




requests库的使用（一篇就够了）_requests使用-CSDN博客


 blog.csdn.net/m0_43404934/article/details/122331463

requests库的使用（一篇就够了）

原创

上善若水。。  已于 2022-04-30 15:41:16 修改

 阅读量9.6w  收藏 2.4k

 点赞数 300

分类专栏：爬虫 文章标签：python http
于 2022-01-05 20:38:30 首次发布

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](#) 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/m0_43404934/article/details/122331463

版权



爬虫 专栏收录该内容

8 篇文章 54 订阅

订阅专栏

urllib库使用繁琐，比如处理网页验证和Cookies时，需要编写Opener和Handler来处理。为了更加方便的实现这些操作，就有了更为强大的requests库。

request库的安装

requests属于第三方库，Python不内置，因此需要我们手动安装。

1、相关链接

- GitHub：<https://github.com/psf/requests>
- PyPI：<https://pypi.org/project/requests/>
- 官方文档：<https://docs.python-requests.org/en/latest/>

- 中文文档：https://docs.python-requests.org/zh_CN/latest/user/quickstart.html

2、通过pip安装

无论是Windows、Linux还是Mac，都可以通过pip这个包管理工具来安装requests。在命令行界面运行如下命令，即可完成requests库的安装：

```
pip3 install requests
```

除了通过pip安装，还可以通过wheel或源码安装，这里不进行叙述。

3、验证安装

在命令行可通过导入import库来测试requests是否安装成功。

```
C:\Users\18758>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>>
```

导入库成功，说明requests安装成功。

基本用法

下面案例使用requests库中的get()方法发送了一个get请求。

```
#导入requests库
import requests
#发送一个get请求并得到响应
r = requests.get('https://www.baidu.com')
#查看响应对象的类型
print(type(r))
#查看响应状态码
print(r.status_code)
#查看响应内容的类型
print(type(r.text))
#查看响应的内容
print(r.text)
#查看cookies
print(r.cookies)
```

这里调用了get()方法实现urlopen()相同的操作，结果返回一个响应对象，然后分别输出响应对象类型、状态码、响应体内容的类型、响应体的内容、Cookies。通过运行结果可以得知：响应对象的类型是requests.models.Response，响应体内容的类型是str，Cookies 的类型是RequestCookieJar。如果要发送其他类型的请求直接调用其对应的方法即可：

```
r = requests.post('https://www.baidu.com')
r = requests.put('https://www.baidu.com')
r = requests.delete('https://www.baidu.com')
r = requests.head('https://www.baidu.com')
r = requests.options('https://www.baidu.com')
```

GET请求

构建一个GET请求，请求http://httpbin.org/get（该网站会判断如果客户端发起的是GET请求的话，它返回相应的信息）

```
import requests
r = requests.get('http://httpbin.org/get')
print(r.text)
```

```
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.26.0",
    "X-Amzn-Trace-Id": "Root=1-61d2e8e9-17518b2601f0c2977b4f05b7"
  },
  "origin": "182.105.80.242",
  "url": "http://httpbin.org/get"
}
```

CSDN @上善若水。。

1) 如果要添加请求参数，比如添加两个请求参数，其中name值是germey，age值是20。虽然可以写成如下形式：

```
r = requests.get('http://httpbin.org/get?name=germey&age=20')
```

但较好的写法是下面这种写法：

```
import requests
data = {
    'name': 'germey',
    'age': 22
}
r = requests.get('http://httpbin.org/get', params=data)
print(r.text)
```

```
{
  "args": {
    "age": "22",
    "name": "germey"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.26.0",
    "X-Amzn-Trace-Id": "Root=1-61d2ea37-505e5b5764eb2b010848e6b5"
  },
  "origin": "182.105.80.242",
  "url": "http://httpbin.org/get?name=germey&age=22"
}
```

CSDN @上善若水。。

通过运行结果可以看出，请求的URL最终被构造成了“http://httpbin.org/get?name=germey&age=20”。

2) 网页的返回内容的类型是str类型的，如果它符合JSON格式，则可以使用json()方法将其转换为字典类型，以方便解析。

```
import requests
r = requests.get('http://httpbin.org/get')
#str类型
print(type(r.text))
#返回响应内容的字典形式
print(r.json())
#dict类型
print(type(r.json()))
```

