# 第2章 接口自动化测试

上一章我们学会了接口测试的一些理论知识，如何抓包以及怎样使用Postman手工测试各种接口。手工测试是自动化测试的基础，当我们使用Postman调通各个接口，清楚接口的各种场景下的返回值后，我们就可以开始使用代码实现接口测试的自动化。一般情况下，接口服务相对稳定，在每个迭代版本中有大量的回归工作。自动化测试脚本可以7\*24小时值守，帮我们监控及定时验证接口服务的正确性。

本章带领大家学习使用Python的第三方requests库构造和发送各种HTTP请求，学习如何处理接口依赖，异步接口，如何进行数据的参数化，如何解析JSON/XML/HTML格式的响应信息，以及断言的一些策略。

## 2.1 使用Requests发送请求

Python3自带的http.client以及urllib.request库都能发送HTTP请求，不过相对来说，以人为本的第三方requests库使用起来更方便。支持HTTPS,内容自动编码界面，会话保持，长连等。

### **2.1.1安装方法**

在安装了Python3环境和pip包管理器的情况下，我们可以直接在终端或命令行使用以下命令安装requests:

pip install requests

#### 验证是否安装成功：

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 05:52:31)

[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin

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

>>> import requests

>>>

打开命令行，输入python或python3，在python shell环境下输入import requests，没有报错即安装成功。

### **2.1.2 Requests简介**

#### 1.请求方法

requests库包含了get()/post()/put()/head()/delete()方法分别对应HTTP的GET/POST/PUT/HEAD/DELETE请求方法。除此之外，requests还提供统一的request()方法和用于会话保持的session()方法。

* requests.get()：用于发送HTTP GET请求
* requests.post(): 用于发送HTTP POST请求
* requests.put()：用于发送HTTP PUT请求
* requests.head(): 用于发送HTTP HEAD请求
* requests.delete(): 用于发送HTTP DELETE请求

这些请求方法的参数和使用方法一致，包含必选参数url，和可选参数params, data, headers, cookies, files, auth, timeout等。

* requests.request(): 统一的请求方法，第一个必选参数为'get'/'post'/...等使用的请求方法，后面的参数和上面的请求方法一致。
* requests.session(): 新建一个会话，用于保持会话

#### 2.请求参数

* 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
* verify: 是否验证HTTPS证书，默认为True
* 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

#### 3.响应解析

* 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，如果响应文本不是合法的JSON字符串，会报错
* 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 ...

### **2.1.3 发送GET请求**

#### 1.发送一个请求分3步

* 组装请求： 请求可能包含url，params(url参数)，data(请求数据)，headers(请求头)，cookies等，最少必须有url
* 发送请求，获取响应：支持get，post等各种方法发送，返回的是一个响应对象
* 解析响应: 输出响应文本

打开Pycharm，新建一个demo项目，项目下新建一个.py文件，输入以下内容：

# 导入requests包

import requests

# 1. 组装请求

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

# 2. 发送请求，获取响应

res = requests.get(url) # res即返回的响应对象

# 3. 解析响应

print(res.text) # 输出响应的文本

#### 2.运行并查看输出的响应内容

2.1.2 发送带参数的GET请求

GET请求的参数（URL参数）可以直接写到URL中发送，如果参数较多，也可以把参数单独提出来构造一个字典格式的变量传递给reqeusts.get()方法的params参数

#### 3.参数直接写到URL中

import requests

# 参数可以写到url里

url = "http://www.tuling123.com/openapi/api"\

"?key=ec961279f453459b9248f0aeb6600bbe&info=你好"

res = requests.get(url)

print(res.text)

#### 4.参数构造成单独的字典

import requests

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

# 字典格式，单独提出来，方便参数的添加修改等操作

params = {"key":"ec961279f453459b9248f0aeb6600bbe","info":"你好"}

res = requests.get(url=url, params=params)

print(res.text)

### **2.1.4 发送表单格式POST请求**

表单格式指网页登录、注册这种需要用户输入提交一种常用格式，HTTP请求头中Content-Type类型为application/x-www-form-urlencoded格式，这种格式，会将表单中的变量进行url编码后放到请求体中传送。我们将要提交的表单数据构造一个字典传递给requests.post()方法的data参数即可。

import requests

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

# POST请求发送的请求体数据，字典格式

data = {"name": "hanzhichao", "age": 18}

# 这里使用post方法，参数和get方法一样

res = requests.post(url=url, data=data)

print(res.text)

发送请求时，requests库会自动将字典格式data参数转化为url编码放入请求体中，并在请求头中添加Content-Type: application/x-www-form-urlencoded声明内容格式

**注意：**如果给data参数传递的不是变量而是字符串，requests会将改字符串作为原始请求数据不做任何编码传递。也不会在请求头中添加内容类型声明。

### **2.1.5 发送JSON格式POST请求**

JSON格式的请求我们需要对比将要发送的JSON字符串构造一个字典格式的数据传递给requests.post()方法的json参数即可。

import requests

url = "http://openapi.tuling123.com/openapi/api/v2"

data = {

"reqType":0,

"perception": {

"inputText": {

"text": "附近的酒店"

},

"inputImage": {

"url": "imageUrl"

},

"selfInfo": {

"location": {

"city": "北京",

"province": "北京",

"street": "信息路"

}

}

},

"userInfo": {

"apiKey": "ec961279f453459b9248f0aeb6600bbe",

"userId": "206379"

}

}

# JSON格式的请求，将数据赋给json参数

res = requests.post(url=url, json=data)

print(res.text)

发送请求时，requests库会自动将字典格式的json参数转化为JSON字符串，并在请求头中添加Content-Type: application/json声明内容类型。

**注意：**Python的字典和JSON字符串格式很相似但稍有区别，字典中可以使用单引号或双引号，JSON中只能使用双引号，字典中可以有注释，最后一项后可以有逗号，JSON不允许有注释，最后一项不能有逗号，字典的False/True/None对应JSON的false/true/null。

另外我们也可以通过向data参数传递原始JSON字符串来发送JSON格式的请求:

import requests

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

headers = {"Content-Type": "application/json;utf-8;"

# 多行文本, 字符串格式，也可以单行（注意外层有引号，为字符串）

data = '''{

"name": "hanzhichao",

"age": 18

}'''

res = requests.post(url=url, data=data, headers=headers)

print(res.text)

data应是合法的JSON字符串格式，由于使用的是原始格式，一般情况下要自己在请求头中添加内容类型声明。

### **2.1.6 上传文件接口**

### **2.1.7 Basic Auth授权接口**

## 2.2 接口依赖

## 2.3 异步接口

## 2.4 请求参数化

### **2.4.1 字符串格式化**

### **2.4.2 字典/序列键值更新**

### **2.4.3 数据文件的使用**

## 2.5 响应解析

### **2.5.1 JSON响应解析**

### **2.5.2 使用JSONPath解析嵌套JSON**

### **2.5.3 XML/HTML响应解析**

### **2.5.4 正则表达式**

## 2.6 响应断言

### **2.6.1 整体断言**

### **2.6.2 包含断言**

### **2.6.3 逐值断言**

### **2.6.4 JSON Schema断言**