## requests库

requests库是用python语言基于urllib编写的,采用Apache2 Licensed开源协议的HTTP库,可以用来**发送HTTP请求获取响应。（我们写的代码使用requests库可以发送Http请求）**

**请求谁？软件，项目，应用，网站**

**搭建过软件系统（考试系统、ERP），发送请求，获取响应，分析响应结果。**

**GET请求：pull拉取 --- 获取新数据**

**POST请求：push推送 --- 注册，登录，添加新数据**

### 1.Httpbin服务（接口测试网站,本地搭建）

Httpbin是一个使用Python + Flask编写的HTTP请求和响应服务(网站，项目，软件),该服务主要用于测试 HTTP库,你可以向他发送请求,然后他会按照指定的规则将你的请求返回(查看你所发送请求中的信息)。

Httpbin支持HTTP/HTTPS请求,支持所有的HTTP请求,能测试HTTP请求和响应的各种信息,比如：cookie、ip、headers 和登录验证等。能模拟302跳转乃至302跳转的次数,还可以返回一个HTML文件或一个XML文件或一个图片文件(还支持指定返回图片的格式)。

官方网站：http://httpbin.org/

开源地址：https://github.com/postmanlabs/httpbin

1.Httpbin服务本地部署

Httpbin是部署在国外的,考虑到有时无法访问的情况和测试时要不断访问 httpbin 网站,请求过多会被拉到黑名单,可以自己在本地搭建部署Httpbin。

Docker部署(推荐)

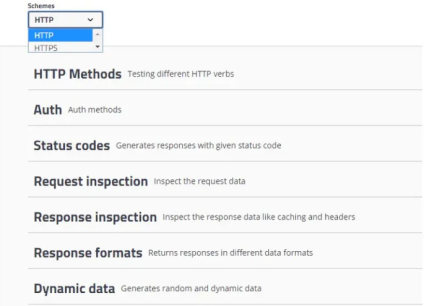
**docker pull kennethreitz/httpbin**

**docker run -p 80:80 kennethreitz/httpbin**

**本地浏览器访问：http://192.168.55.130**

2.Httpbin常用调试接口

访问http://httpbin.org/#/或者本地Httpbin服务http://服务器IP/就会出现如下界面。页面中每个选项都可以点开,里边有对应的接口说明。



常用接口列表：

|  |  |
| --- | --- |
| **Endpoint** | **Descirption** |
| /ip | 返回请求者的IP地址。 |
| /user-agent | 返回传入请求的User-Agent参数。 |
| /headers | 返回传入请求的请求头参数。 |
| /get | GET请求的提交参数。 |
| /post | POST请求的提交参数。 |
| /patch | PATCH请求的提交参数。 |
| /put | PUT请求的提交参数。 |
| /delete | DELETE请求的提交参数。 |

1)返回headers信息

调用接口：**https://httpbin.org/headers**

{

"headers": {

"Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8",

"Accept-Encoding": "gzip, deflate, br",

"Accept-Language": "zh-CN,zh;q=0.9",

"Host": "httpbin.org",

"Upgrade-Insecure-Requests": "1",

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.108 Safari/537.36",

"X-Amzn-Trace-Id": "Root=1-5ff1a4f7-69a84fca0693b09e28d214b0"

}

}

2)返回你使用的访问此链接的IP地址

调用接口：**https://httpbin.org/ip**

{

"origin": "106.35.9.12"

}

3)返回GET请求的详细信息

调用接口：**https://httpbin.org/get?department=1**

{

"args": {

"department": "1"

},

"headers": {

"Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8",

"Accept-Encoding": "gzip, deflate, br",

"Accept-Language": "zh-CN,zh;q=0.9",

"Host": "httpbin.org",

"Upgrade-Insecure-Requests": "1",

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.25 Safari/537.36 Core/1.70.3775.400 QQBrowser/10.6.4209.400",

"X-Amzn-Trace-Id": "Root=1-5ff1a59b-4e514a40023721d276ae07eb"

},

"origin": "106.35.9.12",

"url": "https://httpbin.org/get?department=1"

}

### 2.安装requests库

1.管理员身份打开命令提示符

或者

Win+X,选择终端管理员

2.在命令行界面中输入命令

C:\Users\Administrator>**pip install requests**

3.命令行输入python,在python shell环境下输入import requests没有报错即安装成功。

C:\Users\Administrator>**python**

Type "help", "copyright", "credits" or "license" for more information.

