# 自动化测试脚本

1. 什么是自动化？？怎么定义

简单来说：自动化测试是一项让计算机程序取代人进行软件测试的技术；

广义定义：所有需要借助测试工具辅助进行软件测试的方法都可以称自动化测试，只是区别程度不一样

狭义：利用工具自动识别软件UI元素，并操作元素对象完成测试动作的偏功能性的测试过程。

总体来说呢？？：自动化测试是一项借助测试工具，来识别软件UI元素，并让程序执行测试动作的软件测试技术。

2：为什么要做自动化测试？？

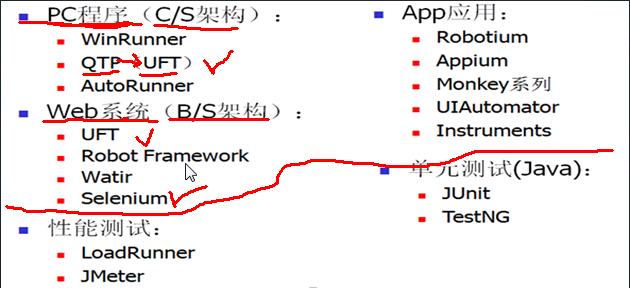
1. 领导要求呗；
2. 适应现在软件行业的趋势，越来越趋势自动化测试方向
3. 简化人工的重复操作
4. 一些人工无法完成的测试（那些是人工无法完成的测试呢？）

比如：性能测试吧？？需要一些大量数据的模拟测试？？这些是人工无法独立完成的吧.......

