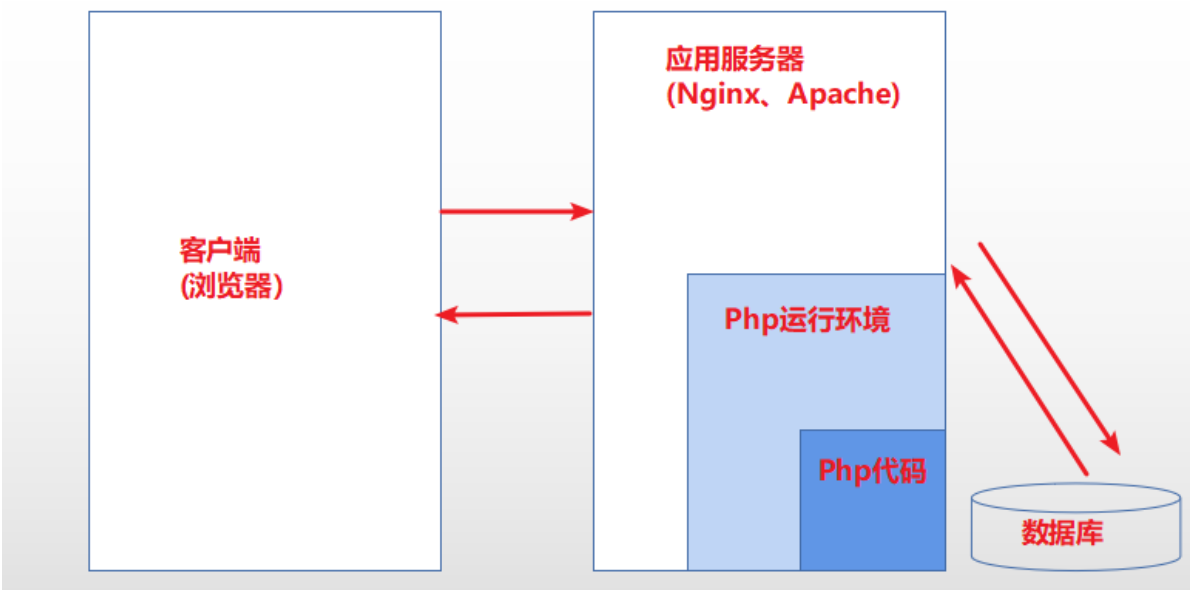
拉勾商城系统Web自动化测试

1 项目介绍及需求

项目描述

拉勾商城商城采用php语言开发的开源电商项目，目的是通过程序打造一个电子化的销售平台



编写自动化测试用例的原则

- 自动化测试用例一般只实现核心业务流程或者重复执行率较高的功能
- 自动化测试用例的选择一般以“正向”逻辑的验证为主

项目需求：

用例编号	模块	标题	优先级	前置条件	操作步骤	预期结果
1	登录	测试手机号码登录成功	P0	登录的账号：18888888888已经被注册 网络畅通	第一步：进入拉勾商城登录页面 第二步：输入账号18888888888 第三步：输入密码123456 第四步：输入图片验证码 第五步：点击登录	登录成功，页面跳转我的商城
2	登录	测试没有注册的手机号码登录	P1	登录的账号：18888999988没有被注册 网络畅通	第一步：进入拉勾商城登录页面 第二步：输入账号18888999988 第三步：输入密码123456 第四步：输入图片验证码 第五步：点击登录	登录失败，提示账号不存在！
3	登录	测试错误密码登录	P2	登录的账号：18888888888已经被注册 网络畅通	第一步：进入拉勾商城登录页面 第二步：输入手机号码18888888888 第三步：输入密码12345 第四步：输入图片验证码 第五步：点击登录	登录失败，提示密码错误！
4	登录	测试访问购物车	P1		第一步：进入首页 第二步：点击搜索框，输入电视 第三步：输入密码123456 第四步：点击搜索框 第五步：详情页点击添加购物车	弹窗提示‘添加成功’

