UI 自动化 day06 随堂笔记

今日内容

- PO 设计模式的剩余内容
  - 代码封装

PO 设计模式

v1 版本

说明：通过测试执行框架 pytest，可以整合所有的同一模块的测试用例脚本，并且需要尽力符合手工测试的操作业务逻辑，最终实现执行单个测试脚本，执行同一模块的所有测试用例

v1.1

> 通过测试方法整合测试脚本

```
"""
整合多个脚本至同一个测试用例中
"""
import pytest
from time import sleep
from selenium import webdriver


class TestLogin(object):
```

```
    """登录测试类"""

    def test_account_does_not_exist(self):
        """账号不存在测试方法"""
        driver = webdriver.Chrome()
        driver.get('http://127.0.0.1/')
        driver.maximize_window()  # 窗口最大化
        driver.implicitly_wait(10)  # 隐式等待

        # 1. 点击首页的‘登录’链接，进入登录页面
        driver.find_element_by_link_text('登录').click()

        # 2. 输入一个不存在的用户名

 driver.find_element_by_id('username').send_keys('13811110001')
        # 3. 输入密码

 driver.find_element_by_id('password').send_keys('123456')
        # 4. 输入验证码

 driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值: 元素对象.text
        msg = driver.find_element_by_class_name('layui-layer-
content').text
        print('错误信息为:', msg)

        sleep(3)
        driver.quit()
```

```
    def test_wrong_password(self):
        """密码错误测试方法"""
        driver = webdriver.Chrome()
        driver.get('http://127.0.0.1/')
        driver.maximize_window()  # 窗口最大化
        driver.implicitly_wait(10)  # 隐式等待

        # 1. 点击首页的‘登录’链接，进入登录页面
        driver.find_element_by_link_text('登录').click()

        # 2. 输入用户名

 driver.find_element_by_id('username').send_keys('13800001111')
        # 3. 输入错误密码
        driver.find_element_by_id('password').send_keys('error')
        # 4. 输入验证码

 driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值：元素对象.text
        msg = driver.find_element_by_class_name('layui-layer-
content').text
        print('错误信息为:', msg)

        sleep(3)
        driver.quit()


if __name__ == '__main__':
    pytest.main(['-s', 'tpshop_login_1.py'])
```

v1.2

> 通过 setup() 和 teardown() 方法优化多个脚本相同的前置后后置操作

```
"""
整合多个脚本至同一个测试用例中
"""
import pytest
from time import sleep
from selenium import webdriver


class TestLogin(object):
    """登录测试类"""

    def setup(self):
        self.driver = webdriver.Chrome()
        self.driver.get('http://127.0.0.1/')
        self.driver.maximize_window()  # 窗口最大化
        self.driver.implicitly_wait(10)  # 隐式等待

    def teardown(self):
        sleep(3)
        self.driver.quit()

    def test_account_does_not_exist(self):
        """账号不存在测试方法"""
        # driver = webdriver.Chrome()
        # driver.get('http://127.0.0.1/')
```

```
        # driver.maximize_window()  # 窗口最大化
        # driver.implicitly_wait(10)  # 隐式等待

        # 1. 点击首页的‘登录’链接，进入登录页面
        self.driver.find_element_by_link_text('登录').click()

        # 2. 输入一个不存在的用户名

 self.driver.find_element_by_id('username').send_keys('13811110001')
        # 3. 输入密码

 self.driver.find_element_by_id('password').send_keys('123456')
        # 4. 输入验证码

 self.driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        self.driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值: 元素对象.text
        sleep(2)
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
        print('错误信息为:', msg)

        # sleep(3)
        # driver.quit()

    def test_wrong_password(self):
        """密码错误测试方法"""
        # driver = webdriver.Chrome()
        # driver.get('http://127.0.0.1/')
```

```
        # driver.maximize_window()  # 窗口最大化
        # driver.implicitly_wait(10)  # 隐式等待

        # 1. 点击首页的‘登录’链接，进入登录页面
        self.driver.find_element_by_link_text('登录').click()

        # 2. 输入用户名

 self.driver.find_element_by_id('username').send_keys('13800001111')
        # 3. 输入错误密码

 self.driver.find_element_by_id('password').send_keys('error')
        # 4. 输入验证码

 self.driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        self.driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值：元素对象.text
        sleep(2)
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
        print('错误信息为:', msg)

        # sleep(3)
        # driver.quit()


if __name__ == '__main__':
    pytest.main(['-s', 'tpshop_login_2.py'])
```

