APIObject 模式与原则

在普通的接口自动化测试中，如果接口的参数，比如 url，headers 等传参改变，或者测试用例的逻辑、断言改变，那么整个测试代码都需要改变。APIObject 设计模式借鉴了 PageObject 的设计模式，可以实现一个优雅、强大的接口测试框架。

理念

APIObject 设计模式可以简单分为 6 个模块，分别是 API 对象、接口测试框架、配置模块、数据封装、Utils、测试用例。

- 接口测试框架：base_api，完成对 API 的驱动
- API 对象：继承 base_api 后，完成对接口的封装
- 配置模块：完成配置文件的读取
- 数据封装：数据构造与测试用例的数据封装
- Utils：其他功能封装，改进原生框架不足
- 测试用例：调用 Page/API 对象实现业务并断言

枯燥的讲述概念可能难以理解，后面的章节都会围绕这个6个模块进行理论的拆解和实例的演示。

APIObject 模式应用

本章将结合企业微信的部门管理，获取部门列表接口作为一个接口测试用例，从没有封装到使用 APIObject 设计模式进行封装改造。将实战与理论结合，更深入理解 APIObject 设计模式。

环境准备

企业微信服务端 API：

https://work.weixin.qq.com/api/doc/90000/90135/90664

不加任何封装和改造的企业微信，获取部门列表接口测试用例：

```
import requests


class TestDemo:

    def test_get_token(self):
        r = requests.get(url="
https://qyapi.weixin.qq.com/cgi-bin/gettoken
",
            params={"corpid": "ww93348658d7c66ef4", "corpsecret":
"T0TFrXmGYel167lnkzEydsjl6bcDDeXVmkUnEYugKIw"})
        return r.json()["access_token"]

    def test_department_list(self):
```

```
        r = requests.get(url="
https://qyapi.weixin.qq.com/cgi-bin/department/list
",
            params={"access_token": self.test_get_token(), "id": 1})
        assert r.json()["errcode"] == 0
        return print(r.json())

```

思路

改造后的文件结构：

```
├── __init__.py
├── api
│   ├── __init__.py
│   ├── base_api.py
│   ├── department.py
│   └── wework.py
├── data
│   └── department_list.yml
├── testcases
│   ├── __init__.py
│   └── test_department_list.py
└── utils
    ├── __init__.py
    └── utils.py

```

- API
  - base_api.py 是封装用来所有 API 的通用方法，比如打印 Log、对断言工具做二次封装等，不牵涉和业务相关的操作；
  - wework.py 继承 base_api 并实现基本业务，之后所有的具体的业务资源继承自 wework，比如 token 的获取等；
  - department 继承自 wework，用来实现对应模块具体的业务逻辑，比如发送请求，请求内有什么参数等等。
- testcases 文件夹内统一存放所有的测试用例，调用 API 对象实现业务并断言；
- utils 文件夹内存放对其他功能封装，改进原生框架不足；
- data 文件夹数据构造与测试用例的数据封装；

此外，还有配置模块与数据封装会在后面的章节进行具体的介绍。

APIObject 实战

utils.py，在此文件中封装一个 jsonpath 方法。

```python
import json

from jsonpath import jsonpath

class Utils:
    @classmethod
    def jsonpath(cls, json_object, expr):
        return jsonpath(json_object, expr)

```

base_api.py，在此文件中调用utils中的jsonpath方法。

```python
from test_wework.utils.Utils import Utils

class BaseApi:
    json_data = None

    def jsonpath(self, expr):
        return Utils.jsonpath(self.json_data, expr)

```

wework.py，继承类BaseApi，实现 token 的获取。将在后面“通用 API 封装”章节中详细讲述函数内部的设计。

```python
class WeWork(BaseApi):
    corpid = "ww93348658d7c66ef4"
    contact_secret = "T0TFrXmGYel167lnkzEydsjl6bcDDeXVmkUnEYugKIw"
    token = dict()
    token_url = "
https://qyapi.weixin.qq.com/cgi-bin/gettoken
"

    @classmethod
    def get_token(cls, secret=contact_secret):
        # 避免重复请求，提高速度
        if secret not in cls.token.keys():
            r = cls.get_access_token(secret)
            cls.token[secret] = r["access_token"]
        return cls.token[secret]

```

```
    @classmethod
    def get_access_token(cls, secret):
        r = requests.get(cls.token_url, params={"corpid": cls.corpid, "corpsecret": secret})
        return r.json()

```

department.py，继承类WeWork，发起一个 get 请求，获取 department 的 list。

```
class Department(BaseApi):
    list_url = "https://qyapi.weixin.qq.com/cgi-bin/department/list"

    def list(self, id):
        self.json_data = requests.get(self.list_url, params={"access_token": WeWork.get_contact_token(), "id": id}).json()
        return self.json_data

```

test_department.py，断言返回值中的第一个 name 是否为"WestWayyt"。

```

class TestDepartment:
    department = Department()

    def test_department_list(self):
        r = self.department.list(1)
        assert self.department.jsonpath(expr="$..name")[0] == "WestWayyt"
```