**Selenium**

1. CssSelector

Selenium官网的Document里极力推荐使用CSS locator，而不是XPath来定位元素，原因是CSS locator比XPath locator速度快，更简洁，正则表达式

没有xpath直接，如果页面有改动，大部分情况就要跟着改

1. webdriver原理

C/S模式设计，server就是打开浏览器（指定端口）， client就是测试的代码。

client 发送httpRequest给server的侦听端口

server根据原生的浏览器提供的api来实现本地化的调用。

（启动的时候用http， 中间用的协议： WebDriver Wire Protocol）

1. Selenium + TestNG 实现多线程并行测试（只支持java）

优点：减少了整体测试所花费的时间，验证被测系统多线程条件下代码的正确性

为什么不用： 1.测试强度不是很大，感觉不需要多线程，觉得unittest就不错，而且特别热爱python就自己写了一个框架。 2.不是很稳定，两浏览器执行的操作虽然分别在不同线程中，但因为引用的是同一个driver，因此会出现两浏览器相互跳跃操作的情况。运行时会相互干扰甚至死掉。python对多线程支持还是欠缺， 当时也考虑用java的线程安全来做，最终因为一些原因没有research下去。

Selenium IDE: 是firefox的插件，可以录制/回放。

Selenium Grid :分布式测试，将selenium脚本分配到不同的测试机上同步运行。

1. 如何提高自动化脚本稳定性(这次可以通过，下次就没办法通过了)

1)网速原因，增加时间等待

2)设置重试机制（主要的原因：页面还没有加载完成，我们就开始进行元素定位。）

1. 提高selenium脚本的执行速度

减少不必要步骤

终止加载

用java配置testNG实现多线程，尽量设置多线程运行，提高执行速度。多线程的时候耦合性要松，尽量没有任何关系，因为多线程的执行顺序是不受控制的。

1. 基础

Driver = webdriver.Chrome(driver path)

driver.get(url)

Xpath //p[contains(@class, '\*\*\*')] 相对+标签+contains（@属性，value）

根据text()

driver.find\_elment\_by\_xpath('//p[contains(text(), '京公网')]')

多个属性组合定位

driver.find\_element\_by\_xpath("//input[@class='s\_ipt' and @name='wd']")

Appium

driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired\_caps)#连接appium 4723是appium

driver.is\_app\_installed("com.lovebizhi.wallpaper")

find\_element\_by\_id("com.lovebizhi.wallpaper:id/nav\_home\_ll").click()

driver.install\_app("D:\\bz.apk")

1. 父,子

Find\_element\_by\_id(‘parent’).find\_element\_by...

Find\_element\_by\_xpath(‘//input/div[@name=’n’]/span/input[1]’)

1. 二次面it定位 select

Element1 = driver.find\_element\_by\_id(‘ID1’)

Element2 = Element1.find\_element\_by\_id(‘ID2’)

1. isElementPresent() 也要用try

element.click()

element.send\_keys

ActionChains(driver)（鼠标事件

#双击ActionChains(driver).double\_click(element).perform()

#右击ActionChains(driver).context\_click(element).perform()

#拖动ActionChains(driver).drag\_and\_drop(element).perform()

#悬停ActionChains(driver).move\_to\_element(element).perform()

1. 怎样去选择一个下拉框中的value＝xx的option？

select\_by\_index(1)

select\_by\_value(“2”)

1. Driver.refresh() 刷新

Driver.back() 后退

Driver.forward() 前进

Driver.maximize\_window() 最大化

Driver.set\_window\_size(100,200) 设置窗口大小

Driver.switch\_to.window(searchwindow)

1. 显式等待：每隔一段时间进行检测，超出最大时间则抛出异常，针对单个控件

WebDriverWait(driver, 5).until(EC.presence\_of\_element\_located( (By.ID, ‘kw’) )

程序默认会每500ms调用一次来查看元素是否已经生成，如果本来元素就是存在的，那么会立即返回。

隐式等待：作用于全局，相当于设置全局在定位元素时的等待超时时间

driver.implicitly\_wait(20)

强制等待 Sleep(2)

1. page object设计模式

是将page对象封装成一个HTML页面，通过提供API来操作页面元素。减少了代码的重复，提高可读性和可维护性

把原来的使用 xpath方式，改为使用 id 查找。

selenium中hidden或者是display ＝ none的元素是否可以定位到？- 不能

selenium中如何保证操作元素的成功率？尽量使用相对地址定位.