但需要注意，如果返回的内容不是JSON格式，调用json()方法便会出现错误，抛出json.decoder.JSONDecodeError异常。

POST请求

1) 发送POST请求。

```
import requests
r = requests.post('http://httpbin.org/post')
print(r.text)
```

2) 发送带有请求参数的POST请求。

```
import requests
data = {
    "name": "germey",
    "age": "22"
}
r = requests.post('http://httpbin.org/post', data=data)
print(r.text)
```

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "age": "22",
    "name": "germey"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "18",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.26.0",
    "X-Amzn-Trace-Id": "Root=1-61d2f73b-4235608929d933a60f1c6e33"
  },
  "json": null,
  "origin": "182.105.80.242",
  "url": "http://httpbin.org/post"
}
```

CSDN @上善若水。。

在POST请求方法中，form部分就是请求参数。

设置请求头

```
import requests
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0',
    'my-test': 'Hello'
}
r = requests.get('http://httpbin.org/get', headers=headers)
print(r.text)
```

```
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "My-Test": "Hello",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0",
    "X-Amzn-Trace-Id": "Root=1-61d2f0eb-5b49b4be08b4de9274c5128e"
  },
  "origin": "182.105.80.242",
  "url": "http://httpbin.org/get"
}
```

CSDN @上善若水。。

响应

1) 发送请求后，返回一个响应，它具有很多属性，通过它的属性来获取状态码、响应头、Cookies、响应内容等。如下：

```
import requests
r = requests.get('https://www.baidu.com/')
#响应内容 (str类型)
print(type(r.text), r.text)
#响应内容 (bytes类型)
print(type(r.content), r.content)
#状态码
print(type(r.status_code), r.status_code)
#响应头
print(type(r.headers), r.headers)
#Cookies
print(type(r.cookies), r.cookies)
#URL
print(type(r.url), r.url)
#请求历史
print(type(r.history), r.history)
```

2) 状态码常用来判断请求是否成功，除了可以使用HTTP提供的状态码，requests库中也提供了一个内置的状态码查询对象，叫做 requests.codes，实际上两者都是等价的。示例如下：

```
import requests
r = requests.get('https://www.baidu.com/')
if not r.status_code==requests.codes.ok:
    print('Request Fail')
else:
    print('Request Successfully')
```

requests.codes对象拥有的状态码如下：

#信息性状态码

```
100 : ('continue',),
101 : ('switching_protocols',),
102 : ('processing',),
103 : ('checkpoint',),
122 : ('uri_too_long', 'request_uri_too_long'),
```

#成功状态码

```
200 : ('ok', 'okay', 'all_ok', 'all_okay', 'all_good', '\\o/', '✓'),
201 : ('created',),
202 : ('accepted',),
203 : ('non_authoritative_info', 'non_authoritative_information'),
204 : ('no_content',),
205 : ('reset_content', 'reset'),
206 : ('partial_content', 'partial'),
207 : ('multi_status', 'multiple_status', 'multi_stati', 'multiple_stati'),
208 : ('already_reported',),
226 : ('im_used',),
```

#重定向状态码

```
300 : ('multiple_choices',),
301 : ('moved_permanently', 'moved', '\\o-'),
302 : ('found',),
303 : ('see_other', 'other'),
304 : ('not_modified',),
305 : ('user_proxy',),
306 : ('switch_proxy',),
307 : ('temporary_redirect', 'temporary_moved', 'temporary'),
308 : ('permanent_redirect',),
```

#客户端请求错误

```
400 : ('bad_request', 'bad'),
401 : ('unauthorized',),
402 : ('payment_required', 'payment'),
403 : ('forbiddent',),
404 : ('not_found', '-o-'),
405 : ('method_not_allowed', 'not_allowed'),
406 : ('not_acceptable',),
407 : ('proxy_authentication_required', 'proxy_auth', 'proxy_authentication'),
408 : ('request_timeout', 'timeout'),
409 : ('conflict',),
410 : ('gone',),
411 : ('length_required',),
412 : ('precondition_failed', 'precondition'),
413 : ('request_entity_too_large',),
414 : ('request_uri_too_large',),
415 : ('unsupported_media_type', 'unsupported_media', 'media_type'),
416 : ('request_range_not_satisfiable', 'requested_range', 'range_not_satisfiable'),
417 : ('expectation_failed',),
418 : ('im_a_teapot', 'teapot', 'i_am_a_teapot'),
421 : ('misdirected_request',),
422 : ('unprocessable_entity', 'unprocessable'),
```

