Table of Contents

UI自动化测试课程 1.1

数据驱动 1.2

- 数据驱动介绍 1.2.1
- 数据驱动实战一 1.2.2
- 数据驱动实战二 1.2.3

UI自动化测试课程

序号	章节	知识点
1	UI自动化测试介绍	1. UI自动化测试
2	Web自动化测试基础	1. Web自动化测试框架 2. 环境搭建 3. 元素定位和元素操作 4. 鼠标和键盘操作 5. 元素等待 6. HTML特殊元素处理 7. 验证码处理
3	移动自动化测试基础	1. 移动自动化测试框架 2. ADB调试工具 3. UIAutomatorViewer工具 4. 元素定位和元素操作 5. 滑动和拖拽事件 6. 高级手势TouchAction 7. 手机操作
4	PyTest框架	1. PyTest基本使用 2. PyTest常用插件 3. PyTest高级用法
5	PO模式	1. 方法封装 2. PO模式介绍 3. PO模式实战
6	数据驱动	1. 数据驱动介绍 2. 数据驱动实战
7	日志收集	1. 日志相关概念 2. 日志的基本方法 3. 日志的高级方法
8	黑马头条项目实战	1. 自动化测试流程 2. 项目实战演练

课程目标

1. 掌握使用Selenium实现Web自动化测试的流程和方法，并且能够完成自动化测试脚本的编写。
2. 掌握使用Appium实现移动自动化测试的流程和方法，并且能够完成自动化测试脚本的编写。
3. 掌握如何通过PyTest管理用例脚本，并使用Allure生成HTML测试报告。
4. 掌握使用PO模式来设计自动化测试代码的架构。
5. 掌握使用数据驱动来实现自动化测试代码和测试数据的分离。
6. 掌握使用logging来实现日志的收集。

数据驱动

目标

1. 理解什么是数据驱动
2. 能够使用数据驱动对代码进行优化

数据驱动介绍

目标

1. 理解数据驱动的概念

---

1. 什么是数据驱动？

数据驱动：是以数据来驱动整个测试用例的执行，也就是测试数据决定测试结果。
比如我们要测试加法，我们的测试数据是1和1，测试结果就是2，如果测试数据是1和2，测试结果就是3。

1.1 数据驱动的特点

- 数据驱动本身不是一个工业级标准的概念，因此在不同的公司都会有不同的解释。
- 可以把数据驱动理解为一种模式或者一种思想。
- 数据驱动技术可以将用户把关注点放在对测试数据的构建和维护上，而不是直接维护脚本，可以利用同样的过程对不同的数据输入进行测试。
- 数据驱动的实现要依赖参数化的技术。

1.2 传入数据的方式（测试数据的来源）

- 直接定义在测试脚本中（简单直观，但代码和数据未实现真正的分离，不方便后期维护）
- 从文件读取数据，如JSON、excel、xml、txt等格式文件
- 从数据库中读取数据
- 直接调用接口获取数据源
- 本地封装一些生成数据的方法

