

Python 接口测试

——by 雷子

个人总结，
仅供学习使用

1. 概念：

接口测试是测试系统组件间接口的一种测试。接口测试主要用于检测外部系统与系统之间以及内部各个子系统之间的交互点。测试的重点是要检查数据的交换，传递和控制管理过程，以及系统间的相互逻辑依赖关系等。

2. 环境准备：

(1.) 安装 python

<https://www.python.org/downloads/> 下载你想用的版本（本文介绍的版本是 3.5 的版本）。安装后，添加系统环境变量。在 cmd 中输入 python，

(2.) 安装 requests 库。

<https://pypi.python.org/pypi/requests/> 下载地址，然后进入 cmd。进入下载目录，安装 python setup.py install

也可以在 cmd 输入 pip install requests 安装，安装完，

import requests 不报错即安装成功

requests 文档中文版本：

http://docs.python-requests.org/zh_CN/latest/user/quickstart.html

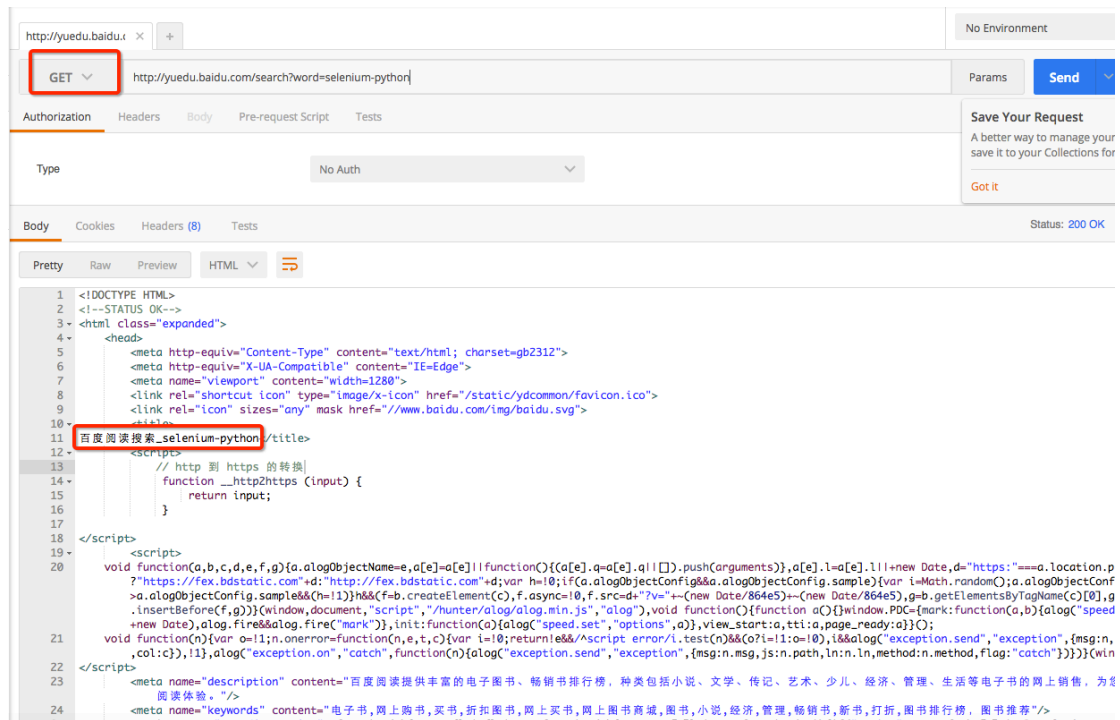
对于接口测试来说，一般分为二种情况，分别是基于 http 协议和基于 web services 协议，但是最常用的是基于 http 协议的

接口测试，其中最常用的 http 方法是 get 和 post，当然还有 put, delete 请求，接口测试的过程就是 client (浏览器) 向 server (服务

