今日学习目标

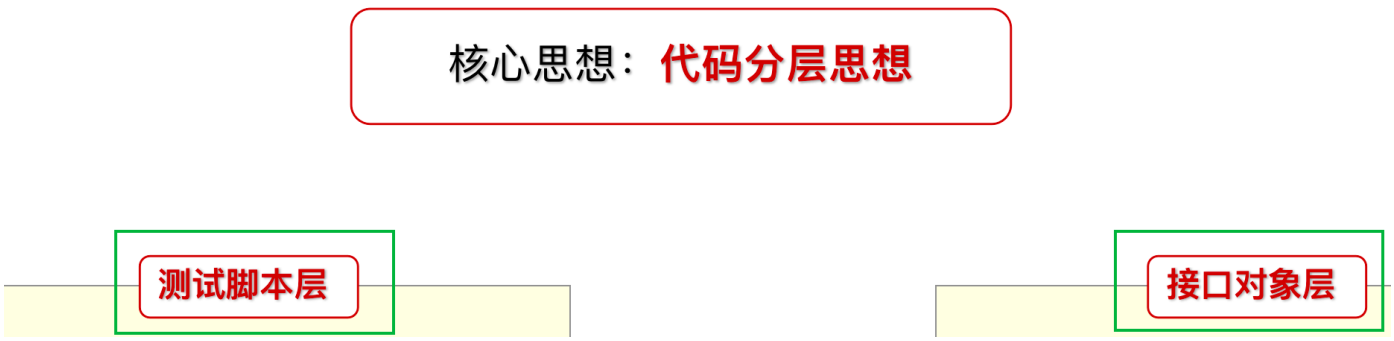
掌握接口对象封装的思想和实现

接口对象封装

解决的问题

- 代码冗余度高（有大量重复代码）
- 代码耦合度高
- 代码维护成本高

核心思想：代码分层



- 分层思想：
  - 将 普通方法实现的 ， 分为 接口对象层 和 测试脚本层。
- 接口对象层：
  - 对 接口 进行封装。封装好之后，给 测试用例层 调用！
  - 面向对象 类 封装 实现。
- 测试用例层：
  - 调用 接口对象层 封装的方法，拿到 响应结果，断言进行接口测试！
  - 借助 unittest 框架实现

封装Tpshop商城

普通方式实现

```
import unittest
import requests
```

```
3
4   class TestTpshopLogin(unittest.TestCase):
5       # 测试 登录成功
6       def test01_login_ok(self):
7           # 创建 session 实例
8           session = requests.Session()
9
10          # 使用实例，调用get 发送获取验证码请求
11          session.get(url="http://tpshop-test.itheima.net/index.php?
    m=Home&c=User&a=verify&r=0.21519623710645064")
12
13          # 使用实例，调用post 发送登录请求
14          resp = session.post(
15              url="http://tpshop-test.itheima.net/index.php?
    m=Home&c=User&a=do_login&t=0.7094195931397276",
16              data={"username": "13012345678", "password": "123456", "verify_code":
    "8888"})
17
18          print("响应结果 =", resp.json())
19
20          # 断言:
21          self.assertEqual(200, resp.status_code)
22          self.assertEqual(1, resp.json().get("status"))
23          self.assertEqual("登陆成功", resp.json().get("msg"))
24
25      # 测试 手机号不存在
26      def test02_tel_not_exists(self):
27          # 创建 session 实例
28          session = requests.Session()
29
30          # 使用实例，调用get 发送获取验证码请求
31          session.get(url="http://tpshop-test.itheima.net/index.php?
    m=Home&c=User&a=verify&r=0.21519623710645064")
32
33          # 使用实例，调用post 发送登录请求
34          resp = session.post(
35              url="http://tpshop-test.itheima.net/index.php?
    m=Home&c=User&a=do_login&t=0.7094195931397276",
36              data={"username": "13847834701", "password": "123456", "verify_code":
    "8888"})
37
38          print("响应结果 =", resp.json())
39
40          # 断言:
41          self.assertEqual(200, resp.status_code)
42          self.assertEqual(-1, resp.json().get("status"))
43          self.assertEqual("账号不存在!", resp.json().get("msg"))
44
45      # 测试 密码错误
```

```
46      def test03_pwd_err(self):
47          # 创建 session 实例
48          session = requests.Session()
49
50          # 使用实例，调用get 发送获取验证码请求
51          session.get(url="http://tpshop-test.itheima.net/index.php?
m=Home&c=User&a=verify&r=0.21519623710645064")
52
53          # 使用实例，调用post 发送登录请求
54          resp = session.post(
55              url="http://tpshop-test.itheima.net/index.php?
m=Home&c=User&a=do_login&t=0.7094195931397276",
56              data={"username": "13012345678", "password": "123456890",
"verify_code": "8888"})
57
58          print("响应结果 =", resp.json())
59
60          # 断言:
61          self.assertEqual(200, resp.status_code)
62          self.assertEqual(-2, resp.json().get("status"))
63          self.assertEqual("密码错误!", resp.json().get("msg"))
64
```