数据驱动实战一

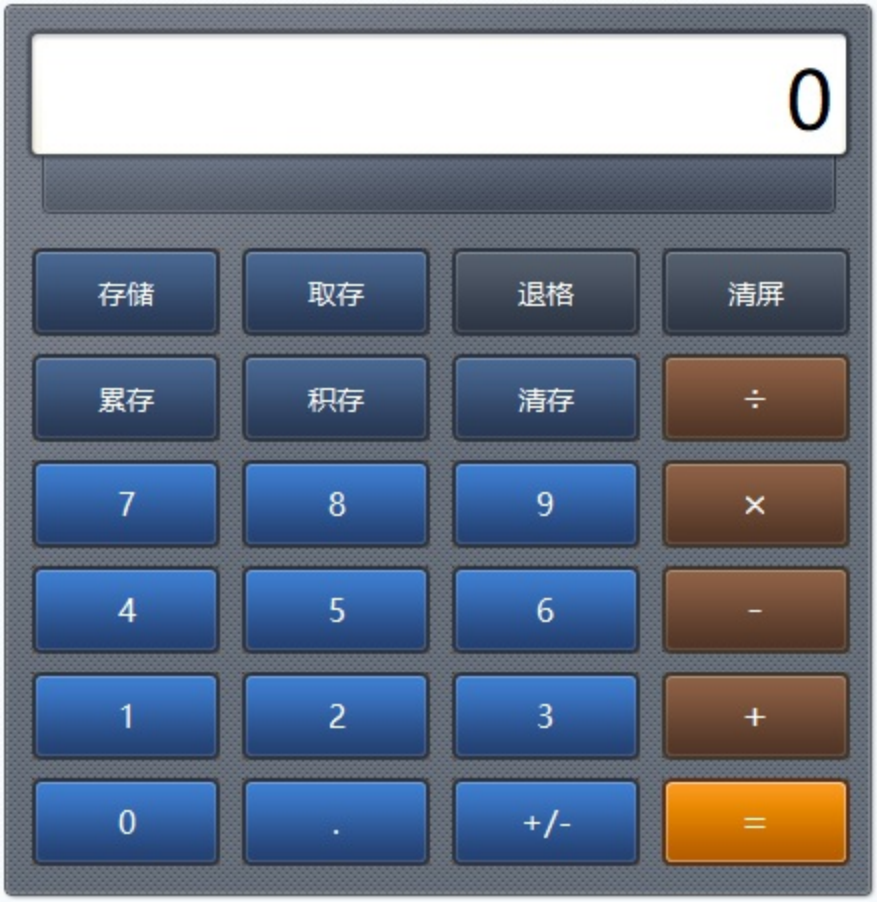
目标

1. 掌握数据驱动的开发流程
2. 掌握如何读取JSON数据文件
3. 巩固PO模式

---

1. 案例

对网页计算器，进行加法的测试操作。通过读取数据文件中的数据来执行用例。
网址：http://cal.apple886.com/



1.1 实现步骤

1. 采用PO模式的分层思想对页面进行封装
2. 编写测试脚本
3. 使用参数化传入测试数据

4. 把测试数据定义到JSON数据文件中

1.2 数据文件

第一个数字加第二个数字等于第三个数字，每一行数据代表一个用例

calculator.json

```
[
    [1, 1, 2],
    [1, 2, 3],
    [1, 0, 1],
    [0, 0, 1],
    [12, 23, 35]
]
```

1.3 示例代码

```
from selenium import webdriver


class DriverUtil:
    """
    浏览器驱动工具类
    """

    _driver = None

    @classmethod
    def get_driver(cls):
        """
        获取浏览器驱动对象，并完成初始化设置
        :return: 浏览器驱动对象
        """
        if cls._driver is None:
            cls._driver = webdriver.Chrome()
            cls._driver.maximize_window()
            cls._driver.implicitly_wait(10)
            cls._driver.get("http://cal.apple886.com/")
        return cls._driver

    @classmethod
    def quit_driver(cls):
        """
        关闭浏览器驱动
        """
        if cls._driver:
            cls._driver.quit()
```

```
            cls._driver = None
```

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

from ddt.calculator.utils import DriverUtil


class CalculatorPage:
    """
    计算器页面-对象库层
    """

    def __init__(self):
        self.driver = DriverUtil.get_driver()

        # 数字按钮
        self.digit_btn = (By.ID, "simple{}")
        # 加法按钮
        self.add_btn = (By.ID, "simpleAdd")
        # 等号按钮
        self.eq_btn = (By.ID, "simpleEqual")
        # 计算结果
        self.result = (By.ID, "resultIpt")

    def find_digit_btn(self, digit):
        location = (self.digit_btn[0], self.digit_btn[1].format(digit))
        return self.driver.find_element(*location)

    def find_add_btn(self):
        return self.driver.find_element(*self.add_btn)

    def find_eq_btn(self):
        return self.driver.find_element(*self.eq_btn)

    def find_result_btn(self):
        return self.driver.find_element(*self.result)


class CalculatorHandle:
    """
    计算器页面-操作层
    """

    def __init__(self):
        self.calculator_page = CalculatorPage()

    def click_digit_btn(self, digit):
```

```
        self.calculator_page.find_digit_btn(digit).click()

    def click_add_btn(self):
        self.calculator_page.find_add_btn().click()