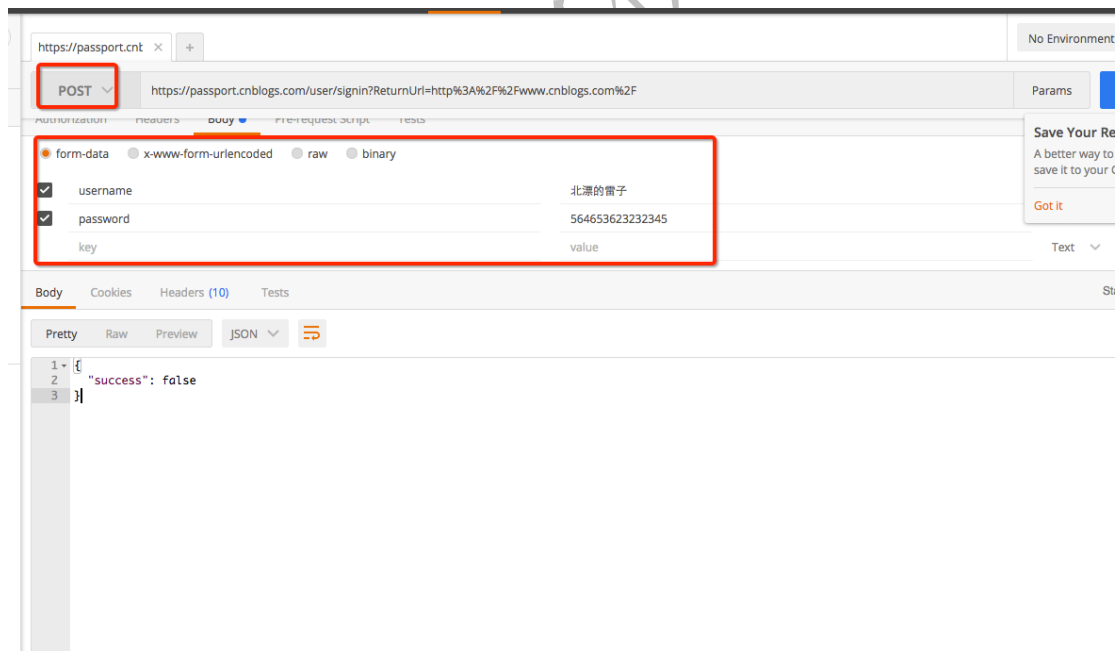
器端) request 一个请求, server 得到请求后, response 返回给 client 响应数据。

GET: 从指定资源获取数据

如在百度搜索输入 selenium-python 返回结果，如图，



post: 向指定的资源要被处理的数据, 以登录博客园为例



显然从结果看我们的登录是失败的。

PUT: 上传指定的 URL, 一般是修改, 可以理解为数据库中的 update。

DELETE: 删除指定资源。

在接口测试中，一般来说, post 创建数据， get 获取创建成功后的所有数据和指定的数据, put 可以对创建成功后的数据

进行修改， delete 是指定的资源。

三：http 状态码含义：

状态码：

1xx: 信息

消息: 描述:

100 Continue 服务器仅接收到部分请求，但是一旦服务器并没有拒绝该请求，客户端应该继续发送其余的请求。

101 Switching Protocols 服务器转换协议：服务器将遵从客户请求转换到另外一种协议。

2xx: 成功

消息: 描述:

200 OK 请求成功（其后是对 GET 和 POST 请求的应答文档。）

201 Created 请求被创建完成，同时新的资源被创建。

202 Accepted 供处理的请求已被接受，但是处理未完成。

203 Non-authoritative Information 文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝。

204 No Content 没有新文档。浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而 Servlet 可以确定用户文档足够新，这个状态代码是很有用的。

205 Reset Content 没有新文档。但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容。

206 Partial Content 客户发送了一个带有 Range 头的 GET 请求，服务器完成了它。

3xx: 重定向

消息: 描述:

