

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

核心思想：代码分层思想

测试脚本层

接口对象层

- 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

```
1 import unittest
2 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())

```

变化 不变

变化 不变

返回 变化 不变

封装实现:

```

1 class TpshopLoginApi(object):
2     # 发送验证码请求
3     @classmethod
4     def get_verify(cls, session):
5         session.get(url="http://tpshop-test.itheima.net/index.php?
m=Home&c=User&a=verify&r=0.21519623710645064")
6
7     # 发送登录请求
8     @classmethod
9     def login(cls, session, login_data):
10        resp = session.post(
11            url="http://tpshop-test.itheima.net/index.php?
m=Home&c=User&a=do_login&t=0.7094195931397276",
12            data=login_data)
13        return resp

```

登录接口测试用例层

优化前:

```

1 import unittest
2 import requests
3
4 from tpshop_login_api import TpshopLoginApi
5
6
7 class TestTpshopLogin(unittest.TestCase):
8
9     # 测试 登录成功
10    def test01_login_ok(self):
11        # 创建 session实例
12        s = requests.Session()
13        # 用实例, 调用自己封装的 获取验证码 接口
14        TpshopLoginApi.get_verify(s)
15        # 调用 自己封装的接口, 登录
16        abc = {"username": "13012345678", "password": "123456", "verify_code":
"8888"}
17        resp = TpshopLoginApi.login(s, abc)
18        print(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         s = requests.Session()
29
30         # 用实例，调用自己封装的 获取验证码 接口
31         TpshopLoginApi.get_verify(s)
32
33         # 调用 自己封装的接口，登录
34         abc = {"username": "13048932745", "password": "123456", "verify_code":
"8888"}
35         resp = TpshopLoginApi.login(s, abc)
36         print(resp.json())
37
38         # 断言
39         self.assertEqual(200, resp.status_code)
40         self.assertEqual(-1, resp.json().get("status"))
41         self.assertIn("账号不存在", resp.json().get("msg"))
42
43         # 测试 密码错误
44         def test03_pwd_err(self):
45             # 创建 session实例
46             s = requests.Session()
47
48             # 用实例，调用自己封装的 获取验证码 接口
49             TpshopLoginApi.get_verify(s)
50
51             # 调用 自己封装的接口，登录
52             abc = {"username": "13012345678", "password": "123456789", "verify_code":
"8888"}
53             resp = TpshopLoginApi.login(s, abc)
54             print(resp.json())
55
56             # 断言
57             self.assertEqual(200, resp.status_code)
58             self.assertEqual(-2, resp.json().get("status"))
59             self.assertIn("密码错误", resp.json().get("msg"))

```

优化后:

```

1  import unittest
2  import requests
3
4  from tpshop_login_api import TpshopLoginApi
5
6  # 封装 通用 的 断言函数
7  def common_assert(self, resp, status_code, status, msg):
8      self.assertEqual(status_code, resp.status_code)

```

```
9     self.assertEqual(status, resp.json().get("status"))
10    self.assertIn(msg, resp.json().get("msg"))
11
12    class TestTpshopLogin(unittest.TestCase):
13        # 添加类属性
14        session = None
15
16        @classmethod
17        def setUpClass(cls) -> None: # 在类中所有测试方法执行之前, 自动执行1次。
18            cls.session = requests.Session()
19
20        def setUp(self) -> None: # 在每个测试方法执行之前, 自动执行1次。
21            # 调用自己封装的接口, 获取验证码
22            TpshopLoginApi.get_verify(self.session)
23
24        # 测试 登录成功
25        def test01_login_ok(self):
26            # 调用自己封装的接口, 登录
27            data = {"username": "13012345678", "password": "123456", "verify_code":
"8888"}
28            resp = TpshopLoginApi.login(self.session, data)
29            # 断言
30            common_assert(self, resp, 200, 1, "登陆成功")
31
32        # 测试 手机号不存在
33        def test02_tel_not_exists(self):
34            # 调用自己封装的接口, 登录
35            data = {"username": "13048392845", "password": "123456", "verify_code":
"8888"}
36            resp = TpshopLoginApi.login(self.session, data)
37            # 断言
38            common_assert(self, resp, 200, -1, "账号不存在")
39
40        # 测试 密码错误
41        def test03_pwd_err(self):
42            # 调用自己封装的接口, 登录
43            data = {"username": "13012345678", "password": "123456890", "verify_code":
"8888"}
44            resp = TpshopLoginApi.login(self.session, data)
45            # 断言
46            common_assert(self, resp, 200, -2, "密码错误")
47
```

