目 录 (四十六期)

测试女巫之 Python 实战问题篇

◆ 作者：王平平

摘要：Pykeyboard 和 Pymouse，Pywinauto 此模块是使用 Python 语言，它是用于控制鼠标键盘以及控制运行在 Windows 上的程序。

一、前言：

我们已经学习了几个常见的模块，也讲解了如何应用到工作中，不知道大家是否跟着女巫真的改善了自己的工作，因为近期女巫在 51Testing 软件测试网上参加了“你来问我来答”以及“51 tesing13 周年庆”这两个活动，遇到了不少学员提出的问题，这期我就将这些问题进行汇整，产出“Python 实战问题篇”，我认为这些问题非常具有代表性，希望可以帮到大家。

二、第一类问题：基础问题（小白问题）

这类问题比较多，一般都是还未接触到 Python 不知道它为何物，它能在哪些方面帮到自己的小白提出的，这样的问题我找出几个有代表性的如下：

a.如何从黑盒测试升级到自动化测试，学习 Python 的方法？

b.你是如何从一个纯黑盒手动测试发展为一个全能测试的？

c.我想自学 Python，现在的工作中又用不到，不知道有没有什么好的学习方法？

d.你好，我从事软件测试 6 年时间了，但是绝大多数都是黑盒测试，再北大青鸟学习过编程，想使自己能有质的提升，请帮忙指点该学习哪方面的知识使自己能有足够竞争力。

上述问题我将其总结：为什么学 Python，以及如何学习 Python？

1.为什么学 Python?

1)我先回顾一下我为什么学 Python。我学习 Python 已经 5 年了，为什么学习 Python

是因为当时公司在做智能手机，希望可以对手机一些测试项目实现自动化，找到了 Monkeyrunner 这个模块，所以才进一步接触到了 Python 这个语言。

2)因为好用呗，我是 5 年前学 Python 的，那时它还没这么火，现在是越来越火，从我这几年的使用看，它的实用性非常强，对于工作中遇到的问题，我都能找到相关的模块去解决，这个真的太牛了！至于实际应用的项目，例如手机测试可以用它的 Monkeyrunner 模块，路由器项目可以用它的 selenium 模块，需要调用运行在操作系统上的软件可以用 pywinauto 模块。如果您再问，那如何具体用到这些工作中，那就请看我的 Python 系列课程吧，那里有详细的说明。

3)因为怕被淘汰,说起淘汰姐就有一把辛酸泪，我所在的公司经历了多少次针对 QA 的裁员，至少 5 次以上。我就是在一次又一次的裁员洗礼下明白了，只是做点点界面的测试真的太不安全了，一有裁员需求，第一个考虑的就是 QA。我认为未来的测试应该是以自动化为主，手动测试为辅。所以测试人员必须要学会编程，测试是一个无穷尽的工作，如何体现测试人员的价值，就是在单位时间内能有更多的产出，有让老板瞠目结舌的产出，我觉得以往的那些只凭手工测试就完成一个项目的测试，会渐渐减少，甚至消失，所以如果你是个测试人员建议现在就开始学习 Python，只要愿意学习，永远不会晚……

4)最后给大家两个应用例子来说明为什么要学 Python

Pywinauto 我们完成的两类开发项目(注意是两类，不是两个，项目的数量是很多滴):

对于工厂开发的生产工具的验证:压力测试验证->这些工具都是安装运行在 Windows 上的，所以首先使用 Spy++查找工具的窗口属性，然后将窗口控件的属性打印出来，先确认是否可以定位。一般都可以^_^，如果不可以请看下面总结的几个问题，然后就可以根据需要开始根据测试流程组织逻辑了。如下图

```
from pywinauto import application
import sys
import os
import time
from pykeyboard import PyKeyboard
from pymouse import PyMouse
m=PyMouse()
k=PyKeyboard()
l=0
time.sleep(60)
for i in range(1,101):
    app=application.Application().connect_(title_re="[redacted]-1 AllTest V1.0.4")
    dlg=app.window_(title_re="[redacted]1 AllTest V1.0.4")
    dlg['Edit3'].TypeKeys(u"0000000Z0000000")
    time.sleep(2)
    dlg.Button.Click()
    time.sleep(350)
    #dlg.print_control_identifiers()
    l=l+1
    result=dlg['Static'].WindowText()
    if result==u'NG':
       n=n+1
       f= open('e:\\aaaa\\result.log','a')
       f.write(str(l)+'time:'+str(result)+'\n')
       f.close()
       break
    else:
         m=m+1
    print l,result
    time.sleep(60)
print l,m,n
passrate=m/(m+n)*100
f= open('e:\\aaaa\\result.log','a')
f.write('Pass:'+str(m)+'\n')
f.write('Fail:'+str(n)+'\n')
f.write('Pass Rate:'+str(passrate)+'%'+'\n')
f.close()
```

b.对于硬件测试的开发：某日硬件同事需要使用 QRCT 此工具(熟悉硬件测试的人员应该很熟这个软件吧)测试产品，麻烦的是需要将产品放到高低温箱中，测试一夜，需要定期点击这个页面上的两个按钮。问题来了：谁能一夜不睡，去做这个无聊且没价值的工作？当然有：Python，而且代码就是如下 12 行，这个理由是不是很震撼？

```
# -*- coding: cp936 -*-
from pywinauto import application
import sys
import os
import time
app=application.Application().connect_(title_re="QRCT -- External Licensed")
dlg=app.window_(title_re="QRCT -- External Licensed")
time.sleep(3)
for i in range(1,100):
   dlg.Button23.Click()
   time.sleep(10)
   dlg.Button22.Click()
   time.sleep(10)
```

2.如何学习 Python?

1)首先明确 Python 是一个语言，一个脚本语言，所有的语言都是从 Hello world 开始，首先都要学习它的基本语法，变量；它的数据结构；它的各种语句写法等等，然后对于这些基本语言，写各种对应的代码，目的是帮助我们更好理解这些基础的知识。

2)问问自己学它是为了什么，目的性一定要强，我们总不会为了考试是吗，总归有相关的工作需求你才会想学 python，根据你的工作需求确认一下你需要学习什么模块，python 是有非常非常丰富的第三方模块，可以帮助你快速完成你所需要的自动化开发工作。

3)Python 有很多模块，这些模块可以帮你实现很多功能，所以千万不要尝试自己写测试库，请先尝试找相关的模块，这些模块可以让你事半功倍，当然它也有自己的架构，例如 unittest 架构等，所以千万千万不要上来就想所有的功能都要靠自己写。另外我还想

多说两句：万事开头难，刚开始学习一定会很枯燥痛苦，但是一定要坚持，撑下去，学语言一定要实践，不能看死书，多做小例子，最好把自己看过的内容，用自己的语言写出文档，这样会加深你的认识，另外学语言不能就事论事，一定要扩展，例如你看到资料上有例子，也在编辑器上写了，也正常运行了，就结束了嘛，不，你还要想，这个例子我如果这样改会怎样？通过不断改例子，会更加加深你的认识，学习起来也不会这样枯燥。

三、第二类问题：Pywinauto 实际应用中的问题（进阶的问题）

我们使用 Pywinauto 开发的项目都是其它部门嗷嗷待哺的项目，说他们嗷嗷待哺，并不是贬低他们，容我解释一下：即恨不得，这一刻提出需求，下一刻就可以稳定运行，当然对于 Python 玩的比较转的我们来说，当然没有问题（再此可以嘚瑟一下）。所以这类项目都是操作简单，但是时效性要求非常高，这种类型的项目使用 Python 实在太适合了！对了，说明一下，如果只是对一个应用程序的界面进行操作，每一次操作不需要将应用程序打开和关闭，可以手动将这个程序打开，然后就直接使用 Connect 函数定位即可，没必要使用 start 函数打开。因为什么？因为其它部门”“嗷嗷待哺”嘛。

如果你的脚本出了错误，嗯，不要乱，（出问题太正常了，问题出现就是提高你能力的机会）请回归问题的本质：想想 Pywinauto 的作用是什么？对了：定位、操作，所以厘清问题也要从这两方面入手，不要发散问题，千万不要发散问题！告诉大家一个秘密：

通过我们的实践，发现出现的问题主要在“定位”上。

一般厘清问题的方法

一定要抓本质，一定要先确认定位是否正常。简单的厘清方法如下：使用 window 函数定位：dlg=app.window_(title_re=” aaa”)然后用 dlg.close()此函数确认这个窗口是否可以被关闭，如果可以关闭就可以认为此窗口可以被正常定位。如果没有被正常定位就需要看下面的几个特定问题的厘清方法了。

特定问题的厘清方法:

1)问题 1: 如果使用 Spy++查到的 title 这个属性来定位窗口，一直出错，该怎么办？

我之前的文章，已经跟大家分享过定位应用程序以及窗口，都是使用 Spy++这个工具查看这个工具的属性：多用 Class 或者 Title 这两个属性，就可以了吗？但是有时就是不能定位，一直有错误信息跳出来，怎么办？要静下心看错误信息，才能慢慢靠近问题

本质，千万不能一出问题，不管三七二十一就到网上一通乱搜，越搜越糊涂，要相信自己可以慢慢厘清。

例如:

app=application.Application().connect_(title_re='µTorrent 3.2')执行后得到错误信息

```
Traceback (most recent call last):
  File "C:\Users\05071104\Desktop\try\utorrent.py", line 12, in <module>
    app=application.Application().connect_(title_re='µTorrent 3.2')
  File "C:\Python27\lib\site-packages\pywinauto\application.py", line 931, i
nnect_
    handle = findwindows.find_window(**kwargs)
  File "C:\Python27\lib\site-packages\pywinauto\findwindows.py", line 63, in
d_window
    raise WindowNotFoundError()
WindowNotFoundError
```

从以上错误信息有充足的理由怀疑'µ'出了问题，应该不是单纯的英文单词，因为错误信息打印出来的 µ 竟然是一个中文字前面加 u 转化为 UTF 格式试试，成功了。(如果要问我，你怎么知道要加 u，我告诉你，因为错误信息说明是编码的问题，这个正常人都能想到，然后再在网上搜有关字符串编码的问题，就很容易找到 u 这个方法)

app=application.Application().connect_(title_re=u'µTorrent 3.2')所以不管 title 是不是中文，前面统统加 u 转化格式，一般这样就不会有这样的问题出现。

2)问题 2: 在电脑 1 开发的脚本，可以正常运行，移植到电脑 2 就无法运行?

我们在开发时遇到上述问题，开发者开始一直纠结是电脑环境的问题，问题一直在发散，一直在发散，最后问题越来越复杂，搞到最后不可收拾.....，最后他告诉我，这个开发任务无法进行，这个小伙子成功在他的小主管的心里留下了：解决实际问题差的印象.....他的问题在于没有回到问题的本质：Python 的作用是定位，操作；我帮他厘清问题就首先回到确认窗口是否定位的思路上(当时他竟然还质疑他的小主管：窗口肯定定位了，因为在他的开发电脑上都是定位的好好的)：真的发现窗口没有被成功定位。开发者就很困惑，为什么在电脑 1 是好的，电脑 2 就不行。他根本没想到是定位出的问题，所以连简单的确认动作也没做！然后再网上一通乱搜，真的越搜越晕，越搜问题越发散。

使用 Spy++发现：窗口的 Title 在电脑 1 和电脑 2 确实发生了变化，所以才会出现这个问题，然后问题解决的就简单了，更改 titles 字串就可以了。

请不要问，为什么，我也不知道为什么，如果各位学员有时间，可以慢慢研究，对

于外部门嗷嗷待哺的需求，实在没时间研究为什么，首要的问题是先解决问题，后续有时间再慢慢了解为什么。

3)问题 3: 通过 start 函数自动调出来的窗口与用户手动打开的窗口不一致也就是说使用 start 函数自动调出来的窗口，后续无法定位其中的控件。

一般情况下，手动打开的窗口与通过函数自动调出的窗口是相同的，但是在我们开发的项目中，就遇到一次这样的情况：不仅手动打开的窗口与通过函数自动调出的窗口不同，而且自动调出的窗口无法定位，更无法定位相关的控件。是不是这样的情况我们就放弃开发自动化了？当然不是，换一个角度考虑问题，这个看起来无法解决的问题就能解决！

解决方式：调出 CMD 窗口使用 dos 命令将此程序调用出来，这种方式调出来的程序与手动打开程序的窗口是一样的就可以解决这个问题。首先我们通过 Pywinauto 将 dos 界面调出来，然后定位 dos 界面，在 dos 界面输入 dos 命令，这些 dos 命令就是进入这个应用程序的路径，并在此路径下运行此应用程序，这样的方式调出来的窗口就与手动打开程序的窗口一致，完美的解决了这个问题。

```
import os
import time
from pykeyboard import PyKeyboard
from pymouse import PyMouse
import cmd
m=PyMouse()
k=PyKeyboard()
appcmd=application.Application().start_('cmd.exe')
dlgcmd=appcmd.window_(class_name_re=u"ConsoleWindowClass")
time.sleep(2)
dlgcmd.TypeKeys('e:',with_spaces=True)
k.tap_key(k.enter_key)
for i in range(1,21):
    appcmd=application.Application().connect_(class_name_re=u"ConsoleWindowClass")
    dlgcmd=appcmd.window_(class_name_re=u"ConsoleWindowClass")
    dlgcmd.TypeKeys('cd E:\\a\Settingtest',with_spaces=True)
    k.tap_key(k.enter_key)
    dlgcmd.TypeKeys('"D18QB LTE Final.exe"',with_spaces=True)
    k.tap_key(k.enter_key)
    time.sleep(1)
    app=application.Application().connect_(title_re="D18QB LTE Final Test V1.0.5")
    dlg=app.window_(class_name="WindowsForms10.Window.8.app.0.378734a")
    dlg['Edit2'].TypeKeys(u"012345")
    #dlgcmd.TypeKeys('cd..',with_spaces=True)
    #k.tap_key(k.enter_key)
    dlgcmd.TypeKeys('cd E:\\a\D18Q7CALATS',with_spaces=True)
    k.tap_key(k.enter_key)
    time.sleep(1)
    dlgcmd.TypeKeys('"Calibration_Test.exe"',with_spaces=True)
    k.tap_key(k.enter_key)
    time.sleep(1)
    app=application.Application().connect_(title_re="D18 Calibration Test V1.0.7")
```


