Python Selenium库的使用

# （一）Selenium基础

入门教程：[Selenium官网教程](http://www.selenium.org.cn/1598.html)

## 1.Selenium简介

Selenium是一个用于测试网站的自动化测试工具，支持各种浏览器包括Chrome、Firefox、Safari等主流界面浏览器，同时也支持phantomJS无界面浏览器。

## 2.支持多种操作系统

如Windows、Linux、IOS、Android等。

## 3.安装Selenium

pip install Selenium

## 4.安装浏览器驱动

Selenium3.x调用浏览器必须有一个webdriver驱动文件

Chrome驱动文件下载：[点击下载chromedrive](http://chromedriver.storage.googleapis.com/index.html)

Firefox驱动文件下载:[点解下载geckodrive](https://github.com/mozilla/geckodriver/releases)r

## 5.配置环境变量

设置浏览器的地址非常简单。 我们可以手动创建一个存放浏览器驱动的目录，如： F:\GeckoDriver , 将下载的浏览器驱动文件（例如：chromedriver、geckodriver）丢到该目录下。

我的电脑–>属性–>系统设置–>高级–>环境变量–>系统变量–>Path，将“F:\GeckoDriver”目录添加到Path的值中。比如：Path字段;F:\GeckoDriver

参考浏览器驱动环境配置

# (二)Selenium 快速入门

## 1.Selenium提供了8种定位方式:

id

name

class name

tag name

link text

partial link text

xpath

css selector

## 2.定位元素的8种方式

参考：[selenium元素定位](http://www.testclass.net/selenium_python/find-element/)

|  |  |  |
| --- | --- | --- |
| 定位一个元素 | 定位多个元素 | 含义 |
| find\_element\_by\_id | find\_elements\_by\_id | 通过元素id定位 |
| find\_element\_by\_name | find\_elements\_by\_name | 通过元素name定位 |
| find\_element\_by\_xpath | find\_elements\_by\_xpath | 通过xpath表达式定位 |
| find\_element\_by\_link\_text | find\_elements\_by\_link\_tex | 通过完整超链接定位 |
| find\_element\_by\_partial\_link\_text | find\_elements\_by\_partial\_link\_text | 通过部分链接定位 |
| find\_element\_by\_tag\_name | find\_elements\_by\_tag\_name | 通过标签定位 |
| find\_element\_by\_class\_name | find\_elements\_by\_class\_name | 通过类名进行定位 |
| find\_elements\_by\_css\_selector | find\_elements\_by\_css\_selector | 通过css选择器进行定位 |

## 3.实例演示

假如我们有一个Web页面，通过前端工具（如，Firebug）查看到一个元素的属性是这样的。

<html>

<head>

<body link="#0000cc">

<a id="result\_logo" href="/" onmousedown="return c({'fm':'tab','tab':'logo'})">

<form id="form" class="fm" name="f" action="/s">

<span class="soutu-btn"></span>

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

#### 通过id定位:

dr.find\_element\_by\_id("kw")

#### 通过name定位:

dr.find\_element\_by\_name("wd")

#### 通过class name定位:

dr.find\_element\_by\_class\_name("s\_ipt")

#### 通过tag name定位:

dr.find\_element\_by\_tag\_name("input")

#### 通过xpath定位:

xpath定位有N种写法，这里列几个常用写法:

dr.find\_element\_by\_xpath("//\*[@id='kw']")

dr.find\_element\_by\_xpath("//\*[@name='wd']")

dr.find\_element\_by\_xpath("//input[@class='s\_ipt']")

dr.find\_element\_by\_xpath("/html/body/form/span/input")

dr.find\_element\_by\_xpath("//span[@class='soutu-btn']/input")

dr.find\_element\_by\_xpath("//form[@id='form']/span/input")

dr.find\_element\_by\_xpath("//input[@id='kw' and @name='wd']")