300 Multiple Choices 多重选择。链接列表。用户可以选择某链接到达目的地。最多允许五个地址。

301 Moved Permanently 所请求的页面已经转移至新的 url。

302 Found 所请求的页面已经临时转移至新的 url。

303 See Other 所请求的页面可在别的 url 下被找到。

304 Not Modified 未按预期修改文档。客户端有缓冲的文档并发出了一个条件性的请求（一般是提供 **If-Modified-Since** 头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。

305 Use Proxy 客户请求的文档应该通过 **Location** 头所指明的代理服务器提取。

306 Unused 此代码被用于前一版本。目前已不再使用，但是代码依然被保留。

307 Temporary Redirect 被请求的页面已经临时移至新的 **url**。

4xx: 客户端错误

消息: 描述:

400 Bad Request 服务器未能理解请求。

401 Unauthorized 被请求的页面需要用户名和密码。

402 Payment Required 此代码尚无法使用。

403 Forbidden 对被请求页面的访问被禁止。

404 Not Found 服务器无法找到被请求的页面。

405 Method Not Allowed 请求中指定的方法不被允许。

406 Not Acceptable 服务器生成的响应无法被客户端所接受。

407 Proxy Authentication Required 用户必须首先使用代理服务器进行验证，这样请求才会被处理。

408 Request Timeout 请求超出了服务器的等待时间。

409 Conflict 由于冲突，请求无法被完成。

410 Gone 被请求的页面不可用。

411 Length Required "**Content-Length**" 未被定义。如果无此内容，服务器不会接受请求。

412 Precondition Failed 请求中的前提条件被服务器评估为失败。

413 Request Entity Too Large 由于所请求的实体的太大，服务器不会接受请求。

414 Request-url Too Long 由于 **url** 太长，服务器不会接受请求。当 **post** 请求被转换为带有很长的查询信息的 **get** 请求时，就会发生这种情况。

415 Unsupported Media Type 由于媒介类型不被支持，服务器不会接受请求。

416 服务器不能满足客户在请求中指定的 **Range** 头。

417 Expectation Failed

5xx: 服务器错误

消息: 描述:

500 Internal Server Error 请求未完成。服务器遇到不可预知的情况。

501 Not Implemented 请求未完成。服务器不支持所请求的功能。

502 Bad Gateway 请求未完成。服务器从上游服务器收到一个无效的响应。

503 Service Unavailable 请求未完成。服务器临时过载或当机。

504 Gateway Timeout 网关超时。

505 HTTP Version Not Supported 服务器不支持请求中指定的 HTTP 协议版本。

四：python 接口之 http 请求

python 的强大之处在于提供了很多的标准库以及第三库，本文介绍 urllib 和第三库的 requests。

Urllib 定义了很多函数和类，这些函数和类能够帮助我们在复杂的情况下获取 url 内容。复杂情况— 基本的和深入的验证，重定向，cookies 等等

Urllib 的 GET 请求代码如下：

```
import urllib.request
url='http://www.baidu.com'
response=urllib.request.Request(url=url)
html=urllib.request.urlopen(response)
print(html.getcode())
print(html.headers)
```

请求结果：

/Library/Frameworks/Python.framework/Versions/3.4/bin/python3.4

/Users/playcrab/PycharmProjects/jiekouceshi/pachong.py

200

Date: Mon, 20 Feb 2017 08:08:36 GMT

Content-Type: text/html; charset=utf-8

Transfer-Encoding: chunked

Connection: Close

Vary: Accept-Encoding

Set-Cookie: BAIDUID=D3E5547ACC26D3908EBB29522BABCCD4:FG=1; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com

Set-Cookie: BIDUPSID=D3E5547ACC26D3908EBB29522BABCCD4; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com

Set-Cookie: PSTM=1487578116; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com