4)问题 4: 与 Pykeyboard 和 Pymouse 如何配合使用还是上面那个代码，有时我们必须调用 windows 自带的程序，我们在操作 windows 自动程序时，与操作第三方程序是不一样的，对于自动程序的控件定位更为困难，如果不相信，可以自己尝试试试看，通过 pywinauto 的方式操作控制面板或者网络连接等界面。所以就需要用到不需要定位的 Pykeyboard 和 Pymouse 两个模块辅助完成界面操作的工作。

例如对于上面的代码：对于 CMD 这样的 windows 系统自带的窗口，输入 enter_key 使用 keyboard 模块,比较方便，那我们为什么不用这个模块呢？所以在使用 pywinauto 时需要根据实际情况来决定，千万不要企图对于一个项目，只是使用 Pywinauto 这一个模块来实现操作界面的工作，是否需要使用 Pykeyboard 或者 Pymouse，关键看使用哪个模块方便。

5)问题 5: 对于“复杂控件”定位问题

例如 Button Edit 这样的控件，没法使用 Spy++找到相关参数进行定位，而是通过 Control type+Number 进行定位的。我们的 Python 系列课程已经讲解，使用 dlg.print_control_identifiers()这个函数将窗口的控件以及控件属性打印出来，找出相关控件的 Number 即可。但是在实际开发中发现，使用打印的控件 number 定位的控件不是期望定位的控件。例如下面是打印的窗口控件及控件属性，使用 Button3 无法定位到 About 按键，这时候怎么办？怎么办？凉拌，使用最原始也是最有用的方法：只能一个个的试，（我认它狠，一个界面能有一个 button，我目前的最高纪录也就是试 10 个 button number，普通的情况试个 3-4 个就能解决问题吧，所以并不会太浪费时间）例如从 Button1 开始一个个的试，这时打印的属性就可以弃之不管。这个情况我们在开发中遇到很多次，所以也见怪不怪了唉所以实践才是检验真理的唯一方法，不要迷信打印出来的属性，它有时会出错的，而且出错的频率还不低呢。

6)问题 6: 控件的属性在不同的电脑上会发生变化

例如 Button3 如果在电脑 1 上可以定位到 About 按键，但是电脑 2 就有可能无法定位到 About 按键，需要重新试，值得庆幸的是，同样的电脑，这些控件的属性是固定的(如果不是固定的真的要崩溃)

四、有关架构问题

问题：使用 python 做自动化主要使用是的哪些工具和框架？

这个问题非常好，框架是个非常好的东西，大家在搭建自己的测试体系时， 要先在网上尝试寻找有没有已经框架，因为框架这个东西是一个可以大大加快你的开发进度的非常非常好的东西，举个例子，对于自动化测试系统，都是执行一个个 Case，不可能一个 case 一个脚本，我们需要将这些脚本组织起来，而且需要执行一个大功能块的 test case，并需要有一个整体的报告产出。这个需求就需要用到 Unittest 此框架。如果需要了解详细内容可以查看博为峰网校上的“Python 系列课程”，其中有详细的讲解。

如果你想做“云”这个高大上的后端开发，就需要用到 Django 此框架，这个框架是我下半年要学习的内容，在此无法给各位更为专业的指导。

综上，你想到使用框架，就是一个非常值得赞赏的想法，因为使用框架进

行开发，就像站在巨人的肩膀上，可以的大大提高你的开发效率，至于到底需要什么框架，就要看你的工作需求了，我目前用的最多的是 unittest。

五、总结

我们这一期主要是围绕着学员以及网友的问题进行提炼，针对大家关心的问题进行总结，首先明确一个基本概念：Python 首先是一个脚本语言，所以当然需要学习语言的基本语法，无论你的实际需求是什么这些基本语法是逃不掉的，但是更为重要的是 Python 有各种模块，每个模块有自己的功能，有的模块是 Python 自带的，有的模块是第三方模块，必须自己另外安装，根据我的验，第三方模块非常重要，一个模块可以解决一个方向的自动化测试。具备了上述知识后就可以着手开发了，在开发中会遇到很多棘手的问题，有不少问题在网上是无法搜到具体的解决方法，我将我们在开发中遇到的奇怪问题总结出来，相信会对你的实际开发有所帮助。

参考文献:

Python 官网：http://www.python.org/getit/
PyUserInput 的官网 https://pypi.python.org/pypi/PyUserInput
(此网站包括如何使用，以及其中的一些常用函数的介绍)
Pywin32 下载官网:
https://sourceforge.net/projects/pywin32/files/pywin32/
Pyhook 下载官网：https://sourceforge.net/projects/pyhook/files/

面向测试人员的 WebUI 自动化测试框架-Pyswat 入门

◆作者：Lamecho

在介绍 Pyswat 框架前，我们先来看看目前行业中有关 Web 前端自动化测试的一些成熟方案都有哪些！个人来看无外乎两种形式，第一种：录制操作生成自动化回放代码（比较有代表的 QTP，selenium IDE 基于火狐插件）；第二种：提供封装好的方法，手写代码执行页面操作（如 selenium，ruby 的 watir 等）。录制的方式入手容易，但不够灵活，而通过手写程序的方式够灵活，上手却比较难。并且网上也有很多团队或个人开发的 Web 自动化测试工具，有些是对原有工具的二次封装，只是看起来语法简单了，实际问题并没很好的解决。而通过工具录制的方式生成的同样是晦涩难看的东西。测试人员赖以生存的是什么，是测试案例啊！所以说自动化测试成本高，维护难，投入产出不划算，是什么原因造成的，归根还是没有一个很好的解决方案。

那么怎么在现有的方案的基础上规划一个好的框架，一方面入手简单，另一方面定制灵活，维护不耗时，投入产出达到基本的平衡呢？这就是我们今天看到的 pyswat 做的事情。

首先，来看测试工程师在做测试时离不开的测试用例，它是测试依据，当然你要做自动化测试同样需要有这样的一份文档，通过这份用例集告诉程序该执行什么，如果你使用的自动化测试方案需要写代码去转化原有的用例文档，那么将是一个浩大的工程，需要懂代码的测试工程师一行一行的敲代码，敲几行还要运行环境调式一下，看看程序能不能跑通，遇到跑不通的地方或者报错的地方还需要花时间去找原因，这样的过程非常耗时。这也是大家认为自动化成本高的一个原因，遇到项目改版或是微调继续上面的步骤吧，等你的程序调通了项目也该上线了吧？

接下来我们看看 pyswat 框架是如何实现 web 端自动化测试的。

第一步—“录制”案例

Pyswat 的使用前需要配置一个环境运行文件“Enviroment.ini”，比如我们现在需要开始录制我们的项目，在环境配置中找到“record”的“url”选项设置为我们访问的链接地址即可，如下图

```
[record]CRLF
file = temp_case.txtCRLF
url = http://xueqiu.com/CRLF
browser = ChormeCRLF
ex-record = 1CRLF
CRLF
[2excel]CRLF
sheet = temp_caseCRLF
file = temp_case.xlsCRLF
smart_mode = 1CRLF
CRLF
[replay]CRLF
file = C:\Users\lamecho\Desktop\pyswat\雪球案例.xlsCRLF
case = 登录,物料CRLF
url = http://xueqiu.com/CRLF
browser = ChormeCRLF
think_time = 0CRLF
wait_time = 3CRLF
retry = 0CRLF
CRLF
[parameters]CRLF
password = 123456CRLF
loginname = admintestCRLF
```


配置完成后，就可以开始录制我们的操作了。运行框架程序“webs_record”将会自动打开浏览器访问链接地址，录制结束后生成一个临时的 excel 表格的案例文件“temp_case.xls”，我们打开来看看里边的内容。

	A	B	C	D	E	F	G	H
1	**stepID**	**Action**	**思考时间**	**坐标**	**属性**	**步骤**	**输入**	**参数**
2	step1	Key_Fun		pys	time_start(登录)			
3	step2	Key_Fun		pys	time_end(登录)			
4	step3	Mouse	1.71	726,379	tagName:INPUT,id:,name:username,value:,placeholder:请输入手机号或者邮箱,class:,href:,innertext:,docker:	请输入手机号或者邮箱	[redacted]	
5	step4	Mouse	2.41	650,446	tagName:INPUT,id:,name:,value:,placeholder:请输入登录密码,class:,href:,innertext:,docker:	请输入登录密码	[redacted]	
6	step5	Mouse	2.58	754,565	tagName:DIV,id:,class:modal__login__btn,innertext:,docker:	点击---->DIV		
7	stepOver	Over						
8								

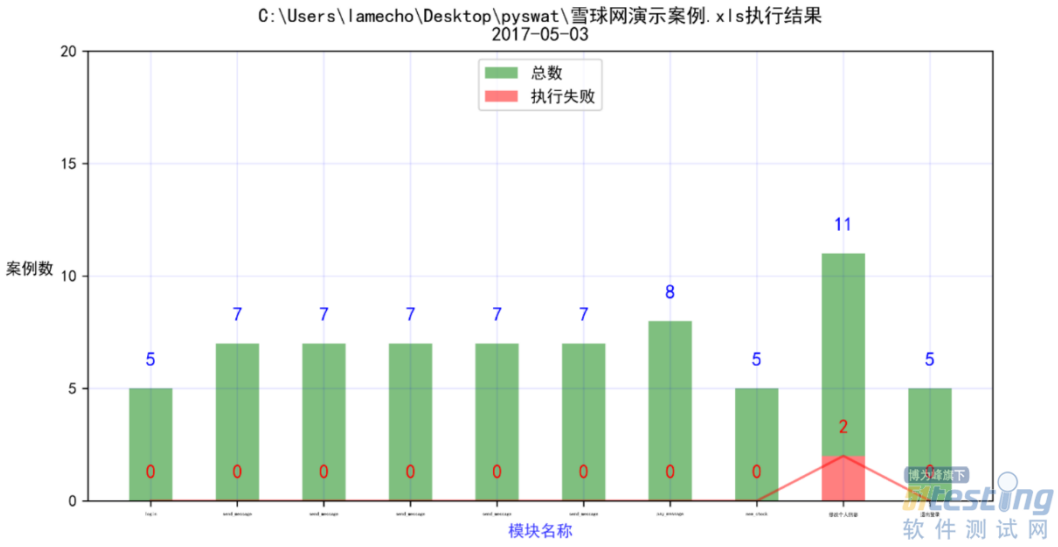
这是录制一个登录的操作，step1 和 step2 是后面手动增加的步骤，目的是测试登录页面的性能，可以先忽略。Step3 开始到 step5，分别录制到了输入登录的手机号，输入密码以及点击登录按键三个步骤。最后一步 stepOver 是终止符，得到这样的 excel 案例后，我们就可以直接通过“webs_replay”运行回放了，不过不要忘记先配置一下回放的案例路径和 case 名称以及回放 url 链接，这些同样是在“Enviroment.ini”配置文件中的 replay 项里做配置。回放过程中程序会记录运行的日志，详细到每一个 step。如下图

