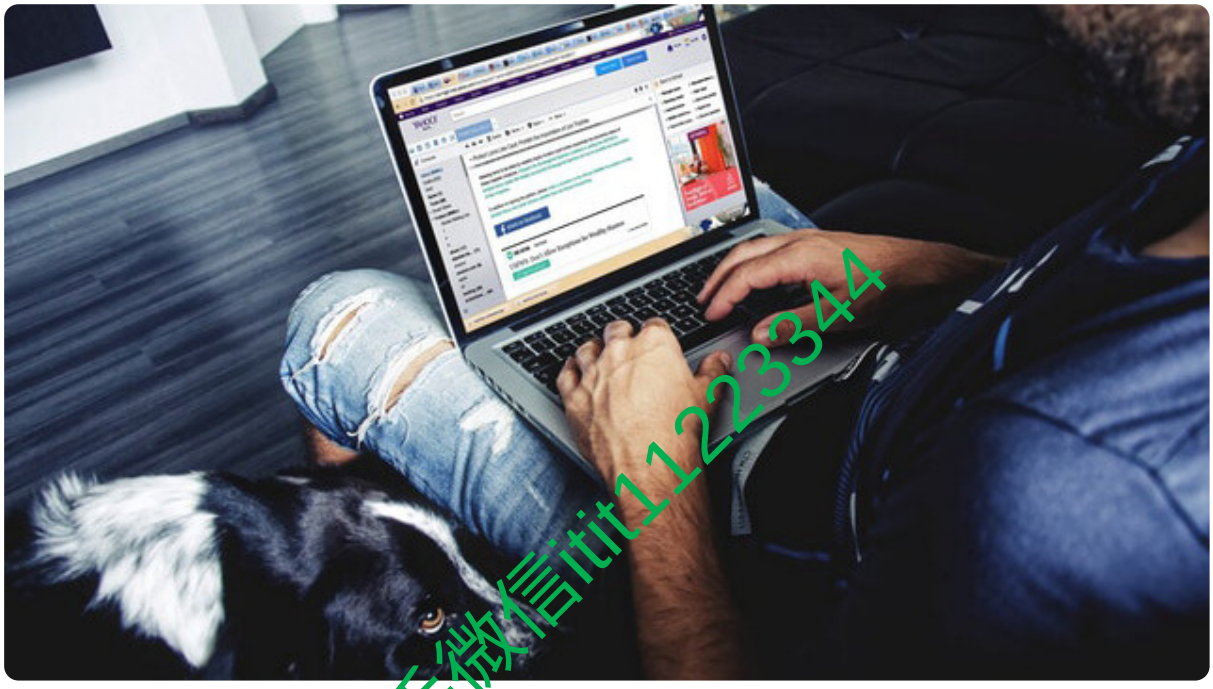
15 驱动方式PK：数据驱动 VS 关键字驱动 VS 行为驱动

更新时间：2019-09-23 10:02:26



> 虚心使人进步，骄傲使人落后。——毛泽东

上一次我们从 UI 自动化测试工具聊到了自动化测试框架，有很多同学就郁闷了：这自动化测试脚本也不是个变形金刚，不能让我们随便一吼脚本就能随时变形状的。更有同学愤怒地告诉我：我还不会写自动化测试脚本呢，风落你就让我开始写框架啊。

当然不是，不会走直接学跑，甚至学飞必然是不提倡的。所以，大家首先要学会如何编写一个完整的 Selenium 脚本，去学习如何使用 Selenium 的一系列 API，如何通过单元测试框架整合并运行起来。

```
from selenium import webdriver
import time
```

```
driver.get("http://XXXX.com")
#等10秒，浏览器打开和网页跳转需要时间
time.sleep(10)
#取ID为txtLoginCode的网页元素(用户名输入元素)
elem_user=driver.find_element_by_id('txtLoginCode')
#清空输入
elem_user.clear()
#键入用户名
elem_user.send_keys('nice_xp')
#取ID为txtPwd的网页元素(密码输入元素)
elem_pass=driver.find_element_by_id('txtPwd')
#清空输入
elem_pass.clear()
#键入密码
elem_pass.send_keys('*****')
#取ID为btnLogin的登录按钮
elem_login=driver.find_element_by_id('btnLogin')
#点击登录按钮
elem_login.click()
exit(0)
```

上边就是我们经过简单学习以后可以编写出的一个简单登录模块的自动化脚本。就像我在开篇里说的，在这个专栏里我不会带着大家去做很多，我希望能够更多给大家一些指导性的东西。所以我们接下来聊一聊把一个完整的脚本变成更好的框架，到底要做什么？

所以这才是我们今天的主旨，所谓的驱动方式其实就是一个自动化测试架构的体系，是不同自动化测试规则下来规划包括方法、数据源、识别方式、各种可重用模块在内的整套测试框架基础体系。可能这样的概念仍然有一些拗口，换个方式来说，要搭建一套自动化测试框架就好比高考考场上你拿到了一个作文题目，优秀的你可以选择用议论文、记叙文、散文、小说甚至是文言文等各种方式来完成你决定命运的 800 字，这里的不同的文体就是我们现在的驱动方式，或者我们叫设计模式，决定了我们后边该如何去开展。