```

423: ('locked'),
424: ('failed_dependency', 'dependency'),
425: ('unordered_collection', 'unordered'),
426: ('upgrade_required', 'upgrade'),
428: ('precondition_required', 'precondition'),
429: ('too_many_requests', 'too_many'),
431: ('header_fields_too_large', 'fields_too_large'),
444: ('no_response', 'none'),
449: ('retry_with', 'retry'),
450: ('blocked_by_windows_parental_controls', 'parental_controls'),
451: ('unavailable_for_legal_reasons', 'legal_reasons'),
499: ('client_closed_request',),

#服务端错误状态码
500: ('internal_server_error', 'server_error', '/o\\', 'x')
501: ('not_implemented',),
502: ('bad_gateway',),
503: ('service_unavailable', 'unavailable'),
504: ('gateway_timeout',),
505: ('http_version_not_supported', 'http_version'),
506: ('variant_also_negotiates',),
507: ('insufficient_storage',),
509: ('bandwidth_limit_exceeded', 'bandwith'),
510: ('not_extended',),
511: ('network_authentication_required', 'network_auth', 'network_authentication')

```

爬取二进制数据

图片、音频、视频这些文件本质上都是由二进制码组成的，所以想要爬取它们，就要拿到它们的二进制码。以爬取百度的站点图标（选项卡上的小图标）为例：

```

import requests
#向资源URL发送一个GET请求
r = requests.get('https://www.baidu.com/favicon.ico')
with open('favicon.ico', 'wb') as f:
    f.write(r.content)

```

使用open()方法，它的第一个参数是要保存文件名（可带路径），第二个参数表示以二进制的形式写入数据。运行结束之后，可以在当前目录下发现保存的名为favicon.ico的图标。同样的，音频和视频也可以用这种方法获取。

文件上传

requests可以模拟提交一些数据。假如某网站需要上传文件，我们也可以实现。


```
import requests
#以二进制方式读取当前目录下的favicon.ico文件，并将其赋给file
files = {'file':open('favicon.ico','rb')}
#进行上传
r = requests.post('http://httpbin.org/post',files=files)
print(r.text)
```

处理Cookies

使用urllib处理Cookies比较复杂，而使用requests处理Cookies非常简单。

1) 获取Cookies。

```
import requests
r = requests.get('https://www.baidu.com')
#打印Cookies对象
print(r.cookies)
#遍历Cookies
for key,value in r.cookies.items():
    print(key+'='+value)
```

通过响应对象调用cookies属性即可获得Cookies，它是一个RequestCookiesJar类型的对象，然后用items()方法将其转换为元组组成的列表，遍历输出每一个Cookies的名称和值。

2) 使用Cookies来维持登录状态。以知乎为例，首先登录知乎，进入一个登录之后才可以访问的页面，在浏览器开发者工具中将Headers 中的Cookies复制下来（有时候这样直接复制的Cookies中包含省略号，会导致程序出错，此时可以在Console项下输document.cookie 即可得到完整的Cookie），在程序中使用它，将其设置到Headers里面，然后发送请求。

