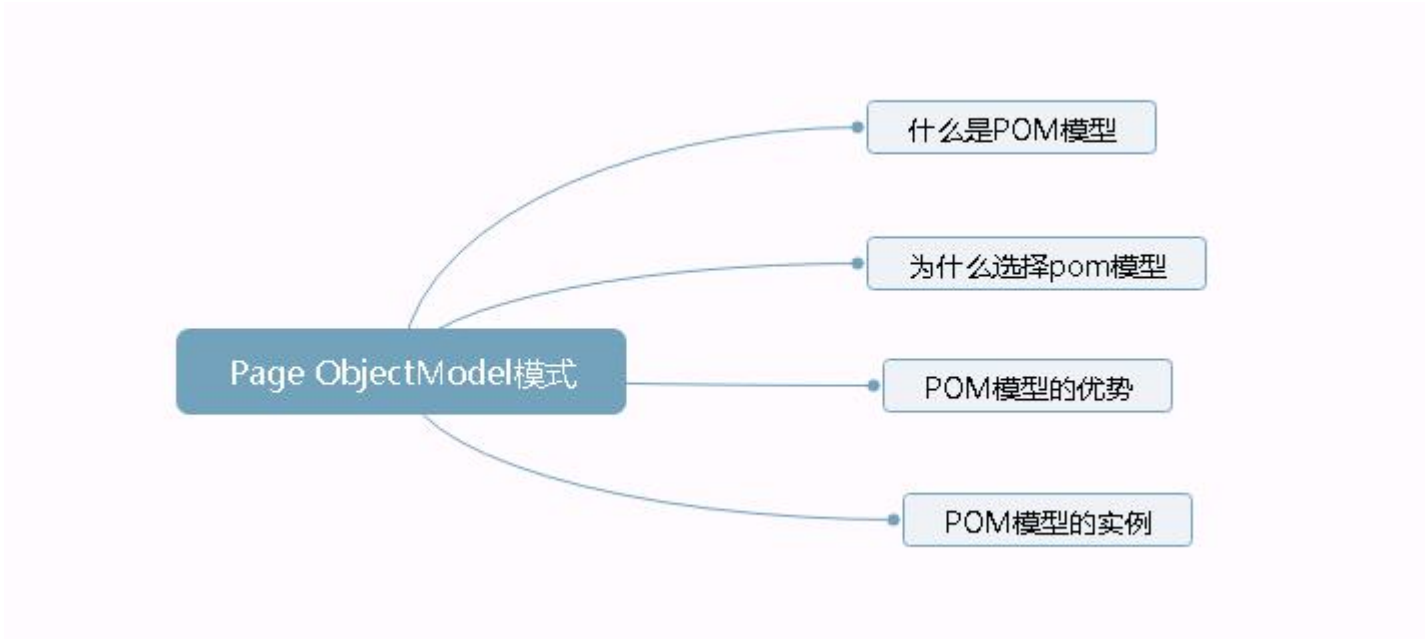
JFZ 金斧子
WWW.JFZ.COM

自动化测试框架(P0模型)

讲师 : 芳姐 | 时间 : 2018-7-12


Page ObjectModel模式
什么是POM模型
为什么选择pom模型
POM模型的优势
POM模型的实例

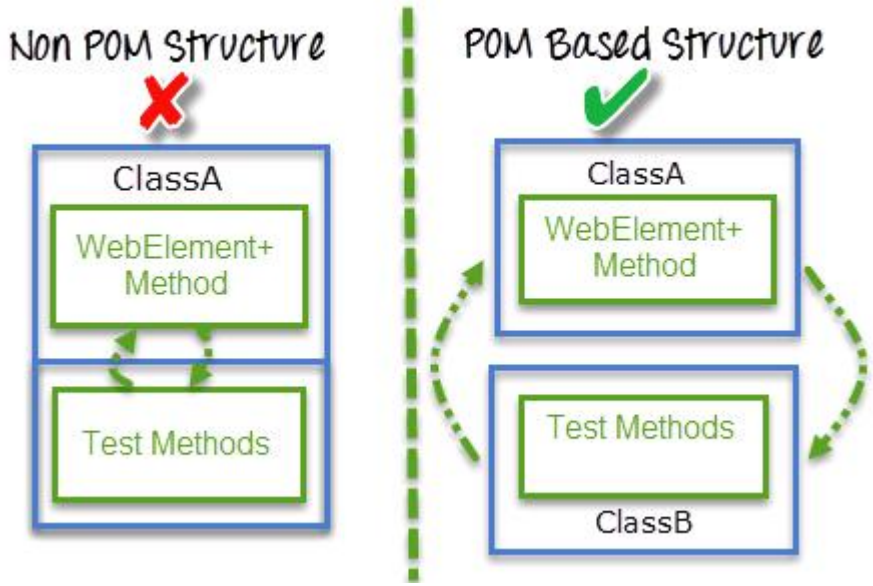
一、什么是Page Object Mode模式

Page Objects是selenium的一种测试设计模式，主要将每个页面看作是一个class。class的内容主要包括属性和方法，属性不难理解，就是这个页面中的元素对象，比如输入用户名的输入框，输入登陆密码的输入框，登陆按钮，这个页面的url等，而方法，主要是指这个页面可以提供的具体功能。

