软件测试教程 自动化测试selenium篇（一）

本课程主要讲述以下内容：

> 什么是自动化测试
>
> 如何实施自动化测试
>
> 发展方向
>
> 什么是selenium
>
> 使用Selenium IDE录制脚本
>
> selenium+python环境搭建

什么是自动化测试

自动化测试介绍

自动化测试指软件测试的自动化，在预设状态下运行应用程序或者系统，预设条件包括正常和异常，最后评估运行结果。将人为驱动的测试行为转化为机器执行的过程。

自动化测试金字塔

自动化测试包括UI自动化，接口自动化，单元测试自动化。按照这个金字塔模型来进行自动化测试规划，可以产生最佳的自贡话测试产出投入比（ROI），可以用较少的投入获得很好的收益。

单元测试

最大的投入应该在单元测试上，单元测试运行的频率也更加高。

java的单元测试框架是Junit，之前讲过，在这里不再赘述。

接口自动化

接口测试就是API测试，相对于UI自动化API自动化更加容易实现，执行起来也更稳定。

接口自动化的有以下特点：

- 可在产品前期，接口完成后介入

- 用例维护量小
- 适合接口变动较小，界面变动频繁的项目

常见的接口自动化测试工具有，RobotFramework，JMeter，SoapUI，TestNG+HttpClient，Postman等。

UI自动化

虽然测试金字塔告诉我们尽量多做API层的自动化测试，但是UI层的自动化测试更加贴近用户的需求和软件系统的实际业务。并且有时候我们不得不进行UI层的测试。

UI自动化的特点：

- 用例维护量大
- 页面相关性强，必须后期项目页面开发完成后介入
- UI测试适合与界面变动较小的项目

UI自动化测试的好处

降低大型系统的由于变更或者多期开发引起的大量的回归测试的人力投入，这可能是自动化测试最主要的任务，特别是在程序修改比较频繁，效果是非常明显的，自动化测试前期人力投入较多，但后期进入维护期后，可节省大量人力，而手工测试后期需要增加大量人力用于回归测试

减少重复测试的时间，实现快速回归测试

创建优良可靠的测试过程，减少人为错误

可以运行更多更繁琐的测试

可以执行一些手工测试困难或不可能进行的测试

更好的利用资源

测试脚本的重用性

UI层自动化测试框架

UI层的测试框架比较多，比如Windows客户端测试的AutoIT，web测试的selenium以及TestPlant eggPlant，Robot framework，QTP等。

测试课程里我们主要以Web UI自动化测试框架Selenium为例进行详细介绍。selenium有以下优点：

- 免费，也不用再为破解软件而大伤脑筋
- 小巧，对于不同的语言它只是一个包而已，而QTP 需要下载安装1个多G 的程序。
- 这也是最重要的一点，不管你以前更熟悉C、java、ruby、python、或都是C# ，你都 可以通过selenium 完成自动化测试，而QTP 只支持VBS
- 支持多平台：windows、linux、MAC ，支持多浏览器：ie、ff、safari、opera、chrome
- 支持分布式测试用例的执行，可以把测试用例分布到不同的测试机器执行，相当于分 发机的功能。

UI自动化测试的适用对象

实施自动化测试的前提条件：需求变动不频繁、项目周期足够长、自动化测试脚本可重复使用。

适合做自动化的项目：

1、产品型项目。产品型的项目，新版本是在旧版本的基础上进行改进，功能变不大的项目，但项目的新老功能都必须重复的进行回归测试。回归测试是自动化测试的强项，它能够很好的验证你是否引入了新的缺陷，老的缺陷是否修改过来了。在某种程度上可以把自动化测试工具叫做回归测试工具。