```

import requests
headers = {
    'Cookie': '_zap=616ef976-1fdb-4b8c-a3cb-9327ff629ff1;
_xsrf=0CCNkbCLtTAIz5BfwhMHBHJWW791ZkK6;
d_c0=\"AKBQTnFIRhSPTpoYIf6mUxSic2UjzSp4BYM=|1641093994\";
__snaker__id=mMv5F3gmBHC9jJg;
gdxidpyhxdE=E%2BNK7sMat0%2F3aZ5Ke%2FSRfBRK7B1QBmCta0wrqJm%2F10NP3VPitkrXCcMiAX3%2FIsS
xUwudQPyuDG0%2B1HGpVnqGq09bX1%2B58o7wmf%2FZewh8xSPg%2FH3T2HoWsrs7ZhsSGND0C0la%2BXkLII
G5XXV85PxV5g99d%5CMph%2BbkX1JQBGhDnL3N0zRf%3A1641094897088; _9755xjdesxxd_=32;
YD00517437729195%3AWM_NI=rMeMx2d5Yt3mg0yHPvuPGTjPnGtjL%2Bn%2FPSBnVn%2FHFAVZnIEABUIPIT
BdsHmMX1iCHfKau04qhW%2Bi5bTy12Cg91vrXMPg0HtnaAylN8zk7MFpoTr%2FTeKVo3%2FKSSM6T5cNSGE%3
D;
YD00517437729195%3AWM_NIKE=9ca17ae2e6ffcd170e2e6ee8bea40f8e7a4b2cf69b3b48fb7c54b979b
8fbaf17e93909b91fb338ebaaeadec2af0fea7c3b92ab293abaefb3aa8eb9795b267a5f0b7a9d37eb7908
9b5e95cae99bc8bcf21aef1a0b4c16696b2e1a9c54b9686a2aac84b828b87b1cc6082bcbda9f0479cefa7
a4cb6e89bfbbb0b77bac89e58ab86a98a7ffd3c26dfbefba93fb4794b981a9f766a39fb78dcd34bab5f9a
ec57cad8cbcd0d76f898aa1d4ae41918d83d7d73fa1929da8c837e2a3;
YD00517437729195%3AWM_TID=Kji43bLtZbRAAAVABFMu4upmK4C%2BEGQH;
KLBRSID=9d75f80756f65c61b0a50d80b4ca9b13|1641268679|1641267986; tst=r;
NOT_UNREGISTER_WAITING=1; SESSIONID=lbWS7Y8pmp5qM1DERkXJCahgQwwyl79eT8XAOC6qC7A;
JOID=V1wXAUwzD9BQH284PTQMxsZMqrkrXmuHBio3Bk1cfuMhV1x9fiHKBjYcaD44XxiWm2kKD5TjJvk-
7iTeM3d6aYA=; osd=VVoQAK0xCddTHm0-
OjcnXMBLqbgpWGYEBygxAU5df0UmVF1_eCbJBzQabz05XR6RmGgICZPgJ_s46SffMXF9aoE=;
Hm_lvt_98beee57fd2ef70ccdd5ca52b9740c49=1640500881,1641093994,1641267987;
Hm_lpv_98beee57fd2ef70ccdd5ca52b9740c49=1641268678',
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101
Firefox/95.0',
    'Host': 'www.zhihu.com'
}
r = requests.get('https://www.zhihu.com/people/xing-fu-shi-fen-dou-chu-lai-de-65-
18', headers=headers)
print(r.text)

```

运行之后，结果中包含了登录后的内容，说明获取登录状态成功。

3) 也可以通过cookies参数来设置，不过这样就需要构造RequestCookieJar对象，而且需要分割以下cookies，相对繁琐，但效果是一样。