页面对象模型 是 为Web UI元素创建Object Repository的设计模式 。
在这个模型下，对于应用程序中的每个网页，应该有相应的页面类。
此Page类将会找到该Web页面的WebElements，并且还包含对这些WebElements执行操作的页面方法。

金斧子

从下图看出，采取了POM设计思路和不采取的区别，左侧把测试代码和页面元素都写在一个类文件，如果需要更改页面，那么就要修改页面元素定位，从而要修改这个类中测试代码，这个看起来和混乱。右侧，采取POM后，主要的区别就是，把页面元素和业务逻辑和测试脚本分离出来到两个不同类文件。ClassA只写页面元素定位，和业务逻辑代码操作的封装，ClassB只写测试脚本，不关心如何元素定位，只写调用ClassA的代码去覆盖不同的测试场景。如果前端页面发生变化，只需要修改ClassA的元素定位，而不需要去修改ClassB中的测试脚本代码。



二、为什么选择POM

我们先看一段代码：

```
driver = webdriver.Chrome()
driver.maximize_window()  # 最大化浏览器
driver.implicitly_wait(8)  # 设置隐式时间等待
driver.get("http://10.1.2.58:8080/login.jsp")
title = driver.title
driver.find_element_by_xpath("//*[@name='username']").send_keys("defang2")
driver.find_element_by_xpath("//*[@name='password']").send_keys("123")
driver.find_element_by_xpath("//*[@class='submit_wrap']").click()
driver.find_element_by_xpath("//*[@id='10000012200328']/a").click()
assert "深圳市金斧子网络科技有限公司-ERP" in driver.title
logging.info("标题"+ driver.title)
time.sleep(5)
assert "德芳客服" in driver.page_source
time.sleep(5)
print(driver.title)
```

这是一个简单的小脚本，脚本维护看起来很简单。但随着时间测试套件的增长。随着你在代码中添加越来越多的行，事情变得艰难。

脚本维护的主要问题是，如果10个不同的脚本使用相同的页面元素，并且该元素中的任何更改，则需要更改所有10个脚本。这是耗时且容易出错的。

更好的脚本维护方法是创建一个单独的类文件，它可以找到Web元素，填充或验证它们。该类可以在使用该元素的所有脚本中重用。将来，如果web元素有变化，我们需要在1个类文件中进行更改，而不是10个不同的脚本。

