类型的数据。
- 由于图片是二进制数据，所以前者在打印时转化为str类型，也就是图片直接转化为字符串，这理所当然会出现乱码。



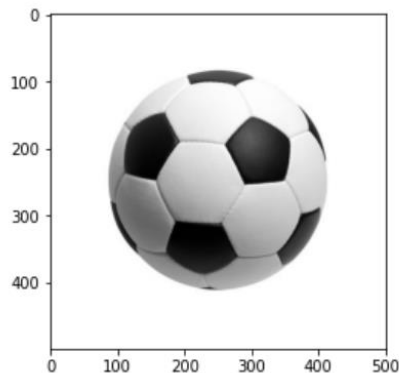
- 下面以代码形式爬取：

```
1 import requests
2
3 r = requests.get("https://img2.baidu.com/it/u=828490327,2767828912&fm=253&f=JPEG")
4 with open('soccer.jpg', 'wb') as f:
5     f.write(r.content)
```

- 将刚才提取到的图片保存下来，用open()方法，它的第一个参数是文件名称，第二个参数代表以二进制写的形式打开，可以向文件里写入二进制数据。
- 运行结束之后，可以在文件夹中找到名为soccer.jpg的图片。
- 右边为读取所保存的图片。

```
1 import matplotlib.pyplot as plt
2 import matplotlib.image as mpimg
3 import numpy as np
4 img = mpimg.imread('soccer.jpg')
5 plt.imshow(img)
```

<matplotlib.image.AxesImage at 0x7f99db572790>



CONTENTS

▶ **PART 1**

库安装和实例展示

▶ **PART 2**

GET请求

▶ **PART 3**

POST请求

▶ **PART 4**

响应



- 另外一种比较常见的请求方式是POST。
- 从获取的结果可以发现，我们成功获得了
 - 返回结果。
 - 其中form部分就是提交的数据，这就证明
 - POST请求成功发送了。

```
1 import requests
2
3 data = {
4     'name': 'abc',
5     'age': 24
6 }
7 r = requests.post('http://httpbin.org/post', data=data)
8 print(r.text)

{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "age": "24",
    "name": "abc"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "15",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.30.0",
    "X-Amzn-Trace-Id": "Root=1-645f917a-7d2d65bf370fbc6c1326f380"
  },
  "json": null,
  "origin": "183.136.238.108",
  "url": "http://httpbin.org/post"
}
```


CONTENTS

▶ **PART 1**

库安装和实例展示

▶ **PART 2**

GET请求

▶ **PART 3**

POST请求

▶ **PART 4**

响应



- 发送请求后，得到的自然就是响应。在上面的实例中，我们使用text和content获取了响应的内容。此外，还有很多属性和方法可以用来获取其他信息，比如状态码、响应头、Cookies等。

```
1 import requests
2
3
4 r = requests.get('https://www.baidu.com')
5 print(r.status_code, type(r.status_code))
```

```
200 <class 'int'>
```

```
1 print(type(r.headers))
2 print(r.headers)
```

```
<class 'requests.structures.CaseInsensitiveDict'>
{'Cache-Control': 'private, no-cache, no-store, proxy-revalidate, no-transform', 'Connection': 'keep-alive', 'Content-Encoding': 'gzip',
'Content-Type': 'text/html', 'Date': 'Sat, 13 May 2023 13:34:15 GMT', 'Last-Modified': 'Mon, 23 Jan 2017 13:23:55 GMT', 'Pragma': 'no-cache',
'Server': 'bfe/1.0.8.18', 'Set-Cookie': 'BDORZ=27315; max-age=86400; domain=.baidu.com; path=/, 'Transfer-Encoding': 'chunked'}
```

```
1 print(type(r.cookies))
2 print(r.cookies)
```

```
<class 'requests.cookies.RequestsCookieJar'>
<RequestsCookieJar[<Cookie BDORZ=27315 for .baidu.com/>]>
```

```
1 print(type(r.url))
2 print(r.url)
```

```
<class 'str'>
https://www.baidu.com/
```

```
1 print(type(r.history))
2 print(r.history)
```

```
<class 'list'>
[]
```



- 状态码常用来判断请求是否成功，而requests提供了一个内置的状态码查询对象requests.codes。

```
1 import requests
2
3
4 r = requests.get('https://www.baidu.com')
5 if r.status_code == requests.codes.ok:
6     print('请求成功!')
7 else:
8     print('请求失败!')
```

请求成功!

- 通过比较返回码和内置的成功的返回码，来保证请求得到了正常响应，输出成功请求的消息，否则程序终止，这里我们用requests.codes.ok得到的是成功的状态码200。

Thank you!