```

import requests

cookies = '_zap=616ef976-1fdb-4b8c-a3cb-9327ff629ff1;
_xsrf=0CCNkbCLtTAIz5BfwhMHBHJWW791ZkK6;
d_c0=\"AKBQTnFIRhSPTpoYIf6mUXSic2UjzSp4BYM=|1641093994\";
__snaker__id=mMv5F3gmBHIC9jJg;
gdxidpyhxdE=E%2BNK7sMat0%2F3aZ5Ke%2FSRfBRK7B1QBmCta0wrqJm%2F10NP3VPitkrXCcMiAX3%2FIsS
xUwudQPyuDGO%2B1HGpVnqGq09bX1%2B58o7wmf%2FZewh8xSPg%2FH3T2HoWsrs7ZhsSGND0C0la%2BXkLII
G5XXV85PxV5g99d%5CMph%2BbkX1JQBGhDnL3N0zRf%3A1641094897088; _9755xjdesxxd_=32;
YD00517437729195%3AWM_NI=rMeMx2d5Yt3mg0yHPvuPGTjPnGtjL%2Bn%2FPSBnVn%2FHFAVZnIEABUIPIT
BdsHmMX1iCHfKau04qhW%2Bi5bTy12Cg91vrXMPg0HtnaAylN8zk7MFpoTr%2FTEkVo3%2FKSSM6T5cNSGE%3
D;
YD00517437729195%3AWM_NIKE=9ca17ae2e6ffcdad170e2e6ee8bea40f8e7a4b2cf69b3b48fb7c54b979b
8fbaf17e93909b91fb338ebaaeadec2af0fea7c3b92ab293abaefb3aa8eb9795b267a5f0b7a9d37eb7908
9b5e95cae99bc8bcf21aef1a0b4c16696b2e1a9c54b9686a2aac84b828b87b1cc6082bcbda9f0479cefa7
a4cb6e89bfbbb0b77bac89e58ab86a98a7ffd3c26dfbefba93fb4794b981a9f766a39fb78dcd34bab5f9a
ec57cad8cbcd0d76f898aa1d4ae41918d83d7d73fa1929da8c837e2a3;
YD00517437729195%3AWM_TID=Kji43bLtZbRAAAVABFMu4upmK4C%2BEGQH;
KLBRSID=9d75f80756f65c61b0a50d80b4ca9b13|1641268679|1641267986; tst=r;
NOT_UNREGISTER_WAITING=1; SESSIONID=lbWSY8pmp5qM1DERkXJCahgQwwyl79eT8XA0C6qC7A;
JOID=V1wXAUwzD9BQH284PTQMxsZMqrkrXmuHBio3Bk1cfuMhV1x9fiHKBjYcaD44XxiWm2kKD5TjJvk-
7iTeM3d6aYA=; osd=VVoQAk0xCddTHm0-
OjcnXMBLqbgpWGYEBygxAU5df0UmVF1_eCbJBzQabz05XR6RmGgICZPgJ_s46SffMXF9aoE=;
Hm_lvt_98beee57fd2ef70ccdd5ca52b9740c49=1640500881,1641093994,1641267987;
Hm_lpvt_98beee57fd2ef70ccdd5ca52b9740c49=1641268678'
jar = requests.cookies.RequestsCookieJar()
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101
Firefox/95.0',
    'Host': 'www.zhihu.com'
}
for cookie in cookies.split(';'):
    key,value = cookie.split('=',1)
    jar.set(key,value)
r = requests.get('https://www.zhihu.com/people/xing-fu-shi-fen-dou-chu-lai-de-65-
18',headers=headers)
print(r.text)

```

会话维持

通过调用get()或post()等方法可以做到模拟网页的请求，但是这实际上是相当于不同的会话，也就是说相当于你用了两个浏览器打开不同的页面。如果第一个请求利用post()方法登录了网站，第二次想获取登录成功后的自己的个人信息，又使用了一次get()方法取请求个人信息，实际上，这相当于打开了两个浏览器，所以是不能成功的获取到个人信息的。为此，需要会话维持，你可以在两次请求时设置一样的Cookies，但这样很繁琐，而通过Session类可以很轻松地维持一个会话。

```
import requests
s = requests.Session()
s.get('http://httpbin.org/cookies/set/number/123456789')
r = s.get('http://httpbin.org/cookies')
print(r.text)
```

首先通过requests打开一个会话，然后通过该会话发送了一个get请求，该请求用于向cookies中设置参数number，参数值为123456789；接着又使用该发起了一个get请求，用于获取Cookies，然后打印获取的内容。

```
{
  "cookies": {
    "number": "123456789"
  }
}
```

成功获取。

SSL证书验证

requests还提供了证书验证功能，当发送HTTP请求的时候，它会检查SSL证书，我们可以使用verify参数控制是否检查SSL证书。

1) 请求一个HTTPS网站时，如果该网站的证书没有被CA机构信任，程序将会出错，提示SSL证书验证错误。对此，只需要将verify参数 设置为False即可。如下：

```
import requests
response = requests.get('https://www.12306.cn', verify=False)
print(response.status_code)
```

也可以指定一个本地证书用作客户端证书，它可以是单个文件（包含密钥和证书）或一个包含两个文件路径的元组。

```
import requests
#本地需要有crt和key文件（key必须是解密状态，加密状态的key是不支持的），并指定它们的路径，
response = requests.get('https://www.12306.cn', cert('/path/server.crt', '/path/key'))
print(response.status_code)
```

2) 在请求SSL证书不被CA机构认可的HTTPS网站时，虽然设置了verify参数为False，但程序运行可能会产生警告，警告中建议我们给它 指定证书，可以通过设置忽略警告的方式来屏蔽这个警告：

```
import requests
from requests.packages import urllib3
urllib3.disable_warnings()
response = requests.get('https://www.12306.cn', verify=False)
print(response.status_code)
```

或者通过捕获警告到日志的方式忽略警告：

```
import logging
import requests
logging.captureWarnings(True)
response = requests.get('https://www.12306.cn', verify=False)
print(response.status_code)
```

代理设置

对于某些网站，在测试的时候请求几次，能正常获取内容。但是一旦开始大规模、频繁地爬取，网站可能会弹出验证码，或者跳转到登录验证页面，更有甚者可能会直接封禁客户端的IP，导致一定时间内无法访问。为了防止这种情况，我们需要使用代理来解决这个问题，这就需要用到proxies参数。

1) 设置代理