v2 版本

> 通过 setup_class() 和 teardown_class() 方法使脚本的执行逻辑跟符合手工测试逻辑

```
"""
整合多个脚本至同一个测试用例中
"""
import pytest
from time import sleep
from selenium import webdriver


class TestLogin(object):
    """登录测试类"""

    def setup_class(self):
        self.driver = webdriver.Chrome()
        self.driver.get('http://127.0.0.1/')
        self.driver.maximize_window()  # 窗口最大化
        self.driver.implicitly_wait(10)  # 隐式等待

    def teardown_class(self):
        sleep(3)
        self.driver.quit()

    def setup(self):
        # self.driver = webdriver.Chrome()
        # self.driver.get('http://127.0.0.1/')
        # self.driver.maximize_window()  # 窗口最大化
        # self.driver.implicitly_wait(10)  # 隐式等待
        # 打开首页
```

```
        self.driver.get('http://127.0.0.1/')
        # 点击登录
        self.driver.find_element_by_link_text('登录').click()

    def teardown(self):
        # sleep(3)
        # self.driver.quit()
        pass

    def test_account_does_not_exist(self):
        """账号不存在测试方法"""

        # 1. 点击首页的‘登录’链接，进入登录页面
        # self.driver.find_element_by_link_text('登录').click()

        # 2. 输入一个不存在的用户名

 self.driver.find_element_by_id('username').send_keys('13811110001')
        # 3. 输入密码

 self.driver.find_element_by_id('password').send_keys('123456')
        # 4. 输入验证码

 self.driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        self.driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值: 元素对象.text
        sleep(2)
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
```

```
        print('错误信息为:', msg)

    def test_wrong_password(self):
        """密码错误测试方法"""

        # 1. 点击首页的‘登录’链接，进入登录页面
        # self.driver.find_element_by_link_text('登录').click()

        # 2. 输入用户名

 self.driver.find_element_by_id('username').send_keys('13800001111')
        # 3. 输入错误密码

 self.driver.find_element_by_id('password').send_keys('error')
        # 4. 输入验证码

 self.driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        self.driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值: 元素对象.text
        sleep(2)
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
        print('错误信息为:', msg)


if __name__ == '__main__':
    pytest.main(['-s', 'tpshop_login_3.py'])
```

方法封装套路

```
1. 确定方法的存放位置：找位置
2. 给方法起个合适名字：起名字
3. 放入要封装的代码内容：放代码
4. 确认是否需要参数和返回值：确必要
5. 调用封装好的方法使用：做调用
```

---

v3 版本

浏览器对象管理类的实现

```
"""
公共方法模块：习惯命名 utils
"""
from selenium import webdriver
from time import sleep


class DriverUtil(object):
    """浏览器对象管理类"""
    driver = None  # 浏览器对象变量初始化状态

    def get_driver(self):
        """获取浏览器对象方法"""
        # 说明：为了防止在同一次测试过程中，调用获取浏览器对象方法时，
        # 都会创建一个新的浏览器对象，因此有必要先判断对象是否存在，
不存在时在创建！
        if self.driver is None:
            self.driver = webdriver.Chrome()
            self.driver.get('http://127.0.0.1/')
```

```
            self.driver.maximize_window()  # 窗口最大化
            self.driver.implicitly_wait(10)  # 隐式等待
        return self.driver

    def quit_driver(self):
        """退出浏览器对象方法"""
        # 说明：必须保证浏览器对象存在，才能执行退出操作
        if self.driver:  # 等价于：if self.driver is not None:
            sleep(3)
            self.driver.quit()
            self.driver = None  # 保险手段：移除对象后，保留对象变量，以备下一次使用


if __name__ == '__main__':
    my_driver = DriverUtil()
    my_driver.get_driver()
    sleep(2)
    my_driver.quit_driver()
```