数据驱动与关键字驱动

最早也是最常用的驱动方式就是数据驱动 DDT（Data-driven testing）和关键字驱动 KDT（Keyword-driven testing）。我曾经跟很多已经在做自动化的同学聊了关于这两个概念的异同，结果很多人都对此有困惑，又或者虽然理解明白，但是仍然不清晰二者的关系。

我们先来看一下数据驱动的简单定义：将脚本里的测试数据剥离出来，存储在独立于脚本之外的数据文件（XML， Excel 等）或数据库里，使脚本中的操作指令和数据分离。举个简单的例子，

数据来进行测试，包括 “风落用拳头打你头 1 下，你觉得有点疼”，“风落用拳头打你头 5 下，你觉得很疼”，“风落用拳头打你头 100 下，你觉得脑震荡了” 等等。输入的数据是结果的决定因素，可以理解为是数据驱动的测试。

再来说关键字驱动：从测试角度来说是把数据驱动再度扩张，将测试逻辑按照关键字进行分解，关键字对应封装的逻辑业务。主要关键字包括三类：被操作对象（Item）、操作（operation）和值（value）。所以按照这样的逻辑，刚才要打你的动作就可以分解为多个关键字：谁要打？用哪打？打谁？打哪儿？打几下？这样就把一个模块拆分的更加细化。所有我们关注的都变成了关键字，可以组成更多更加通用的场景。

那么让我们回到刚才登录模块的场景。在数据驱动的设计下中，就是将脚本中的用户名、密码作为变化的数据，将它提取到数据文件或数据库中，我们就可以用来模拟不同用户的登录过程，在这个测试过程中我们关注的只是数据，只是用户名密码，而对于具体这个模块里我们会先输入哪个，先点击哪个，我们并不关心，这是一个偏黑盒的过程。

而在关键字驱动的设计里，登录脚本是由不同的行为动作组成的，每个动作都会包括被操作对象（用户输入框、密码输入框、登录按钮）、操作（输入、点击）和值（用户名、密码）三者。我们修改关键字就可以改变他执行的动作、顺序，甚至整个登录脚本仅仅是关键字的实现之一，我们可以用同样的一套关键字来进行注册、个人资料修改等等，这是一个更加通用更加白盒的过程。

所以，这样我们通过比较发现：数据驱动和关键字驱动并不是毫无联系的，关键字驱动是更加细化的数据驱动，或者说我们把一切都作为数据，把对象、行为也作为了数据。所以，关键字驱动是数据驱动的细化，数据驱动是关键字驱动的高级封装。这二者之间没有谁更好，谁更优秀，而是视情况而定。

一般情况下，如果我们要使用编码的方式进行程序化编程，就像大家对于开发的了解一样，我们会更多倾向于采用数据驱动，将模块封装起来使用，它的好处在于：

1. **更容易复用**，利用我们再编写代码时模型化的设计，避免重复脚本，减少脚本的维护成本；
2. **脚本更加具备业务属性**，应用程序一旦修改，只需要修改部分业务的脚本而无需调整数据。

当然，这个缺点也很明显，仍然需要测试开发的人员对于自动化测试脚本比较精通，每个脚本也会有多个数据文件，随着项目的进行数据脚本也会越来越多。

如果说数据驱动更加偏向于模块，或者说更利于模型化结构设计的话，一旦我们想要做成一个通用的框架或者平台产品，希望我们的底层能够不受项目的约束，放之四海而皆准，这时候就需要使用关键字驱动的设计思想。

我们可以非常容易地把所有关键字数据存储到数据库或者数据文件中，对于每个动作进行精准匹配，最终产生一个优秀的测试框架或平台。优点很明显的在于自动化测试案例编写者可以不用关注底层实现、不用关注使用了什么语言而是直接按照框架的定义提供各种关键字，让零基础的人员可以快速上手；而当业务逻辑出现改动时，我们也不再需要修改代码，而只是调整关键字就可以达到维护的目标。

因为有了这些优点，所以劣势也更加明显：无法看到底层代码则很难去进行自定义，也很难修改；封装动作时的颗粒度也比较难以把握；同时数据库或者数据文件的复杂度比较高，维护起来可能也比数据驱动更复杂。

行为驱动

行为驱动是最近才开始火爆起来的概念，其实把行为驱动拿到与数据驱动、关键字驱动并列的位置上或许并不是太公平，因为行为驱动是一种新式的软件工程实践，所以这里边我们强调的是抛开代码去观察行为。什么是行为呢？就是我们希望测试人员不关注代码实现场景，用一种“自然语言”来设计测试用例、设计自动化测试脚本，这样的自然语言形式在行为驱动中被称为：剧本（Feature）。

