**Python实现Selenium中元素和浏览器操作方法的二次封装**

1. Python中元素定位的尴尬

如果你之前做过java的selenium的自动化测试，你会发现在python上面做元素定位的时候，遇到一个尴尬的问题：

Java上是这么定义元素和元素操作，

WebElement element

Element.click()

Python是这样做：

Input\_box = (By.ID, ‘su’)

Element = self.driver.find\_element(self.Input\_box)

Element.send\_keys(“selenium\_test”)

所以问题就来了，元素操作太麻烦，在考虑封装元素操作方法的时候，还需要先封装如何识别和定位一个元素。

1. Python实现Selenium一些方法的二次封装

为什么要封装?

封装是为了代码更加安全，复用和健壮角度考虑，方法页面对象和脚本调用方法。其实，在第二篇文章，我们就实现了一个封装，封装一个支持不同浏览器类型的类。这里我们来讲，如果封装一些selenium基本常用的方法到BasePage.py 模块中去。

直接看代码：

BasePage.py 代码（封装一些方法）

*# coding=utf-8***from** selenium.webdriver.support **import** expected\_conditions **as** EC  
**import** os.path  
**from** selenium **import** webdriver  
**from** selenium.webdriver.common.by **import** By  
**from** selenium.webdriver.support.wait **import** WebDriverWait  
  
  
**class** BasePage(object):  
 *"""  
 定义一个页面基类，让所有页面都继承这个类，封装一些常用的页面操作方法到这个类  
 """* **def** \_\_init\_\_(self, driver):  
 self.driver = driver  
  
 *# quit browser and end testing* **def** quit\_browser(self):  
 self.driver.quit()  
  
 *# 浏览器前进操作* **def** forward(self):  
 self.driver.forward()  
  
 *# 浏览器后退操作* **def** back(self):  
 self.driver.back()  
  
 *# 隐式等待* **def** wait(self, seconds):  
 self.driver.implicitly\_wait(seconds)  
  
 *# 打开一个页面* **def** open\_page(self, url):  
 self.driver.get(url)  
  
 *# 最大化浏览器* **def** max\_windows(self):  
 self.driver.maximize\_window()  
  
 *# 设置浏览器大小* **def** set\_windows\_size(self, wide, high):  
 self.driver.set\_window\_size(wide, high)  
  
 *# 点击关闭当前窗口* **def** close(self):  
 self.driver.close()  
  
 *# 刷新* **def** refresh(self):  
 self.driver.refresh()  
  
 *# 执行js脚本* **def** js(self, script):  
 self.driver.execute\_script(script)  
  
 *# 保存图片* **def** get\_windows\_img(self, file\_path):  
 *'''  
 在这里我们把file\_path这个参数写死，直接保存到我们项目根目录的一个文件夹.\Screenshots下  
 '''* file\_path = os.path.dirname(os.getcwd()) + **'\Screenshots'** self.driver.get\_screenshot\_as\_file(file\_path)  
  
 *# 点击接受alert弹出框* **def** accept\_alert(self):  
 self.driver.switch\_to.alert.accept()  
  
 *# 点击取消按钮在alset弹出框* **def** dismiss\_alert(self):  
 self.driver.switch\_to.alert.dismiss()  
  
 *# 跳进 frame* **def** switch\_to\_frame(self, iframe\_name):  
 self.driver.switch\_to.frame(iframe\_name)  
  
 *# 从frame中跳出来* **def** switch\_to\_frame\_out(self):  
 self.driver.switch\_to.default\_content()  
  
 *# 等待元素显示* **def** element\_wait(self, selector, secs=5):  
  
 **if "," not in** selector:  
 **raise** NameError(**"Positioning syntax errors, lack of ','."**)  
  
 by = selector.split(**","**)[0]  
 value = selector.split(**","**)[1]  
  