样板代码实现

```
from selenium import webdriver

import time


class Test:
    def setup(self):
        self.driver = webdriver.Chrome()
        self.driver.get('http://localhost:8081')
        self.driver.implicitly_wait(5)


    def test_login_pass(self):
        # sbtbutton   verify_code
        self.driver.find_element_by_link_text('登录').click()
        self.driver.find_element_by_id('username').send_keys('18888888888')
        self.driver.find_element_by_id('password').send_keys('123456')
        self.driver.find_element_by_id('verify_code').send_keys('8888')
        self.driver.find_element_by_name('sbtbutton').click()
        # 获取提示信息    进行断言
        msg = self.driver.find_element_by_class_name('mu-m-phone').text
        assert msg == '18888888888',"断言提示信息，预期结果【18888888888】，实际结果
【%s】"%msg
        print('登录成功')
        self.driver.find_element_by_link_text('安全退出').click()

    # 账号不存在   layui-layer-content

    def test_account_not_exist(self):
        self.driver.find_element_by_link_text('登录').click()
        self.driver.find_element_by_id('username').send_keys('18888888889')
        self.driver.find_element_by_id('password').send_keys('123456')
        self.driver.find_element_by_id('verify_code').send_keys('8888')
        self.driver.find_element_by_name('sbtbutton').click()
        # class  定位   属性中间的空格并不是空字符串，间隔符号，class属性名称多个，我们可以
取其中一个属性，前提确保重复，
        # css    .layui-layer-content.layui-layer-padding
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
        assert msg == '账号不存在!', "断言提示信息，预期结果【账号不存在!】，实际结果
【%s】"%msg
        print('提示信息验证通过')
```

```
    #  密码错误
    def test_error_password(self):
        self.driver.find_element_by_link_text('登录').click()
        self.driver.find_element_by_id('username').send_keys('18888888888')
        self.driver.find_element_by_id('password').send_keys('12345')
        self.driver.find_element_by_id('verify_code').send_keys('8888')
        self.driver.find_element_by_name('sbtbutton').click()
        # class  定位   属性中间的空格并不是空字符串，间隔符号，class属性名称多个，我们可以
取其中一个属性，前提确保重复，
        # css    .layui-layer-content.layui-layer-padding
        msg = self.driver.find_element_by_class_name('layui-layer-content').text
        assert msg == '密码错误!', "断言提示信息，预期结果【密码错误!】，实际结果
【%s】"%msg
        print('提示信息验证通过')

    # 购物车   提示信息  添加成功        搜索按钮 ecsc-search-button        添加成功
conect-title

    def test_cart_add(self):
        self.driver.find_element_by_id('q').send_keys('电视')
        self.driver.find_element_by_class_name('ecsc-search-button').click()
        self.driver.find_element_by_class_name('p-btn').click()
        #  切换frame

self.driver.switch_to.frame(self.driver.find_element_by_tag_name('iframe'))
        # 验证 打印
        msg_cart = self.driver.find_element_by_xpath('//div[@class="conect-
title"]/span').text
        assert msg_cart == "添加成功","断言提示信息，预期结果【添加成功】，实际结果
【%s】"%msg_cart
        print('提示信息验证通过')
```

2 po模式项目搭建

2.1 编写代码

需求：

采用po模式的设计思想对页面进行封装

使用pytest管理测试用例

创建目录

项目名称：selenium_shop

安装依赖包

- 安装selenium包
- 安装pytest包
- 安装Allure包

封装工具类

- 封装po基类
- 定义page、testcase、data、image

定义页面对象文件

- 公共基类：base.py
- 首页文件：index.py
- 登录文件：login.py
- 购物车文件：shopping.py

base页

```
import os
import time

import yaml
from selenium import webdriver
from selenium.webdriver.remote.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait


class Base:
    def __init__(self,driver :WebDriver=None):
        if driver is None:
            self.driver = webdriver.Chrome()
            self.driver.get('http://www.localhost:8081')
            self.driver.implicitly_wait(5)
        else:
            self.driver=driver

# 查找元素

    def find(self,by,loc):
        return WebDriverWait(self.driver,timeout=30).until(lambda
x:self.driver.find_element(by,loc))

# 点击方法

    def click(self,by,loc):
        return self.find(by,loc).click()

# 输入方法
    def input(self,by,loc,v):
```

```
        return self.find(by,loc).send_keys(v)

# 获取文本信息

    def text(self,by,loc):
        return self.find(by,loc).text

# 截图方法
    def base_get_image(self):

self.driver.get_screenshot_as_file('../image/{}.png'.format(time.strftime("%Y_%m
_%d_%H_%M_%S")))
# 判断元素是否存在

    def base_element_is_exist(self,loc):
        try:
            self.find(loc)
            return True
        except:
            return False
```

index.py

```
from selenium.webdriver.common.by import By

from page.base import Base
from page.login import Login
from page.shopping import Shop


class Index(Base):
    def goto_login(self):
        self.click(By.LINK_TEXT,'登录')
        return Login(self.driver)
    def goto_shop(self,keyword):
        self.input(By.ID,'q',keyword)
        self.click(By.CLASS_NAME,'ecsc-search-button')
        return Shop(self.driver)
```