Set-Cookie: BDSVRTM=0; path=/
www.baidu.com

Set-Cookie: BD_HOME=0; path=/
www.baidu.com

Set-Cookie: H_PS_PSSID=21935_1455_21090_17001_22036; path=/; domain=.baidu.com

P3P: CP=" OTI DSP COR IVA OUR IND COM "

Cache-Control: private

Cxy_all: baidu+547441cee80f5b2514d2439b86d8b151

Expires: Mon, 20 Feb 2017 08:08:33 GMT

X-Powered-By: HPHP

Server: BWS/1.1

X-UA-Compatible: IE=Edge,chrome=1

BDPAGETYPE: 1

BDQID: 0xc3d33f280001f43e

BDUSERID: 0

Urllib 的 Post 请求，代码：

```
import urllib.request

import urllib.parse

url='http://www.tuling123.com/openapi/api'

data={"key": "your", "info": '你好'}

data=urllib.parse.urlencode(data).encode('utf-8')

re=urllib.request.Request(url,data)

html=urllib.request.urlopen(re)

print(html.getcode(),html.msg)

print(html.read())
```

结果：

```
/Library/Frameworks/Python.framework/Versions/3.4/bin/python3.4
/Users/playcrab/PycharmProjects/jiekouceshi/pachong.py
```

200 OK

```
b' {"code":40001,"text": "\xe4\xba\x2\xe7\x88\xb1\xe7\x9a\x84\xef\xbc\x8c\xkey\xe4\xb8\x8d\xe5\xaf\xb9\xe5\x93\xa6\xe3\x80\x82"}'
```

(注：这里不是乱码是输出格式的问题。)

下面介绍下 requests 库的 http 请求、

GET 请求：

```
import requests

r = requests.get('https://www.baidu.com')

print(r.headers)
```

结果：


```
{'Last-Modified': 'Mon, 23 Jan 2017 13:23:55 GMT', 'Transfer-Encoding':  
'chunked', 'Pragma': 'no-cache', 'Cache-Control': 'private, no-cache,  
no-store, proxy-revalidate, no-transform', 'Content-Encoding': 'gzip',  
'Date': 'Mon, 20 Feb 2017 08:30:29 GMT', 'Server': 'bfe/1.0.8.18',  
'Connection': 'keep-alive', 'Content-Type': 'text/html', 'Set-Cookie':  
'BDORZ=27315; max-age=86400; domain=.baidu.com; path=/',  
__bsi=12827760870170119103_00_7_N_N_2_0301_002F_N_N_N_0; expires=Mon,  
20-Feb-17 08:30:34 GMT; domain=www.baidu.com; path=/'}
```

POST 请求:

```
import requests  
  
payload = {'key1': 'value1', 'key2': 'value2'}  
  
r = requests.post("http://httpbin.org/post", data=payload)  
  
print(r.text)
```

结果:

```
{  
  "args": {},  
  "data": "",  
  "files": {},  
  "form": {  
    "key1": "value1",  
    "key2": "value2"  
  },  
  "headers": {  
    "Accept": "*/*",  
    "Accept-Encoding": "gzip, deflate",  
    "Content-Length": "23",  
    "Content-Type": "application/x-www-form-urlencoded",  
    "Host": "httpbin.org",  
    "User-Agent": "python-requests/2.10.0"  
  },  
  "json": null,  
}
```

```
"origin": "180.87.10.156",
"url": "http://httpbin.org/post"
```

以上是利用 urllib 和 requests 发送 GET 和 POST 请求的事例。

五：python 数列化和反序列化

把 python 的对象编码转换为 json 格式的字符串，反序列化可以理解为：把 json 格式

字符串解码为 python 数据对象。在 python 的标准库中，专门提供了 json 库

导入，查看 json 库的主要方法

```
import json
print(json.__all__)
```

见 json 库的主要方法：

```
['dump', 'dumps', 'load', 'loads', 'JSONDecoder', 'JSONEncoder']
```