在自动化测试中，引入了Page Object Model（POM）：页面对象模式来解决，POM能让我们的测试代码变得可读性更好，高可维护性，高复用性。

金羔子

三、POM的优势

1. 把web ui对象仓库从测试脚本分离，业务代码和测试脚本分离。

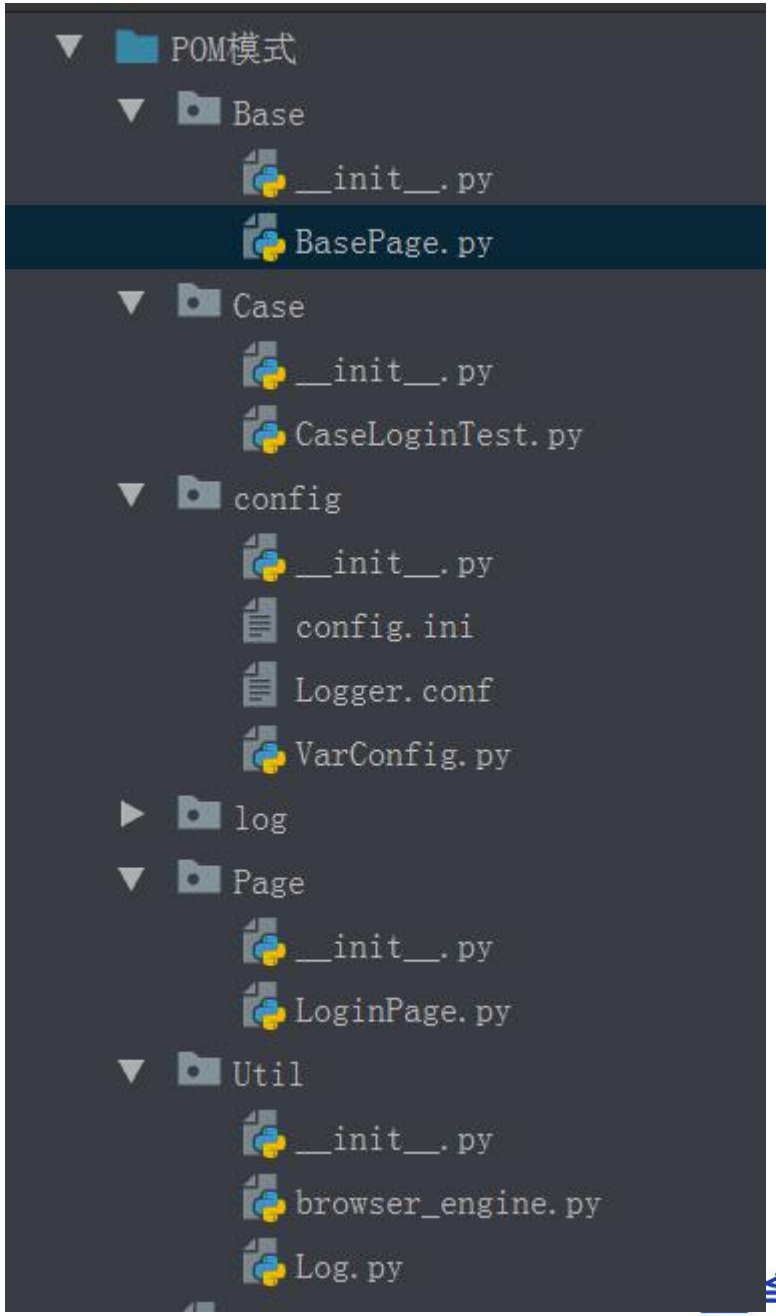
2. 每一个页面对应一个页面类，页面的元素写到这个页面类中。

3. 页面类主要包括该页面的元素定位，和和这些元素相关的业务操作代码封装的方法。

4. 代码复用，从而减少测试脚本代码量。

5. 层次清晰，同时支持多个编写自动化脚本开发，例如每个人写哪几个页面，不影响他人。

6. 建议页面类和业务逻辑方法都给一个有意义的名称，方便他人快速编写脚本和维护脚本。



四、POM的示例

代码还是上面的登录用例

那我们如何进行一个改造升级呢？

改造案例思路：

1. 在工程下面新建6个文件夹目录
   - case：存放测试用例
   - log：存放日志信息
   - page：页面对象库信息
   - config：一些配置文件
   - base：公用的基础类
   - util：公用的方法，比如日志级别设置
2. 我们要分离测试对象（元素对象）和测试脚本（用例脚本），分别创建两个脚本文件，分别为：LoginPage.py 用于定义页面元素对象，每一个元素都封装成组件（可以看做存放页面元素对象的仓库），CaseLoginTest.py 测试用例脚本。
3. 设计实现思想，一切元素和元素的操作组件化定义在Page页面，用例脚本页面，通过调用Page中的组件对象，进行拼凑成一个登录脚本。

金羚子

4. 公共方法放在BasePage.py中，定义一个页面基类，让所有页面都继承这个类，封装一些常用的页面操作方法

5. 定义一个浏览器的类，放在browser_engine.py中，封装打开浏览器初始化的一些动作

```python
class BrowserEngine(object):
    chrome_driver_path = chromeDriverFilePath
    firfox_driver_path = firefoxDriverFilePath
    ie_driver_path = ieDriverFilePath

    def __init__(self, driver):
        self.driver = driver

    def open_browser(self, driver):
        config = configparser.ConfigParser()

        file_path = os.path.dirname(os.path.abspath('.')) + '/config/config.ini'
        config.read(file_path)
        browser = config.get("browserType", "browserName")
        logger.info("选择的浏览器是:%s" % browser)
        url = config.get("testServer", "URL")
        logger.info("测试的服务器地址是:%s" % url)
```

```python
if browser == "Firefox":
    driver = webdriver.Firefox()
    logger.info("firefox 浏览器已经启动")
elif browser == "Chrome":
    driver = webdriver.Chrome()
    logger.info("Chrome 浏览器已经启动")
elif browser == "IE":
    driver = webdriver.Ie()
    logger.info("IE 浏览器已经启动")

driver.get(url)
logger.info("打开的URL地址是:%s" % url)
driver.maximize_window()
logger.info("浏览器最大化")
driver.implicitly_wait(10)
logger.info("等待10秒")
return driver
```