    def click_eq_btn(self):
        self.calculator_page.find_eq_btn().click()

    def get_result(self):
        return self.calculator_page.find_result_btn().get_attribute("value")

    def input_numbers(self, numbers):
        for num in numbers:
            self.click_digit_btn(num)


class CalculatorProxy:
    """
    计算器页面-业务层
    """

    def __init__(self):
        self.calculator_handle = CalculatorHandle()

    def add(self, num1, num2):
        self.calculator_handle.input_numbers(str(num1))
        self.calculator_handle.click_add_btn()
        self.calculator_handle.input_numbers(str(num2))
        self.calculator_handle.click_eq_btn()

    def get_result(self):
        return self.calculator_handle.get_result()
```

```
import json
import time
import pytest

from ddt.calculator.page.calculator_page import CalculatorProxy
from ddt.calculator.utils import DriverUtil


def build_data():
    test_data = []
    with open("../data/calculator.json", encoding='UTF-8') as f:
        test_data = json.load(f)
    print("test_data=", test_data)
    return test_data
```

```python
class TestCalculator:

    def setup_class(self):
        self.driver = DriverUtil.get_driver()
        self.calculatorProxy = CalculatorProxy()

    def teardown_class(self):
        DriverUtil.quit_driver()

    @pytest.mark.parametrize("a,b,expect", build_data())
    def test_add(self, a, b, expect):
        print('a={} b={} expect={}'.format(a, b, expect))

        self.calculatorProxy.add(a, b)

        # 获取计算结果
        result = self.calculatorProxy.get_result()
        print("result=", result)
        assert result == str(expect)
```

数据驱动实战二

目标

1. 掌握数据驱动的开发流程
2. 掌握如何读取JSON数据文件
3. 巩固PO模式

---

1. 案例

对TPshop网站的登录模块进行单元测试

1.1 实现步骤

1. 编写测试用例
2. 采用PO模式的分层思想对页面进行封装
3. 编写测试脚本
4. 定义数据文件，实现参数化

1.2 用例设计

ID	模块	优先级	测试标题	预置条件	步骤描述	测试数据	预期结果	测试结果
login_001	登录	P0	用户名错误	1.打开首页 2.点击登录链接	1.输入用户名 2.输入密码 3.输入验证码 4.点击登录按钮	1.用户名：13099999999 2.密码：123456 3.验证码：8888	提示框提示： 账号不存在!	
login_002	登录	P0	密码错误	1.打开首页 2.点击登录链接	1.输入用户名 2.输入密码 3.输入验证码 4.点击登录按钮	1.用户名：13012345678 2.密码：error 3.验证码：8888	提示框提示： 密码错误!	
login_003	登录	P0	用户名为空	1.打开首页 2.点击登录链接	1.输入密码 2.输入验证码 3.点击登录按钮	1.密码：123456 2.验证码：8888	提示框提示： 用户名不能为空!	
login_004	登录	P0	密码为空	1.打开首页 2.点击登录链接	1.输入用户名 2.输入验证码 3.点击登录按钮	1.用户名：13012345678 2.验证码：8888	提示框提示： 密码不能为空!	
login_005	登录	P0	验证码为空	1.打开首页 2.点击登录链接	1.输入用户名 2.输入密码 3.点击登录按钮	1.用户名：13012345678 2.密码：123456	提示框提示： 验证码不能为空!	
login_006	登录	P0	正常登录	1.打开首页 2.点击登录链接	1.输入用户名 2.输入密码 3.输入验证码 4.点击登录按钮	1.用户名：13012345678 2.密码：123456 3.验证码：8888	登录成功，并跳转到后台管理页面	

1.3 数据文件

```
[
  {
    "desc": "用户名为空",
    "username": "",
    "password": "123456",
    "code": "8888",
    "is_success": false,
    "expect": "用户名不能为空"
  },
  {
    "desc": "密码为空",
    "username": "13012345678",
    "password": "",
    "code": "8888",
    "is_success": false,
    "expect": "密码不能为空"
  },
  {
    "desc": "密码错误",
    "username": "13012345678",
    "password": "error",
    "code": "8888",
    "is_success": false,
    "expect": "密码错误"
  },
  {
    "desc": "登录成功",
    "username": "13012345678",
    "password": "123456",
    "code": "8888",
    "is_success": true,
    "expect": "我的账户"
  }
]
```

1.4 示例代码

