# Requests库说明文档

# 文章推荐：[Requests: 让 HTTP 服务人类](http://cn.python-requests.org/zh_CN/latest/)

# [Python-第三方库requests详解](http://blog.csdn.net/shanzhizi/article/details/50903748)

# Requests 是用[Python](http://blog.csdn.net/shanzhizi" \t "_blank)语言编写，基于urllib，采用Apache2 Licensed开源协议的HTTP库，支持Python3，完全满足HTTP测试需求。

# 安装参考《接口自动化测试框架搭建》 > 扩展库的安装 > Requests库

# 导入模块及基本请求

**import** requests

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

r = requests.post("http://httpbin.org/post")

r = requests.put("http://httpbin.org/put")

r = requests.delete("http://httpbin.org/delete")

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

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

# 传递URL参数

# Requests 允许你使用 params 关键字参数，以一个字符串字典来提供这些参数。举例来说，如果你想传递 key1=value1 和 key2=value2 到 httpbin.org/get ，那么你可以使用如下代码：

payload = {'key1': 'value1', 'key2': 'value2'}

r = requests.get("http://httpbin.org/get", params=payload)

**print**(r.url)

# URL

http://httpbin.org/get?key1=value1&key2=value2

# 响应内容

# 一个http响应的格式通常包括：响应行、响应报头、响应正文

# 响应正文与响应头部之间有一空行间隔。Requests已将http响应封装成Response对象，从Response对象可获取响应正文的内容

# 响应正文文本：使用Response.text，可以获取响应的正文文本内容。

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

**print**(r.text)

# 响应正文文本

{

"origin": "115.236.48.234"

}

# Requests会自动对响应正文进行解码：如果Response.encoding为空，那么requests会猜测响应的编码方式，然后进行解码。如果你可以确定响应的编码方式，也可以先对Response.encoding进行设置，然后再通过Response.text获取响应正文。

# 二进制响应正文：（略-详见推荐文档）

# JSON响应正文：如果响应正文是一json串，可以使用Response.json()方法对响应正文进行json decode操作，并返回一个字典。

# 战旗直播TV端接口-首屏数据获取

R = requests.get("https://apis.zhanqi.tv/static/v2.2/tv/index.json")

**print**(r.json())

# JSON响应正文文本

# {     "code":0,     "message":"OK",     "data":{         "img":"<https://img1.zhanqi.tv/uploads/2017/09/love-2017092910570404571.png>",         "live":{             "id":"203752",             "img":"<https://img2.zhanqi.tv/live/fixed/ff/6f/203752_spic.jpg>",             "title":"拿打游戏当幌子的个人KTV",             "videoId":"203752\_mfP2V",             "line":"eyJjZG5zIjoxNzIsIm91dGNkbnMiOjQyLCJyYXRlIjoiMTcyfDE3MiJ9",             "online":56,             "status":0,             "nickname":"守望先锋职业系列赛"         },         "index":[             {                 "id":"107824",                 "img":"<https://img2.zhanqi.tv/uploads/2017/10/9fea2-2017101110140359189.jpeg>",                 "bpic":"<https://img3.zhanqi.tv/live/20171011/107824_2017-10-11-11-12-29_big.jpg>",                 "title":"云栖大会",                 "videoId":"107824\_F036k",                 "line":"eyJjZG5zIjoyMDIsIm91dGNkbnMiOjQyLCJyYXRlIjoiIn0=",                 "online":17593,                 "status":4,                 "nickname":"战旗官方直播频道"             },             {                 "id":"236116",                 "img":"<https://img1.zhanqi.tv/uploads/2017/09/b7d92-2017092017520293915.jpeg>",                 "bpic":"<https://img3.zhanqi.tv/live/20171011/236116_2017-10-11-11-15-49_big.jpg>",                 "title":"古灵精怪小米娅",                 "videoId":"236116\_PLPQA",                 "line":"eyJjZG5zIjoyMDIsIm91dGNkbnMiOjQyLCJyYXRlIjoiIn0=",                 "online":995,                 "status":4,                 "nickname":"米娅miya"             },             ......             {                 "id":"215291",                 "img":"<https://img3.zhanqi.tv/live/20171011/215291_2017-10-11-11-16-34_big.jpg>",                 "title":"励志上王者的南乔",                 "videoId":"215291\_Rp2aG",                 "line":"eyJjZG5zIjoyMDIsIm91dGNkbnMiOjQyLCJyYXRlIjoiIn0=",                 "online":8787,                 "status":4,                 "nickname":"南乔不可休思i",                 "avatar":"<https://img2.zhanqi.tv/avatar/0d/3ab/71579306_1506784138.jpg>"             }         ]     } }

# 如果响应正文不是一个json串，则会报错。

# 响应状态

# Response对象status\_code属性标识http请求响应的状态码。

# 战旗直播TV端接口-首屏数据获取

r = requests.get("https://apis.zhanqi.tv/static/v2.2/tv/index.json")

**print**(r.status\_code)

# 状态码

200

# 响应的头部

# Response对象的headers属性是一个字典，可以获得http响应结果的头部的相关信息。

# 战旗直播TV端接口-首屏数据获取

r = requests.get("https://apis.zhanqi.tv/static/v2.2/tv/index.json")