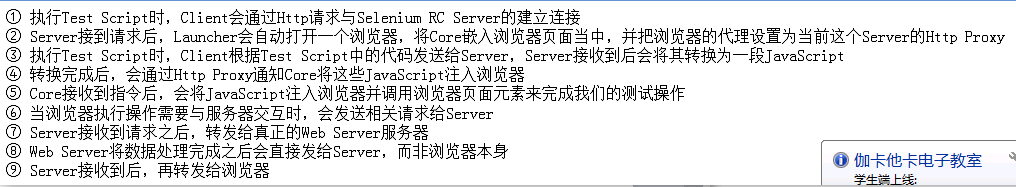
手工测试和自动化测试的比较？？

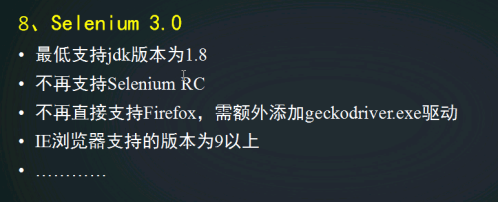


常见的一些自动化测试工具？？



Selenium RC的工作原理？？





Unittest框架



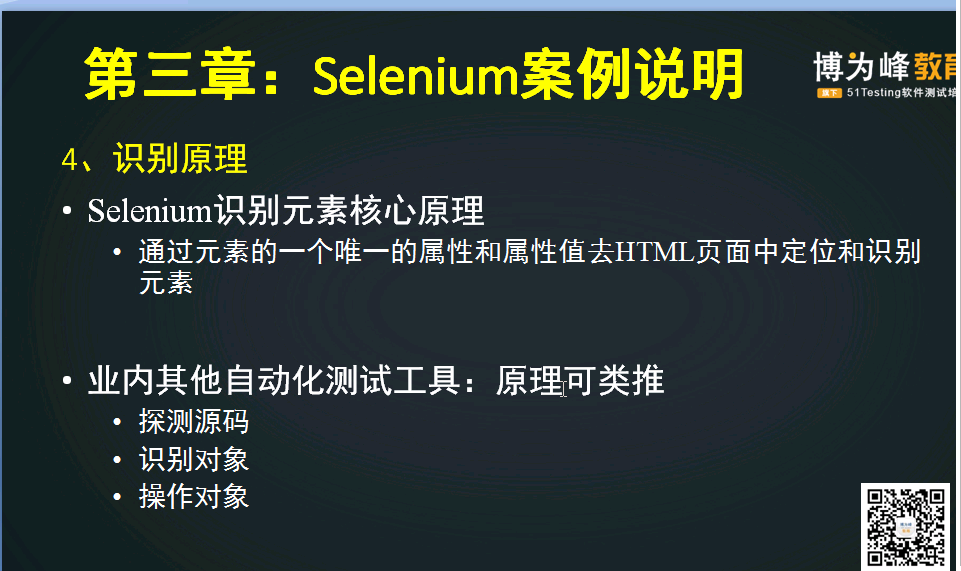
Test fixture测试固件（定以了一些有固定用法的方法）

Test case（测试用例）必须包含有test开头的方法test\_

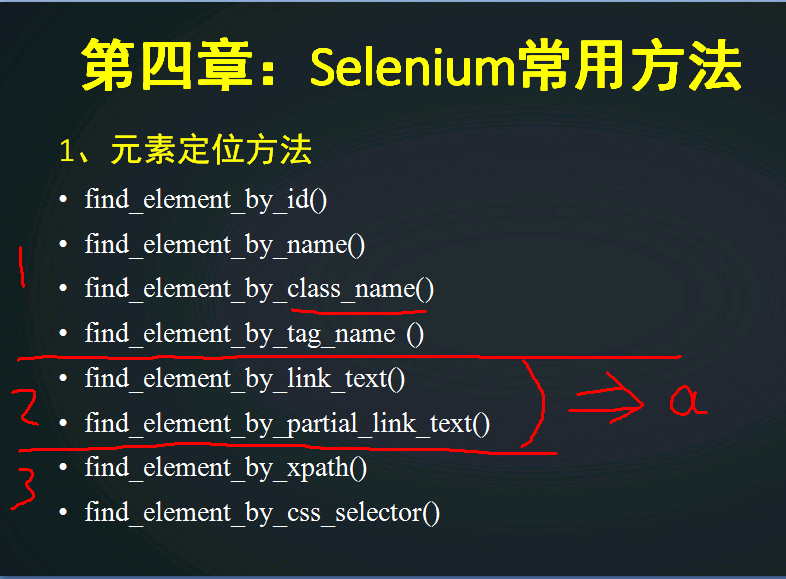
Test suite（测试集合，测试套件）提供了一些方法收集脚本

Test runner（测试运行器）执行测试套件，执行收集起来的脚本。

Selenium识别原理？？

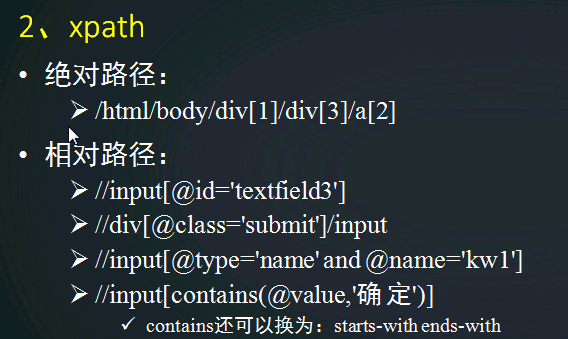


定位元素的常用方法？？？



1. 是常用的
2. A是针对超链接
3. 很牛皮啦，通用的啦

Selenium的路径方法？？？



Selenium元素对象：webelement????



有哪些重点需要了解呢？？

1. 熟悉selenium脚本编写过程
2. 熟练掌握元素定位方法
3. 掌握时间戳带的使用
4. 关于字符串的截取

# 八大元素定位

HTML标签基础：

<input id="kw" name="wd" class="s\_ipt" value="" maxlength="255" autocomplete="off">（举例）

专用于前段开发使用，所谓前段是可以用户肉眼直观所见的，

所有数据交互，逻辑处理安全机制等等（大部分）为后端

前段：HTML JAC

自动化过程中，需要准确的找到元素的准确位置，以便进行直接的自动化操作

身份证： 姓名，性别，住址，照片，身份证号码

1：id元素定位（id "kw"（身份证号码，正常情况下是不可能重复的（不排除开发装怪要弄个重复的）））

2：name元素定位（name "wd"(）

(id、name不一定都是要有的，如果有多个重名，会返回多个元素-->就会报错）

python和java都可以实现自动化，看个人喜好喜欢python或者java实际上两者没有谁好谁不好的说法

3：classname（"s\_ipt"，很少用，容易重名且内容过多，基于元素的class属性进行定位）

4：xpath //\*[@id="kw"]（主流的元素定位方式之一，最为稳当的一种（万金油））

辅助粘贴

手写：//input[@id="kw"]、//\*[@name="wd"]