shopping_card.py

```
from selenium_shop.page.base import Base

from selenium.webdriver.common.by import By
import pytest
import time

class Shopping(Base):
    def shopping(self):
        self.find(By.CSS_SELECTOR, ".p-btn>a").click()
```

```
        time.sleep(5)

        # 购物车页面切换
        self.driver.switch_to.frame('layui-layer-iframe1')

        # 获取购物车添加成功提示信息

        return
self.find(By.XPATH,'//div[@id="addCartBox"]/div[1]/div/span').text
```

login.py

```
from selenium.webdriver.common.by import By

from page.base import Base

class Login(Base):
    def login(self,phone,pwd,ver):
        # self.driver.find_element_by_id('username').send_keys('18888888888')
        self.input(By.ID,'username',phone)
        # self.driver.find_element_by_id('password').send_keys('123456')
        self.input(By.ID,'password',pwd)
        # self.driver.find_element_by_id('verify_code').send_keys('8888')
        self.input(By.ID,'verify_code',ver)
        # self.driver.find_element_by_name('sbtbutton').click()
        self.click(By.NAME,'sbtbutton')

    def page_if_login_success(self):
        return self.base_element_is_exist(self.find(By.LINK_TEXT,'安全退出'))


    def page_exit(self):
        return self.driver.find_element(By.LINK_TEXT,'安全退出').click()
```

编写测试脚本

定义脚本文件 testcase.py

```
import time

import pytest
import yaml

from page.base_data import data_file
from page.index import Index

class Test_index:
    def setup_class(self):
        self.index=Index()

    #参数化
```

```
@pytest.mark.parametrize('phone,pwd,ver,status',yaml.safe_load(open('../data/dat
a.yml','r')))
    def test_login(self,phone,pwd,ver,status):
        self.index.goto_login().login(phone,pwd,ver)
        # 判断是否登录成功
        if status == 'True':
            try:

                time.sleep(3)
                assert True,self.index.goto_login().page_if_login_success()
                print('登录成功')


            except:
                self.index.base_get_image()
                time.sleep(8)

            self.index.goto_login().page_exit()



        else:
            self.index.base_get_image()

    def test_shop(self):
        self.index.goto_shop().shop()

 # 直接传参数
    @pytest.mark.parametrize('keyword',['电视','小米'])
    def test_shop(self,keyword):
        self.index.goto_shop(keyword).shop()

#
# 外部文件传参数
    @pytest.mark.parametrize('args',data_file())
    def test_shop(self,args):
        keyword = args['keyword']
        self.index.goto_shop(keyword).shop()


#
# 灵活传输
    @pytest.mark.parametrize('args',data_file('search_data.yaml','test_search'))
    def test_shop(self,args):
        keyword = args['keyword']
        self.index.goto_shop(keyword).shop()
        assert '添加成功',self.index.goto_shop(keyword).shop_add()
        print('添加成功')
```

