Selenium 常见元素定位方法和操作的学习介绍

这篇文章主要 Selenium+Python 自动测试或爬虫中的常见定位方法、鼠标操作、键盘操作介绍，希望该篇基础性文章对你有所帮助，如果有错误或不足之处，请海涵~

一. 定位元素方法

这里有各种策略用于定位网页中的元素(locate elements)，你可以选择最适合的方案，Selenium 提供了一下方法来定义一个页面中的元素：

- find_element_by_id
- find_element_by_name
- find_element_by_xpath
- find_element_by_link_text
- find_element_by_partial_link_text
- find_element_by_tag_name
- find_element_by_class_name
- find_element_by_css_selector

下面是查找多个元素（这些方法将返回一个列表）:

- find_elements_by_name
- find_elements_by_xpath
- find_elements_by_link_text
- find_elements_by_partial_link_text
- find_elements_by_tag_name
- find_elements_by_class_name
- find_elements_by_css_selector

除了上面给出的公共方法，这里也有两个在页面对象定位器有用的私有方法。这两个私有方法是 find_element 和 find_elements。

常用方法是通过 xpath 相对路径进行定位，同时 CSS 也是比较好的方法。举例：

[html] view plain copy

```
1.  <html>
2.   <body>
3.    <form id="loginForm">
4.     <input name="username" type="text" />
5.     <input name="password" type="password" />
6.     <input name="continue" type="submit" value="Login" />
7.     <input name="continue" type="button" value="Clear" />
8.    </form>
9.  </body>
10. <html>
```

定位 username 元素的方法如下：

[python] view plain copy

```
1.  username = driver.find_element_by_xpath("//form[input/@name='username']")
2.  username = driver.find_element_by_xpath("//form[@id='loginForm']/input[1]")
3.  username = driver.find_element_by_xpath("//input[@name='username']")
```

[1] 第一个 form 元素通过一个 input 子元素，name 属性和值为 username 实现

[2] 通过 id=loginForm 值的 form 元素找到第一个 input 子元素

[3] 属性名为 name 且值为 username 的第一个 input 元素

二. 操作元素方法

在讲述完定位对象(locate elements)之后我们需要对该已定位对象进行操作，通常所有的操作与页面交互都将通过 WebElement 接口，常见的操作元素方法如下：

- clear 清除元素的内容
- send_keys 模拟按键输入
- click 点击元素
- submit 提交表单

举例自动访问 FireFox 浏览器自动登录 163 邮箱。

[python] view plain copy

```
1. from selenium import webdriver
2. from selenium.webdriver.common.keys import Keys
3. import time
4.
5. # Login 163 email
6. driver = webdriver.Firefox()
7. driver.get("http://mail.163.com/")
8.
9. elem_user = driver.find_element_by_name("username")
10. elem_user.clear
```

```
11. elem_user.send_keys("15201615157")
12. elem_pwd = driver.find_element_by_name("password")
13. elem_pwd.clear
14. elem_pwd.send_keys("******")
15. elem_pwd.send_keys(Keys.RETURN)
16. #driver.find_element_by_id("loginBtn").click()
17. #driver.find_element_by_id("loginBtn").submit()
18. time.sleep(5)
19. assert "baidu" in driver.title
20. driver.close()
21. driver.quit()
```

首先通过 name 定位用户名和密码，再调用方法 clear()清除输入框默认内容，如“请输入密码”等提示，通过 send_keys("**")输入正确的用户名和密码，最后通过 click()点击登录按钮或 send_keys(Keys.RETURN)相当于回车登录，submit()提交表单。

PS：如果需要输入中文，防止编码错误使用 send_keys(u"中文用户名")。

三. WebElement 接口获取值

通过 WebElement 接口可以获取常用的值，这些值同样非常重要。

- size 获取元素的尺寸
- text 获取元素的文本
- get_attribute(name) 获取属性值
- location 获取元素坐标，先找到要获取的元素，再调用该方法
- page_source 返回页面源码

- driver.title 返回页面标题
- current_url 获取当前页面的 URL
- is_displayed() 设置该元素是否可见
- is_enabled() 判断元素是否被使用
- is_selected() 判断元素是否被选中
- tag_name 返回元素的 tagName

