16 从脚本到框架： PAGE OBJECT模型与分层框架

更新时间：2019-09-24 17:12:19

> 与有肝胆人共事，从无字句处读书。——周恩来

如果把搭建一套优秀的 UI 测试框架，比作我们的武功修炼的话，上一节中，我们已经开始从各门武功中选好了，自己想要使用的武器；然而无论我们是选择用刀还是用掌，除了武器外，还需要一套完整的武林秘籍，不管是选择 “降龙十八掌” 还是 “黯然销魂掌”，目标都是为了更好的武装自己。测试框架也是一样，不仅仅需要确定自己的架构体系，还需要完善框架的设计模式。

什么是设计模式?

可能测试的小伙伴们对设计模式不太了解，光听这个名词就觉得非常高大上，我在工作中听到这个词最多的地方都是在面试中。

那到底什么是设计模式呢？设计模式是一套被反复使用的、多数人知晓的、经过分类编目的、代

性。

所以换句话来说，设计模式就是别人都在用，用了都说好的一套基础方案，毕竟相比起学习别人已经练成的降龙十八掌比自创一套武林绝学 “风落掌中宝” 要容易的多。

在研发领域里，早在 1995 年就由 “四人帮” 收录总结了 23 种设计模式。但是在测试领域，这样的概念随着测试框架的发展才不断被总结、披露出来。近几年来最火的一种测试框架设计模式就是：Page Object。

PO 模式

所以的 PO 模式其实就是页面对象模式，就是呢我们对每一个页面创建一个类，并且在这个类里对当前页面的属性和操作来构建出一个模型。可以包含页面里按钮啊、输入框的一些属性，也包括点击、输入等一系列动作。比如我们来看这样一个登录页面类：

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

#创建基础类
class BasePage(object):
    #初始化
    def __init__(self, driver):
        self.base_url = 'http://XXXX.com/'
        self.driver = driver
        self.timeout = 30

#定义打开登录页面方法
def _open(self):
    url = self.base_url
    self.driver.get(url)
    self.driver.switch_to.frame('login_frame')  #切换到登录窗口的iframe

    #定义定义open方法，调用_open()进行打开
    def open(self):
        self._open()

#定位方法封装
def find_element(self,*loc):
    return self.driver.find_element(*loc)

#创建LoginPage类
```

```
    password_loc = (By.ID, "p")
    login_loc = (By.ID, "login_button")

#输入用户名
def type_username(self,username):
    self.find_element(*self.username_loc).clear()
    self.find_element(*self.username_loc).send_keys(username)

#输入密码
def type_password(self,password):
    self.find_element(*self.password_loc).send_keys(password)

#点击登录
def type_login(self):
    self.find_element(*self.login_loc).click()

#创建test_user_login()函数
def test_user_login(driver, username, password):
    """测试用户名/密码是否可以登录"""
    login_page = LoginPage(driver)
    login_page.open()
    login_page.type_username(username)
    login_page.type_password(password)
    login_page.type_login()

#创建main()函数
def main():
    driver = webdriver.Edge()
    username = '3494xxxxx'
    password = 'kemixxxx'
    test_user_login(driver, username, password)
    sleep(3)

driver.quit()