登录接口对象层

封装思想：

- 将 动态变化的数据，设计到方法的参数。
- 将 固定不变的，直接写成方法实现。
- 将 响应结果，通过返回值传出。

分析：

```
# 创建 session 实例
session = requests.Session()

# 使用实例，调用get 发送获取验证码请求
session.get(url="http://tpshop-test.itheima.net/index.php?m=Home&c=User&a=verify&r=0.21519623710645064")
# 使用实例，调用post 发送登录请求
resp = session.post(
    url="http://tpshop-test.itheima.net/index.php?m=Home&c=User&a=do_login&t=0.7094195931397276",
    data={"username": "13012345678", "password": "123456", "verify_code": "8888"})

print("响应结果 =", resp.json())
```

不变 变化 不变 变化 不变 返回 不变 变化

封装实现：

```python
class TpshopLoginApi(object):
    # 发送验证码请求
    @classmethod
    def get_verify(cls, session):
        session.get(url="http://tpshop-test.itheima.net/index.php?m=Home&c=User&a=verify&r=0.21519623710645064")

    # 发送登录请求
    @classmethod
    def login(cls, session, login_data):
        resp = session.post(
            url="http://tpshop-test.itheima.net/index.php?m=Home&c=User&a=do_login&t=0.7094195931397276",
            data=login_data)
        return resp
```

登录接口测试用例层

优化前：

```python
import unittest
import requests

from tpshop_login_api import TpshopLoginApi


class TestTpshopLogin(unittest.TestCase):

    # 测试 登录成功
    def test01_login_ok(self):
        # 创建 session实例
        s = requests.Session()
        # 用实例，调用自己封装的 获取验证码 接口
        TpshopLoginApi.get_verify(s)
        # 调用 自己封装的接口，登录
        abc = {"username": "13012345678", "password": "123456", "verify_code": "8888"}
        resp = TpshopLoginApi.login(s, abc)
        print(resp.json())

        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(1, resp.json().get("status"))
        self.assertEqual("登陆成功", resp.json().get("msg"))

    # 测试 手机号不存在
```

```
    def test02_tel_not_exists(self):
        # 创建 session实例
        s = requests.Session()

        # 用实例，调用自己封装的 获取验证码 接口
        TpshopLoginApi.get_verify(s)

        # 调用 自己封装的接口，登录
        abc = {"username": "13048932745", "password": "123456", "verify_code": "8888"}
        resp = TpshopLoginApi.login(s, abc)
        print(resp.json())

        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(-1, resp.json().get("status"))
        self.assertIn("账号不存在", resp.json().get("msg"))

    # 测试 密码错误
    def test03_pwd_err(self):
        # 创建 session实例
        s = requests.Session()

        # 用实例，调用自己封装的 获取验证码 接口
        TpshopLoginApi.get_verify(s)

        # 调用 自己封装的接口，登录
        abc = {"username": "13012345678", "password": "123456789", "verify_code": "8888"}
        resp = TpshopLoginApi.login(s, abc)
        print(resp.json())

        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(-2, resp.json().get("status"))
        self.assertIn("密码错误", resp.json().get("msg"))
```