>>> **import requests**

>>>

或者

pip list 命令查看确认。

C:\Users\Administrator>**pip list**

### 3.发送Get请求

**requests.get(url=url, headers=headers, params=params)**

参数说明：

url：请求url地址

headers：请求头

params：查询字符串

#### 2.3.1.不带参数的Get请求

打开PyCharm,新建一个apitest项目,项目下新建一个Python文件,输入以下内容

#1.导入requests包

import requests

#2.组装请求

url = "http://httpbin.org**/get**" # 这里只有url,字符串格式

#3.发送请求,获取响应

response = **requests.get(url)** # response即返回的响应对象

#4.解析响应

print(response.text) # 输出响应的文本

运行结果：

{

"args": {},

"headers": {

"Accept": "\*/\*",

"Accept-Encoding": "gzip, deflate, br",

"Host": "httpbin.org",

"User-Agent": "python-requests/2.31.0",

"X-Amzn-Trace-Id": "Root=1-661d5028-0a0ff8c84c52e65775ce3fee"

},

"origin": "171.80.82.188",

"url": "http://httpbin.org/get"

}

#### 2.3.2.带参数的GET请求

import requests

#将请求参数写在字典里，通过params传入，params接受字典或序列

**data = {"name": "奥特曼","age": 24}**

# 发出一个get请求，获得响应

response = requests.get("http://httpbin.org/get", **params=data**)

print(response.url) # 打印url

print(response.text) # 打印响应内容

运行结果：

http://httpbin.org/get?name=%E5%A5%A5%E7%89%B9%E6%9B%BC&age=24

{

"args": {

"age": "24",

"name": "\u5965\u7279\u66fc"

},

"headers": {

"Accept": "\*/\*",

"Accept-Encoding": "gzip, deflate, br",

"Host": "httpbin.org",

"User-Agent": "python-requests/2.31.0",

"X-Amzn-Trace-Id": "Root=1-661d50cd-284b6d9f67e31e6c0822b7e2"

},

"origin": "171.80.82.188",

"url": "http://httpbin.org/get?name=\u5965\u7279\u66fc&age=24"

}

#### 2.3.4.响应json

requests对于特定类型的响应，例如：JSON,可以直接获取。

requests里的json方法就是封装了json.loads方法。

import json

import requests

# 发出一个get请求

response = requests.get("http://httpbin.org/get")

# text响应类型

print(type(response.text))

# 直接解析响应json(成字典)

print(**response.json()**) // 默认转换

# 获取响应内容后json进行解析(成字典)

print(**json.loads**(response.text)) // 手工调用json.loads将字符串转换为json对象

# 直接解析后的相应内容类型

print(type(response.json()))

运行结果：

<class 'str'>

{'args': {}, 'headers': {'Accept': '\*/\*', 'Accept-Encoding': 'gzip, deflate, br', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.31.0', 'X-Amzn-Trace-Id': 'Root=1-661d515e-6bdfc8f150e6d0335f95dfb4'}, 'origin': '171.80.82.188', 'url': 'http://httpbin.org/get'}

{'args': {}, 'headers': {'Accept': '\*/\*', 'Accept-Encoding': 'gzip, deflate, br', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.31.0', 'X-Amzn-Trace-Id': 'Root=1-661d515e-6bdfc8f150e6d0335f95dfb4'}, 'origin': '171.80.82.188', 'url': 'http://httpbin.org/get'}

<class 'dict'>

#### 2.3.5.添加头信息headers

需要传入HTTP Header时,传入一个dict作为headers参数。

添加头信息访问：

import requests

# 添加头部信息

**headers** = {

"**User-Agent**": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_12\_4)",

'**Authorization**': 'abcdefghijklmnopqrstuvwxyz1234567890'

}

# 发送请求

response = requests.get("http://httpbin.org/get", **headers=headers**)

# 打印响应

print(response.text)

运行结果：

{

"args": {},

"headers": {

"Accept": "\*/\*",

"Accept-Encoding": "gzip, deflate, br",

"Authorization": "abcdefghijklmnopqrstuvwxyz1234567890",

"Host": "httpbin.org",

"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_12\_4)",

"X-Amzn-Trace-Id": "Root=1-661d53ed-4db6d56d1c64107c4717ffd6"

},

"origin": "171.80.82.188",

"url": "http://httpbin.org/get"