#### 通过css定位:

css定位有N种写法，这里列几个常用写法:

dr.find\_element\_by\_css\_selector("#kw")

dr.find\_element\_by\_css\_selector("[name=wd]")

dr.find\_element\_by\_css\_selector(".s\_ipt")

dr.find\_element\_by\_css\_selector("html > body > form > span > input")

dr.find\_element\_by\_css\_selector("span.soutu-btn> input#kw")

dr.find\_element\_by\_css\_selector("form#form > span > input")

接下来，我们的页面上有一组文本链接。

<a class="mnav" href="http://news.baidu.com" name="tj\_trnews">新闻</a>

<a class="mnav" href="http://www.hao123.com" name="tj\_trhao123">hao123</a>

#### 通过link text定位:

dr.find\_element\_by\_link\_text("新闻")

dr.find\_element\_by\_link\_text("hao123")

#### 通过partial link text定位:

dr.find\_element\_by\_partial\_link\_text("新")

dr.find\_element\_by\_partial\_link\_text("hao")

dr.find\_element\_by\_partial\_link\_text("123")

## 4.Selenium库下webdriver模块常用方法的使用

### 1.控制浏览器操作的一些方法

|  |  |
| --- | --- |
| 方法 | 说明 |
| set\_window\_size() | 设置浏览器的大小 |
| back() | 控制浏览器后退 |
| forward() | 控制浏览器前进 |
| refresh() | 刷新当前页面 |
| clear() | 清除文本 |
| send\_keys (value) | 模拟按键输入 |
| click() | 单击元素 |
| submit() | 用于提交表单 |
| get\_attribute(name) | 获取元素属性值 |
| is\_displayed() | 设置该元素是否用户可见 |
| size | 返回元素的尺寸 |
| text | 获取元素的文本 |

实例演示

from selenium import webdriver

from time import sleep

#1.创建Chrome浏览器对象，这会在电脑上在打开一个浏览器窗口

browser = webdriver.Firefox(executable\_path ="F:\GeckoDriver\geckodriver")

#2.通过浏览器向服务器发送URL请求

browser.get("https://www.baidu.com/")

sleep(3)

#3.刷新浏览器

browser.refresh()

#4.设置浏览器的大小

browser.set\_window\_size(1400,800)

#5.设置链接内容

element=browser.find\_element\_by\_link\_text("新闻")

element.click()

element=browser.find\_element\_by\_link\_text("“下团组”时间")

element.click()

### 2.鼠标事件

在 WebDriver 中， 将这些关于鼠标操作的方法封装在 ActionChains 类提供。

|  |  |
| --- | --- |
| 方法 | 说明 |
| ActionChains(driver) | 构造ActionChains对象 |
| context\_click() | 执行鼠标悬停操作 |
| move\_to\_element(above) | 右击 |
| double\_click() | 双击 |
| drag\_and\_drop() | 拖动 |
| move\_to\_element(above) | 执行鼠标悬停操作 |
| context\_click() | 用于模拟鼠标右键操作， 在调用时需要指定元素定位 |
| perform() | 执行所有 ActionChains 中存储的行为，可以理解成是对整个操作的提交动作 |

实例演示



from selenium import webdriver

#1.引入 ActionChains 类

from selenium.webdriver.common.action\_chains import ActionChains

#1.创建Chrome浏览器对象，这会在电脑上在打开一个浏览器窗口

driver = webdriver.Firefox(executable\_path ="F:\GeckoDriver\geckodriver")

driver.get("https://www.baidu.com")

#2.定位到要悬停的元素

element= driver.find\_element\_by\_link\_text("设置")

#3.对定位到的元素执行鼠标悬停操作

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

#找到链接

elem1=driver.find\_element\_by\_link\_text("搜索设置")

elem1.click()

#通过元素选择器找到id=sh\_2,并点击设置

elem2=driver.find\_element\_by\_id("sh\_1")

elem2.click()

#保存设置

elem3=driver.find\_element\_by\_class\_name("prefpanelgo")

elem3.click()