```
from selenium import webdriver


def get_tips_msg():
    """
    获取弹出框的提示消息
    :return: 消息文本内容
    """
    msg = DriverUtil.get_driver().find_element_by_class_name("layui-layer-content").tex
```

```
t
    return msg


class DriverUtil:
    """
    浏览器驱动工具类
    """

    _driver = None

    @classmethod
    def get_driver(cls):
        """
        获取浏览器驱动对象，并完成初始化设置
        :return: 浏览器驱动对象
        """
        if cls._driver is None:
            cls._driver = webdriver.Chrome()
            cls._driver.maximize_window()
            cls._driver.implicitly_wait(10)
            cls._driver.get("http://localhost")
        return cls._driver

    @classmethod
    def quit_driver(cls):
        """
        关闭浏览器驱动
        """
        if cls._driver:
            cls._driver.quit()
            cls._driver = None
```

```
# login_page.py

from selenium.webdriver.common.by import By

from ddt.tpshop.utils import DriverUtil


class LoginPage:
    """
    登录页面-对象库层
    """

    def __init__(self):
        self.driver = DriverUtil.get_driver()
```

```
        # 用户名输入框
        self.username = (By.ID, "username")
        # 密码
        self.password = (By.ID, "password")
        # 验证码
        self.verify_code = (By.ID, "verify_code")
        # 登录按钮
        self.login_btn = (By.NAME, "sbtbutton")

    def find_username(self):
        return self.driver.find_element(*self.username)

    def find_password(self):
        return self.driver.find_element(*self.password)

    def find_verify_code(self):
        return self.driver.find_element(*self.verify_code)

    def find_login_btn(self):
        return self.driver.find_element(*self.login_btn)


class LoginHandle:
    """
    登录页面-操作层
    """

    def __init__(self):
        self.login_page = LoginPage()

    def input_username(self, username):
        self.login_page.find_username().send_keys(username)

    def input_password(self, pwd):
        self.login_page.find_password().send_keys(pwd)

    def input_verify_code(self, code):
        self.login_page.find_verify_code().send_keys(code)

    def click_login_btn(self):
        self.login_page.find_login_btn().click()


class LoginProxy:
    """
    登录页面-业务层
    """
```

```
    def __init__(self):
        self.login_handle = LoginHandle()

    def login(self, username, password, code):
        self.login_handle.input_username(username)
        self.login_handle.input_password(password)
        self.login_handle.input_verify_code(code)
        self.login_handle.click_login_btn()
```

```
# test_login.py

import json
import time
import pytest

from ddt.tpshop import utils
from ddt.tpshop.page.login_page import LoginProxy
from ddt.tpshop.utils import DriverUtil


# 构建测试数据
def build_data():
    test_data = []
    with open("../data/login.json", encoding='UTF-8') as f:
        json_data = json.load(f)
        for case_data in json_data:
            test_data.append((case_data.get("username"),
                              case_data.get("password"),
                              case_data.get("code"),
                              case_data.get("is_success"),
                              case_data.get("expect")))
    print("test_data=", test_data)
    return test_data


class TestLogin:

    def setup_class(self):
        self.driver = DriverUtil.get_driver()
        self.login_proxy = LoginProxy()

    def teardown_class(self):
        DriverUtil.quit_driver()

    def setup(self):
        # 进入首页
        self.driver.get("http://localhost")
```

```
        # 点击登录链接
        self.driver.find_element_by_link_text("登录").click()

    @pytest.mark.parametrize("username,password,code,is_success,expect", build_data())
    def test_add(self, username, password, code, is_success, expect):
        print('username={} password={} code={} is_success={} expect={}'.
              format(username, password, code, is_success, expect))

        # 登录
        self.login_proxy.login(username, password, code)
        time.sleep(1)

        # 登录成功的用例
        if is_success:
            time.sleep(3)
            assert expect in self.driver.title
        else:
            # 获取提示框消息
            msg = utils.get_tips_msg()
            print("msg=", msg)
            assert expect in msg
```