浏览器对象管理类的优化

```
"""
公共方法模块：习惯命名 utils
"""
from selenium import webdriver
from time import sleep


class DriverUtil(object):
    """浏览器对象管理类"""
```

```
    # 说明：对象变量只需要在类定义内部使用，因此定义为私有
    __driver = None  # 浏览器对象变量初始化状态

    @classmethod
    def get_driver(cls):
        """获取浏览器对象方法"""
        # 说明：为了防止在同一次测试过程中，调用获取浏览器对象方法时，
        # 都会创建一个新的浏览器对象，因此有必要先判断对象是否存在，
不存在时在创建！
        if cls.__driver is None:
            cls.__driver = webdriver.Chrome()
            cls.__driver.get('http://127.0.0.1/')
            cls.__driver.maximize_window()  # 窗口最大化
            cls.__driver.implicitly_wait(10)  # 隐式等待
        return cls.__driver

    @classmethod
    def quit_driver(cls):
        """退出浏览器对象方法"""
        # 说明：必须保证浏览器对象存在，才能执行退出操作
        if cls.__driver:  # 等价于：if self.driver is not None:
            sleep(3)
            cls.__driver.quit()
            cls.__driver = None  # 保险手段：移除对象后，保留对象变
量，以备下一次使用


if __name__ == '__main__':
    # my_driver = DriverUtil()
    # my_driver.get_driver()
    # sleep(2)
    # my_driver.quit_driver()
    # 说明：定义为类方法，可以直接由类对象调用，省略实例化对象步骤
```

```
    DriverUtil.get_driver()  # 获取浏览器对象
    sleep(2)
    DriverUtil.quit_driver()  # 退出浏览器对象
```

浏览器对象管理类的使用

```
"""
整合多个脚本至同一个测试用例中
"""
import pytest
from time import sleep
from selenium import webdriver

from utils import DriverUtil


class TestLogin(object):
    """登录测试类"""

    def setup_class(self):
        # self.driver = webdriver.Chrome()
        # self.driver.get('http://127.0.0.1/')
        # self.driver.maximize_window()  # 窗口最大化
        # self.driver.implicitly_wait(10)  # 隐式等待
        self.driver = DriverUtil.get_driver()  # 获取浏览器对象

    def teardown_class(self):
        # sleep(3)
        # self.driver.quit()
        DriverUtil.quit_driver()  # 退出浏览器对象
```

```
    def setup(self):
        # 打开首页
        self.driver.get('http://127.0.0.1/')
        # 点击登录
        self.driver.find_element_by_link_text('登录').click()

    def teardown(self):
        pass

    def test_account_does_not_exist(self):
        """账号不存在测试方法"""

        # 2. 输入一个不存在的用户名

 self.driver.find_element_by_id('username').send_keys('13811110001')
        # 3. 输入密码

 self.driver.find_element_by_id('password').send_keys('123456')
        # 4. 输入验证码

 self.driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        self.driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值: 元素对象.text
        sleep(2)
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
        print('错误信息为:', msg)

    def test_wrong_password(self):
```

```
        """密码错误测试方法"""

        # 2. 输入用户名

 self.driver.find_element_by_id('username').send_keys('13800001111')
        # 3. 输入错误密码

 self.driver.find_element_by_id('password').send_keys('error')
        # 4. 输入验证码

 self.driver.find_element_by_id('verify_code').send_keys('8888')
        # 5. 点击登录按钮
        self.driver.find_element_by_name('sbtbutton').click()

        # 6. 获取错误提示信息
        # 获取元素文本值: 元素对象.text
        sleep(2)
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
        print('错误信息为:', msg)


if __name__ == '__main__':
    pytest.main(['-s', 'tpshop_login_3.py'])
```

获取弹窗信息方法的封装

```
"""
公共方法模块
"""
```

```python
from selenium import webdriver
from time import sleep


def get_alert_msg():
    """获取弹窗信息方法"""
    sleep(2)
    # msg = self.driver.find_element_by_class_name('layui-layer-content').text
    # msg = driver.find_element_by_class_name('layui-layer-content').text
    msg = DriverUtil.get_driver().find_element_by_class_name('layui-layer-content').text
    return msg


class DriverUtil(object):
    """浏览器对象管理类"""
    __driver = None  # 浏览器对象变量初始化状态

    @classmethod
    def get_driver(cls):
        """获取浏览器对象方法"""
        if cls.__driver is None:
            cls.__driver = webdriver.Chrome()  # 浏览器类型
            cls.__driver.get('http://127.0.0.1/')  # 项目地址
            cls.__driver.maximize_window()  # 窗口最大化
            cls.__driver.implicitly_wait(10)  # 隐式等待
        return cls.__driver

    @classmethod
    def quit_driver(cls):
```