run.log - 记事本

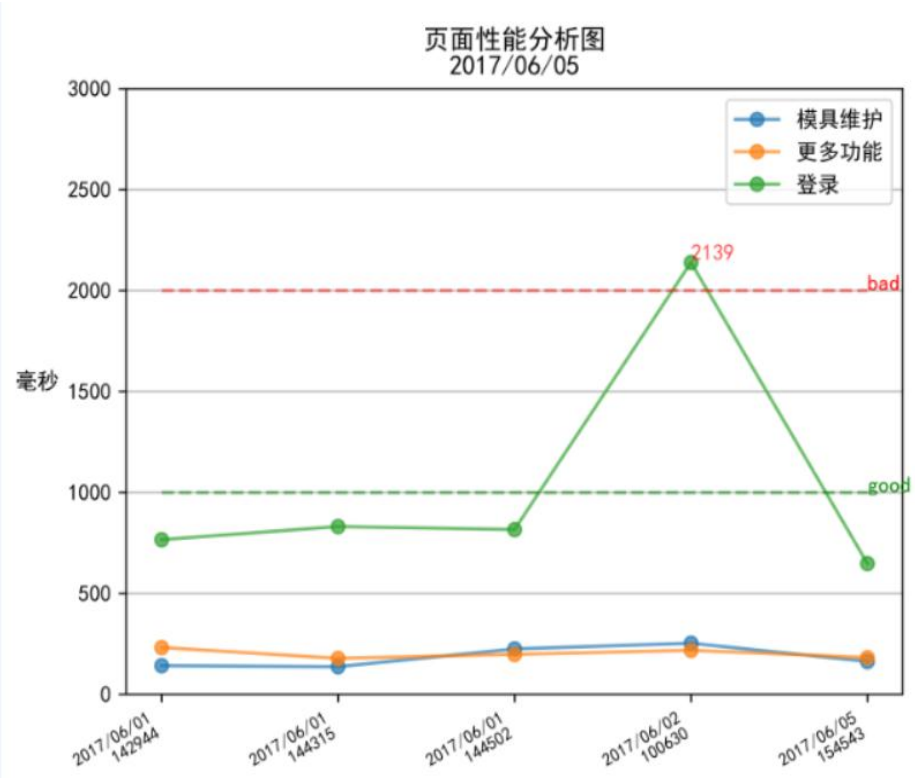
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
2017-06-01 10:26:59 :开始执行 { C:\Users\l  acho\Desktop\pyswat\  .xls }
2017-06-01 10:26:59 :-- Case: [登录] 开始运行
2017-06-01 10:26:59 :---- step1 开始
2017-06-01 10:26:59 :------ 开始 登录 页面性能统计
2017-06-01 10:26:59 :------ 初始化页面性能数据成功
2017-06-01 10:26:59 :---- step1 结束
2017-06-01 10:26:59 :---- step2 开始
2017-06-01 10:27:00 :-------- 页面加载总时间 : 818 毫秒
2017-06-01 10:27:00 :-------- Request请求耗时 : 56 毫秒
2017-06-01 10:27:00 :-------- TCP链接耗时 : 0 毫秒
2017-06-01 10:27:00 :-------- 白屏时间 : 453 毫秒
2017-06-01 10:27:00 :-------- DNS查询耗时 : 0 毫秒
2017-06-01 10:27:00 :-------- DOMready时间 : 804 毫秒
2017-06-01 10:27:00 :-------- 解析DOM树结构耗时 : 139 毫秒
2017-06-01 10:27:00 :-------- 完成 登录 页面性能统计
2017-06-01 10:27:00 :---- step2 结束
2017-06-01 10:27:00 :---- step3 开始
2017-06-01 10:27:01 :---- step3 结束
2017-06-01 10:27:01 :---- step4 开始
2017-06-01 10:27:01 :---- step4 结束
2017-06-01 10:27:01 :---- step5 开始
2017-06-01 10:27:04 :---- step5 结束
2017-06-01 10:27:04 :-- Case: [登录] 运行结束 (Pass) 共执行案例数：5 失败案例数：0
```

这里可以看到日志回放时间，对应的 step。上面提到性能测试是在 1，2 两个 step 完成的，大家可以很清楚的看到测试性能的结果，有总的页面加载时间，和分步骤的时间耗时。案例执行结束后同样会生成一个柱状图，展示整个案例运行的结果。



以及性能分析图表



这样这个自动化实现流程就完成了。

其实个人认为 pyswat 框架最创新的地方就是通过录制的方式记录生成自动化案例，完全不用写一行代码，测试人员上手就非常直接，亲切。这样避免了做自动化实现还需要测试人员学习代码编程方面的知识，通过 pyswat 只需要掌握一点 html 语言的一些相关知识就可以了。

当然以上讲的只是 pyswat 框架最基本的操作知识，如果要用精，用熟的话，pyswat 提供了很多额外的功能去辅助测试人员定位元素或完善我们的自动化测试用例。比如我们上面提到的测试页面性能，用到了 time_start(登录)，time_end(登录)。就只需要增加这两个 step 在用例中就可以了，这样 pyswat 就会在执行案例的同时把页面的性能测试一并同时做了。

下面我就简单举几个这方面的例子。

（1）案例输入参数化。

由于我们录制的输入内容是固定的，如果运行场景需要一个变化的输入，pyswat 提供了多种输入方式，比如生成随机内容，顺序内容，前文参数关联等。

（2）上传文件

写过 selenium 脚本的同学一定对上传这块的处理感到头疼吧，在 pyswat 里封装了 uploadFile(文件路径)方法，只需要将你要上传的文件路径传给方法即可，并且允许多文件同时上传。

find_by_text()

这个方法是可以直接通过页面元素的 text 内容查找元素，这个方法 selenium 是没有的。Selenium 类似的提供一个通过 link_text 查找，而 find_by_text()方法更为强大，只要页面元素有 innertext 值，你就可以通过这个方法去查找到元素。

（3）页面元素存在多个的情况

大家知道有时候我们的页面元素在查找的时候会遇到存在多个的情况，而具体到我们要点击的那个，自己写脚本的话会需要写判断。而 pyswat 提供了多种情况下的指定元素的功能。比如通过指定 at 关联，或是通过在元素属性中添加 index 属性等多种方法实现关联具体的一个元素。

另外 pyswat 框架在识别 iframe，select，或浏览器多窗口（标签页）都进行了处理，做到自动识别。也就是这些问题根本不需要使用者去关注，像对待一般页面元素一样去进行录制就行了，框架会自动识别处理。就目前来说框架在录制上解决了非常多的元素识别问题，基本上能够轻松的识别页面元素。

最后我想说 pyswat 框架非常符合测试人员使用，作为 webUI 的自动化测试框架能够很好，快速的将测试项目自动化方案实现。由于完全不用手写脚本程序，录制的方式可以很轻松的应对项目的环境，元素的变化。

文后提供一个 pyswat 官方的一个使用手册，可以在线阅读：
http://www.51testing.com/html/43/n-3719543.html

❖ 拓展学习

- 高端 Python 自动化测试开发系列课程学习：http://www.atstudy.com/course/374

软件测试心理学

◆ 作者：流氓贵族

心理学一词来源于希腊文，意思是关于灵魂的科学。灵魂在希腊文中也有气体或呼吸的意思，因为古代人们认为生命依赖于呼吸，呼吸停止，生命就完结了。心理学研究涉及知觉、认知、情绪、人格、行为和人际关系等许多领域，也与日常生活的许多领域——家庭、教育、健康等发生关联。这里讨论的心理学并没有理论上的那么复杂，仅仅涉及心理学领域的一丢丢的部分。

我们做的系统的最终目的是为了服务用户，这也就是为什么在测试的时候常常强调要站在用户的角度思考问题。其实不只是测试一个系统或是功能的完成，从产品的理念成型到产品的开发设计到产品的测试，产品的宣传都离不开用户，所以了解用户，揣摩用户的心理就显得至关重要。本片文章着重讨论一下在测试过程中的一些心理学的体现和如何将心理学运用到测试工作中。心理学在测试中主要用于猜测用户的操作行为。

猜测用户行为并不是一件简单的事情，毕竟子非鱼焉知鱼之所想。对于一个刚刚入行的测试人员来说，做到准确的猜测用户行为的确是非常的困难的。那需要有丰富的经验和积累，加上一些无法解释的直觉。根据这些年的工作经验，总结了一些用户的操作行为，这些行为准则对于大多数的用户还是适用的。

1.不要仅仅遵循规则，你不知道用户会怎么操作

这条很好理解，很多时候对于一个新到手的东西，很多人都不会去阅读用户使用手册，这是因为产品对于用户而言已经很熟悉了或者用户使用过同类的产品，还有的用户压根就不喜欢按套路出牌。这样就给测试增加了难度，测试人员需要模拟那些用户可能操作的步骤和使用场景来保证产品的健壮性。

例如：一个上传附件的功能，规定只能上传 jpg 格式的图片。这个功能的测试点是什

么？我们通常会设计这样几个测试点：

- 不上传文件，点击上传
- 上传非 jpg 格式的文件，点击上传
- 上传 jpg 格式图片，图片名称含有特殊字符、中文的图片
- 上传 jpg 格式图片，图片名称与已经上传的图片名称重复
- 上传 jpg 格式图片，图片名称符合常用命名规则

通常来讲，这样的测试点设计已经包含了所有的常用场景。但是，笔者还是遇到了一个意外之中，情理之内的场景。用户是这样操作的：用户将 png 格式的图片通过修改后缀改为符合条件的 jpg 格式文件上传。由于上传只通过文件名称校验，这个披着 jpg 格式的 png 格式图片通过校验，导致运营监控报错。

分析一下这个用户行为其实很好理解，使用这个功能的人员可能不是专业的美工人员，他对于文件的了解仅仅停留在表面，可能是为了工作方便就简单的将文件改成他认为的正确文件传了上去。当然这也算是一个 bug，最后 bug 修复。

总结下来，就是不要严格的遵守规则，因为用户并不会那么做。他们还会用一些方法来规避规则，测试的时候要尽量都考虑到。

2.多去思考用户潜意识遵守的规则行为

有些用户对于产品的使用有潜意识的使用习惯。上条已经讲过很多用户不会阅读使用手册，一些很优秀的产品经理也在致力于设计出让用户脱离使用手册的产品。当然，那是在产品拥有很多忠实用户后才能够实现的产品，就像苹果手机那样。对于一个新推出的产品来说，很多用户还是不了解规则的，但事实上用户也不愿意去了解，仅仅根据自己的经验来使用产品，当他们碰壁了之后才会去阅读使用手册。所以，测试人员要保证异常场景测试覆盖全面并有友好的引导语引导用户正确使用。

例如：一个网站，注册账户有两个身份，一个身份是融资者、一个身份是投资者，两个身份的权限无交集，一个账户只能有一个身份，不能兼有两个身份。该网站的注册页面默认是投资者的身份，注册融资者身份的账户需从投资者注册页面点击链接进入融资者注册页面。这样的注册页面就导致很多想要注册融资者的用户注册成为了投资者，导致了很多用户的流失。最后，修改注册页的注册框为投资者和融资者分 tab 页的形式。

原有的注册页两个账户的身份不明显，用户很容易就按照习惯直接注册账户了，而不是先看是不是自己要注册的身份，而且页面也没有着重表现出注册的用户身份。最终用户注册后，发现网站的内容并不是自己想要的也就放弃了这个网站，注册的账户也就变成了僵尸账户。

总结下来，测试的时候，可以适当的忘记熟悉的规则，按照自己的潜意识去操作。作为一名测试人员，你很熟悉这个产品，但是用户并不是。有时候，他们会任性的按照自己的想法操作。

3.懂逆向思考

逆向思考就是从结果去想起因，这不是用户的思考方式，但是很多用户的操作都是有这种方式引发的。也就是说，对于用户来讲，过程并不重要，他们可以用任何方式和操作方法来达到他们的目的。有些操作对于用户来说是不意思的。

例如：用户想要购买一件商品，他没有时间去挑选或者不知道买什么样的好，就会拜托朋友帮忙。朋友就会发送商品的链接给用户，可以是邮件、微信、短信等的形式发送链接，这个时候，要保证商品的链接是可用的。在测试的时候，这种场景通常会被遗留。遇到过这样的一个问题，通过邮件发送的链接，打开报错页面。

总结一下，从目的去反推、猜想可能的操作步骤和场景来补充测试用例是必要的。尽管遇到这样的场景的概率很小，测试中运用心理学还有几个方面就是引导用户操作、使用功能，预测用户的使用感受。不过这两个方面如果产品经理在设计产品的时候考虑的全面的话，就会在产品设计阶段解决了，但是不能把这两个方面的问题完全依赖于产品经理的设计，测试的时候多考虑一下，培养自己的用户思维。

说白了，测试心理学就是研究用户怎么思考、怎么操作，站在用户的角度去测试产品。然后保证产品尽可能的满足用户的需求。能留住用户的产品才是好产品。

作者简介：原本是学的偏硬件的专业，在找工作的时候听了一次招聘软件测试工程师的宣讲会，遂发觉做软件测试更适合自己，就走上了软件测试这条道，也希望越走越远。做过手机整机测试+WEB 测试+APP 测试。略懂自动化、性能方面，学习 ing 中。

Linux 上安装 Bugfree 系统

◆作者：枫叶

1、整体流程

一、安装 Apache 服务器，安装后启动进程并验证是否正常运行;

二、安装 MySQL 服务器，默认已安装，可以检验是否正常;

三、安装 PHP 服务器，默认已安装，可以检验是否正常;

四、安装 Bugfree 服务器，下载 Bugfree 安装包后解压，配置安装，安装完成后检验本机/其他客户端是否正常访问;

五、配置邮件服务器，注意在...\bugfree\protected\service 这个文件夹下修改 MailService.php 文件

2、详细过程

2.1、安装 Apache 服务器

可以在 SSH 输入

#yum install httpd

提示更新完毕:

httpd.x86_64 0:2.2.15-59.el6.centos

作为依赖被升级:

httpd-tools.x86_64 0:2.2.15-59.el6.centos

完成以上安装后启动进程，

#service httpd start

验证根目录下新建 index.html 文件，用浏览器能正常打开就 OK。

#find / -name httpd.conf

寻找根目录，找到后在 WinScp 中做新建操作。

2.2、安装 MySQL 服务器

检查 mysqld 是否安装:

#service mysqld status

如果没有，则按以下步骤安装:

#yum install mysql++.i686

#yum install mysql-libs.i686

#yum install mysql-server.i686

#yum install php-mysql.i686

我的安装中程序未找到后面两个，且系统中已自带，读者可以依据系统实际情况选择安装。

安装完成后启动进程:

#service mysqld start

再检查服务端口是否调用:

netstat –ntl

从安全角度考虑，需要配置 MySQL 服务器 root 的密码，我的系统中之前已配置，读者请自行查找资料。

2.3、安装 PHP 服务器

默认是已经安装的，可以检验下是否运行正常。

安装路径通常在/etc 目录下，可以参考流程查找。

设置完以上三个，可以将 httpd 和 mysqld 进程设为开机启动:

#chkconfig httpd on

#chkconfig mysqld on

重启系统 shutdown -r now 后生效。

2.4、安装 Bugfree 服务器

下载 Bugfree 安装包后解压，将解压出来的文件夹整个复制到 html 文件夹下，一般路径是/var/www/html/。

Bugfree 初始用户名: admin 初始密码: 123456

2.5、配置邮件服务器

在..\bugfree\protected\service 这个文件夹下，修改 MailService.php，网上有不少这样的资料这里就不赘述了。

Python 自动化测试应用-番外篇

◆ 作者：lamecho 辣么丑

1.1、概要

本文主要内容是教大家学习如何利用 Python 对 Jira 缺陷进行管理操作。

在平时的测试中，大家对于缺陷管理平台一定不陌生，平时的缺陷都要去平台进行操作。今天我们就来看看 Python 是怎么样连接 Jira 进行操作的。

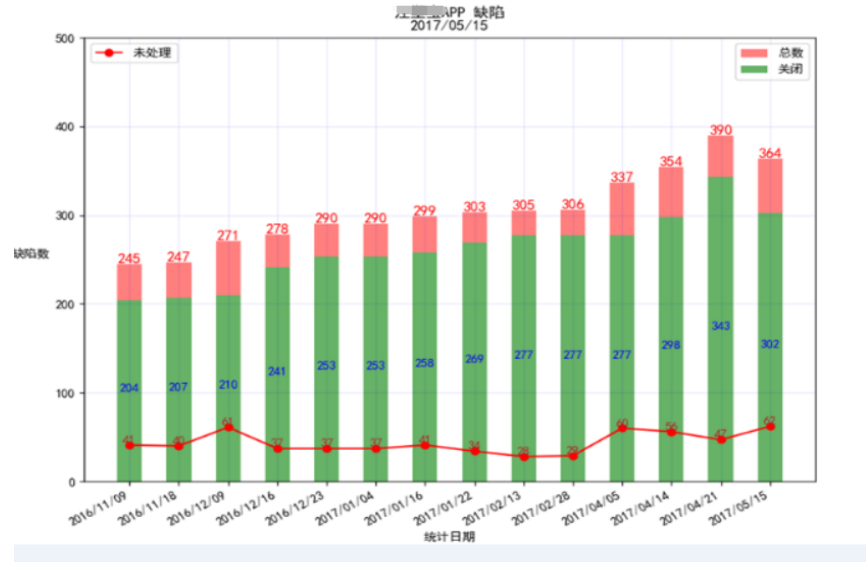
1.2、获取 JIRA 数据

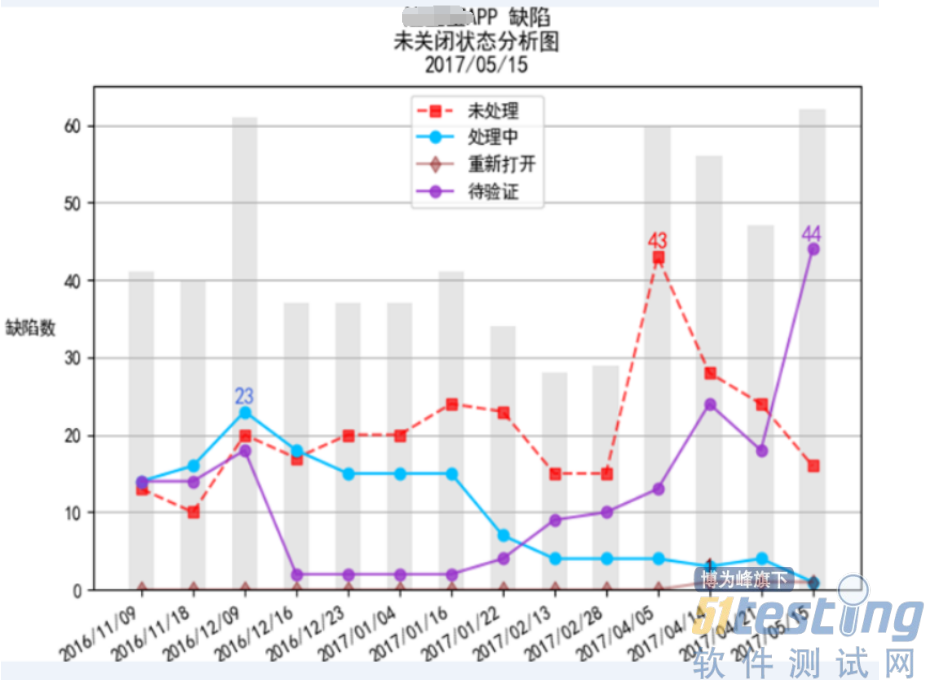
某天领导要看 XX 项目的目前缺陷情况，总共有多少 bug，多少未处理的，未处理的 bug 中严重程度是怎么样的，这些 bug 都属于谁来负责修改等。这一串问题足以把任何一个测试人员击垮，如果平时不做准备的话。

如果你有一个定期统计缺陷信息的习惯的话，我相信你会很淡定的把结果展示给你的领导，像下图这样。

	A	B	C	D	E	F	G
1				APP 缺陷			
2	统计日期	未处理	处理中	重新打开	待验证	关闭	总数
3	2016年11月9日	13	14	0	14	204	245
4	2016年11月18日	10	16	0	14	207	247
5	2016年12月9日	20	23	0	18	210	271
6	2016年12月16日	17	18	0	2	241	278
7	2016年12月23日	20	15	0	2	253	290
8	2017年1月4日	20	15	0	2	253	290
9	2017年1月16日	24	15	0	2	258	299
10	2017年1月22日	23	7	0	4	269	303
11	2017年2月13日	15	4	0	9	277	305
12	2017年2月28日	15	4	0	10	277	306
13	2017年4月5日	43	4	0	13	277	337
14	2017年4月14日	28	3	1	24	298	354
15	2017年4月21日	24	4	1	18	343	390
16	2017年5月15日	16	1	1	44	302	364
17							
18							
19							
20							
21	未解决缺陷总数			等级			
22				严重	重要	次要	微小
23	18			1	13	4	0
24	备注：未解决缺陷=未处理+处理中+重新打开						
25							

	I	J	K	L	M	N	O
1				缺陷			
2	统计日期	未处理	处理中	重新打开	待验证	关闭	总数
3	2016年11月9日	290	30	4	54	270	648
4	2016年11月18日	189	49	5	57	371	671
5	2016年12月9日	202	39	10	52	457	760
6	2016年12月16日	94	24	13	26	662	819
7	2016年12月23日	97	25	13	12	675	822
8	2017年1月4日	76	27	13	31	675	822
9	2017年1月16日	95	13	12	111	695	926
10	2017年1月22日	42	7	10	49	845	953
11	2017年2月13日	102	13	12	17	879	1023
12	2017年2月28日	111	14	11	49	879	1064
13	2017年3月16日	89	11	11	92	881	1084
14	2017年4月5日	105	12	11	88	887	1103
15	2017年4月14日	163	12	16	21	963	1175
16	2017年4月21日	131	11	16	58	977	1193
17	2017年5月15日	123	12	19	33	1029	1216
18							
19							
20							
21	未解决缺陷总数			等级			
22				严重	重要	次要	微小
23	154			14	100	36	4
24	备注：未解决缺陷=未处理+处理中+重新打开						





图表中的数据应该全部来源于缺陷管理平台，如果平时有一个定期统计信息的习惯的话，那么你会很清楚的掌握目前项目的缺陷情况。今天我们重点不是将如何做好项目缺陷的统计，但是自动化的获取这些平台缺陷的信息的目的就是为了这些数据。

首先 Python 操作 Jira 平台只需要利用 Jira 库就可以了，安装也很简单

控制台输入：pip install jira

当安装好 jira 库后，我们就可以连接 Jira 平台了，先来一个简单的脚本：连接 Jira 并获取当前平台所有的项目。