定义一个字典，通过 json 把它序列化为 json 格式的字符串，见实现的代码

```
import json

dict1={'name':'leizi','age':24,'address':'北京'}

print(u'未序列化前的数据类型为:',type(dict1))
print(u'未序列化前的数据:',dict1)

# 对 dict1 进行序列化的处理
str1=json.dumps(dict1)

print(u'序列化后的数据类型为:',type(str1))
print(u'序列化后的数据为:',str1)
```

结果：

未序列化前的数据类型为：<class 'dict'>

未序列化前的数据：{'age': 24, 'name': 'leizi', 'address': '北京'}

序列化后的数据类型为: <class 'str'>

序列化后的数据为: {"age": 24, "name": "leizi", "address":
"\u5317\u4eac"}

我们再反序列化, 把 json 格式的字符串解码为 python 的数据对象, 见实现的代码和输出:

```
import json

dict1={'name':'雷子','age':24,'address':'北京'}

print(u'未序列化前的数据类型为:',type(dict1))
print(u'未序列化前的数据:',dict1)
# 对 dict1 进行序列化的处理
str1=json.dumps(dict1)
print(u'序列化后的数据类型为:',type(str1))
print(u'序列化后的数据为:',str1)
# 对 str1 进行反序列化
dict2=json.loads(str1)
print(u'反序列化后的数据类型:',type(dict2))
print(u'反序列化后的数据: ',dict2)
```

输出

未序列化前的数据类型为: <class 'dict'>

未序列化前的数据: {'age': 24, 'name': '雷子', 'address': '北京'}

序列化后的数据类型为: <class 'str'>

序列化后的数据为: {"age": 24, "name": "\u96f7\u5b50", "address":
"\u5317\u4eac"}

反序列化后的数据类型: <class 'dict'>

反序列化后的数据: {'age': 24, 'name': '雷子', 'address': '北京'}

结合 requests 库, 来看返回的 json 数据, 具体代码为:

```
import json,requests

r=requests.get('http://wthrcdn.etouch.cn/weather_mini?city=北京')

print(r.text,u'数据类型:',type(r.text))

# 对数据进行反序列化的操作

dic=json.loads(r.text)

print(dic,u'数据类型:',type(dic))
```

结果:

```
{"desc": "OK", "status": 1000, "data": {"wendu": "4", "ganmao": "将有一次强降温过程, 且风力较强, 极易发生感冒, 请特别注意增加衣服保暖防寒.", "forecast": [{"fengxiang": "北风", "fengli": "4-5 级", "high": "高温 6\u00b0C", "type": "多云", "low": "低温 -2\u00b0C", "date": "20 日星期一"}, {"fengxiang": "南风", "fengli": "微风级", "high": "高温 1\u00b0C", "type": "小雪", "low": "低温 -3\u00b0C", "date": "21 日星期二"}, {"fengxiang": "北风", "fengli": "3-4 级", "high": "高温 8\u00b0C", "type": "阴", "low": "低温 -2\u00b0C", "date": "22 日星期三"}, {"fengxiang": "南风", "fengli": "微风级", "high": "高温 8\u00b0C", "type": "晴", "low": "低温 -2\u00b0C", "date": "23 日星期四"}, {"fengxiang": "北风", "fengli": "微风级", "high": "高温 10\u00b0C", "type": "晴", "low": "低温 -2\u00b0C", "date": "24 日星期五"}], "yesterday": {"f1": "4-5 级", "fx": "北风", "high": "高温 15\u00b0C", "type": "多云", "low": "低温 -2\u00b0C", "date": "19 日星期日"}, "aqi": "37", "city": "北京"}} 数据类型: <class 'str'>
```

```
{'status': 1000, 'data': {'city': '北京', 'yesterday': {'fx': '北风', 'type': '多云', 'date': '19 日星期日', 'f1': '4-5 级', 'high': '高温 15\u00b0C', 'low': '低温 -2\u00b0C'}, 'forecast': [{'fengxiang': '北风', 'low': '低温 -2\u00b0C', 'fengli': '4-5 级', 'date': '20 日星期一', 'high': '高温 6\u00b0C'}
```

```
' , 'type': '多云'}, {'fengxiang': '南风', 'low': '低温 -3℃', 'fengli': '微风级', 'date': '21 日星期二', 'high': '高温 1℃', 'type': '小雪'}, {'fengxiang': '北风', 'low': '低温 -2℃', 'fengli': '3-4 级', 'date': '22 日星期三', 'high': '高温 8℃', 'type': '阴'}, {'fengxiang': '南风', 'low': '低温 -2℃', 'fengli': '微风级', 'date': '23 日星期四', 'high': '高温 8℃', 'type': '晴'}, {'fengxiang': '北风', 'low': '低温 -2℃', 'fengli': '微风级', 'date': '24 日星期五', 'high': '高温 10℃', 'type': '晴'}], 'ganmao': '将有一次强降温过程, 且风力较强, 极易发生感冒, 请特别注意增加衣服保暖防寒。', 'wendu': '4', 'aqi': '37'}, 'desc': 'OK'} 数据类型: <class 'dict'>
```