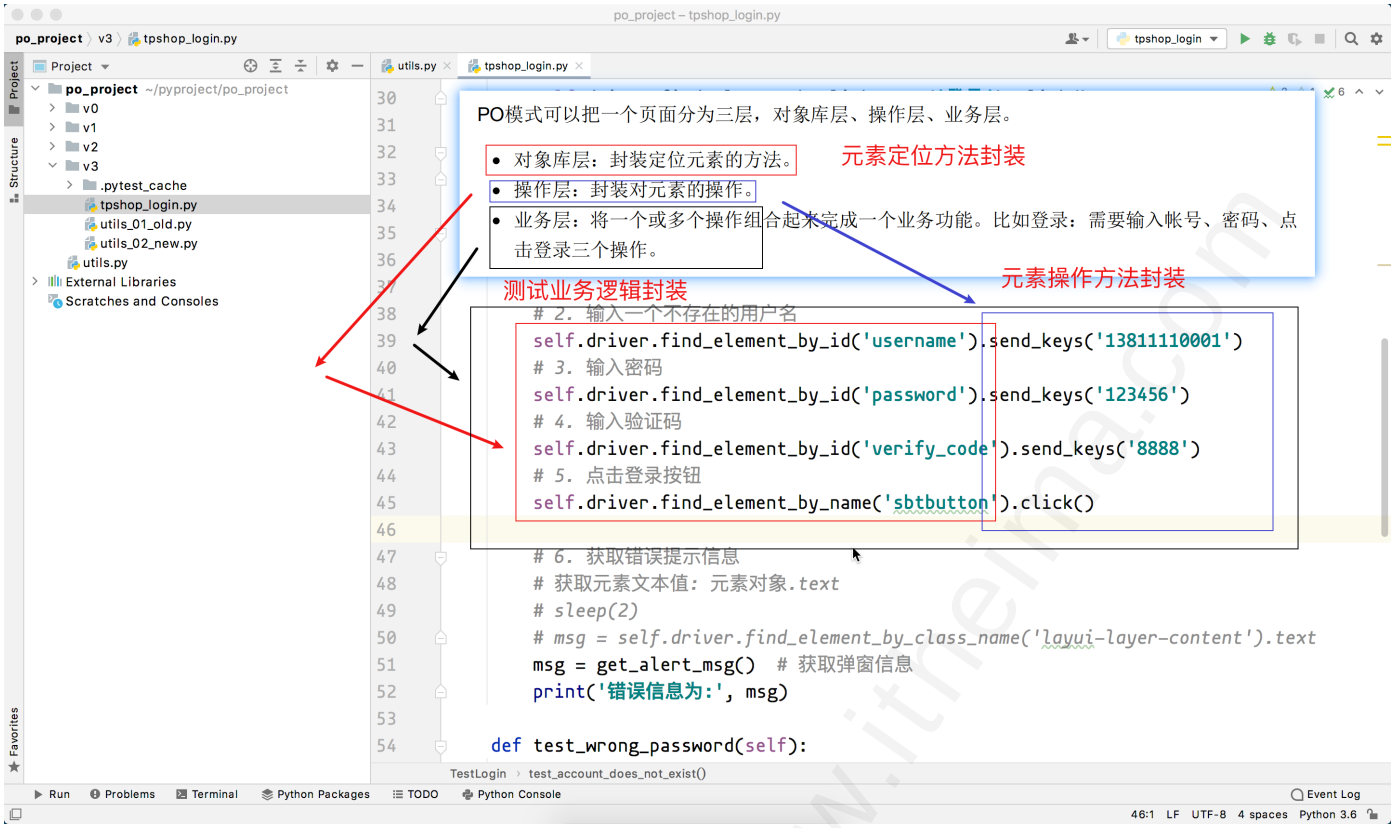
```
        """退出浏览器对象方法"""
        if cls.__driver:
            sleep(3)
            cls.__driver.quit()
            cls.__driver = None


if __name__ == '__main__':
    DriverUtil.get_driver()  # 获取浏览器对象
    sleep(2)
    DriverUtil.quit_driver()  # 退出浏览器对象
```

PO 设计模式

说明: PO 模式又可以叫 POM(P:Page O:Object), 是 UI 自动化测试中一个非常流行的设计模式(代码套路)
核心: 将元素定位及操作和业务逻辑, 拆分三个层面(每个层面对应一个单独的类), 然后通过调用完成最终的测试执行行为的过程