优化后：

```
import unittest
import requests

from tpshop_login_api import TpshopLoginApi

# 封装 通用 的 断言函数
def common_assert(self, resp, status_code, status, msg):
    self.assertEqual(status_code, resp.status_code)
```

```
    self.assertEqual(status, resp.json().get("status"))
    self.assertIn(msg, resp.json().get("msg"))

class TestTpshopLogin(unittest.TestCase):
    # 添加类属性
    session = None

    @classmethod
    def setUpClass(cls) -> None:  # 在 类中 所 有 测试方法执行之前，自动执行1次。
        cls.session = requests.Session()

    def setUp(self) -> None:      # 在 每个 测试方法执行之前，自动执行1次。
        # 调用 自己封装的接口，获取验证码
        TpshopLoginApi.get_verify(self.session)

    # 测试 登录成功
    def test01_login_ok(self):
        # 调用 自己封装的接口，登录
        data = {"username": "13012345678", "password": "123456", "verify_code":
"8888"}
        resp = TpshopLoginApi.login(self.session, data)
        # 断言
        common_assert(self, resp, 200, 1, "登陆成功")

    # 测试 手机号不存在
    def test02_tel_not_exists(self):
        # 调用 自己封装的接口，登录
        data = {"username": "13048392845", "password": "123456", "verify_code":
"8888"}
        resp = TpshopLoginApi.login(self.session, data)
        # 断言
        common_assert(self, resp, 200, -1, "账号不存在")

    # 测试 密码错误
    def test03_pwd_err(self):
        # 调用 自己封装的接口，登录
        data = {"username": "13012345678", "password": "123456890", "verify_code":
"8888"}
        resp = TpshopLoginApi.login(self.session, data)
        # 断言
        common_assert(self, resp, 200, -2, "密码错误")

```

封装断言方法

```
# self.assertEqual(200, resp.status_code)
# self.assertEqual(1, resp.json().get("status"))
# self.assertEqual("登陆成功", resp.json().get("msg"))

# 封装 通用 的 断言函数
def common_assert(self, resp, status_code, status, msg):
    self.assertEqual(status_code, resp.status_code)
    self.assertEqual(status, resp.json().get("status"))
    self.assertIn(msg, resp.json().get("msg"))
```

封装前

封装后

```python
# 封装 通用 的 断言函数
def common_assert(self, resp, status_code, status, msg):
    self.assertEqual(status_code, resp.status_code)
    self.assertEqual(status, resp.json().get("status"))
    self.assertIn(msg, resp.json().get("msg"))

# 调用
common_assert(self, resp, 200, 1, "登陆成功")
common_assert(self, resp, 200, -1, "账号不存在")
common_assert(self, resp, 200, -2, "密码错误")
```

封装iHRM登录

登录接口

普通方式实现

```python
import unittest
import requests

class TestIhrmLogin(unittest.TestCase):
    # 登陆成功
    def test01_login_success(self):
        resp = requests.post(url="http://ihrm-test.itheima.net/api/sys/login",
                             json={"mobile": "13800000002", "password": "123456"})
        print(resp.json())
        # 断言
```

```
        self.assertEqual(200, resp.status_code)
        self.assertEqual(True, resp.json().get("success"))
        self.assertEqual(10000, resp.json().get("code"))
        self.assertIn("操作成功", resp.json().get("message")
```

登录接口对象层

- 思路：
  - 动态变化的，写入参数
  - 固定不变，直接写到方法实现内
  - 响应结果，通过返回值 return
- 分析：

```
resp = requests.post(url="http://ihrm-test.itheima.net/api/sys/login",
                     json={"mobile": "13800000002", "password": "123456"})
print(resp.json())
```

返回 不变 不变 不变 变化

- 封装实现：

```
# 接口对象层
import requests

class IhrmLoginApi(object):
    @classmethod
    def login(cls, json_data):
        resp = requests.post(url="http://ihrm-test.itheima.net/api/sys/login",
                             json=json_data)
        return resp
```