如何去定位属性动态变化的元素？- xpath 通过同级、父级、子级进行定位

点击链接以后，selenium是否会自动等待该页面加载完毕？- 是的

何在定位元素后高亮元素（以调试为目的）？-重置元素属性，给定位的元素加背景、边框

webdriver可以用来做接口测试吗？-有难度，不推荐

1. lazyloding

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

页面的策略，是你不想看就不给你看，我也懒得加载出来，懒得去请求，帮助减少服务器负载。

这给测试确实带来很多麻烦，没有滚轮scroll的方法

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

如果是垂直的话，一般模拟多次按键盘下键可以解决。

如果水平的话就比较讨厌。

1.运气特别好，主页面url变化，url里面get字段里面肯定会有当前页面的index。

2.运气不太好，翻页是javascript动态加载的，那你就去找“下一页”。

3.如果连下一页都没有，你和开发关系特别好，让他帮你预埋一个测试响应，比如右键翻页。

【app 先适配手机 获取屏幕长宽，然后定点滑动swipe。如果不行让开发预埋下键，这些现在虽然不用 ，android框架里应该还有吗】

1. Mock(模拟反馈)

这里要实现一个Count计算类，add() 方法要实现两数相加。但，这个功能我还没有完成。这时就可以借助mock对其进行测试。

count = Count()

count.add = mock.Mock(return\_value=13)

result = count.add(8,5)

self.assertEqual(result,13)

1. 常见控件使用

1) link/button:

element.click()

2) Textbox:

element.send\_keys(‘test’)

3) Upload

Driver.find\_element\_by\_id(‘id’).send\_keys(‘D\test.txt’)

4) Mouse Event:

e1=driver.find\_element\_by\_id(‘kw’)

ActionChains(driver).double\_click()/.context\_click()/.drag\_and\_drop()/.move\_to\_eleme nt()/.perform() # 双击/右击/拖动/悬停/执行actionChains中的行为

5) Dropdown: 二次定位

·<Select>标签的下拉菜单

from selenium.webdriver.support.ui import Select

Select(driver.find\_element\_by\_id(‘gender’)).select\_by\_index(1)

Select(driver.find\_element\_by\_id(‘gender’)).select\_by\_value(“2”)

Select(driver.find\_element\_by\_id(‘gender’)).select\_by\_visible\_text(“Male”)

·非<Select>标签的下拉菜单

Dropdown1 = driver.find\_element\_by\_id(‘id’) #先定位到dropdown

Dropdown1.find\_element\_by\_id(“li2\_input\_2”) #再定位到dropdown中的值

6) Alert:

driver.switch\_to.alert.accept()//接受

driver.switch\_to.alert.dismiss() //取消

Message=driver.switch\_to.alert.text //获取弹窗的文本消息

7) Window

Driver.refresh() 刷新

Driver.back() 后退

Driver.forward() 前进

Driver.maximize\_window() 最大化

Driver.set\_window\_size(100,200) 设置窗口大小

Driver.switch\_to.window(searchwindow)

Driver.switch\_to.frame(ReferenceFrame)

Frame

Driver.switch\_to.parent\_frame() # frame需要一级一级切

Driver.switch\_to.default\_content()

截图

driver.get\_screenshot\_as\_file('E:\sc.jpg')

1. 模块化驱动测试

将重复的操作独立为公共模块，尽量消除重复，提高测试用例的可维护性。

数据驱动测试: 数据参数化

关键字驱动测试：脚本/数据/业务 分离

1. 读取txt文件

File= open(‘E:\wtest.txt’, ‘r’)

Lines = File.readlines()

For line in lines:

User = line.split(‘,’)[0]

Passw=line.split(‘,’)[1]

1. selenium中如何保证操作元素的成功率？也就是说如何保证我点击的元素一定是可以点击的？

首先通过封装find方法，实现wait\_for\_element\_ispresent，这样在对元素进行操作之前保证元素被找到，进而提高成功率。（WebDriverWait）

1. Selenium有几种定位方式？你最偏爱哪一种，为什么？

Selenium有八种定位方式：

与name有关的有三种：name、class\_name、tag\_name

与link相关的有两种：link\_text、partitial\_link\_text

与id有关：id

全能选手：xpath、css\_selector

如果存在id，我一定使用Id，因为简单方便，定位最快。

1. 断言和验证的区别？

1）断言（assert）：测试将会在检查失败时停止，并不运行后续的检查

优点：可以直截了当的看到检查是否通过

缺点：检查失败后，后续检查不会执行，无法收集那些检查结果状态

2）验证（vertify）：将不会终止测试

缺点：你必须做更多的工作来检查测试结果：查看日志——>耗时多，所以更偏向于断言