> 三个层面: 对象库层/操作层/业务层



v4 版本

PO 页面元素封装步骤

1. 对应页面创建页面 PO 代码文件，命名规则：页面功能_page.py，例如首页：index_page.py
2. 定义三个类：对象层(XxxPage)/操作层(XxxHandle)/业务层(XxxTask)
3. 对象层：
   1> init 方法中获取浏览器对象
   2> 自定义方法：封装元素定位方法
   3> 封装元素定位方法需要添加返回值！
4. 操作层：
   1> init 方法获取对象层对象，根据类名写对象变量名
   2> 自定义方法：封装元素操作方法
5. 业务层：
   1> init 方法获取操作层对象，根据类名写对象变量名
   2> 自定义方法：封装测试业务逻辑
6. 在测试用例文件中，实例化业务层对象，调用测试业务方法，执行测试

代码示例: 首页页面

```
"""
首页页面
"""
from utils import DriverUtil


class IndexPage(object):
    """首页对象层"""

    def __init__(self):
        self.driver = DriverUtil.get_driver()  # 获取浏览器对象

    def find_login(self):
        """定位登录方法"""
        # self.driver.find_element_by_link_text('登录')
```

```
        # self.driver = DriverUtil.get_driver()  # 获取浏览器对象
        return self.driver.find_element_by_link_text('登录')


class IndexHandle(object):
    """首页操作层"""

    def __init__(self):
        self.index_page = IndexPage()  # 获取对象层对象

    def click_login(self):
        """点击登录方法"""
        # element = IndexPage()
        # element.find_login().click()
        self.index_page.find_login().click()


class IndexTask(object):
    """首页业务层"""

    def __init__(self):
        self.index_handle = IndexHandle()  # 获取操作层对象

    def go_to_login(self):
        """跳转登录页面方法"""
        self.index_handle.click_login()
```

代码示例: 登录页面

```
"""
登录页面
"""
from utils import DriverUtil


class LoginPage(object):
    """登录对象层"""

    def __init__(self):
        self.driver = DriverUtil.get_driver()  # 获取浏览器对象

    def find_username(self):
        """定位用户名方法"""
        return self.driver.find_element_by_id('username')

    def find_password(self):
        """定位密码方法"""
        return self.driver.find_element_by_id('password')

    def find_verify_code(self):
        """定位验证码方法"""
        return self.driver.find_element_by_id('verify_code')

    def find_login_btn(self):
        """定位登录按钮方法"""
        return self.driver.find_element_by_name('sbtbutton')


class LoginHandle(object):
    """登录操作层"""
```

```
    def __init__(self):
        self.login_page = LoginPage()  # 获取对象层对象

    def input_username(self, name):
        """输入用户名方法"""
        self.login_page.find_username().send_keys(name)

    def input_password(self, pwd):
        """输入密码方法"""
        self.login_page.find_password().send_keys(pwd)

    def input_verify_code(self, code):
        """输入验证码方法"""
        self.login_page.find_verify_code().send_keys(code)

    def click_login_btn(self):
        """点击登录按钮方法"""
        self.login_page.find_login_btn().click()


class LoginTask(object):
    """登录业务层"""

    def __init__(self):
        self.login_handle = LoginHandle()  # 获取操作层对象

    def login_method(self, name, pwd, code):
        """登录方法"""
        self.login_handle.input_username(name)  # 输入用户名
        self.login_handle.input_password(pwd)  # 输入密码
        self.login_handle.input_verify_code(code)  # 输入验证码
        self.login_handle.click_login_btn()  # 点击登录按钮
```