```
import requests

proxies = {
    #该代理服务器在免费代理网站上得到的，这样的网站有很多
    'http': 'http://161.35.4.201:80',
    'https': 'https://161.35.4.201:80'
}
try:
    response = requests.get('http://httpbin.org/get', proxies=proxies)
    print(response.text)
except requests.exceptions.ConnectionError as e:
    print('Error', e.args)
```

```
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.26.0",
    "X-Amzn-Trace-Id": "Root=1-61d3d8aa-50efe9fe7afcdda6172c46fe"
  },
  "origin": "161.35.4.201",
  "url": "http://httpbin.org/get"
}
```

CSDN @上善若水。。

可以发现，我们使用的是代理服务器进行访问的。

2) 如果代理需要使用HTTP Basic Auth，可以使用类似http://user:password@host:port这样的语法来设置代理。

```
import requests
proxies = {
    "http": "http://user:password@161.35.4.201:80"
}
r = requests.get("https://www.taobao.com", proxies=proxies)
print(r.text)
```

3) 除了基本的HTTP代理外，requests还支持SOCKS协议的代理。首先需要安装socks这个库：

```
pip3 install 'requests[socks]'

import requests
proxies = {
    'http': 'socks5://user:password@host:port',
    'https': 'socks5://user:password@host:port'
}
request.get('https://www.taobao.com', proxies=proxies)
```

然后就可以使用SOCKS协议代理了

```
import requests
proxies = {
    'http': 'socks5://user:password@host:port',
    'https': 'socks5://user:password@host:port'
}
requests.get('https://www.taobao.com', proxies=proxies)
```

超时设置

在本机网络状况不好或者服务器网络响应太慢甚至无响应时，我们可能会等待特别久的时间才可能收到响应，甚至到最后收不到响应而报错。为了应对这种情况，应设置一个超时时间，这个时间是计算机发出请求到服务器返回响应的时间，如果请求超过了这个超时时间还没有得到响应，就抛出错误。这就需要使用timeout参数实现，单位为秒。

1) 指定请求总的超时时间

```
import requests
#向淘宝发出请求，如果1秒内没有得到响应，则抛出错误
r = requests.get('https://www.taobao.com', timeout=1)
print(r.status_code)
```

2) 分别指定超时时间。实际上，请求分为两个阶段：连接（connect）和读取（read）。如果给timeout参数指定一个整数值，则超时时间是这两个阶段的总和；如果要分别指定，就可以传入一个元组，连接超时时间和读取超时时间：

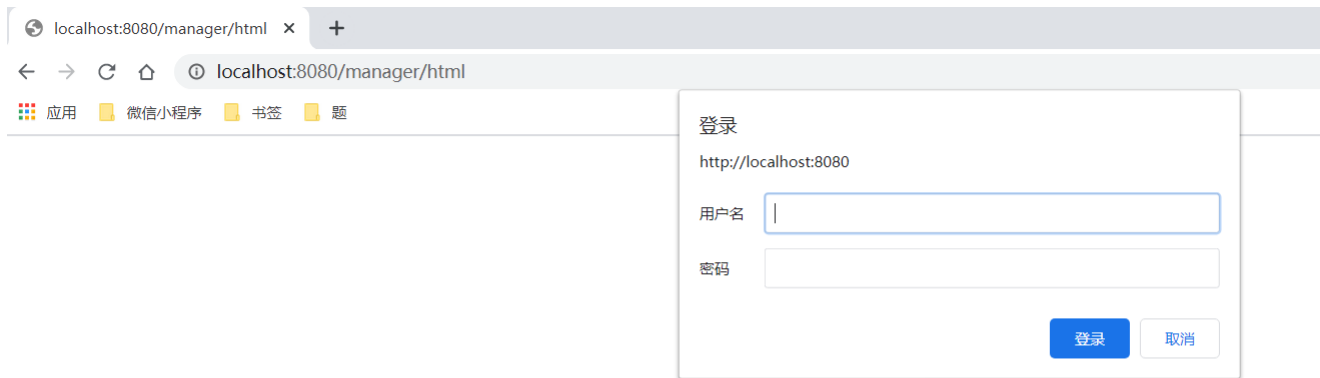
```
import requests
#向淘宝发出请求，如果连接阶段5秒内没有得到响应或读取阶段30秒内没有得到响应，则抛出错误
r = requests.get('https://www.taobao.com', timeout=(5, 30))
print(r.status_code)
```

3) 如果想永久等待，可以直接timeout设置为None，或者不设置timeout参数，因为它的默认值就是None。