```
from jira import JIRA
```

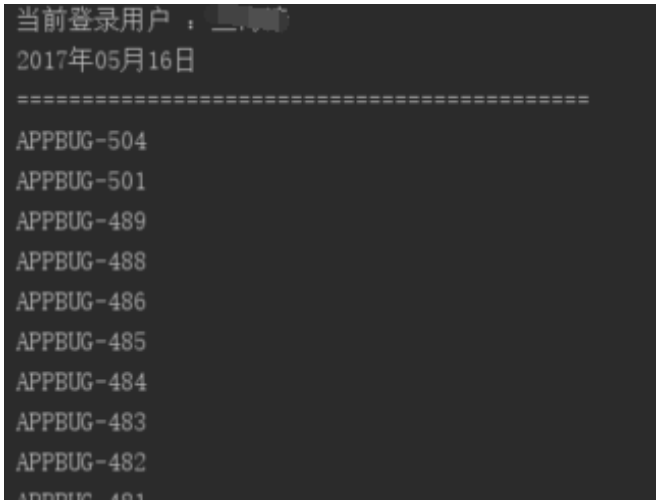
```
jr=JIRA('http://jira 地址',basic_auth=('登录名','密码'))
#连接 jira
print u'当前登录用户 ： '+jr.user(jr.current_user())
#打印当前用户，通过 jr.user(用户名)获取用户名称
print datetime.datetime.now().strftime('%Y 年%m 月%d 日')
print '========================================='
for i in jr.projects():#jr.projects()获取所有项目，返回项目字典
print i,i.name
```

输出结果:

好了，第一步通过 jr.projects 获取到项目名称后，我们就可以获取该项目下的具体缺陷信息了。获取项目下缺陷用到 search_issues('project=项目 name')方法，具体写法:

```
issue=jr.search_issues('project=APPBUG')
for i in iss:
print i,i.id
```

issue 返回的是项目下缺陷的 list，其中包含每条缺陷的 key，id 值。当然我们获取这个列表的长度就是项目目前缺陷总数了。我们看看返回结果:

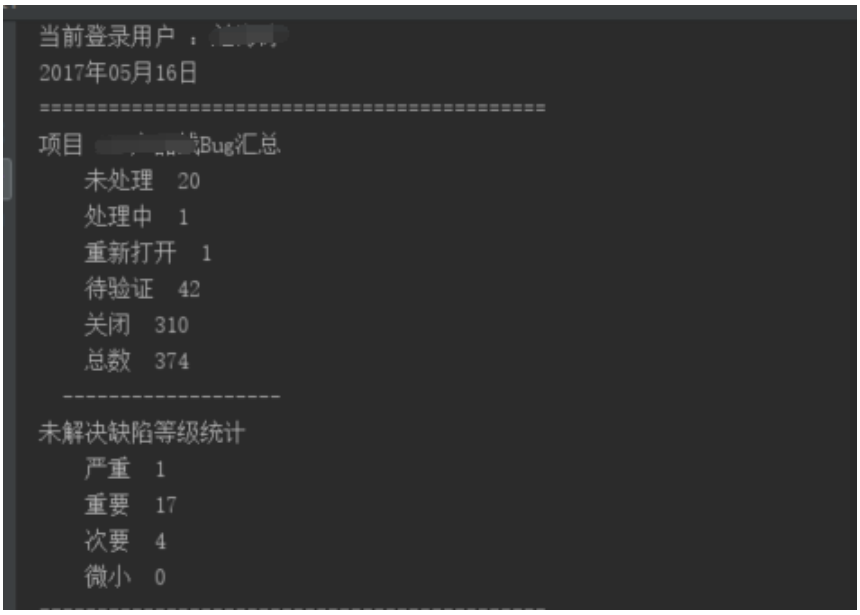


紧接着就是获取某条缺陷的具体信息了，像我们平时录入缺陷所要填的信息，比如：缺陷等级，缺陷状态，缺陷类型，创建时间，缺陷概要，缺陷内容，缺陷创建人，缺陷指派人，附件，备注等等信息。我们通过上面获取到的缺陷 key 值来获取具体的缺陷上，脚本里这样写：

```
isss=jr.issue('APPBUG-486')
#传入具体缺陷的 key 值
print isss.fields.issuetype
#打印缺陷的类型，类型根据自己 jira 平台设置的类型而定，如缺陷，改进等
print isss.fields. summar
#打印缺陷的主题即标题内容
print isss.fields.description
#打印缺陷的描述即具体内容
print isss.fields. status
#打印缺陷的当前状态，如：已解决，关闭
print isss.fields. resolution
#打印缺陷的解决结果，此处是开发人员填写
print isss.fields. priority
#打印缺陷的优先级，如：严重，重要，轻微等
print isss.fields. reporter
#打印缺陷的创建人名称
print isss.fields. assignee
#打印缺陷的指派人名称
print isss.fields. created
```

```
#打印缺陷的创建日期
print isss.fields.updated
#打印缺陷的修改日期
print '%s'%isss.fields.comment.comments
#打印缺陷的备注信息
print isss.fields.attachment
#打印缺陷的附件信息
print '%s'%isss.fields.components
#打印缺陷的所属模块信息
print isss.fields.environment
#打印缺陷的环境信息
```

基本上以上这些缺陷信息都是一个缺陷的必备内容了。此时我们就可以做进一步的开发了，比如获取项目中各种状态的缺陷数，利用 isss.fields. status 获取项目缺陷的状态分别统计个数，如果加上 isss.fields. priority 优先级判断，可以统计对应的状态的缺陷优先级个数。如下图



1.3、录入 Jira 缺陷

上面讲的是关于获取 Jira 缺陷信息的一些操作，当然我们还需要关心一下录入缺陷的操作，毕竟我个人是很讨厌在缺陷平台上一个个添加 bug 的。录入缺陷用到的方法：jr.create_issue(fields= field_list,prefetch=True)。create_issue 就是创建一个新的缺陷，我们只需要往里传入 fields 参数即可。下面看看具体写法：

```
field_list={
'project': {'key':'APPBUG'},
'summary': 'issue 标题',
'description': 'issue 内容',
'issuetype': {'name': u'缺陷'},
'priority': {'id': '3'},
'assignee':{'name': u'username'},
'components':[{'name':u'模块'}],
}
id=jr.create_issue(fields=field_list,prefetch=True)
```

大家可以看到 field_list 传入的是一条缺陷必须的一些属性值，当成功写入一条新的缺陷后返回给 id 的是这条缺陷的 key 值，获取到 key 值后，接下来我们就可以给这条 bug 传入附件了。

```
jr.add_attachment(issue=id,attachment='d:/1.jpg',filename='d:/1.jpg')
```

这里我们通过 add_attachment 方法添加缺陷的附件，issue=传入新缺陷的 key 值，attachment='d:/1.jpg',filename='1.jpg'分别填上文件路径和文件名称就可以了。

1.4、跟踪缺陷

跟踪缺陷的处理情况，用到 transitions(issue)。代码这样实现:

```
transitions= jr.transitions(jr.issue('APPBUG-485'))#传入缺陷对象
print transitions#打印
```

基本上掌握了这些 Jira 的操作，就可以很方便的获取 Jira 缺陷的统计数据，有了这些数据就可以做出精美的图表，并且可以通过 Python 自动的批量录入缺陷，只要把 excel 表的信息读取出来就可以了。

搬走自动化路上的绊脚石

◆作者：邵君兰

在自动化测试的过程中，你是否遇到过以下问题？

1、一个人做自动化的时候挺和谐的，啥毛病没有，多人自动化的时候就发现有很多冲突。

2、脚本到底是拆分好，还是合在一起好，如果拆，要怎么拆？

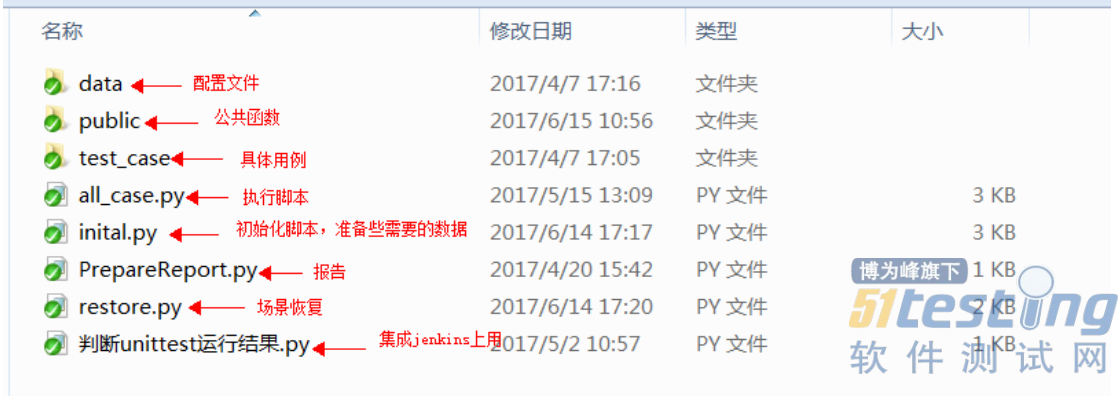
3、自动化脚本做好了，但是做持续集成后，发现有的地方却又不那么自动化，还是很繁琐。变成了一个假的自动化，这也往往导致自动化脚本执行的次数少，价值没有体现的问题。

这些问题在走自动化路上的童鞋应该多少都遇到过，那么如果顺利的去搬走这些绊脚石呢？

搬走法一：搭设框架并制定多人协同的规则

1、我们在做自动化时，确认要用什么框架，比如说采用 Python 的 UnitTest 框架

2、由一个人来负责搭设框架，如下图



2.1、配置文件不可少，经常要变动的参数，不能写死在脚本里的参数，都可以放到 data 下面，比如说服务器地址，由于多个迭代并行开发，经常会分配不同的 ip 地址，如

果写死 ip 就会导致在其他迭代中不能正常运行，所以要先提炼出来放配置文件中。当配置文件中的参数一多时，那么命名规则就尤为重要了，配置文件的规则要有明确定义，比如说我们把服务器地址统一配置成 URL=192.168.*.*，而不要出现多个表示服务器地址的配置名，如又出现一个叫 IP=192.168.*.*。

2.2、写一个脚本的时候会用到很多方法，此时不要写到具体的 case 里面，而是要实现公共的类和函数，放到 public。当不同的人去写 case 时，就可以不用重复代码，去 public 找就可以，日积月累，将越来越丰富。

2.3、执行具体 case 时，我们必须要有前置条件，比如说我的 case 要在登录后才能执行，那么这个登录的帐号我必须确保它是存在的，如果不存在我就自动添加进去，这样脚本的回访率就很高了，所以这个 inital.py 就是干这个的。

2.4、在执行脚本的过程中，我们产生了一些数据，但是这些数据是我们不想要保留的，那么我还可以进行场景恢复，这个 restore.py 就可以干这个。

3、具体的框架出来后，接下来就是各位自动化测试的工程师们按照既定的规则进行填充用例，覆盖更多的场景。

千万不要零散的写脚本，你一脚本我一个脚本，没有规范，这样其实对做持续集成来说会是一个非常难的事情，你可能会花很多的时间去调整脚本，所以一开始就要先搭设框架。

搬走法二：脚本会经历拆拆合合，不断的整合出最适合

脚本 case 还不多的情况下，往往是整在一起比较方便，一个脚本测试一个项目，感觉维护起来很方便，但是当 case 多了以后，发现会出现维护难，甚至是性能有问题，比如 SoapUI 会出现 out of memeroy 的情况，此时就不得不拆。还有当脚本过于庞大时，一旦出现问题时，就要打开一大段代码进行调试，着实也让人很头疼。

那是不是拆的很细呢，一个 case 一个脚本？这个也有弊端，那么多脚本，维护也不方便。所以还是要把握一个度，可能一开始这个度把握不准，那就慢慢的进行拆或整合，就像程序员也要经常的进行重构是一样的道理，总结出一个合适的度，就可以。

比如说我们要做网页的自动化测试 case，可以根据页签或者抽屉来拆分脚本，这样会更加清晰一些。

搬走法三：不断的优化，从半自动变成全自动

之前做 SoapUI 接口时，将配置的参数写到了 SoapUI 脚本里面，这样导致每次修改配置文件，都要去打开具体的脚本，然后修改，非常麻烦。所以就进行了解耦，把需要配置的参数放到 Global，这样只要改这个文件即可，无需打开脚本文件，维护和操作性带来大大的便捷。但是问题来了，当集成到 jenkins 上后，很多的团队在同一台测试机器上跑脚本，SoapUI 只有一个，所以 Global 经常被改的面目全非。这样每次在 jenkins 上跑之前，还要去远程测试机，把 Global 改成正确的才行，很繁琐，且也无法利用 jenkins 的功能，实现一旦代码更改就触发运行脚本的自动化机制。所以导致脚本回放次数低，每次手动执行。对于这个问题大家虽然觉得麻烦，但都习以为常。变成了一个半自动化的自动化测试。

事实上自动化测试不仅仅是 case 的自动化测试，让整个流程都自动化，顺畅的跑起来才是真正的自动化。对于这个问题的解决方案其实很简单，我们把手动远程修改配置文件的操作让它自动化掉，每次在执行这个脚本前，都去修改配置文件，也就不会怕其他人改了配置文件影响到自己的脚本运行。

熟悉 SoapUI 的同学应该不难发现，Global 其实是存在 soapui-settings.xml 文件里面，路径为 C:\Users\Administrator\soapui-settings.xml，那么其实我们只要把需要的配置文件放到 svn 上，然后通过 py 脚本去读取 svn 上的配置文件，然后将内容进行拼装成 xml 的格式写入这个 soapui-settings.xml 即可，当然最后要在 jenkins 上配置好这个 py 脚本，就能全自动了。

自动化测试路上的一些绊脚石确实有时候让人很头疼，但解决每一个测试路上的绊脚石后，都能让自己成长一点，对自动化有了更加深入的理解和认识。追求卓越，让自动化变的更自动化，让测试效率提升，一起收获满满的成就感。

当没有足够的时间去测试的时候，我们应该做些什么

作者：王 鑫

环顾你的测试周期，你是否经常发现你没有足够的时间去测试？刚开始，你会认为一切都在掌控之中，但是很快你就会偶然发现这个问题，“当没有足够的时间去测试的时候，我们应该做些什么”

我一直被这件事困扰，并且这件事毫无乐趣可言。

我思考这件事很长时间，一件事刚开始是非常好的，它是如何这么快变的这么严重，下面是我的分析:

我的测试时间去哪了？

首先，为什么会发生这样的事呢？下面是众多原因中的几种:

1）错误的估算:

如果在开始阶段你就有一个不准确的期望，这件事注定是失败的。一个好的估算需要考虑下面几个方面:

A）准备任务的时间-我们谈论的任务类似下面情况:

识别需要回归的套件并把它们放在一起统一回归

创建测试数据

花时间去准备测试（如冒烟测试、健全测试等）

B）测试用例的维护：测试用例是长期使用价值的资产，在测试执行期间确保测试用例发生较小变更。对于这些新产品小的维护任务，建议分配测试时间的 30%，有可能一些团队和项目并不需要 30%的时间，但是尽量分配一些时间和精力在这项任务上。

C）自由测试/探索性测试-脚本测试是测试估算的主要特性，然而，在世界上即使模型是显性脚本，也没有一个测试团队拒绝探索你的软件。

D）报告/交流-包括分类/站立会议，更新工作管理工具等等

E）意外因素：一般意外因素应该留有你原始估算的 25%-30%的缓冲时间，但是团队很少能够支持，即使是这样，也应该给自己留点调整的余地

F）团队和能力：如果你刚刚带领了一个新的团队，或者他们第一次使用一个新的工具，你需要留出一些时间进行培训。根据你的团队水平来调整你的估计。

2）版本不稳定和其他技术问题

A）冒烟测试、健全测试失败：在 QA 部署的环境中，AUT 阶段基本的测试失败后就不需要那么多的测试团队执行测试。当出现这样现象的时候，我们可以去从事其他项目的工作，但是这也不能填补整个测试周期时间，这是时间浪费的主要原因。

B）测试数据不可用：产品-比如数据是每一个测试项目都必须拥有的，在 QA 环境中没有准时获得测试数据也是另外一个主要因素。有时测试人员可以通过创建和管理自己的测试数据来解决这个问题，但它是费时的而且有可能并不总是在点上。

3）环境问题

部署失败，服务器不断得到超时，更多的类似问题会吞噬你的测试周期。这可能源于这样一个事实，一些公司（并非所有）降低了有效率的 QA 所需良好的工作环境的重要性。他们经常试图摆脱低容量的服务器，使用替代品代替，这确实是一个短暂的修复，没有任何人会喜欢，事实上，这可能会导致他们的测试质量和宝贵的测试时间损失。

4）工作相关关系人间缺少统一

这可能是敏捷或安全团队在接近项目尾声时遇到的罕见问题，但是当开发，实施和 QA 应该接收彼此的交付物的时候很多成员仍然存在分歧或误解，因此，造成延迟。

现在我们了解到以上问题，下面有几种方式去解决这些问题。

测试者如何获得足够的时间进行测试？

1）准确的估算

对于重新估算感到怀疑的时候，不要低估而是需要有一个合理的原因。不要忘记根

据你的团队、工具和进度进行估算判断。估算完成后确保正式确认，保证所有人能够了解并且保持在知情状态中。

2）考虑历史数据-测试管理工具是你最好的朋友

A. 早期测试循环周期需要多长时间

B. 哪种问题原因引起之前测试周期发生变化

C. 在它们通过之前测试用例运行了多少次?

D. 报告了哪些缺陷?

E. 哪些缺陷导致测试中断?

3）在关键时刻问下面这些问题并且制定计划

A. 找出重要的功能是否在你的项目中?

B. 找出项目的高风险模块?

C. 哪些功能模块用户最经常使用?

D. 哪些功能具有很大的安全影响?

E. 哪些功能对用户的财务影响最大?

F. 应用程序的哪些方面对客户最重要?

G. 代码中哪一部分最复杂，因此最容易出错?

H. 应用程序的哪些部分是在匆忙或恐慌模式下开发的?

I. 开发人员认为应用程序的最高风险是什么?

J. 什么类型的问题将引起最坏的影响?

K. 什么类型的问题将引起最多的客户服务的投诉?

L. 什么样的测试可以很容易地覆盖多种功能?

考虑到上面这些点，可以在较少的时间约束下大幅度减少项目释放的风险

4）使用测试管理工具，可以大大减少准备、报告和维护的时间和工作量。

对于最流行的测试管理工具列表，请查看此处

5）对于错误的开发/技术问题，我们没有什么可以做的，但是唯一可以帮助的是查看单元测试结果。这有一个观点无论是开发是成功或者失败，因为哪种测试导致的失败，所以我们不会重蹈覆辙。

如果您的测试管理工具支持 CI 集成，那么您就可以在没有任何干扰的情况下获得这些信息，从而更好地理解应用程序的稳定性。

6）经常衡量你的生产力和过程。不要为了外部团队的利益，而让状态报告成为可交付成果。确保你在密切监视你的日常目标和完成这些目标的能力。

另外，确保不要进入“速度与质量”这一经典问题。因为，当你每天报告 50 个错误时，你可能会显得效率非常高。但是如果大多数人都是无效的，说明你就有问题了。

因此，监视、监视和监视多一点

总结

最后，尽管你采取了所有的措施你依然觉得自己时间紧迫，请需求帮助,大多数团队都愿意参加会议，使事情回到正轨上来。

简单 5 步，轻松搞定 Burpsuit 抓取 Https 数据包

◆作者：王 鑫