### 3.键盘事件

Selenium中的Key模块为我们提供了模拟键盘按键的方法，那就是send\_keys()方法。它不仅可以模拟键盘输入，也可以模拟键盘的操作。

常用的键盘操作如下：

|  |  |
| --- | --- |
| 模拟键盘按键 | 说明 |
| send\_keys(Keys.BACK\_SPACE) | 删除键（BackSpace） |
| send\_keys(Keys.SPACE) | 空格键(Space) |
| send\_keys(Keys.TAB) | 制表键(Tab) |
| send\_keys(Keys.ESCAPE) | 回退键（Esc） |
| send\_keys(Keys.ENTER) | 回车键（Enter） |

组合键的使用

|  |  |
| --- | --- |
| 模拟键盘按键 | 说明 |
| send\_keys(Keys.CONTROL,‘a’) | 全选（Ctrl+A） |
| send\_keys(Keys.CONTROL,‘c’) | 复制（Ctrl+C） |
| send\_keys(Keys.CONTROL,‘x’) | 剪切（Ctrl+X） |
| send\_keys(Keys.CONTROL,‘v’) | 粘贴（Ctrl+V） |
| send\_keys(Keys.F1…Fn) | 键盘 F1…Fn |

### 4.获取断言信息

不管是在做功能测试还是自动化测试，最后一步需要拿实际结果与预期进行比较。这个比较的称之为断言。通过我们获取title 、URL和text等信息进行断言。

|  |  |
| --- | --- |
| 属性 | 说明 |
| title | 用于获得当前页面的标题 |
| current\_url | 用户获得当前页面的URL |
| text | 获取搜索条目的文本信息 |

实例演示

from selenium import webdriver

from time import sleep

driver = webdriver.Firefox(executable\_path ="F:\GeckoDriver\geckodriver")

driver.get("https://www.baidu.com")

print('Before search================')

# 打印当前页面title

title = driver.title

print(title)

# 打印当前页面URL

now\_url = driver.current\_url

print(now\_url)

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

driver.find\_element\_by\_id("su").click()

sleep(1)

print('After search================')

# 再次打印当前页面title

title = driver.title

print(title)

# 打印当前页面URL

now\_url = driver.current\_url

print(now\_url)

# 获取结果数目

user = driver.find\_element\_by\_class\_name('nums').text

print(user)

#关闭所有窗口

driver.quit()

打印输出结果

Before search================

百度一下，你就知道

https://www.baidu.com/

After search================

selenium\_百度搜索

https://www.baidu.com/s?ie=utf-8&f=8&rsv\_bp=0&rsv\_idx=1&tn=baidu&wd=selenium&rsv\_pq=a1d51b980000e36e&rsv\_t=a715IZaMpLd1w92I4LNUi7gKuOdlAz5McsHe%2FSLQeBZD44OUIPnjY%2B7pODM&rqlang=cn&rsv\_enter=0&rsv\_sug3=8&inputT=758&rsv\_sug4=759

搜索工具

百度为您找到相关结果约7,170,000个

### 5.设置元素等待:

[参考文献](http://www.testclass.net/selenium_python/element-wait/)

### 6.定位一组元素

定位一组元素的方法与定位单个元素的方法类似，唯一的区别是在单词element后面多了一个s表示复数。

实例演示

from selenium import webdriver

from time import sleep

driver =webdriver.Firefox(executable\_path ="F:\GeckoDriver\geckodriver")

driver.get("https://www.baidu.com")

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

driver.find\_element\_by\_id("su").click()

sleep(1)

#1.定位一组元素

elements = driver.find\_elements\_by\_xpath('//div/h3/a')

print(type(elements))

#2.循环遍历出每一条搜索结果的标题

for t in elements:

print(t.text)

element=driver.find\_element\_by\_link\_text(t.text)

element.click()

sleep(3)

driver.quit()

### 7.多表单切换

在Web应用中经常会遇到frame/iframe表单嵌套页面的应用，WebDriver只能在一个页面上对元素识别与定位，对于frame/iframe表单内嵌页面上的元素无法直接定位。这时就需要通过switch\_to.frame()方法将当前定位的主体切换为frame/iframe表单的内嵌页面中。

|  |  |
| --- | --- |
| 方法 | 说明 |
| switch\_to.frame() | 将当前定位的主体切换为frame/iframe表单的内嵌页面中 |
| switch\_to.default\_content() | 跳回最外层的页面 |

<html>

<body>

...

