web自动化测试框架训练营之实战集成篇

1.pytest前后置（夹具）

夹具的作用：在**用例执行之前**和**用例执行之后**，需要做的准备工作之前和收尾工作

主要用于固定测试环境，以及清理回收资源。

pytest内部提供了多种类型的夹具：

- 已经定义好的夹具类型一共有一下几种类型:
  - 函数
  - 方法
  - 类
  - 模块

自定义夹具：**fixture**

使用fixture灵活调度固定的测试环境

fixture基本介绍：

- 是pytest当中的一个装饰器
- @pytest.fixture（夹具的作用域，参数化，自动使用）

```
@pytest.fixture(scope="function", autouse=True)
def go_shop():
    print("使用自动保持登录的id")
    yield
    print("登录成功之后的用户id")
```

fixture一般在项目中进行集中管理

- 整个项目中会定义一个py模块：conftest.py专门用来放置自定义的fixture，文件名固定而且不能修改
- 在使用fixture不需要进行导包，可以直接使用，不管是在项目根目录还是用例目都会自动识别进行调用
- 执行顺序：从最外面到最里面，以及从最上面到最下面，定位fixture然后执行。

2.POM模式设计

- P：page代表页面
- O：object代表对象
- M：module代表模型
- **以页面对象为模型进行封装**和使用的模式

- 核心思想：
  - 对页面**元素**进行封装成**类的属性**

```
# 页面类属性的封装：需要被操作的元素
# - 输入账号
ipt_username = (By.XPATH,
"/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[2]/td[2]/input").send_keys(
    "admin")
# - 输入密码
ipt_password = (By.XPATH,
"/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[3]/td[2]/input").send_keys(
    "msjy123")
# - 输入验证码
ipt_code = (By.XPATH,
"/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[5]/td[2]/input").send_keys(co
de)
# - 点击登录
# input(":::")
btn_login_submit = (By.XPATH, '//*[@id="login_btn"]')
```

  - 对**用例执行流程**设计成类的**实例方法**

```
# 对用例执行流程设计成类的实例方法
def login(self,code):
    self.driver.find_element(*self.ipt_username).send_keys("admin")
    self.driver.find_element(*self.ipt_password).send_keys("msjy123")
    self.driver.find_element(*self.ipt_code).send_keys(code)
    self.driver.find_element(*self.btn_login_submit).click()
    # 获取实际结果进行断言
```

  - 通过定义好的页面类实**例化一个对象**，通过**对象调用实例方式执行用例**

```
def test_001(driver):
    # 实例化页面对象
    login_page = BackgroundLoginPage(driver)
    # 通过页面对象调用方法执行用例步骤
    login_page.login("8888")
    # 获取实际结果进行断言
```

- 核心作用：
  - 可以减少代码的冗余，而且方便后去维护，若页面元素发生变化，只需要调整页面封装的类属性即可
  - 提高用例脚本的**维护下和可持续性**

封装后台登录页面类：

```
# 封装后台登录页面类：
import time

from selenium.webdriver.common.by import By


class BackgroundLoginPage():
```

```
    # 定义一个实例属性获取驱动
    def __init__(self, driver):
        self.driver = driver

    # 页面类属性的封装：需要被操作的元素
    # - 输入账号
    ipt_username = (By.XPATH,
"/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[2]/td[2]/input")
    # - 输入密码
    ipt_password = (By.XPATH,
"/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[3]/td[2]/input")
    # - 输入验证码
    ipt_code = (By.XPATH,
"/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[5]/td[2]/input")
    # - 点击登录
    # input(":::")
    btn_login_submit = (By.XPATH, '//*[@id="login_btn"]')

    # 对用例执行流程设计成类的实例方法
    def login(self):
        self.driver.find_element(*self.ipt_username).send_keys("admin")
        self.driver.find_element(*self.ipt_password).send_keys("msjy123")
        self.driver.find_element(*self.ipt_code).send_keys("8888")
        self.driver.find_element(*self.btn_login_submit).click()
        # 获取实际结果进行断言
```

用例执行与调用：

```
from pom import BackgroundLoginPage


# 创建webdriver对象
# 获取驱动对象
def test_001(driver):
    # 实例化页面对象
    login_page = BackgroundLoginPage(driver)
    # 通过页面对象调用方法执行用例步骤
    login_page.login()
    # 获取实际结果进行断言
```

fixture固定测试环境：

- 前置：获取驱动，进入被测页面
    - 用例执行.....
- 后置：关闭驱动

```
@pytest.fixture
def driver():
    print("用例开始执行之前，获取驱动")
    driver = webdriver.Chrome()
```

```
    # 通过驱动访问页面
    driver.get("http:")

    # 将页面最大化
    driver.maximize_window()
    yield driver
    # 用例执行完成之后，关闭驱动
    print("用例执行完成之后，关闭驱动")
    driver.quit()
```

3.通过自定义fixture完成保持登录

目的：

- **被测前置：需要登录成功之后进行自动化测试用例脚本的执行**

```
@pytest.fixture(scope="session")
def admin_driver():
    print("用例开始执行之前，获取驱动")
    driver = webdriver.Chrome()

    # 通过驱动访问页面
    driver.get("http:")

    # 将页面最大化
    driver.maximize_window()
    # # 使用cookie信息
    load_cookie(driver)
    # 判断是否登录成功
    if is_login(driver):
        # 需要正常登录
        login_page = BackgroundLoginPage(driver)
        # 获取验证码图片
        login_page.save_img()
        # 识别验证码
        code = imgcode('./data/code.png')
        print(code)
        # 完成正常流程登录
        login_page.login(code)
    yield driver
    # 保存cookie信息
    sava_cookie(driver)
```

- 后置：登录成功之后执行的用例脚本：

```
def test_002(admin_driver):
    print("让fixture完成登录操作")
    print("开始执行登录之后的用例脚本")
```