封装断言方法

封装前

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

封装后

```
# 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"))
```

```
1 # 封装 通用 的 断言函数
2 def common_assert(self, resp, status_code, status, msg):
3     self.assertEqual(status_code, resp.status_code)
4     self.assertEqual(status, resp.json().get("status"))
5     self.assertIn(msg, resp.json().get("msg"))
6
7 # 调用
8 common_assert(self, resp, 200, 1, "登陆成功")
9 common_assert(self, resp, 200, -1, "账号不存在")
10 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

```
1 import unittest
2 import requests
3
4 class TestIhrmLogin(unittest.TestCase):
5     # 登陆成功
6     def test01_login_success(self):
7         resp = requests.post(url="http://ihrm-test.itheima.net/api/sys/login",
8                               json={"mobile": "13800000002", "password": "123456"})
9         print(resp.json())
10        # 断言
```

```

11         self.assertEqual(200, resp.status_code)
12         self.assertEqual(True, resp.json().get("success"))
13         self.assertEqual(10000, resp.json().get("code"))
14         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", 不变
                        json={"mobile": "13800000002", "password": "123456"}) 不变
print(resp.json()) 变化

```

- 封装实现：

```

1  # 接口对象层
2  import requests
3
4  class IhrmLoginApi(object):
5      @classmethod
6      def login(cls, json_data):
7          resp = requests.post(url="http://ihrm-test.itheima.net/api/sys/login",
8                                json=json_data)
9          return resp

```

登录接口测试用例层

```

1  import unittest
2
3  from ihrm_login_api import IhrmLoginApi
4
5  # 定义测试类
6  class TestIhrmLogin(unittest.TestCase):
7      # 测试方法 - 登录成功
8      def test01_login_success(self):
9          # 调用 自己封装 login
10         login_data = {"mobile": "13800000002", "password": "123456"}
11         resp = IhrmLoginApi.login(login_data)
12
13         print("登录成功:", resp.json())
14         # 断言
15         self.assertEqual(200, resp.status_code)
16         self.assertEqual(True, resp.json().get("success"))

```



```
17         self.assertEqual(10000, resp.json().get("code"))
18         self.assertIn("操作成功", resp.json().get("message"))
19
20     # 测试方法 - 手机号未注册
21     def test02_mobile_not_register(self):
22         # 调用 自己封装 login
23         login_data = {"mobile": "1384780932", "password": "123456"}
24         resp = IhrmLoginApi.login(login_data)
25
26         print("手机号未注册:", resp.json())
27         # 断言
28         self.assertEqual(200, resp.status_code)
29         self.assertEqual(False, resp.json().get("success"))
30         self.assertEqual(20001, resp.json().get("code"))
31         self.assertIn("用户名或密码错误", resp.json().get("message"))
32
33     # 测试方法 - 密码错误
34     def test03_pwd_error(self):
35         # 调用 自己封装 login
36         login_data = {"mobile": "13800000002", "password": "890"}
37         resp = IhrmLoginApi.login(login_data)
38
39         print("密码错误:", resp.json())
40         # 断言
41         self.assertEqual(200, resp.status_code)
42         self.assertEqual(False, resp.json().get("success"))
43         self.assertEqual(20001, resp.json().get("code"))
44         self.assertIn("用户名或密码错误", resp.json().get("message"))
45
46     # 测试方法 - 手机号为空
47     def test04_mobile_is_none(self):
48         # 调用 自己封装 login
49         login_data = {"mobile": None, "password": "123456"}
50         resp = IhrmLoginApi.login(login_data)
51
52         print("手机号为空:", resp.json())
53         # 断言
54         self.assertEqual(200, resp.status_code)
55         self.assertEqual(False, resp.json().get("success"))
56         self.assertEqual(20001, resp.json().get("code"))
57         self.assertIn("用户名或密码错误", resp.json().get("message"))
58
59     # 测试方法 - 多参
60     def test12_more_params(self):
61         # 调用 自己封装 login
62         login_data = {"mobile": "13800000002", "password": "123456", "abc": "123"}
63         resp = IhrmLoginApi.login(login_data)
64
65         print("手机号为空:", resp.json())
```