相对路径和绝对路径：

相对路径：//input[@id="kw"]、//\*[@name="wd"] 表示id="kw"的路径，name="wd"的路径

绝对路径：类似文件路径（类似找文件夹的路径差不多）

/btml/body/div[2]/div[2]/div[5]/div/form/spath/input/

5：tag定位 （tag=input ，这个基本上不用，因为一个标签可能有很多input botton等等）

6：link\_text（）定位页面中的超文本链接，实现元素定位

7：partial\_link\_text() 类似于sql语音中的like %value%模糊查询

是对link定位的一种补充，当链接上的文本内容比较长的时候，可以取文本的一部分进行定位，当然这部分可以唯一地标识这个链接

8: css\_selector 除非是用IE浏览器做web测试，否则都可以使用xpath来实现定位

tag定位：页面html文档下的各种标签，find\_element\_by\_tag\_name("input")

tag往往用来定义一类功能，所以通过tag识别某个元素的概率很低。任意打开一个页面，都会发现大量的<div>、<input>、<a>等tag，所以tag name定位很少用

以上的方式稍有局限，且经常页面没有id，name这些属性值，class name重复性较高，link定位有针对性，所以Xpath与Css定位更灵活些。

对于其他的定位元素的方法这里就不多说了，主要来说说这个万能的定位元素的方法 xpath

相对路径：

通过元素的id,name,class这些属性来定位（函数接收的是字符串所以括号里面的一定要加引号）

driver.find\_element\_by\_xpath("//\*[@id='kw']").send\_keys("by\_xpath") driver.find\_element\_by\_xpath("//\*[@name='wd']").send\_keys("by\_xpath") driver.find\_element\_by\_xpath("//\*[@class='s\_ipt']").send\_keys("by\_xpath")

如果同一个属性同名较多的时候，可以指定标签定位更准确，如下：

定义input标签下的id属性，还是拿百度做例子吧：

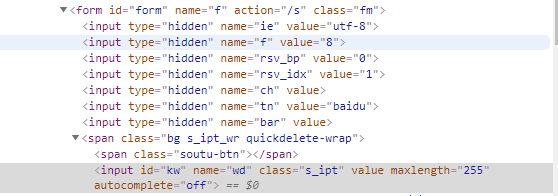


driver.find\_element\_by\_xpath("//input[@id='kw']").send\_keys("by\_xpath")

Xpath层级

如果一个元素的属性不是很明显，可以通过父元素来找它，如果父元素的属性也不是很明显，就通过父元素的父元素，即爷爷元素来找，如下：

同样的还是拿百度来做例子：



看上面的图，假如input标签和span标签下没有我们可以定位的元素，当然这里有，我说的是假如没有，那么我们可以定位form的标签的元素来实现

in\_put=driver.find\_element\_by\_xpath(*"//form[@id='form']/span/input"*).send\_keys(*'51testing'*)

注意：id是form标签的id，然后往下，一层一层的，还有要注意的就是span一定是在form标签里面，input也一定是在span标签里面哦！！

Xpath的模糊匹配：

#匹配文字：

driver.find\_element\_by\_xpath("//\*[contains(text(), 'hao123')]").click()

self.driver.find\_element\_by\_xpath('//\*[text()="退出"]').click() #没有指定标签，查找整个页面代码中的‘退出’两个字

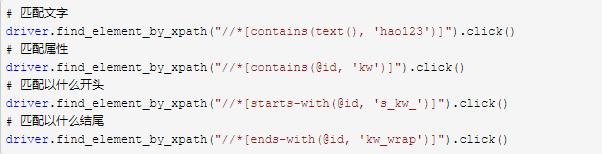


用contains，寻找页面中style属性值包含有sp.gif这个关键字的所有div元素,其中@后面可以跟该元素任意的属性名。

self.driver.find\_element\_by\_xpath('//div[contains(@style,"sp.gif")]').click()

用start-with，寻找style属性以position开头的div元素,其中@后面可以跟该元素任意的属性名

self.driver.find\_element\_by\_xpath('//div[start-with(@style,"position")]').click()



注意：xpath(*"//form[starts-with(@id,'form')]/span/input"*)

