用selenium做自动化，有时候会遇到需要模拟鼠标操作才能进行的情况，比如单击、双击、点击鼠标右键、拖拽等等。而selenium给我们提供了一个类来处理这类事件——**ActionChains**

selenium.webdriver.common.action\_chains.**ActionChains**(driver)

这个类基本能够满足我们所有对鼠标操作的需求。

## ****1.ActionChains基本用法****

首先需要了解ActionChains的执行原理，当你调用ActionChains的方法时，不会立即执行，而是会将所有的操作按顺序存放在一个队列里，当你调用perform()方法时，队列中的时间会依次执行。

这种情况下我们可以有两种调用方法：

链式写法

menu = driver.find\_element\_by\_css\_selector(".nav")

hidden\_submenu = driver.find\_element\_by\_css\_selector(".nav #submenu1")

ActionChains(driver).move\_to\_element(menu).click(hidden\_submenu).perform()

分步写法

menu = driver.find\_element\_by\_css\_selector(".nav")

hidden\_submenu = driver.find\_element\_by\_css\_selector(".nav #submenu1")

actions = ActionChains(driver)

actions.move\_to\_element(menu)

actions.click(hidden\_submenu)

actions.perform()

两种写法本质是一样的，ActionChains都会按照顺序执行所有的操作。

## ****2.ActionChains方法列表****

**click**(on\_element=None) ——单击鼠标左键

**click\_and\_hold**(on\_element=None) ——点击鼠标左键，不松开

**context\_click**(on\_element=None) ——点击鼠标右键

**double\_click**(on\_element=None) ——双击鼠标左键

**drag\_and\_drop**(source, target) ——拖拽到某个元素然后松开

**drag\_and\_drop\_by\_offset**(source, xoffset, yoffset) ——拖拽到某个坐标然后松开

**key\_down**(value, element=None) ——按下某个键盘上的键

**key\_up**(value, element=None) ——松开某个键

**move\_by\_offset**(xoffset, yoffset) ——鼠标从当前位置移动到某个坐标

**move\_to\_element**(to\_element) ——鼠标移动到某个元素

**move\_to\_element\_with\_offset**(to\_element, xoffset, yoffset) ——移动到距某个元素（左上角坐标）多少距离的位置

**perform**() ——执行链中的所有动作

**release**(on\_element=None) ——在某个元素位置松开鼠标左键

**send\_keys**(\*keys\_to\_send) ——发送某个键到当前焦点的元素

**send\_keys\_to\_element**(element, \*keys\_to\_send) ——发送某个键到指定元素

接下来用示例来详细说明和演示每一个方法的用法：

## ****3.代码示例****

**1. 点击操作**

示例网址<http://sahitest.com/demo/clicks.htm>

代码：

# -\*- coding: utf-8 -\*-

from selenium import webdriver

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

from time import sleep

driver = webdriver.Firefox()

driver.implicitly\_wait(10)

driver.maximize\_window()

driver.get('http://sahitest.com/demo/clicks.htm')

click\_btn = driver.find\_element\_by\_xpath('//input[@value="click me"]') # 单击按钮

doubleclick\_btn = driver.find\_element\_by\_xpath('//input[@value="dbl click me"]') # 双击按钮

rightclick\_btn = driver.find\_element\_by\_xpath('//input[@value="right click me"]') # 右键单击按钮

ActionChains(driver).click(click\_btn).double\_click(doubleclick\_btn).context\_click(rightclick\_btn).perform() # 链式用法

print driver.find\_element\_by\_name('t2').get\_attribute('value')

sleep(2)

driver.quit()

结果：

[CLICK][DOUBLE\_CLICK][RIGHT\_CLICK]

**2.鼠标移动**

示例网址<http://sahitest.com/demo/mouseover.htm>

示例代码：

# -\*- coding: utf-8 -\*-

from selenium import webdriver

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

from time import sleep

driver = webdriver.Firefox()

driver.implicitly\_wait(10)

driver.maximize\_window()

driver.get('http://sahitest.com/demo/mouseover.htm')

write = driver.find\_element\_by\_xpath('//input[@value="Write on hover"]')

# 鼠标移动到此元素，在下面的input框中会显示“Mouse moved”

blank = driver.find\_element\_by\_xpath('//input[@value="Blank on hover"]')