}

#### 2.3.6.添加和获取cookie信息

requests对Cookie做了特殊处理，不必解析Cookie就可以轻松获取指定的Cookie

在请求中传入Cookie,只需准备一个dict传入cookies参数

import requests

# 添加头部信息

headers = {

"User-Agent": "HUAWEI MATE60 PRO",

'Authorization': 'abcdefghijklmnopqrstuvwxyz1234567890'

}

# cookies 以字典形式提供，cookies 值以字典形式提供，key为cookie\_name，value为cookie\_value

**cookies = {"username":"admin","password":"123456"}**

# 发送请求

response = requests.get("http://httpbin.org/get", headers=headers,**cookies=cookies**)

# 打印cookie

print(response.request.headers.get("Cookie"))

运行结果：

username=admin; password=123456

### 4.发送POST请求

**requests.post(url=url, headers=headers, data=params)**

参数说明：

url：请求url地址

headers：请求头

data：发送编码为表单形式的数据

#### 2.4.1.带参数的POST请求

发送POST请求,只需要把get()方法变成post(),传入data参数作为POST请求的数据

import requests

#参数写在字典里

**data** = {"name": "马斯克","age": 45}

#请求时将字典参数赋给data参数

response = requests.**post**("http://httpbin.org/post", **data=data**)

#打印响应

print(response.text)

运行结果：

{

"args": {},

"data": "",

"files": {},

"form": {

"age": "45",

"name": "\u9a6c\u65af\u514b"

},

"headers": {

"Accept": "\*/\*",

"Accept-Encoding": "gzip, deflate, br",

"Content-Length": "39",

"Content-Type": "application/x-www-form-urlencoded",

"Host": "httpbin.org",

"User-Agent": "python-requests/2.31.0",

"X-Amzn-Trace-Id": "Root=1-661d57cd-5dfbc792701b7da94a2ffe18"

},

"json": null,

"origin": "171.80.82.188",

"url": "http://httpbin.org/post"

}

#### 2.4.2.传递JSON数据

requests默认使用application/x-www-form-urlencoded对POST数据编码。如果要传递JSON数据，可以直接传入json参数

第1种方式：

将字典格式的数据赋给post方法的JSON参数(会自动将字典格式转为合法的JSON文本并添加到headers)

import requests

#参数写在字典里

**data** = {"company": "openAI","product": "ChatGPT4.0"}

#请求时将字典参数赋给data参数

response = requests.**post**(url="http://httpbin.org/post", **json=data**) # JSON格式的请求,将数据赋给json参数

#打印响应

print(response.text)

第2种方式：

声明字典格式的数据,然后再用json.dumps()方法把字典类型转换为合法的JSON字符串格式

import requests

import json

url = "http://httpbin.org/post"

**data** = {"company": "Google","product": "Gemini"}

# 在请求头里声明我们发送的格式

**headers = {"Content-Type":"application/json"}**

# 将字典格式的data变量转换为合法的JSON字符串传给post的data参数

response = requests.post(url=url, **data=json.dumps(data)**, **headers=headers**)

print(response.text)

#### 2.4.3.文件上传

文件上传需要用到请求参数里的files参数

在读取文件时,使用'rb'即二进制模式读取,这样获取的bytes长度才是文件的长度。

import requests

# rb,以只读的方式打开二进制文件

**files** = {"files": **open("a.jpeg", "rb")**}

# 发送post请求携带文件

response = requests.post("http://httpbin.org/post", **files=files**)

# 响应内容

print(response.text)

request可以把字符串当作文件进行上传

import requests

# 必需显式的设置文件名

**files** = {'file': **('test.txt', b'Hello Requests.')**}

# 发送post请求携带文件

response = requests.post("http://httpbin.org/post", **files=files**)

# 响应内容

print(response.text)

### 5.session会话维持（了解）

会话对象requests.Session能够跨请求地保持某些参数，比如cookies，即在同一个Session实例发出的所有请求都保持同一个cookies,而requests模块每次会自动处理cookies，这样就很方便地处理登录时的cookies问题。

import requests

headers = {'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8',

'Accept-Encoding': 'gzip, deflate, compress',

'Accept-Language': 'en-us;q=0.5,en;q=0.3',

'Cache-Control': 'max-age=0',

'Connection': 'keep-alive',

'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:22.0) Gecko/20100101 Firefox/22.0'}

# 创建session对象

**s = requests.Session()**

s.headers.update(headers) # 使用session访问并设置number参数

**s.get**("http://httpbin.org/cookies/set/number/123456")

# 使用同一个会话发送post请求,可以保持登录状态

response = **s.get**("http://httpbin.org/cookies")

print(response.text)