我们可以来看一个简单的计算器的剧本示例：

```
@math
Feature: Web Selenium Test
url:https://cuketest.github.io/apps/bootstrap-calculator/

  Background: calculator
    Given open url "https://cuketest.github.io/apps/bootstrap-calculator/"
    When I click 1 + 1
    Then I click the =
    And I should  get the result "2"
    And History panel should has "1 + 1 = 2" text

  Scenario: history
    When I click the "C"
    Then History panel should be null
```

```
///// Your step definitions /////
// use this.Given(), this.When() and this.Then() to declare step definitions
let { Given, When, Then } = require('cucumber')
let { driver } = require('../support/web_driver')
let { By } = require('selenium-webdriver')
let assert = require('assert')
Given(/^open url "([^"]*)"$/, function (arg1) {
    return driver.get(arg1);
});
Then(/^I click the =$/, function () {
    return driver.findElement({ linkText: '=' }).click();
});
Then(/^I should  get the result "([^"]*)"$/, function (arg1) {
    driver.findElement({ id: 'calculator-result' }).getText().then(text => {
        return assert.deepEqual(text, arg1);
    });
});
Then(/^History panel should has "([^"]*)" text$/, function (arg1) {
    driver.findElement({ id: 'calc-history-list' }).getText().then(text => {
        return assert.deepEqual(text, arg1);
    });
});
When(/^I click the "([^"]*)"$/, function (arg1) {
    return driver.findElement({ linkText: arg1 }).click();
});

Then(/^History panel should be null$/, function () {
    return driver.findElement({ id: 'calc-history-list' }).getText().then(functio
        return assert.deepEqual(text, '');
    });
});
When(/^I click (\d+) \+ (\d+)$/, function (arg1, arg2) {
    console.log(arg1, arg2)
    driver.findElement(By.css('[data-key="49"]')).click();
    driver.findElement(By.css('[data-constant="SUM"]')).click();
    return driver.findElement(By.css('[data-key="49"]')).click();
});
```


测试开发会用到自然语言和 JavaScript，自然语言用于描述测试用例，JavaScript 用来编写测试代码。两者之间通过一定的语法关联起来。如果跟前边比起来，可以感觉到似乎这是一种特殊的关键字驱动模式，将关键字驱动中的**关键字**封装成为了剧本语言，从而可以把用户或者客户真正的通过 Feature 文件联系在一起了。各个角色，包括 QA、BA、开发、测试、客户、用户都可以通过这一媒介，将测试用例直接转化为自动化测试脚本。

同时，再扩展一步，更可以作为客户、需求与开发之间的沟通桥梁，从行为驱动测试框架发展成为行为驱动开发，从而达到更高效的沟通。这样，无论谁都可以先编写测试剧本，然后指引开发实现，最后再通过剧本执行自动化验证，这样的好处是：

- 用户故事与用例一体化，减少浪费
- 节省成本
- 减少中间的转达过程，需求更明确

总结

其实，三种驱动思想各有千秋，不存在谁好谁坏的分别，在不同的应用场景下，我们可以选择不同的驱动模式。数据驱动和关键字驱动作为架构设计的老牌劲旅，在我们设计测试框架和平台时会起到非常重要的作用，而一旦我们将自动化测试上升为一种新的软件工程实践，行为驱动这样特殊的关键字驱动模式又能够更好地为整个软件工程过程服务。

未来测试行业的弄潮儿们，你们可以从数据驱动开始尝试，不断优化完善自己的框架设计，最终全面掌握三种驱动模式。未来的测试领域，也许还有更多更新的设计模式在等着你发掘、创造，如果你有什么好的想法，欢迎跟大家一起讨论。

← 14 回溯UI自动化测试工具的前世与今生

16 从脚本到框架： PAGE OBJECT模型与分层框架 →

精选留言 4

欢迎在这里发表留言，作者筛选后可公开显示

土豆稀饭

风落老师讲的真的是太好了，很系统，很广度的让我梳理清楚了自动化这块的知识，大佬果然是大佬

2 回复 2019-10-06

风落几番 回复 **土豆稀饭**

感谢支持~~夸的我都不好意思了/(/ /•/ω/•/ /)/

慕丝8558034

打卡：数据驱动 关键字驱动 行为驱动

1 回复 2019-09-24

仲夏rww

老师，看到您有Java+selenium3.0的课程，有，没有Python+selenium3.0的课程，很想学呀~~~

1 回复 2019-09-23

风落几番 回复 **仲夏rww**

你好~python是其他老师授课的课程哈，自动化的思想其实都是一致滴~~

回复 2019-09-24 13:43:44

qq_freedom_65 回复 **仲夏rww**

mushishi有呢，但是声音没有风落好听

回复 2020-03-13 11:53:59

仲夏rww

老师讲的很好，但是我只能概念上理解，还不是很清楚，感觉很高大上，又很懵懂。体会到自己未来的路还很长，先从提升自己实力与技能开始。感恩老师的分享！

1 回复 举报 2019-09-23

风落几番 回复 **仲夏rww**

从现在做起慢慢丰满自己的翅膀哈，积累自己的知识，逐步提升自己~

回复 2019-09-24 13:44:22

仲夏rww 回复 **风落几番**

感谢老师回复，跟着老师好好学习！

回复 2019-09-24 13:50:50

千学不如一看，千看不如一练