**print**(r.headers)

# 响应头部

{'Expires': 'Wed, 11 Oct 2017 03:31:58 GMT', 'Date': 'Wed, 11 Oct 2017 03:30:18 GMT', 'Server': 'Tengine/2.1.2', 'Content-Type': 'text/html; charset=utf-8', 'Transfer-Encoding': 'chunked', 'X-Powered-By': 'PHP/5.5.25', 'Access-Control-Allow-Origin': '\*', 'Access-Control-Allow-Methods': 'POST,GET', 'Access-Control-Allow-Headers': 'x-requested-with,content-type', 'CacheType': 'source-nx', 'websource': 'webapp47, backupGF\_upstream01', 'Cache-Control': 'max-age=100', 'Content-Encoding': 'gzip', 'X-Via': '1.1 jxingdxin48:10 (Cdn Cache Server V2.0)', 'Connection': 'keep-alive', 'Cdn-Src-Port': '17923', 'Strict-Transport-Security': 'max-age=31536000;includeSubdomains'}

# 定制请求头

# Requests支持定制http请求的头部，只需要构造一个字典，然后传给requests.get()的headers参数即可。

# 发送POST请求

# 一个http请求包括三个部分，为别为请求行，请求报头，消息主体，类似以下这样：请求行、 请求报头 、消息主体

# HTTP协议规定post提交的数据必须放在消息主体中，但是协议并没有规定必须使用什么编码方式。服务端通过是根据请求头中的Content-Type字段来获知请求中的消息主体是用何种方式进行编码，再对消息主体进行解析。具体的编码方式包括：

# application/x-www-form-urlencoded：以form表单形式提交数据。

# application/json：以json串提交数据。

# multipart/form-data：一般使用来上传文件。

# 以form形式发送post请求

# form形式

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

d = {'key1': 'value1', 'key2': 'value2'}

r = requests.post(url, data=d)

**print**(r.text)

# 输出：

{

"args": {},

"data": "",

"files": {},

"form": {

"key1": "value1",

"key2": "value2"

},

"headers": {

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

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

"Connection": "close",

"Content-Length": "23",

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

"Host": "httpbin.org",

"User-Agent": "python-requests/2.18.4"

},

"json": null,

"origin": "115.236.48.234",

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

}

# 以json形式发送post请求

**import** requests

**import** json

# json形式

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

s = json.dumps({'key1': 'value1', 'key2': 'value2'})

r = requests.post(url, data=s)

**print**(r.text)

# 输出

{

"args": {},

"data": "{\"key1\": \"value1\", \"key2\": \"value2\"}",

"files": {},

"form": {},

"headers": {

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

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

"Connection": "close",

"Content-Length": "36",

"Host": "httpbin.org",

"User-Agent": "python-requests/2.18.4"

},

"json": {

"key1": "value1",

"key2": "value2"

},

"origin": "115.236.48.234",

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

}

# 以multipart形式发送post请求

**import** requests

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

files = {'file': open('report.txt', 'rb')}

r = requests.post(url, files=files)

**print**(r.text)

# 输出

{

"args": {},

"data": "",

"files": {

"file": "hello world!!!!"

},

"form": {},

"headers": {

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

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

"Connection": "close",

"Content-Length": "161",

"Content-Type": "multipart/form-data; boundary=c38092cb1b31472ca6ce7718ba61a683",

"Host": "httpbin.org",

"User-Agent": "python-requests/2.18.4"

},

"json": null,

"origin": "115.236.48.234",

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

}

# Cookie设置

# 如果某个响应中包含一些 cookie，你可以快速访问它们：

>>> url = 'http://example.com/some/cookie/setting/url'

>>> r = requests.get**(**url**)**

>>> r.cookies**[**'example\_cookie\_name'**]**

'example\_cookie\_value'

# 要想发送你的cookies到服务器，可以使用 cookies 参数：

>>> url = 'http://httpbin.org/cookies'

>>> cookies = dict**(**cookies\_are='working'**)**

>>> r = requests.get**(**url**,** cookies=cookies**)**

>>> r.text

'{"cookies": {"cookies\_are": "working"}}'

# Cookie 的返回对象为 RequestsCookieJar，它的行为和字典类似，但界面更为完整，适合跨域名跨路径使用。你还可以把 Cookie Jar 传到 Requests 中：

>>> jar = requests.cookies.RequestsCookieJar**()**

>>> jar.set**(**'tasty\_cookie'**,** 'yum'**,** domain='httpbin.org'**,** path='/cookies'**)**

>>> jar.set**(**'gross\_cookie'**,** 'blech'**,** domain='httpbin.org'**,** path='/elsewhere'**)**

>>> url = 'http://httpbin.org/cookies'

>>> r = requests.get**(**url**,** cookies=jar**)**

>>> r.text

'{"cookies": {"tasty\_cookie": "yum"}}'

# 错误与异常

# 遇到网络问题（如：DNS 查询失败、拒绝连接等）时，Requests 会抛出一个 ConnectionError 异常。

# 如果 HTTP 请求返回了不成功的状态码， [Response.raise\_for\_status()](http://cn.python-requests.org/zh_CN/latest/api.html" \l "requests.Response.raise_for_status" \o "requests.Response.raise_for_status) 会抛出一个 HTTPError 异常。

# 若请求超时，则抛出一个 Timeout 异常。

# 若请求超过了设定的最大重定向次数，则会抛出一个 TooManyRedirects 异常。

# 所有Requests显式抛出的异常都继承自 requests.exceptions.RequestException 。