2、机械并频繁的测试。每次需要输入相同、大量的一些数据，并且在一个项目中运行的周期比较长。比如兼容性测试。

以下情况的项目不适合做自动化测试：

1、需求变动频繁的项目，自动化脚本不能重复使用，维护成本太大，性价比低

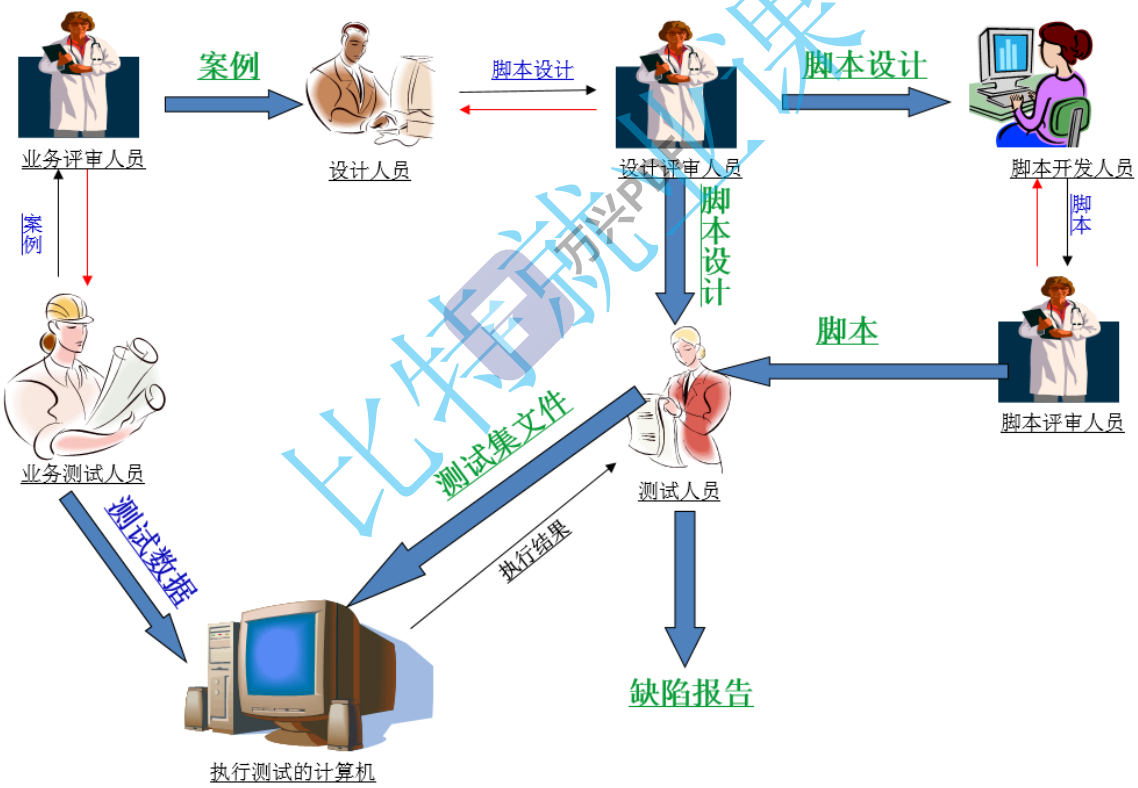
2、项目周期短，自动化脚本编制完成后使用次数不多，性价比低

3、交互型较强的项目，需要人工干预的项目，自动化无法实施

如何实施自动化测试

单纯的讲，自动化测试的具体实现，应该是包含下面七个过程的。

1. 分析：总体把握系统逻辑，分析出系统的核心体系架构。
2. 设计：设计测试用例，测试用例要足够明确和清晰，覆盖面广而精
3. 实现：实现脚本，有两个要求一是断言，二是合理的运用参数化。
4. 执行：执行脚本远远没有我们想象中那么简单。脚本执行过程中的异常需要我们仔细的去分析原因。
5. 总结：测试结果的分析，和测试过程的总结是自动化测试的关键。
6. 维护：自动化测试脚本的维护是一个难以解决但又必须要解决的问题。
7. 分析：在自动化测试过程中深刻的分析自动化用例的覆盖风险和脚本维护的成本。



自动化测试需要了解的技能

了解被测试系统的基本业务

了解业务的技术框架

懂得功能测试

懂得一种编程语言

懂数据库、操作系统

了解常见的测试框架

……

selenium介绍

Selenium是web应用中基于UI的自动化测试框架，支持多平台、多浏览器、多语言。

早期的selenium RC已经被现在的webDriver所替代，可以简单的理解为selenium1.0+webdriver构成现在的Selenium2.0。现在我们说起selenium，一般指的是Selenium2.0。它有由Selenium IDE，Webdriver，Selenium Grid组成。