五：使用 python 进行组织编写接口测试用例

接口测试其实就是几个步骤。

1. 拿到接口的 url 地址
2. 查看接口是用什么方式发送
3. 添加请求头, 请求体
4. 发送查看返回结果, 校验返回结果是否正确

明白了接口测试的测试步骤, 那么我们就可以来组织我们的代码。

```
import requests

# 接口的 url
url = "http://fanyi.baidu.com/v2transapi"

# 接口的参数
params = {
    "from": "en",
    "to": "zh",
    "query": "test"
}
```

```
# 发送接口

r = requests.request("post", url, params=params)

# 打印返回结果

print(r.text)

# 其实到上面就已经完了，因为百度不是我自己写的接口，为了让结果看的更加清楚一点，我取来翻译的
# 字段

import json

d = json.loads(r.text)

print(d['liju_result']['tag'])
```

结果：

['试验', '测验', '考验', '化验', '考查', '受试验', '受测验', '受考验', '测得结果']

（结果很长，截取最后的）

修改参数再次请求：

```
import requests

url = "http://fanyi.baidu.com/v2transapi"

params = {

    "from": "en",

    "to": "zh",

    "query": "study" #

}

r = requests.request("post", url, params=params)

import json

d = json.loads(r.text)

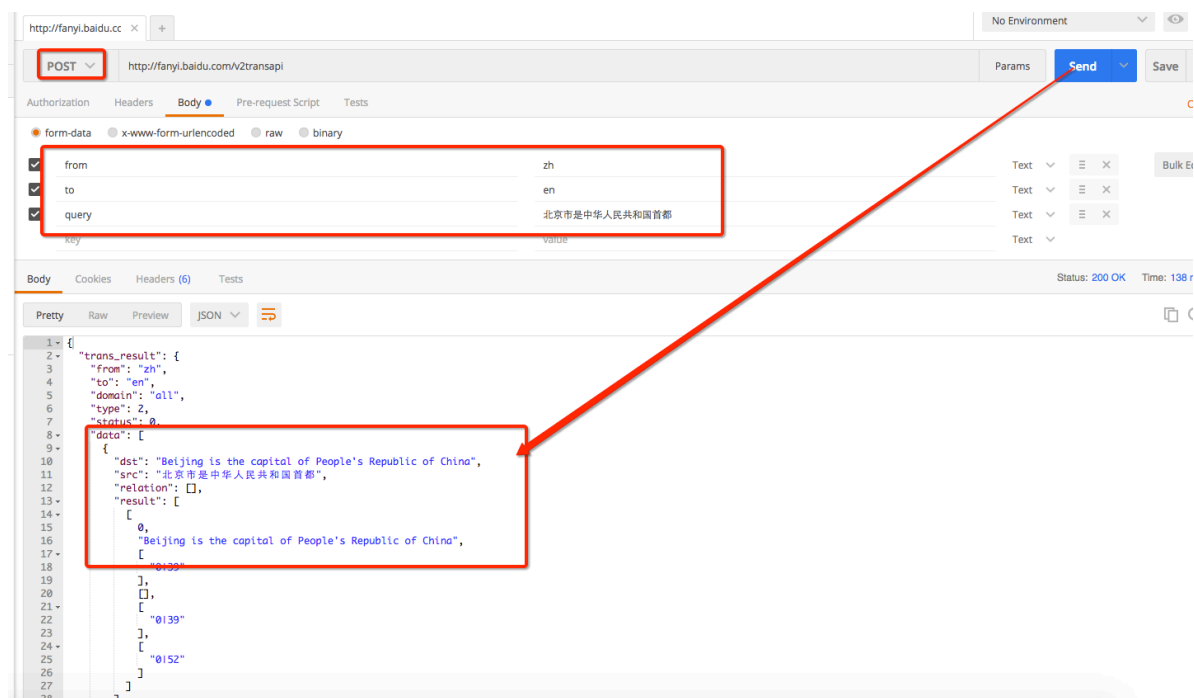
print(d['liju_result']['tag'])
```