登录接口测试用例层

```
import unittest

from ihrm_login_api import IhrmLoginApi

# 定义测试类
class TestIhrmLogin(unittest.TestCase):
    # 测试方法 - 登录成功
    def test01_login_success(self):
        # 调用 自己封装 login
        login_data = {"mobile": "13800000002", "password": "123456"}
        resp = IhrmLoginApi.login(login_data)

        print("登录成功:", resp.json())
        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(True, resp.json().get("success"))
```

```
        self.assertEqual(10000, resp.json().get("code"))
        self.assertIn("操作成功", resp.json().get("message"))

    # 测试方法 - 手机号未注册
    def test02_mobile_not_register(self):
        # 调用 自己封装 login
        login_data = {"mobile": "1384780932", "password": "123456"}
        resp = IhrmLoginApi.login(login_data)

        print("手机号未注册:", resp.json())
        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(False, resp.json().get("success"))
        self.assertEqual(20001, resp.json().get("code"))
        self.assertIn("用户名或密码错误", resp.json().get("message"))

    # 测试方法 - 密码错误
    def test03_pwd_error(self):
        # 调用 自己封装 login
        login_data = {"mobile": "13800000002", "password": "890"}
        resp = IhrmLoginApi.login(login_data)

        print("密码错误:", resp.json())
        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(False, resp.json().get("success"))
        self.assertEqual(20001, resp.json().get("code"))
        self.assertIn("用户名或密码错误", resp.json().get("message"))

    # 测试方法 - 手机号为空
    def test04_mobile_is_none(self):
        # 调用 自己封装 login
        login_data = {"mobile": None, "password": "123456"}
        resp = IhrmLoginApi.login(login_data)

        print("手机号为空:", resp.json())
        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(False, resp.json().get("success"))
        self.assertEqual(20001, resp.json().get("code"))
        self.assertIn("用户名或密码错误", resp.json().get("message"))

    # 测试方法 - 多参
    def test12_more_params(self):
        # 调用 自己封装 login
        login_data = {"mobile": "13800000002", "password": "123456", "abc":"123"}
        resp = IhrmLoginApi.login(login_data)

        print("手机号为空:", resp.json())
```

```
        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(True, resp.json().get("success"))
        self.assertEqual(10000, resp.json().get("code"))
        self.assertIn("操作成功", resp.json().get("message"))

    # 测试方法 - 无参
    def test14_none_params(self):
        # 调用 自己封装 login
        login_data = None
        resp = IhrmLoginApi.login(login_data)

        print("手机号为空:", resp.json())
        # 断言
        self.assertEqual(200, resp.status_code)
        self.assertEqual(False, resp.json().get("success"))
        self.assertEqual(99999, resp.json().get("code"))
        self.assertIn("抱歉，系统繁忙，请稍后重试", resp.json().get("message"))
```

封装断言方法

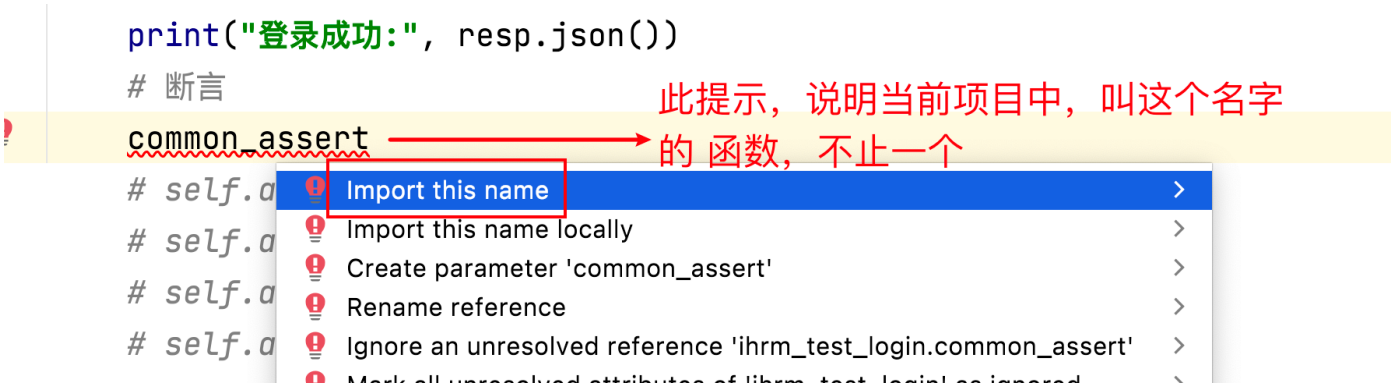
1. 创建 文件 assert_util.py
2. 在 文件内，定义 common_assert() 函数
3. 直接粘贴 unittest框架中的断言代码，修改参数。
4. 回到 unittest框架 实现的 测试脚本中， 调用该函数，实现断言，传递 实际参数。

```
# 定义 断言 函数
# def common_assert():
#     self.assertEqual(200, resp.status_code)   # self / resp 报错
#     self.assertEqual(True, resp.json().get("success"))
#     self.assertEqual(10000, resp.json().get("code"))
#     self.assertIn("操作成功", resp.json().get("message"))

def common_assert(self, resp, status_code, success, code, message):
    self.assertEqual(status_code, resp.status_code)
    self.assertEqual(success, resp.json().get("success"))
    self.assertEqual(code, resp.json().get("code"))
    self.assertIn(message, resp.json().get("message"))

# 调用演示
        common_assert(self, resp, 200, True, 10000, "操作成功")
        common_assert(self, resp, 200, False, 20001, "用户名或密码错误")
        common_assert(self, resp, 200, False, 99999, "抱歉，系统繁忙，请稍后重试")
```