分别做一下介绍：

1，Selenium IDE

Selenium IDE一个用于Selenium测试的完成集成开发环境，可以直接录制在浏览器的用户操作，并且能回放，编辑和调试测试脚本。调试过程中可以逐步进行或调整执行的速度，并且可以在底部浏览日志出错信息。

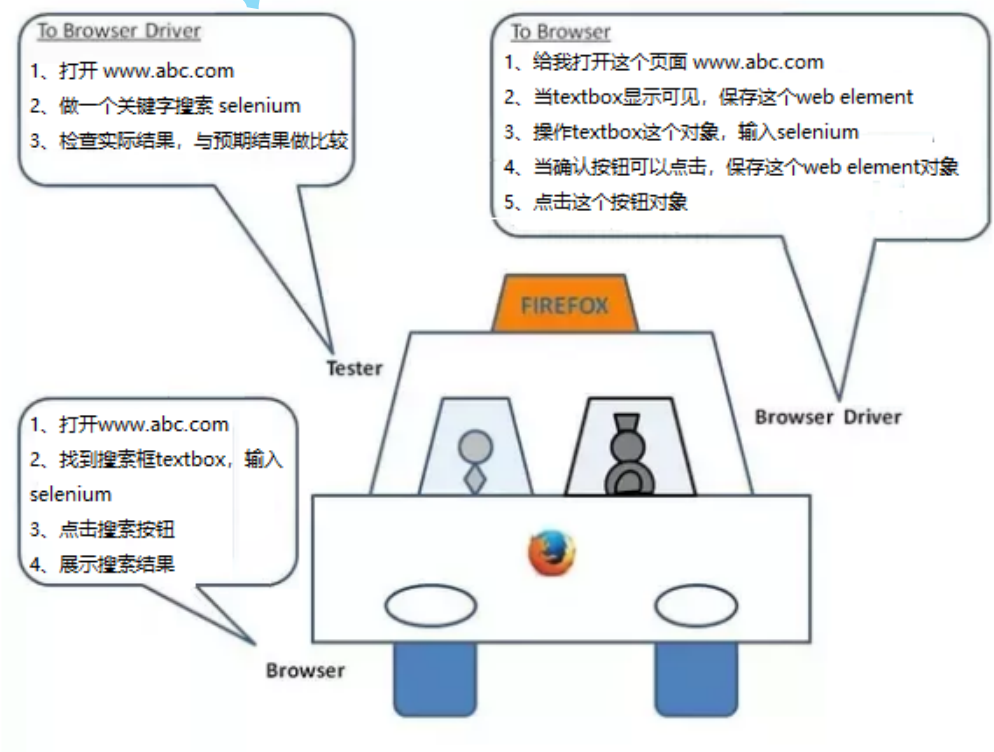
录制的测试脚本可以以多种语言导出，比如java，C#，Python，Ruby等，方便掌握不同语言的测试人员操作。

2，Webdriver

Selenium RC 在浏览器中运行 JavaScript 应用，会存在环境沙箱问题，而WebDriver可以跳出JavaScript的沙箱，针对不同的浏览器创建更健壮的，分布式的，跨平台的自动化测试脚本。基于特定语言（Java，C#，Python，Ruby，Perl，JavaScript等）绑定来驱动浏览器对Web元素进行操作和验证。

webdriver的工作原理：

- 启动浏览器后，selenium-webdriver会将目标浏览器绑定到特定的端口，启动后的浏览器则作为webdriver的remote server。
- 客户端(也就是测试脚本)，借助ComandExecutor发送HTTP请求给sever端（通信协议：The WebDriver Wire Protocol，在HTTP request的body中，会以WebDriver Wire协议规定的JSON格式的字符串来告诉Selenium我们希望浏览器接下来做什么事情）。
- Sever端需要依赖原生的浏览器组件，转化Web Service的命令为浏览器native的调用来完成操作。



