Jenkins+Python+Selenium+Unittest自动化测试框架v2.0

一、什么是web自动化测试?

自动化测试就是将手工测试的过程，转换成代码来执行。

Web是指的web网页，它是在浏览器当中呈现的页面。

功能测试，都是站在用户的角度来测试系统的功能。而用户接触到就是系统的页面，通过在页面上的各种操作来使用其功能。

比如说登陆页面，测试人员是在登陆页面中，第一步：找到用户名的输入框，输入用户名。第二步：找到密码输入框，输入密码。第三步，找到登陆按钮，并点击。不同的登陆数据，会带来不同的测试结果。

从这个过程可见，我们做页面的功能测试，特别依赖于页面的元素。因为所有操作都是对页面的某一个元素(比如输入框，比如按钮)。

二、为什么要做web自动化?

开发人员也不能保证，转给测试的软件版本，功能都是正常的（国内现状，基本上没有开发人员会做比较全面的自测工作）。尤其是历史的功能。假设这次上线时的版本，相对于上次上线时的版本，新增了10个功能。那么测试人员会在这期间，着重测这10个新功能。而除此之外的490个老功能，需要花大量的时间去回归测试。它并不能代替我们的手工测试，毕竟代码是我们人写的，按我们人的思维去做事情。而手工测试是我们大脑高度运转，想象无限。所以新开发的功能，一般还是手工测试为主。那回归测试的历史老功能，基本不会变。重复去做的回归测试工作，就让自动化测试代替。

三、什么样的项目适合做web自动化?

如果测试对象是B/S架构的软件，那么web自动化测试就是其中的一种应用。

web自动化本质上是站在用户的角度，和用户一样在页面上找到某个元素对元素进行各种操作，再在页面观察操作的结果是否正确。

所以在web自动化中，不涉及到数据库、接口这些底层相关的内容。直接从页面上看结果。因为就用户而言，在使用一个网站时只关注网站页面的状态。

所以web自动化，以页面为主，非常的依赖于页面的元素。

元素的变化会直接导致自动化用例执行失败。

因此如果是需求非常频繁变动的项目或功能，就完全不需要考虑web自动化了。你写的自动化脚本才写出来，页面就已经变了，得花时间重新更新脚本。

另外，从自动化背景中可知，web自动化主要应用是在项目长期迭代过程中的一种手段。所以如果项目周期短、功能少也可以不用考虑web自动化了。

因此，项目周期长，项目当中已基本稳定的功能模块可以考虑做web自动化。

但是，如果项目本身是以数据为主的，更多的应该考虑接口自动化更合适。比如说以报表为主的系统。

一、环境搭建：

python3及以上

https://www.python.org/downloads/

selenium pip install selenium

webdriver与Chrome对应版本如下：

https://npm.taobao.org/mirrors/chromedriver

Jenkins（Centos 7） 安装 略，可自行安装

二、框架组成：

Test_case:用于存放、管理用例

Test_data:存放参数化数据、上传的图片等

Test_report: 测试报告的存放

Common: 配置文件、邮件模块、封装函数

三、UI自动化涉及的应用：

1.Selenium 用于web自动化，可自行查阅。

2.Unittest 单元测试框架，提供用例组织执行、断言方法、丰富的日志

3.Jenkins 集成自动化框架、实现远程构建、定时无人构建、构建后触发邮件

4.~~HTMLTestRunner生成测试报告~~

5.用例运行加入失败重运行机制、跳过运行机制

四、部分代码

1.登录：

```python
#-*-coding:utf-8-*-
#@Time    :2020/11/11 15:04
#@Author  :JS_ErvinChiu
#@Email   :qiuxiongfei@jushiwangedu.com
#@File    :Test_Case.py
#@Software:PyCharm
#from selenium.webdriver.support.select import Select
#from selenium.webdriver.common.keys import Keys

import win32con
import win32gui
from selenium import webdriver
from unittestreport import rerun
from unittestreport import TestRunner
import unittest
import time
import HTML
import os
import HTMLTestRunnerNew

class public_def():

    #登录开始
    def login(self):
        driver = webdriver.Chrome()
        driver.get("https://betaweb.jushixl.net.cn/#/home")
        driver.implicitly_wait(10)
        driver.maximize_window()
        time.sleep(5)
        driver.find_elements_by_class_name("nav_item")[3].click()
        time.sleep(3)
        driver.find_element_by_xpath('//*[@id="app"]/div[2]/div/div/div/div[1]/div[1]/div').click()
        driver.find_element_by_xpath('//*[@id="app"]/div[2]/div/div/div/div[2]/div[2]/input').send_keys("18515817789")
        driver.find_element_by_xpath('//*[@id="app"]/div[2]/div/div/div/div[2]/div[4]/input').send_keys("a123456")
        time.sleep(3)
        driver.find_element_by_xpath('//*[@id="app"]/div[2]/div/div/div/div[2]/button').click()
```