<iframe id="x-URS-iframe" ...>

<html>

<body>

...

<input name="email" >

126邮箱登录框的结构大概是这样子的，想要操作登录框必须要先切换到iframe表单。

from selenium import webdriver

driver = webdriver.Chrome()

driver.get("http://www.126.com")

driver.switch\_to.frame('x-URS-iframe')

driver.find\_element\_by\_name("email").clear()

driver.find\_element\_by\_name("email").send\_keys("username")

driver.find\_element\_by\_name("password").clear()

driver.find\_element\_by\_name("password").send\_keys("password")

driver.find\_element\_by\_id("dologin").click()

driver.switch\_to.default\_content()

driver.quit()

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。

……

#先通过xpth定位到iframe

xf = driver.find\_element\_by\_xpath('//\*[@id="x-URS-iframe"]')

#再将定位对象传给switch\_to.frame()方法

driver.switch\_to.frame(xf)

……

driver.switch\_to.parent\_frame()

### 8.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，可以实现在不同的窗口之间切换。

|  |  |
| --- | --- |
| 方法 | 说明 |
| current\_window\_handle | 获得当前窗口句柄 |
| window\_handles | 返回所有窗口的句柄到当前会话 |
| switch\_to.window() | 用于切换到相应的窗口，与上一节的switch\_to.frame()类似，前者用于不同窗口的切换，后者用于不同表单之间的切换。 |

实例演示

from selenium import webdriver

import time

driver = webdriver.Chrome("F:\Chrome\ChromeDriver\chromedriver")

driver.implicitly\_wait(10)

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

#1.获得百度搜索窗口句柄

sreach\_windows = driver.current\_window\_handle

driver.find\_element\_by\_link\_text('登录').click()

driver.find\_element\_by\_link\_text("立即注册").click()

#1.获得当前所有打开的窗口的句柄

all\_handles = driver.window\_handles

#3.进入注册窗口

for handle in all\_handles:

if handle != sreach\_windows:

driver.switch\_to.window(handle)

print('跳转到注册窗口')

driver.find\_element\_by\_name("account").send\_keys('123456789')

driver.find\_element\_by\_name('password').send\_keys('123456789')

time.sleep(2)

driver.quit()

### 9.警告框处理

在WebDriver中处理JavaScript所生成的alert、confirm以及prompt十分简单，具体做法是使用 switch\_to.alert 方法定位到 alert/confirm/prompt，然后使用text/accept/dismiss/ send\_keys等方法进行操作。

|  |  |
| --- | --- |
| 方法 | 说明 |
| text | 返回 alert/confirm/prompt 中的文字信息 |
| accept() | 接受现有警告框 |
| dismiss() | 解散现有警告框 |
| send\_keys(keysToSend) | 发送文本至警告框。keysToSend：将文本发送至警告框。 |

实例演示

from selenium import webdriver

from selenium.webdriver.common.action\_chains import ActionChains

import time

driver = webdriver.Chrome("F:\Chrome\ChromeDriver\chromedriver")

driver.implicitly\_wait(10)

driver.get('http://www.baidu.com')

# 鼠标悬停至“设置”链接

link = driver.find\_element\_by\_link\_text('设置')

ActionChains(driver).move\_to\_element(link).perform()

# 打开搜索设置

driver.find\_element\_by\_link\_text("搜索设置").click()

#在此处设置等待2s否则可能报错

time.sleep(2)

# 保存设置

driver.find\_element\_by\_class\_name("prefpanelgo").click()

time.sleep(2)

# 接受警告框

driver.switch\_to.alert.accept()

driver.quit()