如果不使用对象发送请求,单独发一个post请求呢?

requests.get()或post()每次都会建立一个新会话。

### 6.身份验证（确保接口安全）

#### 2.6.1.基本身份认证(HTTP Basic Auth)

reqeusts支持Basic Auth（基本授权）和Digist Auth（摘要授权）

Oauth2.0参考

<https://cn.python-requests.org/zh_CN/latest/user/authentication.html>

import requests

from requests.auth import HTTPBasicAuth

response = requests.get('https://httpbin.org/hidden-basic-auth/user/passwd', **auth=('user', 'passwd')**)

print(response.json())

运行结果：

{'authenticated': True, 'user': 'user'}

#### 2.6.2.appid或token（令牌）方式认证

appid: 系统为合法用户赋予的访问id,固定的字符串,一般经过加密以确保HTTP传输中的安全。

token: 即令牌,固定或需要动态申请（有一定有效期）,一般由用户信息及申请时间计算加密而成,用于验证接口访问的权限。

token与session的区别

session是存在服务器的,服务端通过验证客户端的请求所携带的session值在服务会话中是否存在,来验证用户是否合法。

token: 是按一定算法加密计算出来的,服务端通过解密客户端所携带的token值来验证用户是否合法。

注册任意一个开发者平台(微信开发者平台、百度开发者平台、饿了么开发者平台),创建应用,根据相应的授权方式获取token,并使用token正常访问一个接口。

import requests

import json

# 申请apikey

app\_key = 'kPoFYw85FXsnojsy5bB9hu6x'

secret\_key = 'l7SuGBkDQHkjiTPU3m6NaNddD6SCvDMC'

img\_url = '//upload-images.jianshu.io/upload\_images/7575721-40c847532432e852.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240'

# 获取token

get\_token\_url = 'https://aip.baidubce.com/oauth/2.0/token?grant\_type=client\_credentials&client\_id={}&client\_secret={}'.format(**app\_key**,**secret\_key**)

token = requests.get(url=get\_token\_url).json().get("access\_token") # 从获取token接口的响应中取得token值

# 识别图片文字

orc\_url = 'https://aip.baidubce.com/rest/2.0/ocr/v1/general\_basic?access\_token={}'.format(token)

data = {"url": img\_url}

res = requests.post(url=orc\_url, data=data)

print(json.dumps(res.json(), indent=2, ensure\_ascii=False)) # 格式化输出

### 7.requests库详解

#### 2.7.1.请求

requests.get()

requests.post()

requests.put()

...

**requests.session()**: 用于保持会话（session）

除requests.session()外,其他请求方法的参数都包含

url,params,data, headers,cookies,files,auth,timeout等等。

参数说明：

**url**: 字符串格式,参数也可以直接写到url中。

**params**：url参数,字典格式。

**data**: 请求数据,字典或字符串格式。

**headers**: 请求头,字典格式。

**cookies**: 字典格式,可以通过携带cookies绕过登录。

**files**: 字典格式,用于混合表单（form-data）中上传文件。

**auth**: Basic Auth授权,数组格式 auth=(user,password)。

timeout: 超时时间（防止请求一直没有响应,最长等待时间）,数字格式,单位为秒。

#### 2.7.2.响应

res.status\_code: 响应的HTTP状态码

res.reason: 响应的状态码含义

**req.text：**响应的文本格式,按req.encoding解码

req.content: 响应的二进制格式

req.encoding: 解码格式,可以通过修改req.encoding='utf-8'来解决一部分中文乱码问题

req.apparent\_encoding：真实编码,由chardet库提供的明显编码

**req.json():** （注意,有括号）,响应的json对象（字典）格式,慎用！如果响应文本不是合法的json文本,或报错

req.headers: 响应头

req.cookies: 响应的cookieJar对象,可以通过req.cookies.get(key)来获取响应cookies中某个key对应的值

import requests

res = requests.get("https://www.baidu.com")

print(res.status\_code, res.reason) # 200 OK

print(res.text) # 文本格式,有乱码

print(res.content) # 二进制格式

print(res.encoding) # 查看解码格式 ISO-8859-1

print(res.apparent\_encoding) # utf-8

res.encoding='utf-8' # 手动设置解码格式为utf-8

print(res.text) # 乱码问题被解决

print(res.cookies.items()) # cookies中的所有的项 [('BDORZ', '27315')]

print(res.cookies.get("BDORZ")) # 获取cookies中BDORZ所对应的值 27315