2.上传本地图片：

```python
    #=========================================================上传图片=========================================================
    def upload_chrome(self, filepath):
        # 一级窗口
        dialog = win32gui.FindWindow("#32770", "打开")
        # 二级窗口
        ComboBoxEx32 = win32gui.FindWindowEx(dialog, 0, "ComboBoxEx32", None)
        # 三级窗口
        comboBox = win32gui.FindWindowEx(ComboBoxEx32, 0, "ComboBox", None)
        # 四级窗口--文件路径输入
        edit = win32gui.FindWindowEx(comboBox, 0, "Edit", None)
        # 二级窗口-打开按钮
        button = win32gui.FindWindowEx(dialog, 0, "Button", "打开(&O)")
        # 操作--添加发送文件路径
        win32gui.SendMessage(edit, win32con.WM_SETTEXT, None, filepath)
        # 点击打开按钮
        # time.sleep(3)
        win32gui.SendMessage(dialog, win32con.WM_COMMAND, 1, button)
        time.sleep(5)
```

3.观看回放课：

```
    @rerun(count=2, interval=5)
    #@unittest.skip("测试跳过用例")
    def test_watch_replay(self):

        self.Login()
        driver = self.driver
        time.sleep(3)
        driver.find_elements_by_class_name("nav_item")[2].click()  # 进入我的班级列表
        time.sleep(3)
        driver.find_elements_by_class_name("fast_button")[0].click()  # 进入班级
        time.sleep(5)
        # 选择全部课（默认定位全部课程可省略）
        driver.find_element_by_xpath(
            "/html/body/div[1]/div[2]/div[2]/div[2]/div[2]/div/div[1]/div[2]/div/div[1]/div[1]/div/i").click()
        windows_handle = driver.window_handles  # 获取当前所有handles
        driver.switch_to.window(windows_handle[-1])  # 切换到最新窗口
        time.sleep(5)
        driver.switch_to.frame([0][0])  # 切换到frame
        # 定位获取时间进度
        time_test = driver.find_elements_by_class_name("time")[0].text#获取初始时间
        print( "\n初始时间为%r:"%time_test)
        time.sleep(10)
        time_test01 = driver.find_elements_by_class_name("time")[0].text#获取进度时间
        print("进度时间为%r:"%time_test01)
        # self.assertEqual(time_test,init_time)
        try:
            assert (time_test == time_test01), '测试结果：Test Pass!!'
        except AssertionError as msg:
            print(msg)

        else:
            print("测试结果:Test Fail!!!")
```

4.订单测试&运行生成测试报告：

```
    def test_Order(self):

        self.Login()
        driver = self.driver
        time.sleep(3)
        driver.find_elements_by_class_name("nav_item")[1].click()  # 进入班级课程
        time.sleep(3)
        driver.find_elements_by_class_name("pic")[0].click()
        time.sleep(5)
        # driver.find_elements_by_class_name("el-button sku_buy_btn el-button--primary")[0].click()
        driver.find_element_by_xpath('//*[@id="app"]/div[2]/div[2]/div/button').click()
        time.sleep(3)
        driver.find_element_by_xpath(
            '//*[@id="app"]/div[2]/div/div[3]/div[5]/div[4]/label/span[1]/span').click()  # checkbox
        time.sleep(3)
        #driver.find_element_by_xpath('//*[@id="app"]/div[2]/div/div[3]/div[6]/div').click()#提交订单
        driver.find_element_by_xpath('//*[@id="app"]/div[2]/div/div[3]/div[7]/div').click()
        time.sleep(3)
        driver.find_element_by_xpath('//*[@id="app"]/div[2]/div[2]/div[4]/div/div[3]').click()
        #driver.find_element_by_xpath('//*[@id="app"]/div[2]/div[2]/div[4]/div/div[1]/div[6]/span').click()  # 选择线下转账（取消功能）
        time.sleep(5)
        ordertext = driver.find_element_by_xpath('//*[@id="app"]/div[2]/div[2]/div[1]/span[1]').text
        ordertext2 = '订单提交成功，请尽快付款！'

        try:
            assert (ordertext == ordertext2), '测试结果：Test Fail'

        except AssertionError as msg:
            print(msg)
        else:
            print("\n%r\n测试结果:Test Pass!!"%ordertext)
```

```
#添加测试用例类
suite = unittest.defaultTestLoader.loadTestsFromTestCase(Test_JS_Cases)
#生成测试报告
runner = TestRunner(suite=suite)
runner.run()
```

测试报告及邮件预览：

聚师网UI自动化测试报告

2551451515 <2551451515@qq.com> 15:26

发送给邱雄飞

共1个附件（ Js_AutoUI_Report.html） 查看附件 详情

Hi，你好！此邮件为自动发送，不必回复~详情请查看附件，如有疑问可联系测试组~

附件1个（16KB） 附件中未发现病毒 举报附件

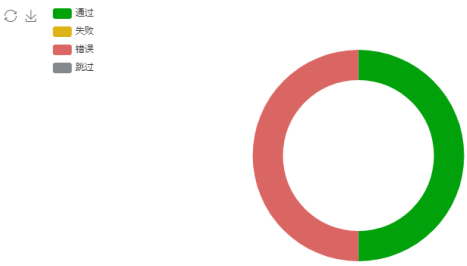
Js_AutoUI...port.html
16KB

UI自动化测试报告

测试结果汇总

测试人员	邱雄飞	成功用例	2
开始时间	2020-11-19 15:07:52	失败用例	0
执行时间	250.92 S	错误用例	2
用例总数	4	跳过用例	0
描述信息	聚师网UI自动化测试报告		

