JSON 请求

在接口的请求中常常会碰到需要发送 json 格式的请求，这种情况下，既可以使用关键字参数 data，也可以使用关键字参数 json 来传递 json 请求。

JSON 请求的发送

使用 data 关键字发送 json 请求，需要使用 json.dumps 对传入的变量进行转码：

```
>>> import json>>> import requests>>> r = requests.post('http://httpbin.org/post', data=json.dumps({'key': 'value'}))>>> print(r.request.headers)
{'User-Agent': 'python-requests/2.22.0', 'Accept-Encoding': 'gzip, deflate',\ 'Accept': '*/*', 'Connection': 'keep-alive', 'Content-Length': '16'}
```

使用 json 关键字参数发送请求：

```
>>> import requests>>> r = requests.post('http://httpbin.org/post', json = {'key':'value'})>>> print(r.request.headers){'User-Agent': 'python-requests/2.22.0', 'Accept-Encoding': 'gzip, deflate',\ 'Accept': '*/*', 'Connection': 'keep-alive', 'Content-Length': '16',\  'Content-Type': 'application/json'}
```

对比两次请求可以看出，如果请求的参数选择是**json**，那么 “Content-Type” 自动变为 “application/json”。

JSON 响应断言

在之前的的章节已经简单介绍了如何断言接口的响应值，而在本章节，主要介绍如何通过 JsonPath 解决断言问题。JsonPath 提供了强大的 JSON 解析功能，使用它自带的类似正则表达式的语法，可以更便捷灵活的用来获取对应的 JSON 内容。

JsonPath 语法

工欲善其事必先利其器，如果想要很好的使用 JsonPath，必须先对其语法有一定的了解。可以看到下表还加入了 XPath 进行对比，这两者的定位方式，有着非常多的相似之处。

XPath	JSONPath	描述
/	$	根节点对象/元素
.	@	当前的对象/元素
/	. or []	匹配下级元素
..	n/a	匹配上级元素，JSONPath不支持
//	..	

		递归方式匹配所有子元素
*	*	通配符，匹配所有对象/元素，无论其名称如何
@	n/a	属性访问，json结构体没有这个特性.
[]	[]	下标运算符。JSONPath从0开始
|	[,]	连接的操作符，多个结果拼接成列表返回
[]	?()	过滤器（脚本）表达式。
n/a	()	脚本表达式，使用基础脚本引擎。

比如同样一个字段，XPath 中的语法是:

```
/store/book[0]/title
```

JsonPath 的语法是:

```
$.store.book[0].title$['store']['book'][0]['title']
```

> 更多内容请访问：https://goessner.net/articles/JsonPath/

JsonPath 实战练习

下面是一组 JSON 结构，分别通过 JsonPath 和 XPath 的方式提取出来：

```
{ "store": {    "book": [       { "category": "reference",        "author": "Nigel Rees"
,        "title": "Sayings of the Century",        "price": 8.95      },      { "categor
y": "fiction",        "author": "Evelyn Waugh",        "title": "Sword of Honour",
  "price": 12.99      },      { "category": "fiction",        "author": "Herman Melville
",        "title": "Moby Dick",        "isbn": "0-553-21311-
3",        "price": 8.99      },      { "category": "fiction",        "author": "J. R. R
. Tolkien",        "title": "The Lord of the Rings",        "isbn": "0-395-19395-
8",        "price": 22.99      }    ],    "bicycle": {      "color": "red",      "price"
: 19.95    }  }}
```

下表列出了 XPath 与 JsonPath 的对比：

JSONPath	结果
$.store.book[*].author	store中所有book的author
$..author	所有的author
$.store.*	store中所有元素
$.store..price	store中所有的price
$..book[2]	book列表中的第三个

$..book[-1:]	book列表中的倒数第一个
$..book[:2]	book列表中的前两个
$..book[?(@.isbn)]	所有有isbn的book
$..book[?(@.price<10)]	所有价格低于10的书
$..*	所有json结构体中的元素

Python 与 JsonPath 组合断言接口

环境准备：

```
pip install jsonpath
```

下面是一个 get 请求实现了，断言响应值中 login 字段为 VipMagic 所对应的 node_name 为“性能常识”。

```
import requestsfrom jsonpath import jsonpathr = requests.get(\"
https://testerhome.com/api/v3/topics.json?limit=2
").json()assert jsonpath(r, \"$..topics[?
(@.user.login == 'VipMagic')].node_name")[0] == '性能常识'
```