一般国内的软件基本上采用 http 协议几年通信，但是对于银行等涉密 APP 现在基本上采用 https 协议。以前 Charles 工具只能抓取 http 包，当 APP 采用 https 协议时却无从下手，针对这情况，找到一款替代 Chrales 抓包的工具是目前趋势。下面总结了从环境搭建到 Burpsuite 抓包的全过程，希望对大家的测试工作起到帮助。

一、Java 环境搭建

Java 环境搭建是运行 Burpsuite 的前提，只有搭建好 java 环境才能看展后续工作，这里请读者自行搭建，不作详细描述。

二、Burpsuite 下载安装

1、Burpsuite 下载最常用的有两种方式，即官网下载和破解版安装，如果只针对于抓取 https 来说，我们在官网：https://portswigger.net/burp 下载免费版即可（事情简单化原则）

2、下载完成后，找到 burpsuite_free_windows-x64_v1_7_21.exe 文件，双击运行，一直下一步，安装完成

三、Proxy 配置

1、在开始菜单找到刚刚安装好的 Burpsuite，点击【next】-【start Burp】如下图

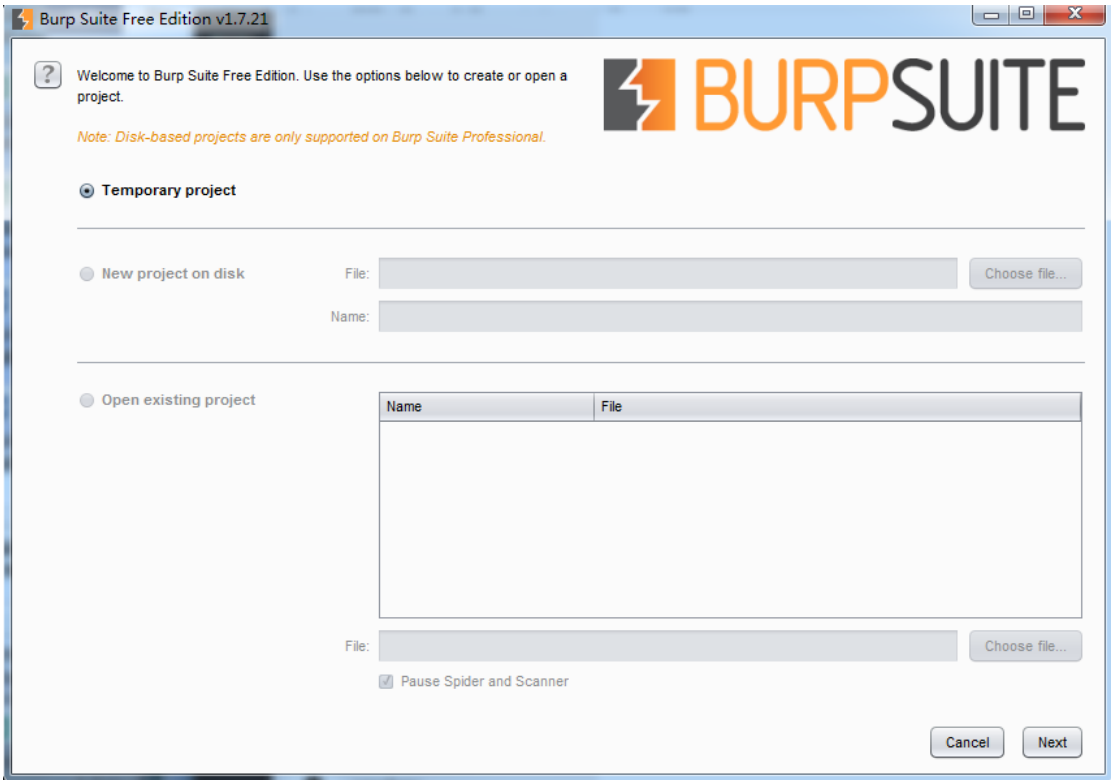


图 1 安装选择

2、代理配置

点击 Proxy，点击 Intercept is on 按钮（Intercept 是用来拦截请求的，抓包通常不用）

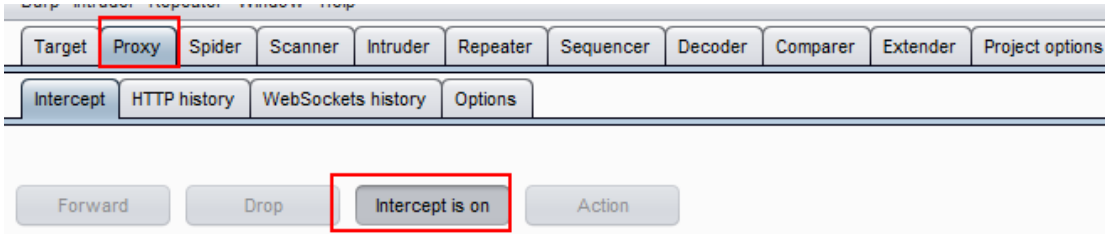


图 2 关掉 intercept

a) 点击 Options tab 页，点击 Add 按钮

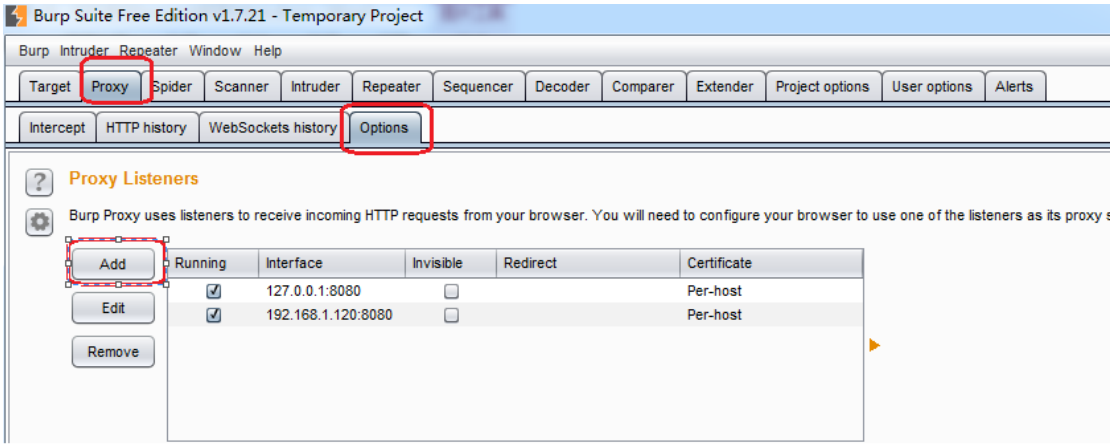


图 3 配置 options

b) 填写代理端口 Bind to port，通常用 8080 就行，不过要确保填写的端口本机没有占用；点击 Specific address 下拉列表，选择本机做代理的 ip 地址，然后点击 OK

图 4 选择本机代理

确保刚刚添加的代理 Running 打钩。

四、手机配置代理

进入网络设置-WLAN-长按连接网络选择【修改网络】-【高级选项】-代理（手动）、

主机名（电脑 ip 地址）、端口（8080）-【保存】，代理配置成功

图 5 手机配置代理

五、手机下载证书安装

关于证书下载这部分，看到网上很多介绍使用 firefox 下载后转换，再传到手机端的方法，个人认为很麻烦，当你手机配置好代理手机浏览器访问 http://burp，然后点击 CA certificate 进行安装 CA 证书。

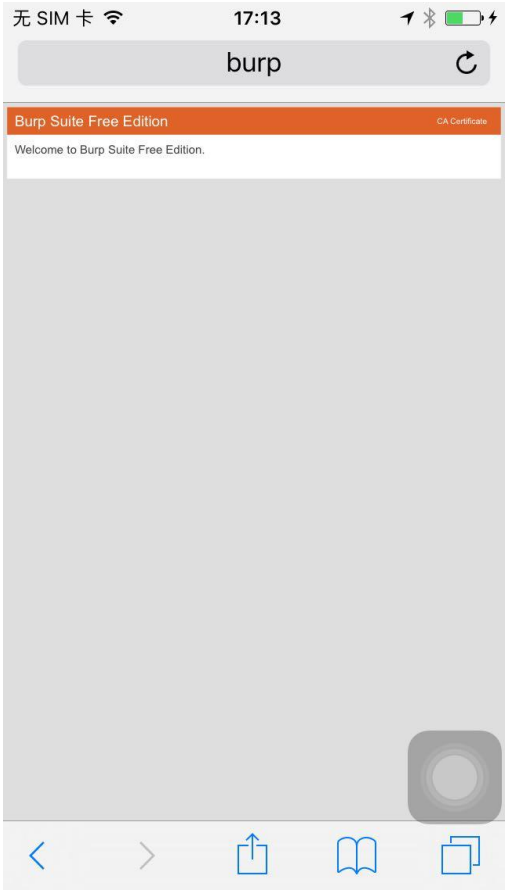


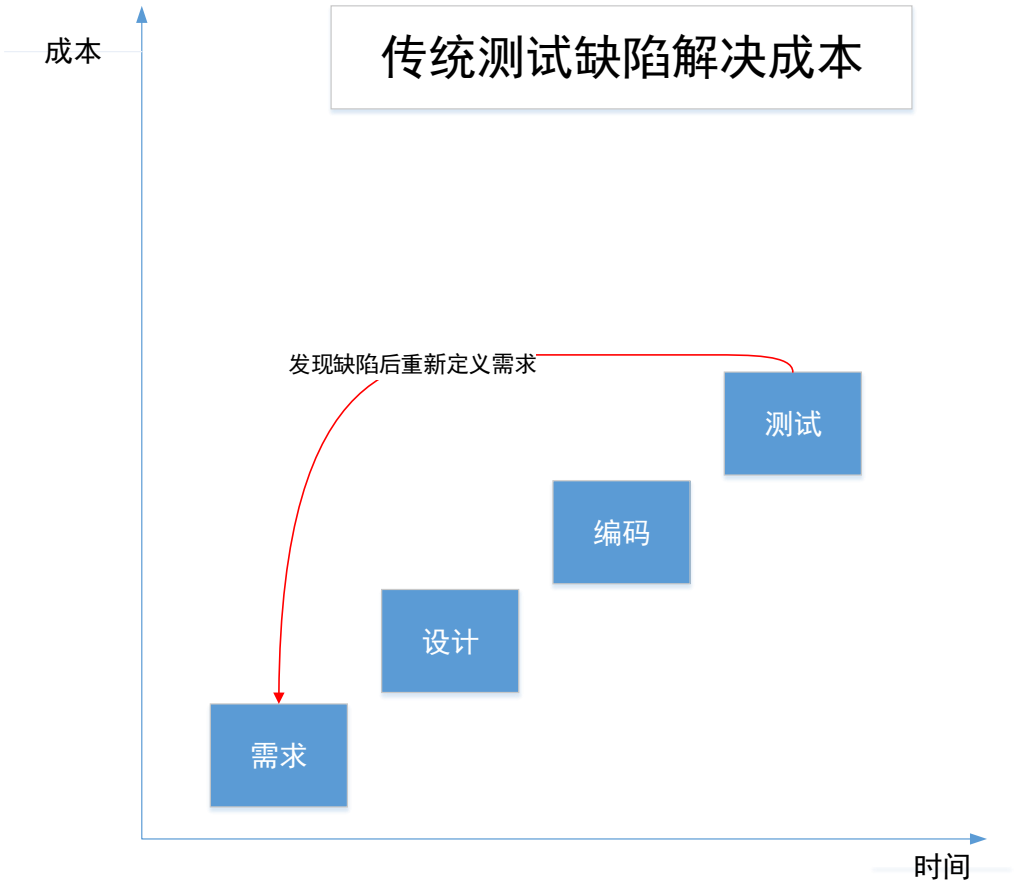
图 6 证书下载

完成以上 5 个步骤，就可以使用 burpsuit 抓取 https 数据包了，怎么样是不是很简便

漫谈“测试左移”思想

◆ 作者：Allen

在我们进行软件测试过程中无时无刻考虑时间，人力等资源投入对我们测试范围的影响，通常随时间增长，解决缺陷的成本也是成正比的，如下图所示，发现缺陷后，一般需重新返回需求审视，经过设计，编码，再提测，这样带来的效果就是研发疲于修复缺陷，沟通。测试则回归缺陷，同时验证是否有新问题产生。项目组成员也开始对产品的信心有了不一样的看法。那么是否有一种思想或策略来解决这种测试的这种“囧”境？答案是肯定的。



测试左移，是一种测试的思想，顾名思义就是让测试的活动向前端推动，不仅仅只是被动式的接受提测版本进行系统测试。测试左移可以从以下几个方向开展：

1、需求评审，评审的目的在于初期准确把握产品商业价值的体现，从必要性，背景，可维护扩展性等几个方向考虑。笔者也经常在评审的时候问参与的测试人员，为什么要实现这个产品/功能？但往往很少能听到比较中肯的答案。实际上需求的精准分析影响可以减少后期测试资源的投入，且可以在这个阶段自动化专家和业务专家共同讨论，提出可落地性的自动化解决方案。

2、设计分析，在这里我们可以很细致的了解到具体某一个功能实现的方式，采用哪种架构设计？所采用的客户端支持哪些？对数据的吞吐能力有什么指标？从设计上不光要分析出显式设计，还要主动思考如果这样设计的话会不会出现别的问题，和设计人员积极沟通后，会得到一些隐式设计（通常这部分涉及 UI）。这些需求汇总可以帮助我们增加测试的覆盖面。

3、单元/接口测试，这个对应着编码阶段。应用软件一般是后台代码先完成，这时可以由研发或测试开发工程师进行单元测试的执行，如 CodeReview。代码无硬伤后，测试开发工程师可以着手进行接口测试框架的封装，测试代码的编写，可以配合相关平台进行每日构建与接口自动化测试。前台代码完成后，可以对已稳定的 UI 控件进行识别，关键字封装，数据驱动等一系列 UI 级别自动化测试架构的封装。

4、系统版本提测，相信经过前面几个阶段，后台的问题基本已经得到了及时的解决，在 UI 设计阶段我们也已完成 UI 基本自动化的封装，在版本提测后在这里可以重点关注 UI 变化，同时 UI 自动化测试验证主线流程，且增加对 UI 页面易用性，兼容性等方向的测试。毕竟让最终用户用着“舒服”也是产品价值的一种体现。

测试左移思想是“第一时间发现缺陷，第一时间解决”，目的是在测试过程中降低缺陷修复的成本，且在测试后期可以更灵活的进行测试资源再分配。尽管如此许多测试人员可能认为会这样看起来会增加工作量，但这些工作量和产品价值的实现而言，还是有意义的，毕竟产品最终稳定的交付对测试人员来说就是一份最好的考核结果，而且这种思想还可以给我们测试团队带来“隐形”的收益：

1、测试“核心”价值的体现：测试人员不仅仅是一个质量的把关员，他可以存在产品整个生命周期内，提供专业的解决方案让产品更好的实现商业价值。

2、编码能力的提升：测试人员不再是“点点点”的代言人，同样可以编写代码来实现产品的高效测试，如自动化，性能，安全测试等。

3、创新习惯养成：需求分析多了，对产品功能有了深度的认识，可以着手于创新项目如测试平台，工具，流程推广等高级测试技能的实现。

尽管测试左移的的思想并不是一种很“新潮”的理念，但据笔观察一些公司甚至是一些知名公司并没有很好贯彻和执行这种思想，无论是测试管理人员还是测试执行人员都要有意识的将这些优秀的思想落地，并积极思考团队的未来，逐渐提升或加强测试团队在公司的“核心”价值。

成功的测试自动化执行需要的 5 个支柱

◆译者：于芳

概述:

有关什么构成了一个对测试自动化的“合适执行”的讨论常常聚焦于应当使用什么工具，但是那只是等式的一部分。巴斯德基科斯特拉详列了四件应当考虑的事情，以及他们怎样有益于测试自动化的成功，和没有对这四件事的其中一件进行合适的关注所带来的风险。

对于期望快速发布质量的机构来说，运行自动化测试是软件开发生命周期的一个重要部分。然而，测试自动化只有在合理恰当执行的情况下才能成功。对于什么构成了测试自动化的恰当执行的讨论常常聚焦在应当使用什么工具来做这项工作，或者使用最好的（即使已经有了这件东西）或者最有效的方式来使用特定的工具来完成给定的任务。

在我看来，尽管使用的工具是整个测试自动化方程的一部分。任何成功的测试自动化指向都是基于五个不同的部分建立的。

在本文中，我们会来看一下这些部分的每个部分，他们怎样助益于你的测试自动化执行使之成功，以及没有好好关注他们的任何一项所会带来的风险。

1、测试自动化工具

测试自动化工具虽然不是测试自动化执行得以成功的唯一因素，工具显然对你自动化工作的整体结果有着重要的影响。选择一个与你要测试的程序不够兼容的工具，或者不符合你们自动化团队所需要的技能集将很可能会导致不理想的结果。

然而，比工具的选择更重要的是，问自己你到底想用什么来完成你的自动化测试，然后决定一个最有效的达成结果的方式。一个首要的问题需要问的是一个特定的功能片或商业逻辑需要在什么层面上进行验证。

你想确保你的顾客可以打开你们的线上商店，搜索一个具体的产品，然后进行下单

付款的操作吗？你很可能想要使用端对端的用户接口驱动的测试来检验这一点。如果你在验证一个决定顾客是否被允许来购买给定的无品（比如，由于国家层面的规制）的逻辑片的正确性，那么你很可能需要能够编写潜入进较低级别的待测程序的测试脚本，例如一个接口甚至一个单独的代码类。这构成了一个针对该测试的不一样的范围和方法，因此，需要一个不同的工具。

简而言之，确保首先你知道要做自动化测试的程序需要验证什么，然后再花费时间去研究怎样取得想要的结果。记住迫使工具去做不是它被设计来做的事情会有很大的风险。

2、测试数据

对于任何严肃的测试自动化方案的另外一个重要因素是采用的管理测试数据的方法。测试的范围越广，测试数据管理就相应地变得越重要和要求高。

在单元测试里，你可以通过模拟所有你的测试依靠的数据来过活，而当你开始启动集成或端对端的测试时，你会需要有特定的数据来做测试。而且，为了让事情更复杂化，你常会需要其他与你的相互连接的程序进行交互的数据处于一个特定的状态。

- 对于这些类型的测试需要以下几种方式来处理测试数据
- 在测试建立阶段创建需要的测试数据
- 开始测试之前询问系统里已存在的测试数据
- 测试执行之前初始化待测程序的数据库

这里的每个方法都有其潜在的陷阱:

在测试建立阶段创建测试数据增加了测试执行时间，增加了在测试执行开始之前的失败风险，还会导致很多无用的测试数据如果没有合理的数据清理程序的话。

当你在测试开始之前查询系统里已存在的测试数据时，因为意外地使用了无效的测试数据或系统里的测试数据属性不对而导致了风险。

在测试执行前初始化数据库给你留下了数据库的截图来管理和保持数据最新—也就是说，你甚至需要首先被允许执行数据库复原程序。

助益没有正确的处理集成和断对端测试的测试数据的方法。然而，选择错的程序，或者根本没提出测试数据的问题，将很可能会导致产出的测试自动化方案不可重用，不

可维护或扩展性不好。

3、测试环境

巨石在快速地挡住恐龙的路。现在的 IT 系统由多种相互连接的组件，服务和程序组合在一起来交付传递商业价值。然而，对于测试，这不总是好消息：必须为了可能引起很多头疼、沮丧以及测试时间延时的集成和端对端测试去管理和依仗于依赖性的可行性，尤其是那些在你的掌控圈范围之外的东西。还有，可靠的和可管理的测试环境是当你向创建和使用自动化测试作为测试方法的一部分的关键。

降低失败或不存在的测试环境风险的一个方法是在测试环境中使用类似于桩、模拟和服务可视化来复制关键却难以访问的依赖项的模拟技术。有足够模拟实际的依赖项行为的模拟器来完成你想要执行的测试用例可以极大地加速自动化测试的执行—因此，加速开发的速度。

更进一步，当虚拟环境被恰当地搭建起来（例如，通过平衡容器集装箱化），重建一个相同的测试环境的新实例，用相同的测试数据和其他特征来完成测试，使得从自动化执行转化到真正的持续性测试成为可能，这又反之成为一个前提如果你在期望采用持续性发布作为更灵活对增加的市场需求更好回应的方法的话。

4、报告

一次自动化测试执行产生的结果应当是任何一个严固地自动化测试方法的必不可少的部分。创建好的测试结果报告常常被忽略，然而却是在任何一个测试自动化项目里潜在的节省时间的任务。好的汇报报告不仅仅是展示测试执行的数量，通过数量，失败数据，当然有这些比没有好得多。