图表展示



详细信息

编号	测试类 所有	测试方法	用例描述	执行时间	执行结果 所有	详细信息
1	Test_JS_Cases	test_Buy_classes_one	None	40.6s	成功	查看详情
2	Test_JS_Cases	test_Buy_classes_two	None	55.5s	成功	查看详情

```
'订单提交成功，请尽快付款！'

测试结果:Test Pass!!!

test_Buy_classes_two (Test_case.Test_Case.Test_JS_Cases)执行---->【通过】
```

3	Test_JS_Cases	test_Order	None	80.7s	错误	查看详情

```
Traceback (most recent call last):
  File "D:\Python38\lib\unittest\case.py", line 60, in testPartExecutor
    yield
  File "D:\Python38\lib\unittest\case.py", line 676, in run
    self._callTestMethod(testMethod)
  File "D:\Python38\lib\unittest\case.py", line 633, in _callTestMethod
    method()
  File "D:\Python38\lib\site-packages\unittestreport\core\reRun.py", line 41, in decorator
    run_count(count, interval, func, *args, **kwargs)
  File "D:\Python38\lib\site-packages\unittestreport\core\reRun.py", line 23, in run_count
    raise e
  File "D:\Python38\lib\site-packages\unittestreport\core\reRun.py", line 18, in run_count
    func(*args, **kwargs)
  File "D:\PycharmProjects\JS_UIAuto_Test\Test_case\Test_Case.py", line 125, in test_Order
```

```
    self.Login()
  File "D:\PycharmProjects\JS_UIAuto_Test\Test_case\Test_Case.py", line 58, in Login
    driver.find_elements_by_class_name("nav_item")[3].click()
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webelement.py", line 80, in click
    self._execute(Command.CLICK_ELEMENT)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webelement.py", line 633, in _execute
    return self._parent.execute(command, params)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 321, in execute
    self.error_handler.check_response(response)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 242, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.ElementNotInteractableException: Message: element not interactable
  (Session info: chrome=86.0.4240.198)


====用例执行失败===
Traceback (most recent call last):
  File "D:\Python38\lib\site-packages\unittestreport\core\reRun.py", line 18, in run_count
    func(*args, **kwargs)
  File "D:\PycharmProjects\JS_UIAuto_Test\Test_case\Test_Case.py", line 135, in test_Order
    driver.find_element_by_xpath(
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 394, in find_element_by_xpath
    return self.find_element(by=By.XPATH, value=xpath)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 976, in find_element
    return self.execute(Command.FIND_ELEMENT, {
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 321, in execute
    self.error_handler.check_response(response)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 242, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.NoSuchElementException: Message: no such element: Unable to locate element: {"method":"xpath","selector":"//*[@id="app"]/div[2]/div/div[3]/div[5]/div[4]/label/span[1]/span"}
  (Session info: chrome=86.0.4240.198)


==========开始第0次重运行==========
====用例执行失败===
Traceback (most recent call last):
  File "D:\Python38\lib\site-packages\unittestreport\core\reRun.py", line 18, in run_count
    func(*args, **kwargs)
  File "D:\PycharmProjects\JS_UIAuto_Test\Test_case\Test_Case.py", line 125, in test_Order
    self.Login()
  File "D:\PycharmProjects\JS_UIAuto_Test\Test_case\Test_Case.py", line 58, in Login
    driver.find_elements_by_class_name("nav_item")[3].click()
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webelement.py", line 80, in click
    self._execute(Command.CLICK_ELEMENT)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webelement.py", line 633, in _execute
    return self._parent.execute(command, params)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 321, in execute
    self.error_handler.check_response(response)
  File "D:\Python38\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 242, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.ElementNotInteractableException: Message: element not interactable
  (Session info: chrome=86.0.4240.198)


test_Order (Test_case.Test_Case.Test_JS_Cases)执行—>【错误Error】
```

4	Test_JS_Cases	test_watch_replay	None	74.2s	错误	查看详情

PS：邮件可自定义配置接收邮箱，密码为邮箱授权码，并非邮箱密码

自动化测试脚本运行在本地环境，Jenkins运行脚本之前，需建立与本地连接，通过节点配置

自动化脚本目前分为两部分：邮件模块（独立）、用例模块（用例、报告），通过Jenkins批处理命令，构建时分别运行两个脚本：1用例脚本2.邮件脚本（邮件通过sort() 方法取到最新报告，并上传发送邮件,本次测试报告不以时间戳区分，新报告覆盖）

六、Jenkins集成自动化：

自动化演示.mp4
12.8 MB