if __name__ == '__main__':
    main()
```


我们可以看出，整个登录页被我们的 Page 类切割成了多层，包括基础参数：URL、超时、driver 等。页面的元素属性：username、password 输入框和登录按钮，每个页面元素的操作以及整合出带有业务逻辑的登陆操作。而外部呢，也就是我们的 main 函数调用方法，则在更上层使用这个页面对象。

所以总结一下，PO 这种设计模式，就是把自动化测试代码以页面进行组织，将同一个页面上的

行，简单的调用和集合。如此一来，PO 模式的优势很明显：

- 当页面的元素、属性或者操作发生变化，只需要找到对应页面类中进行修改，维护更容易；
- 提高代码重用性，结构更加清晰。

当然还有一点优势，我隐去了：在面试时对于 *PO* 模式的阐述能够为你的自动化经历加分。为什么这么说呢？刚才的登陆界面很是简单，然而我们通过 PO 模式写了这么洋洋洒洒的一大段，ok，那么我们来试想一下这样一个场景：如果现在的页面中有 10 个、20 个甚至更多元素的时候，PO 模式该如何使用？

一套简单的操作，可能某个页面只需要一个点击事件就可以 hold 住的话，PO 模式会不会有浪费？一个基本的业务，如果横跨非常多的页面，又当如何？一个系统，比如电商网站，包含太多的页面，又该怎么组织？

在这些情况下，看起来清晰明了的 PO 模式会显得过于笨重，而且维护效果很难。我很多次在面试的时候，询问用 PO 模式期望撬开面试大门的同学，几乎成为了话题终结者。所以在我看来，虽然 PO 模 式看起来很高大上，但是和 JAVA 的很多设计模式类似，在实际工作中、大型项目中应用的并不广泛，即便硬要使用，也会发现效果与预期相差甚远。

PS：对于这样的热门思想，你当然可以不用它，但是不妨碍你认识了解甚至精通它。

那么，有没有更好的方式来针对上面的实际问题呢？

分层模式

这是我惯于采用的模式，被很多朋友和同学吐槽过：这也太简单、太 low 了吧。曾经有那么一段时间我也这么想，希望把这样的结构模式变得更加绚丽，赋予更突出的意义。直到有一天，和一个朋友聊到 EJB 的没落，为什么如此强大的应用规范，在几年之内就被 Hibernate、Spring 弯道超车呢？结论就是大道至简。越简单的架构设计，越容易理解的框架才能发挥出最大的威力。所以，我让框架回归了原来的样子：

我们的封装方式不拘泥于页面，也不拘泥于项目，可以更加灵活的掌握。核心思想在于将公共业务模块和非业务方法底层分离封装，再把元素、数据参数、配置变量分别做为不同类型的数据提取，以最上层的方式进行调用。

其实思路上与 PO 模式也有异曲同工之处，然而具体实现上则大不相同。在这样的模式中，维护更加有针对性：基于页面元素调整就维护 Elment 层，基于业务操作修改则操作 Task 层，用例行为变化修改 TestCase 层，测试数据优化调整 Data 层，有的放矢的进行框架的维护。

当然，篇幅所限，我们不太可能在这样一篇文章里展开来说明每一层应该如何划分，但我相信，聪明的同学可以从这样的模式中 get 到一些，想要更深入了解的话，可以到这里了解更细节的内容。

总结

设计模式就是再一次站在巨人的肩膀上，从前辈总结下来的成熟“兵法”中挑选符合你使用的场景。小时候看倚天屠龙记，印象最深的一段就是：

> 张无忌在殿上缓缓踱了一个圈子，沉思半晌，又缓缓踱了半个圈子，抬起头来，满脸喜色，叫道：“这我可全忘了，忘得干干净净的了。”张三丰道：“不坏，不坏！忘得真快，你这就请八臂神剑指教罢！”

这是张三丰教张无忌太极剑的场景。为什么要全都忘记，是因为金庸武侠中剑法的最高境界就是“无招胜有招”。所以，在我看来，设计测试框架最顶尖的状态，也是忘掉自己所学习的设计模式。这样的忘掉不是要你不学不看，而是在学习了已有的设计模式设计以后，将它们都放在自己的思维中，碰到不同的场景和问题，能够因势利导，选择不同的“打法”，甚至能够在实践中不断磨合改进，创造出属于自己的“巅峰绝学”。当然，一定不要忘记大功告成的那一天通知我！

← 15 驱动方式PK：数据驱动 VS 关键字驱动 VS 行为驱动

17 框架提升指南：从持续集成到测试数据收集 →

精选留言 4

欢迎在这里发表留言，作者筛选后可公开显示

啊哈哈略知一二

请问横跨多个页面的某些元素是否可以放到一个BasePage类中，然后各个page类都是继承自BasePage的

0 回复 2020-03-05

风落几番 回复 **啊哈哈略知一二**

在PO模型里不同页面的元素肯定是要分开的，这样才能够清晰明了啊~

回复 2020-03-06 13:48:05

风落几番 回复 **啊哈哈略知一二**

至于你说的basepage，我觉得没有太大的实际意义，每个页面再继承出去，多加一层，没什么作用啊

回复 2020-03-06 13:49:23

Python工程师

所以的 PO 模式其实就是页面对象模式 是不是应该所有?

1 回复 2020-01-31

在页面体量不大，但是改动频繁的项目上，PO和分层哪个更合适呢？

1 回复 2019-11-15

风落几番 回复 **简单随风**

我个人觉得实用性上PO不如纯分层，但是两个维护上差距不大。如果是对于某个页面的变动频繁，那肯定是PO更好，如果是全面的重写，我觉得还是分层更舒服一点~总体还是习惯问题哈，没有太明显的优劣之分

回复 2019-11-18 14:34:18

最爱pepsi 回复 **风落几番**

分层这个还是没看清楚，能用代码举例子么

回复 2020-01-31 20:46:04

风落几番 回复 **最爱pepsi**

可能这样用代码说也不是一两句能说清楚的，可以看下我写的这篇手记，http://www.imooc.com/article/38895 看看是不是能对你有点帮助哈

回复 2020-01-31 20:51:06

仲夏rww

有点深奥，看了之后觉得很管用，虽然我也没到设计框架的水平，相信自己在后面的学习中一定能够掌握。老师的课程很棒，感谢老师。

2 回复 2019-09-25

千学不如一看，千看不如一练