```
import requests
#向淘宝发出请求，如果连接阶段5秒内没有得到响应或读取阶段30秒内没有得到响应，则抛出错误
r = requests.get('https://www.taobao.com', timeout=None)
print(r.status_code)
```

身份验证

访问某网站时，可能会遇到如下的验证页面：



CSDN @上善若水。。

1) 此时可以使用requests自带的身份验证功能，通过HTTPBasicAuth类实现。

```
import requests
from requests.auth import HTTPBasicAuth
r =
requests.get('http://localhost:8080/manager/html',auth=HTTPBasicAuth('admin','123456'
))
print(r.status_code)
```

如果用户名和密码正确的话，返回200状态码；如果不正确，则返回401状态码。也可以不使用HTTPBasicAuth类，而是直接传入一个元组，它会默认使用HTTPBasicAuth这个类来验证。

```
import requests
from requests.auth import HTTPBasicAuth
r = requests.get('http://localhost:8080/manager/html',auth=('admin','123456'))
print(r.status_code)
```

2) requests还提供了其他验证方式，如OAuth验证，不过需要安装oauth包，安装命令如下：

```
pip3 install requests_oauthlib
```

使用OAuth验证的方法如下：

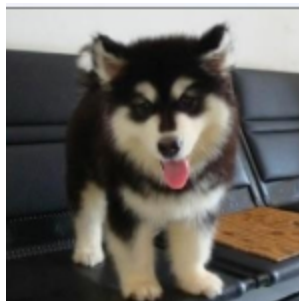

```
import requests
from requests_oauthlib import OAuth1
url = 'https://api.twitter.com/1.1/account/verify_credentials.json'
auth =
OAuth1("YOUR_APP_KEY", "YOUR_APP_SECRET", "USER_OAUTH_TOKEN", "USER_OAUTH_TOKEN_SECRET")
requests.get(url, auth=auth)
```

Prepared Request

在学习urllib库时，发送请求如果需要设置请求头，需要通过一个Request对象来表示。在requests库中，存在一个与之类似的类，称为Prepared Request。

```
from requests import Request, Session
url = 'http://httpbin.org/post'
data = {
    'name': 'germey'
}
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0)
Gecko/20100101 Firefox/95.0'
}
s = Session()
req = Request('POST', url, data=data, headers=headers)
prepped = s.prepare_request(req)
r = s.send(prepped)
print(r.text)
```

这里引入了Request，然后用url、data和headers参数构造了一个Request对象，这时需要再调用Session的prepare_request()方法将其转换为一个Prepared Request对象，然后调用send()方法发送。这样做的好处是：可以利用Request将请求当作独立的对象来看待，这样在进行队列调度时会非常方便，后面会用它来构造一个Request队列。



热门推荐

既然选择远方，便只顾风雨兼程！

02-28  5万+

文章目录[requests库一、基本概念1、简介2、获取3、http 协议3.1 URL3.2 常用 http 请求方法二、使用方法1、基本语法requests 库中的方法2、具体使用方法2.1 get2.1.1 基本语法2.1.2 常用参数2.2 post2.2.1 基本语法2.2.2 常用参数2.3 response2.4 **head**2.4.1 基本语法2.5 put](#)[requests库一、基本概念 1、简介 requests 模块是 python 基于 urllib，采用 A](#)

[爬虫 - requests](#)

[Waller_的博客](#)

11-25  1119

[介绍 使用requests可以模拟浏览器的请求，比起python内置的urllib模块，requests模块的api更加便捷（本质就是封装了urllib3） 注意：requests库发送请求将网页内容下载下来以后，并不会执行js代码，这需要我们自己分析目标站点然后发起新的request请求 安装 >: pip3 install requests 使用 各种请求方式：常用的就是...](#)

[request库详解](#)

[qq_41845823的博客](#)

08-08  8829

[Request库 request库中文文档：https://docs.python-](#)

[requests.org/zh_CN/latest/user/quickstart.html 视频链接：](#)

[https://www.bilibili.com/video/BV1Us41177P1?p=2 实例引入 import requests response = requests.get\("https://www.baidu.com/"\) .print\(type\(response\)\).print\(type\(response](#)