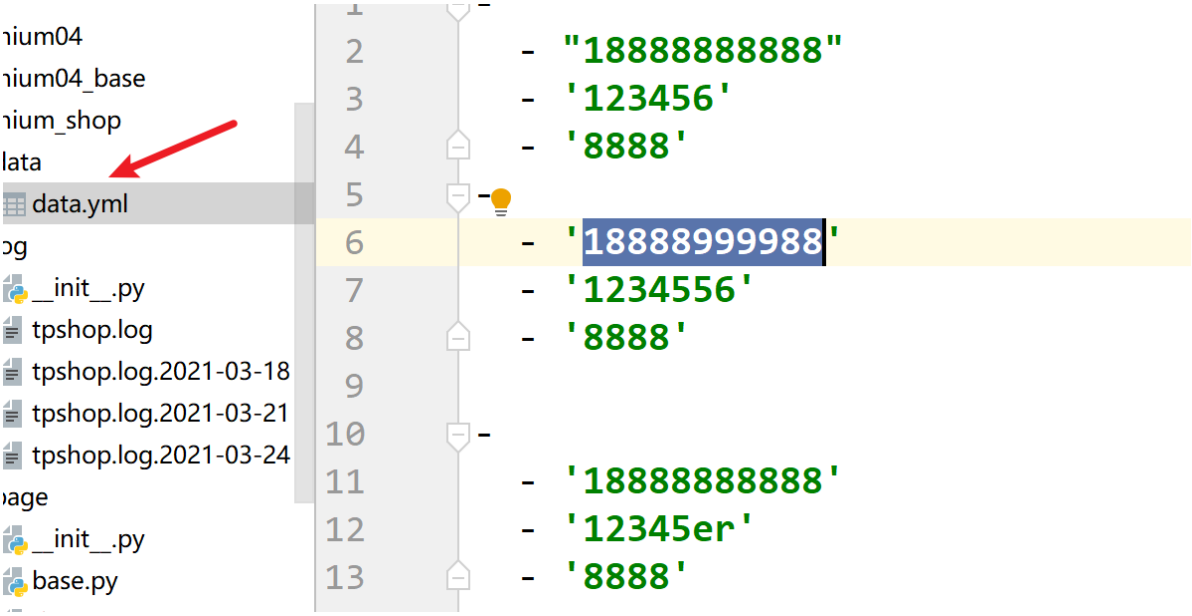
执行脚本使用pytest 命令行格式

2.2 完善代码

添加数据驱动

修改测试脚本，使用@pytest.mark.parametrize（）方法，对登录数据读取

定义数据文件　yaml



2.3 生成测试报告

2.3.1 pytest --html

r.html

Report generated on 25-Mar-2021 at 20:47:20 by pytest-html v2.1.1

Environment

JAVA_HOME	D:\developer tools\Java\jdk1.8.0_181
Packages	{"pluggy": "0.13.1", "py": "1.8.1", "pytest": "5.4.3"}
Platform	Windows-10-10.0.17134-SP0
Plugins	{"allure-pytest": "2.8.16", "html": "2.1.1", "metadata": "1.8.0", "ordering": "0.6", "rerunfailures": "9.0"}
Python	3.7.2

Summary

4 tests ran in 91.06 seconds.

(Un)check the boxes to filter the results.

3 passed, 0 skipped, 1 failed, 0 errors, 0 expected failures, 0 unexpected passes, 0 rerun

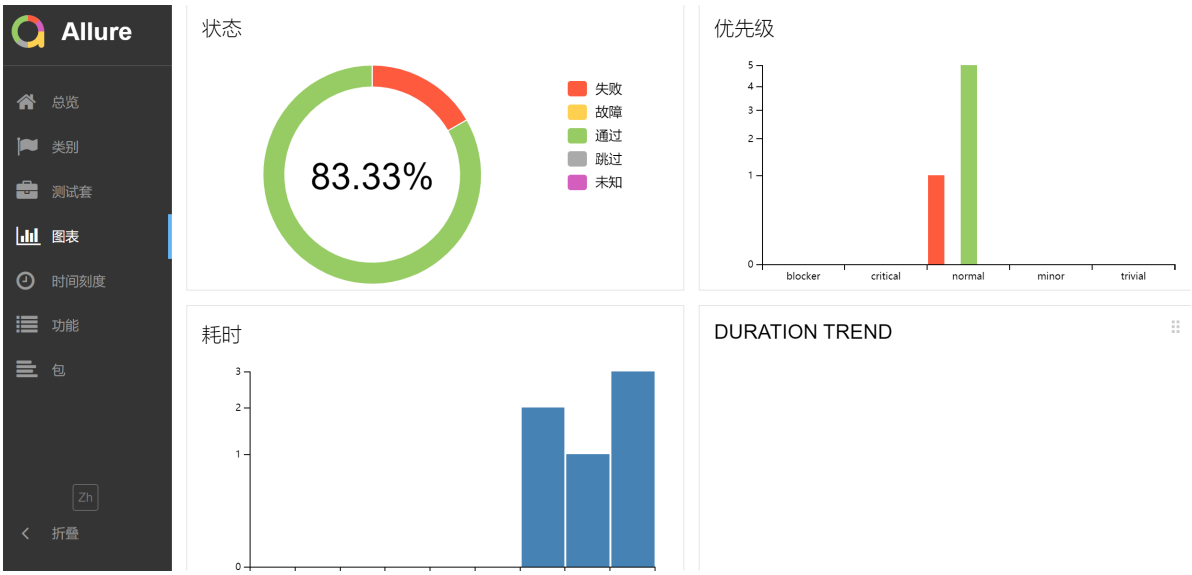
Results

Show all details / Hide all details

Result	Test	Duration
Failed *(show details)*	test_case.py::Test_Index::test_card	8.85
Passed *(show details)*	test_case.py::Test_Index::test_login[18888888888-123456-8888]	6.63
Passed *(show details)*	test_case.py::Test_Index::test_login[18888999988-1234556-8888]	7.63
Passed *(show details)*	test_case.py::Test_Index::test_login[18888888888-12345er-8888]	6.63

2.3.2 使用Allure生成测报告

报告展示图



测试用例详细信息