3，selenium Grid

selenium Grid是一个服务器，提供对浏览器实例访问的服务器列表，管理各个节点的注册和状态信息。可以实现在同一时刻不同服务器上执行不同的测试脚本。

如何使用selenium IDE录制脚本

安装Selenium IDE

1、安装Firefox 17.0 - 40.*，因为firefox更新过快，selenium的不同版本对firefox的支持不同。

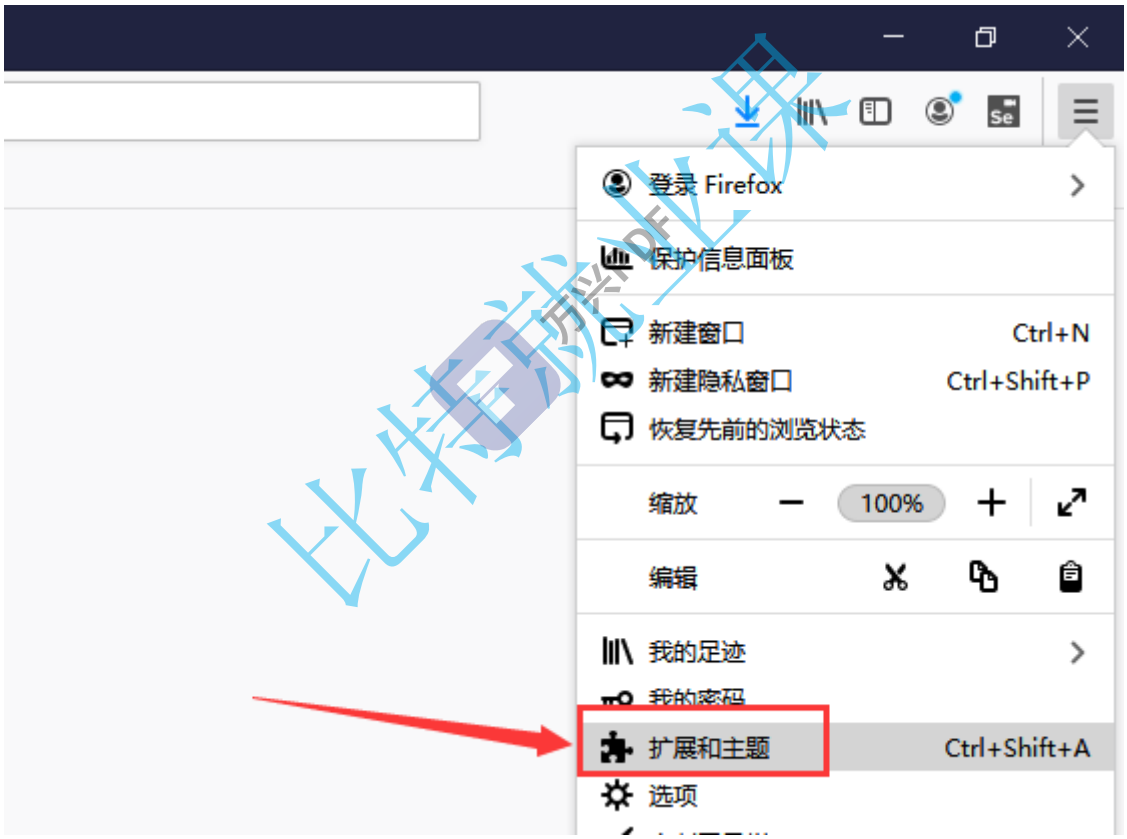
注意，安装完毕后，在选项菜单取消自动更新。为了兼容selenium以及ide，建议使用54版本。

下载地址：http://ftp.mozilla.org/pub/firefox/releases/

2、安装selenium IDE

下载地址：https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/

或者直接到火狐浏览器的扩展与主题中直接Selenium IDE搜索并下载



注意：IDE仅作为辅助工具来快速生成用例或者个人测试使用。实际很少用它来管理自动化测试用例。

如下以本机安装的禅道为例介绍IDE的使用。

1、打开Firefox-工具-选择selenium ide：

2、点击 File 菜单，弹出下拉列表，选择 New Test Case，此时左中部 Test Case 窗口会增加一个 Untitle 2 的测试案例，右键点击'Property'，在弹出窗口中重命名为"TestDemo"

