在 APIObject 设计模式中，需要一个 base_api 作为其他 API 步骤的父类，把通用功能放在这个父类中，供其他的 API 直接继承调用。这样做的优点在于，减少重复代码，提高代码的复用性。

## 通用 API 封装实战

在上一章节在演示使用 API-Object 设计模式对脚本进行改造时提到了 base_api。不过在上一章，仅仅只是封装了一个 utils 中的一个简单方法。并没有完全体现出 base_api 的实际作用。
接下来，我们通过通用接口协议的定义与封装实战，来实际体会一下 base_api 的巧妙之处。

- base_api.py

在代码内，对 request 进行一层封装，当然在这里还看不出来具体的优势:

```
import requests

class BaseApi:

    def request(self, method, url, **kwargs):
        self.json_data = requests.request(method=method, url=url, **kwargs)
        return self.json_data

```

- wework.py

继承于类 BaseApi，可以**直接调用**父类中的 request 方法（不需要导入 requests 库），从而发起一个 get 请求：

```
from test_interface.test_wework.api.base_api import BaseApi

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

    def get_access_token(self):
        r = self.request(method="get", url=self.token_url,
                         params={"corpid": self.corpid, "corpsecret":
self.contact_secret})
        return r.json()
```

```
13
14
```

- test_wework.py

继承于类 WeWork，主要目的只是为了检查上面的 get_access_token(self) 是否成功：

```python
from test_interface.test_wework.api.wework import WeWork

class TestWeWork(WeWork):

    def test_get_access_token(self):
        r = self.get_access_token()
        assert r["errcode"]==0

```

以上，在上面的案例中，在 base_api.py 中对 requests 进行了多一层的封装，这样只要是属于 BaseApi 这个类的子类，都可以无需引用而直接调用 requests 库。从而发起各种各样的请求，实现了通用接口协议的定义与封装。