### 10.下拉框选择操作

导入选择下拉框Select类，使用该类处理下拉框操作。

from selenium.webdriver.support.select import Select

Select类的方法

|  |  |
| --- | --- |
| 方法 | 说明 |
| select\_by\_value(“选择值”) | select标签的value属性的值 |
| select\_by\_index(“索引值”) | 下拉框的索引 |
| select\_by\_visible\_testx(“文本值”) | 下拉框的文本值 |

有时我们会碰到下拉框，WebDriver提供了Select类来处理下拉框。 如百度搜索设置的下拉框，如下图：

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-7WUxrkEj-1571973999629)(http://orru5lls3.bkt.clouddn.com/select.png)]

from selenium import webdriver

from selenium.webdriver.support.select import Select

from time import sleep

driver = webdriver.Chrome("F:\Chrome\ChromeDriver\chromedriver")

driver.implicitly\_wait(10)

driver.get('http://www.baidu.com')

#1.鼠标悬停至“设置”链接

driver.find\_element\_by\_link\_text('设置').click()

sleep(1)

#2.打开搜索设置

driver.find\_element\_by\_link\_text("搜索设置").click()

sleep(2)

#3.搜索结果显示条数

sel = driver.find\_element\_by\_xpath("//select[@id='nr']")

Select(sel).select\_by\_value('50') # 显示50条

sleep(3)

driver.quit()

### 11.文件上传

对于通过input标签实现的上传功能，可以将其看作是一个输入框，即通过send\_keys()指定本地文件路径的方式实现文件上传。

通过send\_keys()方法来实现文件上传:

from selenium import webdriver

import os

driver = webdriver.Firefox()

file\_path = 'file:///' + os.path.abspath('upfile.html')

driver.get(file\_path)

# 定位上传按钮，添加本地文件

driver.find\_element\_by\_name("file").send\_keys('D:\\upload\_file.txt')

driver.quit()

### 12.cookie操作

有时候我们需要验证浏览器中cookie是否正确，因为基于真实cookie的测试是无法通过白盒和集成测试进行的。WebDriver提供了操作Cookie的相关方法，可以读取、添加和删除cookie信息。

WebDriver操作cookie的方法:

|  |  |
| --- | --- |
| 方法 | 说明 |
| get\_cookies() | 获得所有cookie信息 |
| get\_cookie(name) | 返回字典的key为“name”的cookie信息 |
| add\_cookie(cookie\_dict) | 添加cookie。“cookie\_dict”指字典对象，必须有name 和value 值 |
| delete\_cookie(name,optionsString) | 删除cookie信息。“name”是要删除的cookie的名称，“optionsString”是该cookie的选项，目前支持的选项包括“路径”，“域” |
| delete\_all\_cookies() | 删除所有cookie信息 |

实例演示

from selenium import webdriver

import time

browser = webdriver.Chrome("F:\Chrome\ChromeDriver\chromedriver")

browser.get("http://www.youdao.com")

#1.打印cookie信息

print('=====================================')

print("打印cookie信息为：")

print(browser.get\_cookies)

#2.添加cookie信息

dict={'name':"name",'value':'Kaina'}

browser.add\_cookie(dict)

print('=====================================')

print('添加cookie信息为：')

#3.遍历打印cookie信息

for cookie in browser.get\_cookies():

print('%s----%s\n' %(cookie['name'],cookie['value']))

#4.删除一个cookie

browser.delete\_cookie('name')

print('=====================================')

print('删除一个cookie')

for cookie in browser.get\_cookies():

print('%s----%s\n' %(cookie['name'],cookie['value']))

print('=====================================')

print('删除所有cookie后：')

#5.删除所有cookie,无需传递参数

browser.delete\_all\_cookies()

for cookie in browser.get\_cookies():

print('%s----%s\n' %(cookie['name'],cookie['value']))

time.sleep(3)

browser.close()