6. 在LoginPage.py脚本中封装登录元素对象，并集成基础类

```
class LoginPage(Action):
    user_name = "xpath=>//*[@name='username']"
    pass_word = "xpath=>//*[@name='password']"
    login_Button = "xpath=>//*[@class='login-btn']"
    def userNameObj(self, inputContent):
        try:
            self.input_string(self.user_name, inputContent)
        except Exception as e:
            raise e
    def passwordObj(self, inputContent):
        try:
            self.input_string(self.pass_word, inputContent)
        except Exception as e:
            raise e
    def loginButton(self):
        try:
            self.click(self.login_Button)
        except Exception as e:
            raise e
```

7. 在case文件夹下面新建CaseLoginTest.py脚本

```
class CaseLoginTest(unittest.TestCase):
    def setUp(self):
        browse = BrowserEngine(self)
        self.driver = browse.open_browser(self)

    def test_login1(self):

        loginpage = LoginPage(self.driver)

        try:

            # 找到用户名和密码输入框，并输入测试数据
            loginpage.userNameObj("defang1")
            loginpage.passwordObj("123")

            # 找到登录按钮，并单击
            loginpage.loginButton()
            loginpage.sleep(3)
            # 断言期望结果是否出现在页面源代码中
            loginpage.assert_string_in_pagesource("德芳理财")
        except NoSuchElementException as e:...
```

8. 引入日志

1、在config文件夹下面新建Logger.conf文件，定义日志的模板信息，以及日志的文件位置

2、在util文件夹下面新建Log.py文件，调用日志的配置文件(Logger.conf)

```
logging.config.fileConfig(parentDirPath + "\config\Logger.conf")
# 选择一个日志格式
logger = logging.getLogger("example02")
def debug(message):
    # 定义debug级别日志打印方法
    logger.debug(message)

def info(message):
    # 定义info级别日志打印方法
    logger.info(message)
def warning(message):
    # 定义warning级别日志打印方法
    logger.warning(message)
```

9. 在需要打印日志的地方加入日志代码

```
def assert_string_in_pagesource(self, assertString):
    # 断言页面源码是否存在某关键字或关键字字符串

    try:
        assert assertString in self.driver.page_source, "%s not found in page source!" % assertString
        logger.info("查找预期结果成功，%s" % assertString)
    except AssertionError as e:
        raise AssertionError(e)
    except Exception as e:
        raise e
```

10.查看运行结果

```
Launching unittests with arguments python -m unittest E:/python/ERP_selenium UI自动化(Python)/第十课/POM模式/Ca
2018-07-11 11:05:59 browser_engine.py[line:33] INFO 测试的服务器地址是:https://qal-erp.jfz.com
2018-07-11 11:06:04 browser_engine.py[line:40] INFO Chrome 浏览器已经启动
2018-07-11 11:06:06 browser_engine.py[line:46] INFO 打开的URL地址是:https://qal-erp.jfz.com
2018-07-11 11:06:07 browser_engine.py[line:48] INFO 浏览器最大化
2018-07-11 11:06:07 browser_engine.py[line:50] INFO 等待10秒
2018-07-11 11:06:08 BasePage.py[line:60] INFO 找到了元素  成功xpath 该值为: //*[@name='username']
2018-07-11 11:06:08 BasePage.py[line:73] INFO 在输入框输入 defang1
2018-07-11 11:06:08 BasePage.py[line:60] INFO 找到了元素  成功xpath 该值为: //*[@name='password']
2018-07-11 11:06:08 BasePage.py[line:73] INFO 在输入框输入 123
2018-07-11 11:06:08 BasePage.py[line:60] INFO 找到了元素 登录 成功xpath 该值为: //*[@class='login-btn']
2018-07-11 11:06:08 BasePage.py[line:81] INFO 点击该按钮成功
2018-07-11 11:06:11 BasePage.py[line:104] INFO 等待 3 秒
2018-07-11 11:06:11 BasePage.py[line:95] INFO 查找预期结果成功, 德芳理财
2018-07-11 11:06:11 CaseLoginTest.py[line:48] INFO 登录'defang1',期望'德芳理财',成功
```

总结

- POM是selenium webdriver自动化测试实践对象库设计模式
- POM使得测试脚本更易于维护
- POM通过对象库方式进一步优化了元素、用例、数据的维护组织

金羔子

作业：

1、使用pom模型完成登录

金羚子

Thanks!

科　技　提　升　投　资　品　质

金斧子
WWW.JFZ.COM