举例代码如下：

[python] view plain copy

```
1. from selenium import webdriver
2. from selenium.webdriver.common.keys import Keys
3. import time
4.
5. driver = webdriver.PhantomJS(executable_path="G:\phantomjs-1.9.1-windows\phantomjs.exe")
6. driver.get("http://www.baidu.com/")
7.
8. size = driver.find_element_by_name("wd").size
9. print size
10. #尺寸: {'width': 500, 'height': 22}
11.
12. news = driver.find_element_by_xpath("//div[@id='u1']/a[1]").text
13. print news
```

```
14. #文本: 新闻
15.
16. href = driver.find_element_by_xpath("//div[@id='u1']/a[2]").get_attribute('href')
17. name = driver.find_element_by_xpath("//div[@id='u1']/a[2]").get_attribute('name')
18. print href,name
19. #属性值: http://www.hao123.com/ tj_trhao123
20.
21. location = driver.find_element_by_xpath("//div[@id='u1']/a[3]").location
22. print location
23. #坐标: {'y': 19, 'x': 498}
24.
25. print driver.current_url
26. #当前链接: https://www.baidu.com/
27. print driver.title
28. #标题: 百度一下， 你就知道
29.
30. result = location = driver.find_element_by_id("su").is_displayed()
31. print result
32. #是否可见: True
```

其中图片解释如下图所示。

四. 鼠标操作

在现实的自动化测试中关于鼠标的操作不仅仅是 click()单击操作 ,还有很多包含在 ActionChains 类中的操作。如下：

- context_click(elem) 右击鼠标点击元素 elem，另存为等行为
- double_click(elem) 双击鼠标点击元素 elem，地图 web 可实现放大功能
- drag_and_drop(source,target) 拖动鼠标，源元素按下左键移动至目标元素释放
- move_to_element(elem) 鼠标移动到一个元素上
- click_and_hold(elem) 按下鼠标左键在一个元素上
- perform() 在通过调用该函数执行 ActionChains 中存储行为

举例如下图所示，获取通过鼠标右键另存为百度图片 logo。代码：

[python] view plain copy

```
1. import time
2. from selenium import webdriver
3. from selenium.webdriver.common.keys import Keys
4. from selenium.webdriver.common.action_chains import ActionChains
5.
6. driver = webdriver.Firefox()
7. driver.get("http://www.baidu.com")
8.
9. #鼠标移动至图片上 右键保存图片
```

```
10. elem_pic = driver.find_element_by_xpath("//div[@id='lg']/img")
11. print elem_pic.get_attribute("src")
12. action = ActionChains(driver).move_to_element(elem_pic)
13. action.context_click(elem_pic)
14.
15. #重点:当右键鼠标点击键盘光标向下则移动至右键菜单第一个选项
16. action.send_keys(Keys.ARROW_DOWN)
17. time.sleep(3)
18. action.send_keys('v') #另存为
19. action.perform()
20.
21. #获取另存为对话框(失败)
22. alert.switch_to_alert()
23. alert.accept()
```

效果如下图所示，通过 xpath 定位到图片位置并右击鼠标，在弹出的菜单中选择“另存为图片”。但是如何点击“另存为对话框”的“保存”按钮是个难点，目前刚学习阶段，境界没到无法解决。原因：

WebDriver cannot directly interact with dialog windows this is because dialog windows are the domain of the operating system and not the webpage.

五. 键盘操作

前面讲述了鼠标操作，现在讲述键盘操作。在 webdriver 的 Keys 类中提供了键盘所有的按键操作，当然也包括一些常见的组合键操作如 Ctrl+A(全选)、Ctrl+C(复制)、Ctrl+V(粘贴)。更多键参考官方文档对应的编码。

- send_keys(Keys.ENTER) 按下回车键
- send_keys(Keys.TAB) 按下 Tab 制表键
- send_keys(Keys.SPACE) 按下空格键 space
- send_keys(Kyes.ESCAPE) 按下回退键 Esc
- send_keys(Keys.BACK_SPACE) 按下删除键 BackSpace
- send_keys(Keys.SHIFT) 按下 shift 键
- send_keys(Keys.CONTROL) 按下 Ctrl 键
- send_keys(Keys.ARROW_DOWN) 按下鼠标光标向下按键
- send_keys(Keys.CONTROL,'a') 组合键全选 Ctrl+A
- send_keys(Keys.CONTROL,'c') 组合键复制 Ctrl+C
- send_keys(Keys.CONTROL,'x') 组合键剪切 Ctrl+X
- send_keys(Keys.CONTROL,'v') 组合键粘贴 Ctrl+V