概述

上篇我们很简单的分享了如何基于flask搞一个支持http GET\POST\HEAD\DELETE方法的服务，大家可以根据这个简单的实例进行扩展。

下面我们基于flask来构建一个简单的restful风格的API服务出来，以便大家进一步了解和掌握，说不定哪天你就需要自己去实现一个简单的mock server以便让你的测试更加顺畅。

注意

1. 实现一个简单的restful api
2. 简单到就像没有任何封装
3. 不要问我什么是restful风格

安装

使用以下命令安装flask-restful

```
pip install flask-restful
```

示例

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


__author__ = "苦叶子"

from flask import Flask
from flask_restful import reqparse, abort, Api, Resource
import random
import time


# 生成随机字符串
def random_str(lenght):
    # 待选随机数据
    data = "1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

    # 用时间来做随机播种
    random.seed(time.time())
    time.sleep(0.3)

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

```
    return salt

# 初始化
app = Flask(__name__)
api = Api(app)

# 初始化源数据
# 随机生成
USERS = {
    "user1": {
        "username": random_str(10),
        "password": random_str(16),
        "token": random_str(32)
    },
    "user2": {
        "username": random_str(10),
        "password": random_str(16),
        "token": random_str(32)
    },
    "user3": {
        "username": random_str(10),
        "password": random_str(16),
        "token": random_str(32)
    }
}

# 判断用户id是否存在
def abort_if_user_not_exist(user_id):
    if user_id not in USERS:
        abort(404, message="user {%s} is not exist" % user_id)

parser = reqparse.RequestParser()
parser.add_argument("username", type=str)

# 用户管理
class User(Resource):
    # 获取指定用户信息
    def get(self, user_id):
        abort_if_user_not_exist(user_id)

        return USERS[user_id]

    # 删除指定用户
    def delete(self, user_id):
        abort_if_user_not_exist(user_id)

        del USERS[user_id]

        return "", 204

    # 新增/修改用户
```

```
    def put(self, user_id):
        args = parser.parse_args()
        print(args)

        user = {"username": args["username"],
            "password": random_str(16),
            "token": random_str(32)}

        USERS[user_id] = user

        return user, 201

    # 新增/修改用户
    def post(self, user_id):
        args = parser.parse_args()
        print(args)
        user = {"username": args["username"],
            "password": random_str(16),
            "token": random_str(32)}

        USERS[user_id] = user

        return user, 201

# 查询所有用户信息
class UserList(Resource):
    def get(self):
        return USERS


# 新增资源
api.add_resource(UserList, "/user")
api.add_resource(User, "/user/<user_id>")


# 主入口程序
if __name__ == "__main__":
    app.run(debug=True)
```

小结

如果把上述USERS数据源替换为数据库，你发现是不是就变成了一个基本的管理系统了？自己去尝试吧，come on 小白

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：