概述

近期在写接口测试系列时，总感觉很不对劲，主要是缺乏一个比较完整的用于API测试的环境，四处找公开的API真心不靠谱，尝试了不少公开的，总觉得少了些什么，所以决定自己搭建一个，后续所有的实例都会基于本文所构建的mock server进行API实例演示。

选型

为了让大家了解python的强大，我们flask来做一个简单的server

如果你需要更多的了解flask请参见官方中文文档:

http://docs.jinkan.org/docs/flask

安装

安装flaask

```
pip install flask
```

支持

支持GET, POST, PUT, PATCH, DELETE 等http方法

看一个简单的flask代码

```
#-*- coding:utf-8 -*-

__author__ = "苦叶子"

from flask import Flask

app = Flask(__name__)

@app.route('/api')
def index():
    return "Hello, World!"

if __name__ == '__main__':
    app.run()
```

将上述代码保存至run.py, 使用一下命令运行

```
python run.py runserver
```

打开浏览器输入http://localhost:5000 即可在浏览器看到Hello， World几个单词的输出。

基本示例

这里我们构建一个简单的server，后续我们的接口测试分享实战都会基于这个server来进行交互实战。

为了让大家显得简洁，我不会添加异常等容错处理。

下面我们基于flask实现HTTP的GET\POST\HEAD等方法，用于后续的测试，然后也可以基于这个代码进一步扩展成restful风格的API。

我们先简单的实现，大家根据需要自行学习扩展.

```
#-*- coding:utf-8 -*-


__author__ = "苦叶子"

from flask import Flask
from flask import jsonify
from flask import request, Response
import random
import time

app = Flask(__name__)

"""
    这里所有的接口我们才去返回json串
    所有的json传对应的value值都为随机的
"""

# 生成随机字符串
def random_str():
    # 待选随机数据
    data =
"1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&*()_+=-"

    # 用时间来做随机播种
    random.seed(time.time())

    # 随机选取数据
    sa = []
    for i in range(8):
        sa.append(random.choice(data))
    salt = ''.join(sa)

    return salt

# 构建response
def make_response():
    content = '{"result": "%s", "data": "%s"}' % (random_str(), random_str())
    resp = Response(content)
    resp.headers["Access-Control-Origin"] = '*'

    return resp
```

```python
# http GET
@app.route("/query", methods=["GET"])
def query():
    print("////////////////////////////")
    return jsonify(
        username=random_str(),
        password=random_str()
        )

# http POST
@app.route("/update", methods=["POST"])
def update():

    return make_response()


# http delete
@app.route("/delete", methods=["DELETE"])
def delete():

    return make_response()

# http head
@app.route("/head", methods=["HEAD"])
def head():

    return make_response()

if __name__ == "__main__":

    app.run(debug=True)
```

说明：

1. 注意POST\HEAD\DELETE方法，响应头均被加入了Access-Control-Origin属性，其值为：*
2. 注意即便给HEAD方法添加了响应内容，但你在实际接收到的内容是木有响应内容的，请思考为什么
3. 上述仅用于简单的测试，不讨论其优雅、靠谱、高大上等等可能性

对应的flask代码及jmeter测试代码请参见：

链接: https://pan.baidu.com/s/1pLv7Mo7

密码: iha4

扫一扫关注微信公众号：