结果：

['学习', '研究', '课题', '书房', '结论', '考虑', '沉思', '努力', '想出']

’]

ps: 我们看看利用工具测试该接口:



接下来我们来引入 unittest 库优化代码:

```
import requests,unittest,json

class Testbaiduapi(unittest.TestCase):

    def setUp(self):

        url = "http://fanyi.baidu.com/v2transapi"

    def testzhen(self):

        params = {

            "from":"en",

            "to":"zh",

            "query": "study" #

        }

        url = "http://fanyi.baidu.com/v2transapi"

        r = requests.request("post", url, params=params)

        r=json.loads(r.text)
```

```

        assert u'学习' in r['liju_result']['tag']

    def testzhen1(self):

        params = {

            "from": "en",

            "to": "h",

            "query": "stud" #

        }

        url = "http://fanyi.baidu.com/v2transapi"

        r = requests.request("post", url, params=params)

        r=json.loads(r.text)

        assert u'学' in r['liju_result']['tag']

    def tearDown(self):

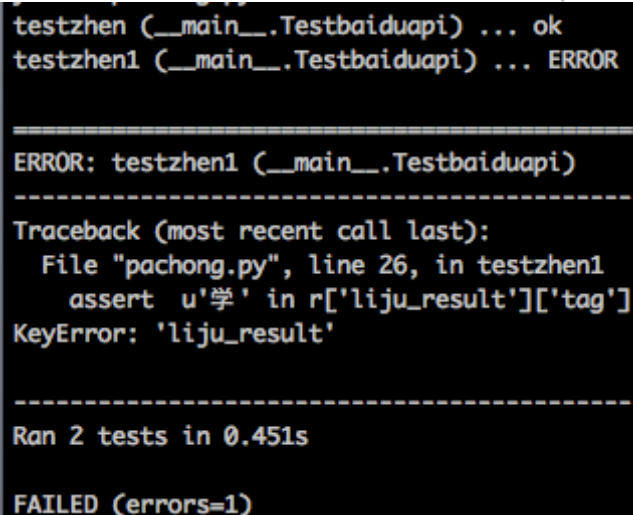
        pass

if __name__ == '__main__':

    unittest.main(verbosity=2)

```

结果:



```

testzhen (__main__.Testbaiduapi) ... ok
testzhen1 (__main__.Testbaiduapi) ... ERROR

=====
ERROR: testzhen1 (__main__.Testbaiduapi)
-----
Traceback (most recent call last):
  File "pachong.py", line 26, in testzhen1
    assert u'学' in r['liju_result']['tag']
KeyError: 'liju_result'

-----
Ran 2 tests in 0.451s

FAILED (errors=1)