@id, ‘form’ 中间有个空格，否则就会报错哦，不要问我为什么，语法就是这样的

Xpath索引

如果一个元素它的兄弟元素跟它标签一样，这时候无法通过层级定位，可以通过索引定位，如下：

driver.find\_element\_by\_xpath("//select[@id='nr']/option[1]").click() driver.find\_element\_by\_xpath("//select[@id='nr']/option[2]").click() driver.find\_element\_by\_xpath("//select[@id='nr']/option[3]").click()

这个没有深入的研究过（遇到了再说吧）

#### Css

常见符号：

#表示id选择器

.表示class选择器

>表示子元素，层级

一个空格也表示子元素，但是是所有的后代子元素，相当于xpath中的相对路径

Css:属性定位

1）css可以通过元素的id、class、标签这三个常规属性直接定位到

如下是百度输入框的的html代码：

<input id="kw" class="s\_ipt" type="text" autocomplete="off" maxlength="100" name="wd"/>

css用#号表示id属性,如：#kw

css用.表示class属性，如：.s\_ipt

css直接用标签名称，无任何标示符，如：input

Css其他属性

[name=wd] [autocomplete='off'][maxlength='255']

driver.find\_element\_by\_css\_selector(*"input[autocomplete='off']"*)

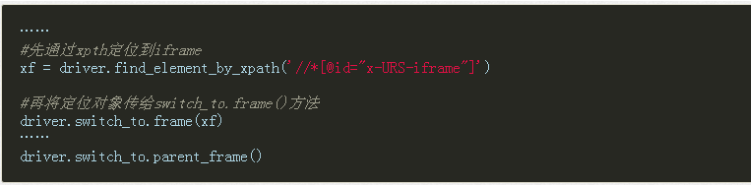
Selenium 多表单切换处理（iframe/frame）

在web应用中，前台网页的设计一般会用到iframe/frame表单嵌套页面的应用。简单的就是一个页面签嵌套多个HEML/JSP文件。selenium webdriver  只能在同一页面识别定位元素，可以狭隘的理解成只能识别当前所在位置的页面上的元素。对于不同的iframe/frame表单中的元素是无法直接定位的。需要结合switchTo().frame()方法切换到指定的frame/iframe中。switchTo().frame()默认的是取表单的ID和name属性。如果没有id和name ,可通过Xpath路径定位。



126邮箱登录框架大概就是这个样子

switch\_to.frame() 默认可以直接取表单的id 或name属性。如果iframe没有可用的id和name属性，则可以通过下面的方式进行定位。



driver.get(*"http://mail.163.com/"*)

form\_id = driver.find\_element\_by\_name(*""*)

driver.switch\_to.frame(form\_id)

通过name定位

frame是整个页面的框架，iframe是内嵌的网页元素，也可以说是内嵌的框架，可以用它将一个HTML文档嵌入在另一个HTML中显示

如果完成了在当前表单上的操作可以通过 switch\_to\_default\_content()方法返回到上一层表单。该方法不用指定某个表单的返回，默认对应与它最近的 switch\_to\_frame()方法。

**多窗口切换**

在页面操作过程中有时候点击某个链接会弹出新的窗口，这时就需要主机切换到新打开的窗口上进行操作。WebDriver提供了switch\_to.window()方法，可以实现不同的窗口之间的切换。

几种常见的方法？

获得所有窗口句柄 .windows\_handles---> handles为一个数组：handles=[窗口1，窗口2,....]

#获得当前窗口句柄.current\_window\_handl

#switch\_to\_window().切换窗口函数

第一种方法？？

# 获取打开的多个窗口句柄

windows = driver.window\_handles

# 切换到当前最新打开的窗口driver.switch\_to.window(windows[-1])

实际举例说明：

#获取当前浏览器所有的窗口

wins = driver.window\_handles

#切换窗口，切换为最新打开的窗口

driver.switch\_to\_window(wins[-1])

#切换为最初打开的窗口

driver.switch\_to\_window(wins[0])

第二种方法？？

# 获得打开的第一个窗口句柄（当前窗口）

window\_1 = driver.current\_window\_handle

# 获得打开的所有的窗口句柄

windows = driver.window\_handles

# 切换到最新的窗口

for current\_window in windows:

if current\_window != window\_1:

driver.switch\_to.window(current\_window)

举例说明？？

window\_1 = driver.current\_window\_handle

windows = driver.window\_handles

for current\_window in windows:

if current\_window != window\_1:

driver.switch\_to\_window(current\_window)