3、点击 IDE 的右上部录制按钮（小红点）开始手动录制

4、在地址栏中输入待测试的网址(禅道)如http://127.0.0.1，输入用户名和密码，进行登录操作。这时可以看到IDE进行了录制操作。

5、在页面中点击右键，可以增加检查点。

)

6、录制结束后，点击录制按钮（小红点），结束本次手动录制。在selenium IDE 中，选中一个 Test Case，点击 File 菜单，下拉列表中选择“Export Test Case As”-“python2/ unittest/ WebDriver”;导出为Testdemo.py文件。

7、将该脚本在python中运行并调试。

```
# -*- coding: utf-8 -*-
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import NoAlertPresentException
import unittest, time, re

class Test(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.base_url = "http://127.0.0.1/"
        self.verificationErrors = []
        self.accept_next_alert = True

    def test_1(self):
        driver = self.driver
        driver.get(self.base_url + "/")
        driver.find_element_by_id("zentao").click()
        driver.find_element_by_id("account").clear()
        driver.find_element_by_id("account").send_keys("admin")
        driver.find_element_by_name("password").clear()
        driver.find_element_by_name("password").send_keys("Bitekeji2018")
        driver.find_element_by_id("submit").click()
        self.assertEqual("admin",
driver.find_element_by_link_text("admin").text)
        driver.find_element_by_link_text(u"退出").click()
```

```
    def is_element_present(self, how, what):
        try: self.driver.find_element(by=how, value=what)
        except NoSuchElementException as e: return False
        return True

    def is_alert_present(self):
        try: self.driver.switch_to_alert()
        except NoAlertPresentException as e: return False
        return True

    def close_alert_and_get_its_text(self):
        try:
            alert = self.driver.switch_to_alert()
            alert_text = alert.text
            if self.accept_next_alert:
                alert.accept()
            else:
                alert.dismiss()
            return alert_text
        finally: self.accept_next_alert = True

    def tearDown(self):
        self.driver.quit()
        self.assertEqual([], self.verificationErrors)

if __name__ == "__main__":
    unittest.main()
```

至此，我们就完成了一个test case的生成。

selenium+python环境搭建

1，下载和安装

下载 python http://python.org/getit/

- 安装python，双击安装包

  环境配置：

  方式一：在安装的时候直接勾选add python to path 选项；

  方式二：安装完成后，配置环境变量，在path中添加python安装路径。

- 安装setuptools

  打开cmd（开始---cmd 回车）

命令：pip install setuptools

- 安装selenium

  打开cmd（开始---cmd 回车）

  命令：pip install selenium

注意：如果直接下载不下来，可以使用镜像：

pip install selenium -i https://mirrors.aliyun.com/pypi/simple

有时候遇到pip不是最新的版本，也可以使用镜像更新一下：

python -m pip install --upgrade pip -i https://pypi.douban.com/simple

- 安装驱动

  火狐驱动 geckodriver.exe

  下载地址：https://github.com/mozilla/geckodriver/releases，请根据系统版本选择下载；（如Windows 64位系统） 下载解压后将getckodriver.exe复制到Python的安装目录Scripts文件夹下；

  安装谷歌驱动chrome driver

  下载地址：http://npm.taobao.org/mirrors/chromedriver/

  下载解压，你会得到一个chromedriver.exe 文件，放到安装Python的目录的Scripts文件夹下。

  只有安装了对应的driver才能运行对应的浏览器

建议python版本3..8

案例：

运行以下脚本：

```
# coding = utf-8
from selenium import webdriver
driver =webdriver.Firefox()
driver.get('http://www.baidu.com')
print driver.title
driver.quit()
```

结果如下：

```
Traceback (most recent call last):
  File "C:\Users\liujiey\Desktop\chandao.py", line 12, in setUp
    self.driver = webdriver.Firefox()
  File "C:\Python27\lib\site-packages\selenium\webdriver\firefox\webdriver.py",
line 152, in __init__
    self.service.start()
  File "C:\Python27\lib\site-packages\selenium\webdriver\common\service.py",
line 83, in start
    os.path.basename(self.path), self.start_error_message)
WebDriverException: Message: 'geckodriver' executable needs to be in PATH.
```

原因：没有安装对应的火狐浏览器的驱动