如何开展Web项目的UI自动化测试，方案如何？

作者：深圳51TESTING

组织建设：

建立自动化测试的组，理想状态下有4个人员，测试开发、中高级自动化测试工程师、2个初级自动化工程师；非理想的情况下，可能只有一个人。

- 测试开发：

  `基础答案`：自动化框架的建设，确定自动化框架的设计模式、第三方代码工具的封装、中间公共模块的设计和调用、测试用例、测试套件的管理和执行、测试报告和测试结果的输出（文件输出和邮件通知）

  `可选高级`：如果可能的话，需要搭建持续集成服务器（CI，Continuous Integration Server）的环境，进行持续交付和自动化的冒烟测试等。

  有自动化方案的实施经验、有开发背景、以及持续集成的背景等。

- 中高级自动化测试工程师：

  配合测试开发人员，实施测试框架的建设。主要负责中间公共模块的实现和实例化等，以及部分高难度和流程复杂的自动化用例脚本编写和调试等工作。

  有参与过自动化方案的建设、脚本编写经验丰富、会代码调试、懂Web测试等。

- 初级自动化测试工程师：
  根据中间公共模块的设计，进行实例化公共模块、方法组合，实现自动化用例脚本的编写。
  有计算机编程思维、有代码经验、可以读懂脚本和HTML等。

- **若只有我一个人：**
  首先实现自动化用例的维护和执行。在这个基础上不断的抽取实现公共模块的设计以及测试报告的生成等工作。通过经验的积累，以及后续人员的补充，早日做好自动化框架的建设工作。

技术选择

Selenium WebDriver、Python(unittest) Java(JUnit)、CI Server

技术方案：
选择`Python + Selenium`的技术方案。

首先技术工具是免费的，Python的工具用`PyCharm`社区版，Selenium的`WebDriver`是开源工具。利用比较简洁的Python语言进行自动化测试，对于人员的学习成本来讲比较实用，学习时间短，有优势。

另外Python自带的`unittest`单元测试框架可以很方便的实现自动化用例的设计和执行以及自动化用例套件的管理等任务。Python是纯面向对象的语言，后续也可以过渡到`Java + Selenium`进行更加丰富的自动化测试。
此外，可以选择`Jenkins`作为持续集成服务器，配合Python+Selenium的方案进行自动化冒烟测试。

适合自动化的项目模块：

1. 任务测试明确，不会频繁变动
2. 每日构建后的测试验证
3. 比较频繁的回归测试
4. 软件系统界面稳定，变动少
5. 需要在多平台上运行的相同测试案例、组合遍历型的测试、大量的重复任务
6. 软件维护周期长
7. 项目进度压力不太大
8. 被测软件系统开发比较规范，能够保证系统的可测试性
9. 具备大量的自动化测试平台
10. 测试人员具备较强的编程能力

硬件：
硬件的要求不高，主要需要独立的测试环境。另外测试人员用的电脑最好是Windows桌面操作系统，需要安装Firefox浏览器，避免47.0的最新版本。测试人员最好也使用Chrome浏览器辅助进行Web元素的定位。

Selenium 学习总结

Selenium IDE (火狐的插件)

1. 录制局部脚本、
2. 修改编辑脚本（插入命令、注释）
3. 导出脚本（支持 C#、Java、Python、Ruby）
    - 单元测试框架

- C#：`NUnit`
- Java: `JUnit` | `TestNG`
- Python：`unittest`
- Ruby : test-unit

1. 学习研究 WebDriver

Python + Selenium WebDriver

1. 搭建环境步骤 a 安装 python3.x（windows xp 不支持 python3.5+） b 设置环境变量 path（安装时候也可以解决 勾上 add python.ext to path） c 安装Selenium 用pip命令安装 pip install -U selenium 有无问题？
    1. 路径，pip可能定位不到，pip也可能定位到别的文件夹（perl） 方案：cd c:\python34\scripts
    2. 外网ip问题。交换机的原因，局域网里面是同一个ip，造成安装超时
    3. 火狐的版本 46.0以及以下
2. 普通的使用 定位方式 `by id`, `name`, `class_name`, `tag_name`, `css_selector`, `xpath`, `link_text`, `partial_link_text` 主要用的 是 id,name, css_selector, xpath, link_text

鼠标的操作
定位的问题：
有无问题？

```
a <frame>
    f1 = find_element_by_css_selector(frame)
    switch_to.frame(f1)
b <select>
    s1 = find_element_by_css_selector(select)
    ss1 = Select(s1)
    ss1.select_by_index(0)
    ss1.select_by_value("人事部")
c 编码问题 utf-8
d 时间等待问题 sleep(5)
```

单元测试框架
unittest

解决了什么问题？

```
unittest.TestCase
    前置条件
    清理
    测试过程步骤
    断言 assertEqual(期望值，实际结果，错误提示)
    运行测试 test_开头的方法
unittest.TestSuite
    addTest(xxx("test_batch_login"))
    自定义的添加测试用例，并执行
unittest.TextTestRunner
    run(suite)
    执行创建并维护好的测试套件

test_runner  test_suite  test_case
测试运行器     测试套件      测试用例
```

模块化操作

面向对象的实践：
1. 编写一个类，类里面描述公共的方法
2. 实例化这个类，调用这个类的方法
有无问题？
类的构造方法，需要传递 webdriver.Firefox()

```
    self.common = RanzhiCommon(self.browser, self.base_url)
```

数据驱动测试
读取csv，循环每一行数据进行操作
读取MySQL，用例存到MySQL中

1. 中文编码的问题。

```
    # coding="utf-8"
    读csv的时候，添加 encoding='utf-8'
    csv.reader(open("xxx.csv", "r", -1, encoding="utf-8")
```

2. 循环放到最外层
3. 文件路径在控制台读取不到的问题，需要用绝对路径来指向文件

封装WebDriver

- 避免第三方代码的威胁，防止大批量修改测试用例
- 节约人力的成本（如果封装以后，不需要每个自动化测试工程师都会WebDriver）
- 标准化自动化用例的操作，只需要调用公共的标准模块就好。