导包注意事项：



Tpshop商城参数化

准备工作

分析 tpshop 商城测试用例：

```
# 测试 登录成功
def test01_login_ok(self):
    data = {"username": "13012345678", "password": "123456", "verify_code": "8888"}
    resp = TpshopLoginApi.login(self.session, data)
    common_assert(self, resp, 200, 1, "登陆成功")

# 测试 手机号不存在
def test02_tel_not_exists(self):
    data = {"username": "13048392845", "password": "123456", "verify_code": "8888"}
    resp = TpshopLoginApi.login(self.session, data)
    common_assert(self, resp, 200, -1, "账号不存在")

# 测试 密码错误
def test03_pwd_err(self):
    data = {"username": "13012345678", "password": "123456890", "verify_code": "8888"}
    resp = TpshopLoginApi.login(self.session, data)
    common_assert(self, resp, 200, -2, "密码错误")
```

只有数据不同。
可以将数据 提取，用一个 测试方法 传递不同的数据。展开接口测试

提取每个测试用例 使用的 测试数据 和 断言数据。

```
data = [
    {
        "req_body": {"username": "13012345678", "password": "123456", "verify_code": "8888"},
        "status_code": 200,
        "status": 1,
        "msg": "登陆成功"
    },
    {
        "req_body": {"username": "13048392845", "password": "123456", "verify_code": "8888"},
        "status_code": 200,
        "status": -1,
        "msg": "账号不存在"
    },
    {
        "req_body": {"username": "13012345678", "password": "123456890", "verify_code": "8888"},
        "status_code": 200,
        "status": -2,
        "msg": "密码错误"
    }
]
```

将 每个 测试用例，使用的 请求体数据、断言数据，封装到 {} 中， 有几个 测试用例，有几个 {}

封装函数，将 数据 转换为 元组列表。

```python
# 定义函数，读取 [{},{},{}] --> [(),(),()]
def read_json_data():
    list_data = []
    for item in json_data:
        tmp = tuple(item.values())
        list_data.append(tmp)

    # 循环结束后，将 list_data 为 [(),(),()] 数据， 返回
    return list_data
```

参数化步骤

1. **导包 from parameterized import parameterized**
2. **在 通用测试方法，上一行，添加 @parameterized.expand()**
3. **给 expand() 传入 [(),(),()]（调用 转换 [{},{},{}] --> [(),(),()] 的函数）**
4. **修改 通用测试方法，添加形参，个数、顺序，与 [{},{},{}] 中 { } 内的所有 key 完全一一对应。**
5. **在 通用测试方法内，使用形参。**

代码实现

```python
from parameterized import parameterized

class TestTpshopLogin(unittest.TestCase):
    # 添加类属性
    session = None

    @classmethod
    def setUpClass(cls) -> None:
        cls.session = requests.Session()

    def setUp(self) -> None:
        # 调用 自己封装的接口，获取验证码
        TpshopLoginApi.get_verify(self.session)

    # 测试 tpshop 登录
    @parameterized.expand(read_json_data())
    def test_tpshop_login(self, req_body, status_code, status, msg):
        resp = TpshopLoginApi.login(self.session, req_body)
        common_assert(self, resp, status_code, status, msg)
```

作业

1. 完成《接口测试-第07天-作业.md》，下次课抽查！
2. 复习 Python 最后阶段课程《UnitTest框架》中，“参数化” 和 “生成测试报告” 相关知识。