### 13.调用JavaScript代码

虽然WebDriver提供了操作浏览器的前进和后退方法，但对于浏览器滚动条并没有提供相应的操作方法。在这种情况下，就可以借助JavaScript来控制浏览器的滚动条。WebDriver提供了execute\_script()方法来执行JavaScript代码。

用于调整浏览器滚动条位置的JavaScript代码如下：

<!-- window.scrollTo(左边距,上边距); -->

window.scrollTo(0,450);

window.scrollTo()方法用于设置浏览器窗口滚动条的水平和垂直位置。方法的第一个参数表示水平的左间距，第二个参数表示垂直的上边距。其代码如下：

from selenium import webdriver

from time import sleep

#1.访问百度

driver=webdriver.Firefox(executable\_path ="F:\GeckoDriver\geckodriver")

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

#2.搜索

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

driver.find\_element\_by\_id("su").click()

#3.休眠2s目的是获得服务器的响应内容，如果不使用休眠可能报错

sleep(2)

#4.通过javascript设置浏览器窗口的滚动条位置

js="window.scrollTo(100,450);"

driver.execute\_script(js)

sleep(3)

driver.close()

通过浏览器打开百度进行搜索，并且提前通过set\_window\_size()方法将浏览器窗口设置为固定宽高显示，目的是让窗口出现水平和垂直滚动条。然后通过execute\_script()方法执行JavaScripts代码来移动滚动条的位置。

滚动条上下左右滚动代码演示

from selenium import webdriver

from time import sleep

driver=webdriver.Firefox(executable\_path ="F:\GeckoDriver\geckodriver")

driver.set\_window\_size(400,400)

driver.get("https://www.baidu.com")

#2.搜索

# driver.find\_element\_by\_id("kw").send\_keys("selenium")

# driver.find\_element\_by\_id("su").click()

#3.休眠2s目的是获得服务器的响应内容，如果不使用休眠可能报错

sleep(10)

#4 滚动左右滚动条---向右

js2 = "var q=document.documentElement.scrollLeft=10000"

driver.execute\_script(js2)

sleep(15)

#5 滚动左右滚动条---向左

js3 = "var q=document.documentElement.scrollLeft=0"

driver.execute\_script(js3)

sleep(15)

#6 拖动到滚动条底部---向下

js = "var q=document.documentElement.scrollTop=10000"

driver.execute\_script(js)

sleep(15)

#7 拖动到滚动条底部---向上

js = "var q=document.documentElement.scrollTop=0"

driver.execute\_script(js)

sleep(15)

driver.close()

### 14.窗口截图

自动化用例是由程序去执行的，因此有时候打印的错误信息并不十分明确。如果在脚本执行出错的时候能对当前窗口截图保存，那么通过图片就可以非常直观地看出出错的原因。WebDriver提供了截图函数get\_screenshot\_as\_file()来截取当前窗口。

截屏方法：

|  |  |
| --- | --- |
| 方法 | 说明 |
| get\_screenshot\_as\_file(self, filename) | 用于截取当前窗口，并把图片保存到本地 |

from selenium import webdriver

from time import sleep

driver =webdriver.Firefox(executable\_path ="F:\GeckoDriver\geckodriver")

driver.get('http://www.baidu.com')

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

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

sleep(2)

#1.截取当前窗口，并指定截图图片的保存位置

driver.get\_screenshot\_as\_file("D:\\baidu\_img.jpg")

driver.quit()

### 15.关闭浏览器

在前面的例子中我们一直使用quit()方法，其含义为退出相关的驱动程序和关闭所有窗口。除此之外，WebDriver还提供了close()方法，用来关闭当前窗口。例多窗口的处理，在用例执行的过程中打开了多个窗口，我们想要关闭其中的某个窗口，这时就要用到close()方法进行关闭了。

|  |  |
| --- | --- |
| 方法 | 说明 |
| close() | 关闭单个窗口 |
| quit() | 关闭所有窗口 |

————————————————

版权声明：本文为CSDN博主「凯耐」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_36279318/java/article/details/79475388