```

在 python 中, 提供了 HTMLTestRunner.py 来生成测试报告, 把该文件下载后, 直接放到 python 安装文件的 lib 的目录下,

就可以导入该模块使用了, 见该实现的代码:


```

import requests,unittest,json,HTMLTestRunner

class Testbaiduapi(unittest.TestCase):

    def setUp(self):

        url = "http://fanyi.baidu.com/v2transapi"

    def testzhen(self):

        params = {

            "from":"en",

            "to":"zh",

            "query": "study" #

        }

        url = "http://fanyi.baidu.com/v2transapi"

        r = requests.request("post", url, params=params)

        r=json.loads(r.text)

        assert u'学习' in r['liju_result']['tag']

    def testzhen2(self):

        params = {

            "from":"en",

            "to":"h",

            "query": "stud" #

        }

        url = "http://fanyi.baidu.com/v2transapi"

        r = requests.request("post", url, params=params)

        r=json.loads(r.text)

        assert u'学' in r['liju_result']['tag']

    def tearDown(self):

        pass

if __name__=='__main__':

    report_dir= r's.html'

    re_open= open(report_dir,'wb')

    suite=unittest.TestLoader().loadTestsFromTestCase(Testbaiduapi)

    runner=HTMLTestRunner.HTMLTestRunner(

        stream=re_open,

        title=u'百度翻译 api 接口测试报告',

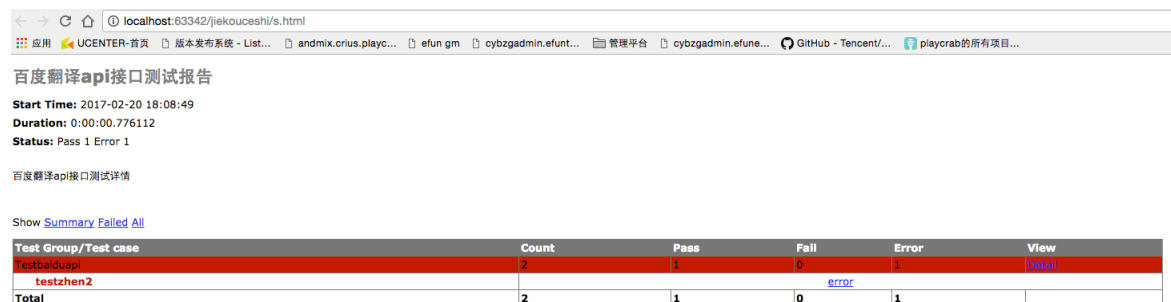
```

```
description=u'百度翻译 api 接口测试详情'

)

runner.run(suite)
```

执行后，会在当前目录下生成测试报告，截图如下：



The screenshot shows a web browser window with the address bar displaying 'localhost:63342/jiekouceshi/s.html'. The page title is '百度翻译api接口测试报告'. Below the title, the following information is displayed:

- Start Time: 2017-02-20 18:08:49
- Duration: 0:00:00.776112
- Status: Pass 1 Error 1

Below this information, there is a link 'Show Summary Failed All'. The main content of the page is a table with the following data:

Test Group/Test case	Count	Pass	Fail	Error	View
Testbaiduapi	2	1	0	1	Detail
testzhen2				error	
Total	2	1	0	1	

其他接口的测试方法也是这个思路，

作者寄语：

前进的道路我们充满着迷茫，

前进的每一步我们都会有收获。

路在脚下，我们决定不了我们的出身，但是我们可以努力改变我们未来。

告别昨天失败的自己，努力拼搏今天，成就美好明天

ps：如有疑问可以联系作者

email: leileilil126@163.com

qq:952943386 (qq 群: 194704520)