 **if** by == **"id"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.ID, value)))  
 **elif** by == **"name"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.NAME, value)))  
 **elif** by == **"class"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, value)))  
 **elif** by == **"link\_text"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.LINK\_TEXT, value)))  
 **elif** by == **"xpath"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.XPATH, value)))  
 **elif** by == **"css"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, value)))  
 **else**:  
 **raise** NameError(  
 **"Please enter the correct targeting elements,'id','name','"  
 "class','link\_text','xpath','css'."**)  
  
  
 *# 定位元素方法* **def** get\_element(self, selector):  
  
 **if ',' not in** selector:  
 **return** self.driver.find\_element\_by\_id(selector)  
 selector\_by = selector.split(**','**)[0]  
 selector\_value = selector.split(**','**)[1]  
  
 **if** selector\_by == **"i" or** selector\_by == **'id'**:  
 element = self.driver.find\_element\_by\_id(selector\_value)  
 **elif** selector\_by == **"n" or** selector\_by == **'name'**:  
 element = self.driver.find\_element\_by\_name(selector\_value)  
 **elif** selector\_by == **"c" or** selector\_by == **'class\_name'**:  
 element = self.driver.find\_element\_by\_class\_name(selector\_value)  
 **elif** selector\_by == **"l" or** selector\_by == **'link\_text'**:  
 element = self.driver.find\_element\_by\_link\_text(selector\_value)  
 **elif** selector\_by == **"p" or** selector\_by == **'partial\_link\_text'**:  
 element = self.driver.find\_element\_by\_partial\_link\_text(selector\_value)  
 **elif** selector\_by == **"t" or** selector\_by == **'tag\_name'**:  
 element = self.driver.find\_element\_by\_tag\_name(selector\_value)  
 **elif** selector\_by == **"x" or** selector\_by == **'xpath'**:  
 element = self.driver.find\_element\_by\_xpath(selector\_value)  
 **elif** selector\_by == **"s" or** selector\_by == **'selector\_selector'**:  
 element = self.driver.find\_element\_by\_css\_selector(selector\_value)  
 **else**:  
 **raise** NameError(**"Please enter a valid type of targeting elements."**)  
  
 **return** element  
  
 **def** type(self, selector, text):  
 el = self.get\_element(selector)  
 el.clear()  
 el.send\_keys(text)  
  
 **def** clear(self, selector):  
 self.element\_wait(selector)  
 el = self.get\_element(selector)  
 el.clear()  
  
 **def** click(self, selector):  
 self.element\_wait(selector)  
 el = self.get\_element(selector)  
 el.click()

解释：

主要有三部分

1. 元素定位封装
2. Driver相关方法，例如 self.driver.quit()
3. 元素相关方法，点击，输入

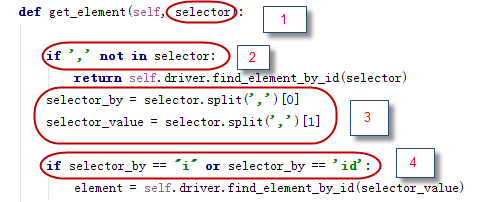
主要来介绍两个方法

1. def get\_element(self)

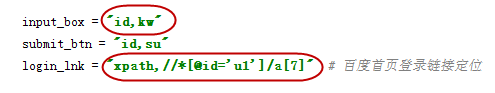
**def** get\_element(self, selector):  
  
 **if ',' not in** selector:  
 **return** self.driver.find\_element\_by\_id(selector)  
 selector\_by = selector.split(**','**)[0]  
 selector\_value = selector.split(**','**)[1]  
  