- 对于一个测试执行报告要使之真的有价值的话，需要让那些测试在被执行可视化（助益使用清晰不模糊的方式来命名测试用例是任何好的报告的基础），不仅是结果如何（成功或失败），还有在哪个地方失败的测试用例出现了错误，并尽可能详细地记录下细节。
- 这与仅仅通过拷贝随便什么东西和所有东西到你的测试报告里来提高那个一个信息过载是不同的—那将会不必要地延误找到测试失败的根源。一个好的报告显示有些地方出错，在测试的哪个部分出现错误（在哪个步骤），错误信息是什么（取决于你的报告的听众，这可以简单如一个轨迹跟踪，但是在其他情形下你可能需要提供可供非

技术人员也能看懂的错误信息），和待测程序在失败时刻的状态（例如，使用截图来做用户接口驱动的测试）。

助益一个良好的报告测试可能需要对每次测试执行创建不止一个报告。如果你的测试时持续发布构建流水的一部分，你可能想要创建低级别的能够由你的构建引擎编译的报告来决定该构建是否可继续。但是你可能也想创建一个用 HTML 格式的可读的报告，使用文本化的描述来完成测试目的，同样有人性化的信息和截图在遇到失败的测试时使用。这都取决于听众是谁。

5、技术

最后，也是争议最重要的难题之一是，创建一个强大高效的测试自动化方案是负责执行自动化测试的人。没有技能娴熟的自动化咨询师，架构师，工程师和开发人员来关注本文提到的测试自动化其他方面，你很可能很快就玩完了。

你的测试自动化团队理想地应当是在测试领域有所建树，同样在软件开发领域技能颇丰，这样可以回答为什么测试自动化会首先是一个合适的解决方案以及什么测试应当被自动化，这样意味着他们知道怎样创建一个机枪打又可维护的测试自动化执行方案。这不是指每个测试自动化团队的成员都需要在这两个区域技能娴熟，但是从整体来看，你的团队需要永远一个健康平衡两方面能力的成员以更好的交付产品。

把东西放到一起

一个好的测试自动化方案需要更多地考虑驱动测试的工具。要让自动化真正成功，你需要多多揣摩测试数据策略，如何管理测试环境，和你向观众呈现自动化测试执行的结果。然而，最重要的是，这是有关组建一个知道怎样做到以上这些的人的团队的事情。

学霸的测试之路

◆ 作者：含羞草

一、前言

时间在一分一秒的流逝，在你谈笑风生时，在你逛街休闲时，在你游戏打闹时，甚至在你吃饭睡觉时。你是否也曾感叹自己一直碌碌无为，你是否也曾回想过去的光辉岁月，你是否也曾感伤现实的残酷与人情的冷漠。对于逝去的青春，尽管我们有悲欢离合，有阴晴圆缺，有喜怒哀乐，过去终将是过去，再也不能重新来过。我们能做的就是从过去的青春岁月中自我反省、积累经验、提升自己，从而将这些岁月痕迹转化成能力，形成自己的特色，才能展望未来，建立属于自己的品牌。

很荣幸一直以来 51testing 软件测试网站的陪伴，回想一下距离我在 51testing 软件测试网站进行第一次文章投稿已将近一年时间。在这一年里，我总共参与了 5 期的文章投稿，大大小小地总结了一年时间里的测试学习历程。感谢 51testing 软件测试网站这样一个平台，让我有机会与大家一起分享一些测试方面的学习心得。分享是进步的源泉，一方面我们将自己的学习笔记与心得体会分享给别人，让别人能有所收获，少走一些弯路；另一方面我们也能从别人分享的文章中了解和学习别人的一些经验，不断进行自我完善。

回顾这几期的投稿主题，都是关于测试技术方面的学习笔记，确实自己也在不断地学习，一直在不断努力，渴望在测试方面有所成长。而这些学习笔记就见证了自己在测试方面的成长，在岁月的长河中留下了我的痕迹。一直以来，都不希望自己的人生平平淡淡，总想着留下点什么。可是留下什么呢？一直也没有特别的才能，从小到大一直为学业而奋斗，感觉除了学习没有什么其他的业余爱好。在结束了学生时代之后，希望能更多地培养业余爱好，慢慢将爱好发展成事业。希望发表文章会是一个好的开端。

二、测试历程

1、 软件测试理论的学习

对于测试理论的学习，主要侧重于软件开发模型的学习以及软件测试方法的学习。以下主要对这两方面进行介绍:

1） 软件开发模型

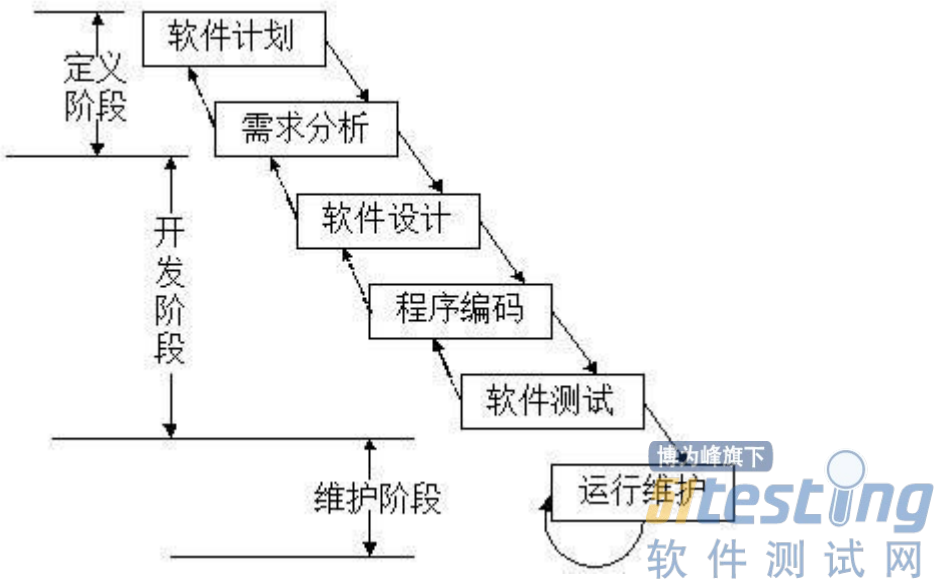
随着互联网技术的不断发展，测试越来越受企业的重视，那么如何学习测试技术呢？我认为：首先要熟悉软件开发模型。说到软件开发模型，不得不谈到以下两个经典的开发模型：瀑布模型和 V 模型。

（1）瀑布模型

瀑布模型，顾名思义就是将软件生存周期的各项活动规定为按固定顺序而连接的若干阶段工作，形如瀑布流水，最终得到软件产品。

瀑布模型核心思想是按工序将问题化简，将功能的实现与设计分开，便于分工协作，即采用结构化的分析与设计方法将逻辑实现与物理实现分开。将软件生命周期划分为制定计划、需求分析、软件设计、程序编写、软件测试和运行维护等六个基本活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。

具体的瀑布模型示意图如下所示:



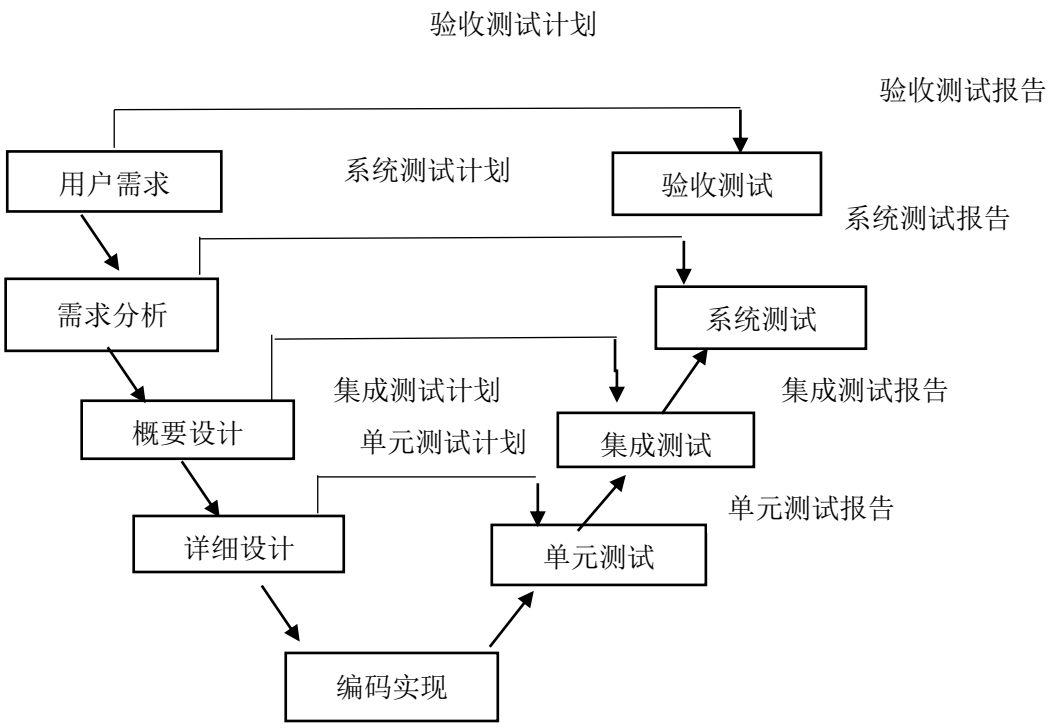
在瀑布模型中，测试是软件开发的最后阶段，当测试发现了需求方面的问题时，可能导致工作产品的大量返工，产品交付可能因此延期。这在现实的测试工作中是经常遇到的。

（2）V 模型

V 模型是软件开发过程中的一个重要模型，由于其模型构图形似字母 V，所以又称软件测试的 V 模型。V 模型的全称实际为 RAD（Rap Application Development，快速应用开发）模型。

V 模型的核心思想是通过开发和测试同时进行的方式来缩短开发周期，提高开发效率。将 V 模型大体可以划分为以下几个不同的阶段步骤：需求分析、概要设计、详细设计、软件编码、单元测试、集成测试、系统测试、验收测试。

具体的 V 模型示意图如下所示：



在 V 模型中，软件测试从项目需求分析阶段开始，贯穿于整个软件开发过程活动中。测试人员能够尽早进入项目，熟悉产品。对设计出高质量的测试用例非常有帮助。同时，如果早期能够发现更多缺陷，有利于大幅度降低成本。

具体的各个测试阶段特点如下：

- 单元测试

单元测试是对软件中的基本组成单位进行的测试，如一个模块、一个过程等等。它是软件动态测试的最基本的部分，也是最重要的部分之一，其目的是检验软件基本组成单位的正确性。因为单元测试需要知道内部程序设计和编码的细节知识，一般应由程序

员而非测试员来完成，往往需要开发测试驱动模块和桩模块来辅助完成单元测试。因此应用系统有一个设计很好的体系结构就显得尤为重要。

一个软件单元的正确性是相对于该单元的规约而言的。因此，单元测试以被测试单位的规约为基准。单元测试的主要方法有控制流测试、数据流测试、排错测试、分域测试等等。

- 集成测试

集成测试（也叫组装测试，联合测试）是单元测试的逻辑扩展，是在软件系统集成过程中所进行的测试。其主要目的是检查软件单位之间的接口是否正确。它根据集成测试计划，一边将模块或其他软件单位组合成越来越大的系统，一边运行该系统，以分析所组成的系统是否正确，各组成部分是否合拍。实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。一些局部反映不出来的问题，在全局上很可能暴露出来。集成测试的策略主要有自顶向下和自底向上两种。具体两种集成测试策略的优缺点就不在此处描述，有兴趣的可以自己升入探讨。

- 系统测试

系统测试是针对软件产品系统进行的测试，主要验证整机系统是否满足了系统需求规格的定义。系统测试是将通过确认测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他元素结合在一起，在实际运行的环境下，对计算机系统进行的测试。软件系统测试方法很多，主要有功能测试、性能测试、强度测试、安全测试、其他测试等等。各种测试的差异在此处不做赘述，有兴趣的伙伴可以查阅相关文档进行学习探讨。

- 验收测试

验收测试旨在向软件的购买者展示该软件系统满足其用户的需求。它的测试数据通常是系统测试的测试数据的子集。所不同的是，验收测试常常有软件系统的购买者代表在现场，甚至是在软件安装使用的现场。这是软件在投入使用之前的最后测试。

- 回归测试

回归测试是指当测试人员提出 bug，开发人员针对 bug 修改了旧代码后，需要测试人员重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。回归测试主要验证两个方面的内容：一是所做的修改没有引入新的错误，没有在原来 bug 的基础上使

系统增加新的 bug；二是原来发现的 bug 在进行修复之后已经不存在了。

回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。因此，通过选择正确的回归测试策略来改进回归测试的效率和有效性是很有意义的。正是由于回归测试的重要性，所以对回归测试来说自动化测试是必要的。测试的自动化程度越高，回归的周期就越短，效果越明显，软件的质量也越高。回归测试自动化一直是测试人员以及管理者的期望所在。

2） 软件测试方法

谈到软件测试方法，不得不谈到三种基本的软件测试方法：白盒测试、黑盒测试、灰盒测试。一下简单介绍这三种测试方法的特点：

- 白盒测试

白盒测试也称结构测试或逻辑驱动测试，是指基于一个应用代码的内部逻辑知识，即基于覆盖全部代码、分支、路径、条件的测试，它是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于软件验证。

白盒测试需要全面了解程序内部逻辑结构，对所有逻辑路径进行测试。白盒测试是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字。但即使每条路径都测试了仍然可能有错误。第一，穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序。第二，穷举路径测试不可能查出程序中因遗漏路径而出错。第三，穷举路径测试可能发现不了一些与数据相关的错误。

白盒测试可以借助一些工具来完成如：Junit Framework、Jtest 等。白盒测试方法群：同行评审、需求审查、代码审查、接口测试（调用测试和返回测试，需要结合等价类和因果图方法）等。

- 黑盒测试

黑盒测试是指不基于内部设计和代码而基于需求和功能性的测试，黑盒测试也称功能测试或数据驱动测试，它是在已知产品所应具有的功能，通过测试来检测每个功能是

否都能正常使用，在测试时，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，测试者在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息，并且保持外部信息（如数据库或文件）的完整性。黑盒测试方法主要有等价类划分、边值分析、因—果图、错误推测等，主要用于软件确认测试。

黑盒测试主要着眼于程序外部结构、不考虑内部逻辑结构、针对软件界面和软件功能进行测试。黑盒测试是穷举输入测试，只有把所有可能的输入都作为测试情况使用，才能以这种方法查出程序中所有的错误。实际上测试情况有无穷多个，人们不仅要测试所有合法的输入，而且还要对那些不合法但是可能的输入进行测试。