测试用例的最终代码样式

```
"""
整合多个脚本至同一个测试用例中
"""
import pytest
from utils import DriverUtil, get_alert_msg
from v4.index_page import IndexTask
from v4.login_page import LoginTask


class TestLogin(object):
    """登录测试类"""

    def setup_class(self):
        self.driver = DriverUtil.get_driver()  # 获取浏览器对象
        self.index_task = IndexTask()  # 实例化首页业务层对象
        self.login_task = LoginTask()  # 实例化登录页面业务层对象

    def teardown_class(self):
        DriverUtil.quit_driver()  # 退出浏览器对象

    def setup(self):
        # 打开首页
        self.driver.get('http://127.0.0.1/')
        # 点击登录
        # self.driver.find_element_by_link_text('登录').click()
        self.index_task.go_to_login()  # 跳转登录

    def teardown(self):
```

```
        pass

    def test_account_does_not_exist(self):
        """账号不存在测试方法"""

        # # 2. 输入一个不存在的用户名
        #
self.driver.find_element_by_id('username').send_keys('13811110001')
        # # 3. 输入密码
        #
self.driver.find_element_by_id('password').send_keys('123456')
        # # 4. 输入验证码
        #
self.driver.find_element_by_id('verify_code').send_keys('8888')
        # # 5. 点击登录按钮
        # self.driver.find_element_by_name('sbtbutton').click()
        self.login_task.login_method('13811110001', '123456', '8888')

        # 6. 获取错误提示信息
        msg = get_alert_msg()  # 获取弹窗信息
        print('错误信息为:', msg)

    def test_wrong_password(self):
        """密码错误测试方法"""

        # # 2. 输入用户名
        #
self.driver.find_element_by_id('username').send_keys('13800001111')
        # # 3. 输入错误密码
```

```
        #
self.driver.find_element_by_id('password').send_keys('error')
        # # 4. 输入验证码
        #
self.driver.find_element_by_id('verify_code').send_keys('8888')
        # # 5. 点击登录按钮
        # self.driver.find_element_by_name('sbtbutton').click()
        self.login_task.login_method('13800001111', 'error',
'8888')

        # 6. 获取错误提示信息
        msg = get_alert_msg()  # 获取弹窗信息
        print('错误信息为:', msg)


if __name__ == '__main__':
    pytest.main(['-s', 'tpshop_login.py'])
```

---

v5 版本

PO 文件对象层代码优化

```
class LoginPage(object):
    """登录对象层"""

    def __init__(self):
        self.driver = DriverUtil.get_driver()  # 获取浏览器对象
        # 说明: 将元素的定位方法及特征值封装成属性, 能够实现集中管理
目标元素的定位方法及特征值
        self.name = (By.ID, 'username')  # 用户名
        self.pwd = (By.ID, 'password')  # 密码
```

```
        self.code = (By.ID, 'verify_code')  # 验证码
        self.btn = (By.NAME, 'sbtbutton')  # 登录按钮

    def find_username(self):
        """定位用户名方法"""
        # return self.driver.find_element_by_id('username')
        # return self.driver.find_element(By.ID, 'username')
        return self.driver.find_element(self.name[0],
self.name[1])

    def find_password(self):
        """定位密码方法"""
        # return self.driver.find_element_by_id('password')
        return self.driver.find_element(self.pwd[0],
self.pwd[1])

    def find_verify_code(self):
        """定位验证码方法"""
        # return self.driver.find_element_by_id('verify_code')
        return self.driver.find_element(self.code[0],
self.code[1])

    def find_login_btn(self):
        """定位登录按钮方法"""
        # return self.driver.find_element_by_name('sbtbutton')
        return self.driver.find_element(self.btn[0],
self.btn[1])
```

PO 文件操作层代码优化

```
class LoginHandle(object):
    """登录操作层"""

    def __init__(self):
        self.login_page = LoginPage()  # 获取对象层对象

    def input_username(self, name):
        """输入用户名方法"""
        # 说明：在执行输入操作前，应该先执行清空操作
        self.login_page.find_username().clear()
        self.login_page.find_username().send_keys(name)

    def input_password(self, pwd):
        """输入密码方法"""
        self.login_page.find_password().clear()
        self.login_page.find_password().send_keys(pwd)

    def input_verify_code(self, code):
        """输入验证码方法"""
        self.login_page.find_verify_code().clear()
        self.login_page.find_verify_code().send_keys(code)

    def click_login_btn(self):
        """点击登录按钮方法"""
        self.login_page.find_login_btn().click()
```

今日任务

- 页面 PO 代码封装练习: TPShop 商城注册页面 -> 代码截图提交
- 综合练习: 首页搜索商品 -> 商品详情页面添加购物车 -> 断言判断添加结果

1. 路径 1: 通过 PO 封装页面, 编写测试用例, 完成断言判断
2. 路径 2: 不通过 PO 封装页面, 直接定位元素操纵元素, 编写测试用例, 完成断言判断