对于两个以上的窗口切换呢？？？？

？？？？？我已经会了

说说焦点切换：

Switch\_to

selenium做自动化的过程中，经常会遇到alert、frame和新的window，这是经常是switch\_to家族大展拳脚的时候，先看看switch\_to家族的成员：

**alert** ——返回浏览器的Alert对象，可对浏览器alert、confirm、prompt框操作

**default\_content**() ——切到主文档

**frame**(*frame\_reference*) ——切到某个frame #上面已经说了

**parent\_frame**() ——切到父frame，这个方法也不常被人所知，但有多层frame的时候很有用，不过这里要提一句，一般这种嵌套多层的      frame都是有问题的，会影响到性能，可以提给开发，让其改进

**window**(*window\_name*) ——切到某个浏览器窗口  #上面已经说了

### switch\_to.active\_element等同于    driver.switch\_to\_active\_element()

## ****嵌套frame的操作(switch\_to.parent\_frame())****

**弹窗问题？？？**

**Selenium 字符串切割**

在Python中自带的一个切割方法split(),这个方法不带参数，就默认按照空格去切割字段，如果带参数，就按照参数去切割

以百度为例代码如下：

from selenium import webdriver

import time

class**getsubstring**(object):

def**get\_search\_result**(*self*):

driver = webdriver.Chrome()

driver.maximize\_window()

driver.implicitly\_wait(8)

driver.get(*"http://www.baidu.com"*)

driver.find\_element\_by\_id(*'kw'*).send\_keys(*'selenium'*)

driver.find\_element\_by\_id(*'su'*).click()

time.sleep(3)

search\_result\_string = driver.find\_element\_by\_xpath(*"//\*[@id='2']/h3/a"*).text

print(search\_result\_string)

new\_string = search\_result\_string.split(*u'('*)[1] #*[1]代表切割右边部分*

print (new\_string)

last\_result = new\_string.split(*u'科'*)[0] #*[0]代表切割参照参数的左边部分*

print (last\_result)

getstring = getsubstring()

getstring.get\_search\_result()

Python自动化函数封装

单元测试框架

**TestCase（测试用例）:**

**TestSuite（测试套件）:**多个测试用例test case集合就是TestSuite，TestSuite可以嵌套TestSuite

自动化自动报告生成

--stream:测试报告写入文件的存储区域

--title:测试报告的主题

--description：测试报告的描述

report\_path是存放测试报告的地址

unittest.main()方法会搜索该模块下所有以test开头的测试用例方法，并自动执行它们，**unittest.main()**:使用它可以将一个单元测试模块变为可直接运行的测试脚本，main()方法使用TestLoader类来搜索所有包含在该模块中以“test”命名开头的测试方法，并自动执行。

自动化报告生成：

将HTMLTestrunner.py放在python安装目录的Lib路劲下面。

自动化测试生成单元测试报告的方法：

1. 写完单元测试用例：

Class BaiduTest(unittest.TestCase):

def setUp(self):

pass

def test\_baidu(self):

pass

def tearDown(self):

pass

if \_\_name\_\_ == “\_\_main\_\_”:

#定义测试套件（容器）

test=unittest.TestSuit()

#添加测试用例到测试套件中（容器）

test.addTest(BaiduTest(“test\_baidu”)

#定义报告的路径

File\_path=r“E:\bbs\result.html” # E:\bbs\为E盘下的bbs文件夹中

File\_result=open(file\_path,’wb’)

#定义测试报告

runner=HTMLTestRunner.HTMLTestRunner(stream=file\_result,title=’百度测试报告’,description=‘用例执行情况’)

#运行测试用例

Runner.run(test)

#关闭报告文件

file\_result.close()

以上就是自动化单元测试报告生成的方式

关于自动化selenium中def \_init\_（self）的用法：

在调用class类时，如果在def内部去调用，则使用self方法，如果在def外部调用，则括号内需要传入其他的值？

Selenium中获取文本值：

可以通过定位到具体的元素后通过text方法获取文本信息，例如:

driver.find\_element\_by\_xpath(“//div[‘ip’]”).text

直接通过定位到的元素的属性直接获取属性名称，如输入框提示信息等：

driver.find\_element\_by\_id(“xxx”).get\_attribute(yyy)



在不使用UNITTEST单元测试框架的情况下、如何进行断言

From unittest import TeseCase

………..

TestCase.assertEqual()