 **if** selector\_by == **"i" or** selector\_by == **'id'**:  
 element = self.driver.find\_element\_by\_id(selector\_value)  
 **elif** selector\_by == **"n" or** selector\_by == **'name'**:  
 element = self.driver.find\_element\_by\_name(selector\_value)  
 **elif** selector\_by == **"c" or** selector\_by == **'class\_name'**:  
 element = self.driver.find\_element\_by\_class\_name(selector\_value)  
 **elif** selector\_by == **"l" or** selector\_by == **'link\_text'**:  
 element = self.driver.find\_element\_by\_link\_text(selector\_value)  
 **elif** selector\_by == **"p" or** selector\_by == **'partial\_link\_text'**:  
 element = self.driver.find\_element\_by\_partial\_link\_text(selector\_value)  
 **elif** selector\_by == **"t" or** selector\_by == **'tag\_name'**:  
 element = self.driver.find\_element\_by\_tag\_name(selector\_value)  
 **elif** selector\_by == **"x" or** selector\_by == **'xpath'**:  
 element = self.driver.find\_element\_by\_xpath(selector\_value)  
 **elif** selector\_by == **"s" or** selector\_by == **'selector\_selector'**:  
 element = self.driver.find\_element\_by\_css\_selector(selector\_value)  
 **else**:  
 **raise** NameError(**"Please enter a valid type of targeting elements."**)  
  
 **return** element

解释：

支持通过八种方式的元素定位方法，找到这个元素后，返回这个元素对象。这里需要解释下前面那段字符串分割的代码的意思。



在看看页面类元素定位截图



原理：

图1中的1部分selector 指的就是图2中的input\_box

图1中的 2部分“，”这个逗号就是图二中第一个红圈里的逗号，这个是一个写法，如果你用 “=” 不太好看，在同一行有两个等号的表达式很别扭，虽然第二个“=”是放在双引号里。

图1中的3 部分，是一个字符串分割方法，通过逗号分开，[0]代表获取逗号前面的片段，[1]代表获取逗号后的片段，例如input\_box= “id,kw”

通过切割后，此时

selector\_by = id selector\_value = kw

图1中4部分，调用selenium find by方法去找元素。

1. def element\_wait(self)

Selenium中有关于wait有两个方法

隐式等待 driver.implicitly\_wait(10)

这里我们就要讨论的是显示等待： WebDriverWait

WebDriverWait(self.driver, secs,1).until(EC.presence\_of\_element\_located((By.ID, value)))

意思大概是，最多等待 secs秒时间，判断这个元素是否显示（presence）

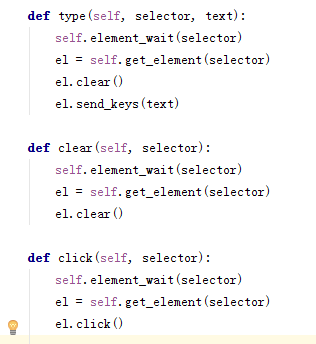
特别注意需要导入这个包

**from** selenium.webdriver.support **import** expected\_conditions **as** EC

直接上代码：

**def** element\_wait(self, selector, secs=5):  
  
 **if "," not in** selector:  
 **raise** NameError(**"Positioning syntax errors, lack of ','."**)  
  
 by = selector.split(**","**)[0]  
 value = selector.split(**","**)[1]  
  
 **if** by == **"id"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.ID, value)))  
 **elif** by == **"name"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.NAME, value)))  
 **elif** by == **"class"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, value)))  
 **elif** by == **"link\_text"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.LINK\_TEXT, value)))  
 **elif** by == **"xpath"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.XPATH, value)))  
 **elif** by == **"css"**:  
 WebDriverWait(self.driver, secs, 1).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, value)))  
 **else**:  
 **raise** NameError(  
 **"Please enter the correct targeting elements,'id','name','"  
 "class','link\_text','xpath','css'."**)

写这个方法的好处是，当我们需要操作元素，先执行这个方法，判断这个元素在不在页面显示，如果显示再执行其他方法。



上图中添加element\_wait（）就是这个目的

**看看页面类如何去调用这些封装的方法**



现在明白了第四篇文章中关于这个元素定位表达式的新写法的问题吧