软件测试的方法根据软件工程的组织和实现方式，有很大差别，有些是比较技术化的方法，有些则是工程方法，主要分为：黑盒测试方法群：等价类划分、边界值、因果图、基路径法、专家测试法、smoking、场景测试等。

- 灰盒测试

灰盒测试，是介于白盒测试与黑盒测试之间的测试。可以这样理解，灰盒测试关注输出对于输入的正确性，同时也关注内部表现。但这种关注不像白盒那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态，有时候输出是正确的，但内部其实已经错误了，这种情况非常多，如果每次都通过白盒测试来操作，效率会很低，因此需要采取这样的一种灰盒的方法。

2、 软件测试类型

软件测试根据需要会有不同的类型和场景，主要包括以下几中类型：功能测试、性能测试、安全测试、数据库测试、自动化测试等等。以下简单概括以下各个类型的测试工作。

功能测试是相对基础的软件测试，类似于黑盒测试，不需要了解系统内部设计和代码，根据需求进行的测试。这也是软件测试行业最初级的测试。一般刚进入软件测试行业，涉及最多的就是功能测试。这一阶段主要的任务就是熟悉需求文档，根据需求文档编写测试用例，执行测试用例，最后将 bug 进行反馈至 bug 管理系统进行修复，修复完成之后进行回归测试。这一切的过程都是手动进行的。

性能测试是利用测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性

能指标进行测试。负载测试和压力测试都属于性能测试，两者可以结合进行。我们可以通过负载测试确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。压力测试是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。进行这方面的测试需要了解整个待测试系统的拓扑结构，了解整个系统的压力所在，对服务器进行相应的压力测试和负载测试。最终生成测试报告，通过分析测试报告数据实现系统的优化。

安全测试是在 IT 软件产品的生命周期中，特别是产品开发基本完成到发布阶段，对产品进行检验以验证产品符合安全需求定义和产品质量标准的过程。随着大数据时代的发展，数据安全越来越受企业的重视。一个完整的 web 安全体系测试可以从部署与基础结构，输入验证，身份验证，授权，配置管理，敏感数据，会话管理，加密，参数操作，异常管理，审核和日志记录等几个方面入手。安全性漏洞种类主要包括：SQL 注入、跨站式攻击、文件上传漏洞、跨站请求伪造等等，至于每一种类的特点，可以参考我的相关文章《安全测试学习总结》。这篇文章详细介绍了各个种类的特点以及如何进行防御。

数据库测试是依据数据库设计规范对软件系统的数据库结构、数据表及其之间的数据调用关系进行的测试。无论是在 Web、桌面应用、客户端服务器、企业和个人业务，都需要数据库在后端操作。同样的在金融、租赁、零售、邮寄、医疗领域中，数据库也是不可缺少的。从测试过程的角度来说我们也可以把数据库测试分为：单元测试、集成测试、系统测试。单元测试侧重于逻辑覆盖，相对于复杂的代码来说，数据库开发的单元测试相对简单些，可以通过语句覆盖和走读的方式完成。集成测试主要是针对接口进行的测试，对数据库测试来说，需要考虑的是数据项的修改操作、数据项的增加操作、数据项的删除操作、数据表增加满、数据表删除空、删除空表中的记录、数据表的并发操作、针对存储过程的接口测试、结合业务逻辑做关联表的接口测试。

自动化测试是把人从繁琐的手工测试的重复性的工作中解放出来，是把以人为驱动的测试行为转化为机器执行的一种过程。通常，在设计了测试用例并通过评审之后，由测试人员根据测试用例中描述的规程执行测试，得到实际结果与期望结果的比较。在此过程中，为了节省人力、时间或硬件资源，提高测试效率，便引入了自动化测试的概念。不是所有的产品都适合进行自动化测试，这就需要测试人员在进行自动化测试之前对产品进行需求分析。根据产品需求判断是否适合进行自动化测试，设计出自动化测试用例，从而搭建自动化测试的框架。

三、面对的问题

1、 测试过程中经常进行重复性测试

功能测试是基础的黑盒测试，不需要了解具体的代码，只需要验证页面功能、页面样式以及数据的可靠性、浏览器的兼容性。这期间存在大量的重复性工作，比如回归测试、UI 测试都是重复性比较高的测试。这就需要我们设计的测试用例能最大范围的覆盖测试模块。对于重复性比较高的测试工作，需要我们利用一些工具帮助我们进行测试。例如 QTP 进行自动化测试，通过录制脚本，修改脚本实现测试的自动化。

2、 性能测试工具 LoadRunner 的安装与破解

通常我们需要对系统的负载、压力进行测试，就需要使用性能测试工具对一些系统指标进行监控，通过这些指标的参数，分析系统的性能，例如 LoadRunner。但是一般对于这种商业性测试工具，不能长期使用，需要进行安装破解。关于 LoadRunner 的一些详细学习与介绍可以参考之前我的文章关于 LoadRunner 的学习笔记，相信在学习过程中能帮助你排疑解惑，弄明白一些基本的问题。

3、 测试工作比较枯燥乏味

相对于开发与设计，测试工作相对枯燥、乏味，因为开发与设计都会有成就感，但是测试工作就显得单调一些，测试人员一直在给开发找 bug，完善系统。测试过程中会遇到许多问题，我们需要的是耐心、恒心还有信心。要把枯燥的事情变得有趣，要从枯燥乏味中找寻乐趣。我认为最好的方法就是给自己安排学习计划，穿插着测试与学习，这样既能不断提高测试技能，又能完成测试任务，提出问题，开发解决问题从而改善系统的体验。

四、学到的经验

1、 明确自己的目标

任何行业任何职位都有其存在的价值，不能轻视任何职位的发展前景。测试也是如此，随着互联网技术的发展，测试也越来越受公司与企业的重视。当然测试人员自身也不能放松对自己的要求，否则技术很容易跟不上时代的发展需要，自身的地位也得不到捍卫，自身价值也得不到体现。

不管别人怎么看待测试工作，测试人员一定要认清自己的方向，找准自己的定位，

否则会一直停留在功能测试手动测试的阶段停滞不前。如果不想公司里的人看清自己，就需要将自己的优势发挥到极致，让自己在公司变得不可替代。只有被需要的工作才是最值得付出的工作，只有被公司需要的能力才是无可替代的能力。

在 2 年的测试生涯当中，我也曾经动摇自己的目标，是不是应该去做开发或者设计，曾经还在开发上花费相当多的时间去学习。可是后来想想，技术不是目的，技术无穷尽，没有人能完全掌握所有的技术，关键是处理问题的能力、与人沟通的能力。不管世界再怎么变化，希望我们最终都能不忘初心，想想最初的理想与愿景。

2、 只相信自己看到的

在很多情况下，开发或者领导会告诉你“这个功能很小，简单测一下就好了”等等的话。在我看来，只相信自己看到的。对于测试而言，不管别人说了什么，不能不信也不能完全相信。所有的系统不可能做到绝对没有 bug，我们能做的就是亲自测试验证。不管开发怎么说怎么做，我们都应该始终保持认真严谨的态度，不轻易放过一个小功能。不能因为某个功能小而轻视对该功能的测试，也不能因为别人的言语改变自己的测试原则与方法。

只相信自己看到的，一切没有经过自己的测试验证的说法都是没有依据且值得被怀疑的。测试人员始终要以怀疑的眼光看待每一个功能点，并且制造出各种出乎意料的数据验证功能的可靠性、正确性。测试不怕出错，就怕不出错。如果我们验证各个功能都没有发现错误，不能说明系统完善，只能说明测试用例不够全面，测试数据不具有代表性。

在上线之前发现的 bug 都会是我们走向成功的印记，就怕上线之前发现不了 bug，上线之后一堆 bug。这才是测试人员以及公司的悲哀。

3、 干一行爱一行敬一行

既然选择了远方，就应该风雨兼程。既然选择了测试这样一个职业，就应该尊重这个职位，热爱这个职位并将其作为终身事业去做，勇于承担责任、勇于挑战自己，进行潜能开发。

测试这个职位看似简单，想要做好做精也是有一定难度的，一方面需要技术的支持，另一方面需要对项目的全局管理、把控。但是不管作为测试的你在公司充当一个什么角色，从事什么样的测试工作，你都应该相信你是独一无二、不可替代的。你需要继续提

高自己的测试技能，树立在公司的威信，形成自己的测试风格与方法。不管做什么都要爱一行敬一行，从工作中找寻到乐趣，从而更好得享受生活。

4、 要敢于承担责任

俗话说的好：心有多大舞台就有多宽。在测试生涯中不要怕承担责任，要知道你承担多大责任就决定你在公司的重要性。对于日常的测试过程当中，要勇于表现自己的能力，让领导关注我们。当然勇于承担责任的前提就是自己的测试技术过硬，能解决大多数问题。即使自己解决不了问题也能促进问题的解决，这样的工作能力才是受大家尊重的。

勇于承担责任还督促我们在测试的道路上不断提升自身的技能，多与其他同事进行沟通交流。有时候就是需要逼自己，不逼自己有时候真不到自己到底有多大的潜能。包括我自己，不是吃不了苦，最怕的就是太安逸的环境下找不到方向。吃苦是小，吃苦获得的效果是大。如果花费的时间没有在某些方面发挥到作用，那么对于我们而言就会打击自信心、积极性。

五、测试之路小结

1、 身心具备

任何道路不可能一直一帆风顺，所以需要做好两方面的准备。一方面做好心理准备：准备着吃苦耐劳，准备着战胜枯燥与繁琐。如果刚入这行就没有做好心理准备，在测试过程中很容易缺少耐心。面对重复性的测试工作，感觉枯燥无味，看不见希望。另一方面做好终身学习的准备：准备着将精力集中在工作当中，不断提升自己的各方面技能，准备着接触更多成功人士，不断提升自己的个人气质与精神素养。

测试道路上会遇到很多各个方面的问题，遇到问题其实并不可怕，关键是怎么解决问题，从解决各个问题当中提升自己处理问题的能力。而我们与成功人士的最大差别就在于处理问题的发散性思维，遇到问题没有解决问题的思维方式，只能停留在遇到问题的忧虑当中。不管你在测试的哪一个职位上，希望你能全身心投入到你所从事的行业，为公司创造价值，这样你才能实现自己的价值。

2、 没有最好只有更好

在测试的道路上没有最好的系统，任何系统都会存在一些大大小小的问题，对于测试的要求也是。不能不在乎一些小问题，也不能将小问题抓太紧。这当中需要有一个度

的问题，把握好这个度的话，就能在公司当中建立自己的威信，成为不可替代的一员。过了这个度的话，就会被认为是固执，古板，不懂得变通。所以在测试道路上的你们，希望能做最好的自己，为自己的测试生涯留下奋斗的足迹。要知道，一切皆有可能，要敢想敢做敢于挑战自己，挖掘自己的潜能。

六、后记

很高兴能在这里与大家分享我的测试历程，曾经是学霸的我相信在工作当中也能崭露头角，寻找自己想要的人生。不管未来如何，希望大家一旦做出了选择，就要坚持不懈的走下去。遇到什么问题，我们来解决问题。不能半途而废，否则将会意识无成。越努力越幸运！希望我的总结能帮助大家确定测试方向，少走一些弯路，找到自己的方向，更快地实现自己的目标。

❖ 拓展学习

- 如何开展软件测试：http://www.atstudy.com/course/277
- 性能测试与 LoadRunner 基础：http://www.atstudy.com/course/7