```

66         # 断言
67         self.assertEqual(200, resp.status_code)
68         self.assertEqual(True, resp.json().get("success"))
69         self.assertEqual(10000, resp.json().get("code"))
70         self.assertIn("操作成功", resp.json().get("message"))
71
72     # 测试方法 - 无参
73     def test14_none_params(self):
74         # 调用 自己封装 login
75         login_data = None
76         resp = IhrmLoginApi.login(login_data)
77
78         print("手机号为空:", resp.json())
79         # 断言
80         self.assertEqual(200, resp.status_code)
81         self.assertEqual(False, resp.json().get("success"))
82         self.assertEqual(99999, resp.json().get("code"))
83         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。

```

1  # 定义 断言 函数
2  # def common_assert():
3  #     self.assertEqual(200, resp.status_code)    # self / resp 报错
4  #     self.assertEqual(True, resp.json().get("success"))
5  #     self.assertEqual(10000, resp.json().get("code"))
6  #     self.assertIn("操作成功", resp.json().get("message"))
7
8  def common_assert(self, resp, status_code, success, code, message):
9      self.assertEqual(status_code, resp.status_code)
10     self.assertEqual(success, resp.json().get("success"))
11     self.assertEqual(code, resp.json().get("code"))
12     self.assertIn(message, resp.json().get("message"))
13
14 # 调用演示
15     common_assert(self, resp, 200, True, 10000, "操作成功")
16     common_assert(self, resp, 200, False, 20001, "用户名或密码错误")
17     common_assert(self, resp, 200, False, 99999, "抱歉, 系统繁忙, 请稍后重试")

```

导包注意事项:

```
print("登录成功:", resp.json())
```

```
# 断言
```

```
common_assert
```

此提示, 说明当前项目中, 叫这个名字的函数, 不止一个

```
# self.a
```

```
Import this name
```

```
# self.a
```

```
Import this name locally
```

```
# self.a
```

```
Create parameter 'common_assert'
```

```
# self.a
```

```
Rename reference
```

```
Ignore an unresolved reference 'ihrm_test_login.common_assert'
```

```
# 断言
```

```
common_assert
```

选择“import this name”后, 再从列表中选择

```
# self.as
```

```
Import from ...
```

```
# self.as
```

```
assert_util.common_assert()
```

```
# self.as
```

```
tpshop_test_login.common_assert()
```

```
# self.assertIn("操作成功", resp.json().get("message"))
```

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")
```

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

```
# 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")
```

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。

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 `[(0,0,0)]` (调用 转换 `[{} , {} , {}] --> [(0,0,0)]` 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

```
1  from parameterized import parameterized
2
3  class TestTpshopLogin(unittest.TestCase):
4      # 添加类属性
5      session = None
6
7      @classmethod
8      def setUpClass(cls) -> None:
9          cls.session = requests.Session()
10
11     def setUp(self) -> None:
12         # 调用 自己封装的接口, 获取验证码
13         TpshopLoginApi.get_verify(self.session)
14
15     # 测试 tpshop 登录
16     @parameterized.expand(read_json_data())
17     def test_tpshop_login(self, req_body, status_code, status, msg):
18         resp = TpshopLoginApi.login(self.session, req_body)
19         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。