# 鼠标移动到此元素，会清空下面input框中的内容

result = driver.find\_element\_by\_name('t1')

action = ActionChains(driver)

action.move\_to\_element(write).perform()

移动到write，显示“Mouse moved”

print result.get\_attribute('value')

# action.move\_to\_element(blank).perform()

action.move\_by\_offset(10, 50).perform()

# 移动到距离当前位置(10,50)的点，与上句效果相同，移动到blank上，清空

print result.get\_attribute('value')

action.move\_to\_element\_with\_offset(blank, 10, -40).perform()

# 移动到距离blank元素(10,-40)的点，可移动到write上

print result.get\_attribute('value')

sleep(2)

driver.quit()

结果

Mouse moved

Mouse moved

一般很少用位置关系来移动鼠标，如果需要，可参考下面的链接来测量元素位置

<http://jingyan.baidu.com/article/eb9f7b6d87e2ae869264e847.html>

**3.拖拽**

示例网址<http://sahitest.com/demo/dragDropMooTools.htm>

代码：

# -\*- coding: utf-8 -\*-

from selenium import webdriver

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

from time import sleep

driver = webdriver.Firefox()

driver.implicitly\_wait(10)

driver.maximize\_window()

driver.get('http://sahitest.com/demo/dragDropMooTools.htm')

dragger = driver.find\_element\_by\_id('dragger') # 被拖拽元素

item1 = driver.find\_element\_by\_xpath('//div[text()="Item 1"]') # 目标元素1

item2 = driver.find\_element\_by\_xpath('//div[text()="Item 2"]') # 目标2

item3 = driver.find\_element\_by\_xpath('//div[text()="Item 3"]') # 目标3

item4 = driver.find\_element\_by\_xpath('//div[text()="Item 4"]') # 目标4

action = ActionChains(driver)

action.drag\_and\_drop(dragger, item1).perform() # 1.移动dragger到目标1

sleep(2)

action.click\_and\_hold(dragger).release(item2).perform() # 2.效果与上句相同，也能起到移动效果

sleep(2)

action.click\_and\_hold(dragger).move\_to\_element(item3).release().perform()

# 3.效果与上两句相同，也能起到移动的效果

sleep(2)

# action.drag\_and\_drop\_by\_offset(dragger, 400, 150).perform() # 4.移动到指定坐标

action.click\_and\_hold(dragger).move\_by\_offset(400, 150).release().perform()

# 5.与上一句相同，移动到指定坐标

sleep(2)

driver.quit()

结果：

dropped dropped dropped dropped

一般用坐标定位很少，用上例中的方法1足够了，如果看源码，会发现方法2其实就是方法1中的drag\_and\_drop()的实现。**注意：拖拽使用时注意加等待时间，有时会因为速度太快而失败。**

**4.按键**

模拟按键有多种方法，能用win32api来实现，能用SendKeys来实现，也可以用selenium的WebElement对象的send\_keys()方法来实现，这里ActionChains类也提供了几个模拟按键的方法。

示例网址<http://sahitest.com/demo/keypress.htm>

代码1：

# -\*- coding: utf-8 -\*-

from selenium import webdriver

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

from time import sleep

driver = webdriver.Firefox()

driver.implicitly\_wait(10)

driver.maximize\_window()

driver.get('http://sahitest.com/demo/keypress.htm')

key\_up\_radio = driver.find\_element\_by\_id('r1') # 监测按键升起

key\_down\_radio = driver.find\_element\_by\_id('r2') # 监测按键按下

key\_press\_radio = driver.find\_element\_by\_id('r3') # 监测按键按下升起

enter = driver.find\_elements\_by\_xpath('//form[@name="f1"]/input')[1] # 输入框

result = driver.find\_elements\_by\_xpath('//form[@name="f1"]/input')[0] # 监测结果

# 监测key\_down

key\_down\_radio.click()

ActionChains(driver).key\_down(Keys.CONTROL, enter).key\_up(Keys.CONTROL).perform()

print result.get\_attribute('value')

# 监测key\_up

key\_up\_radio.click()

enter.click()

ActionChains(driver).key\_down(Keys.SHIFT).key\_up(Keys.SHIFT).perform()

print result.get\_attribute('value')

# 监测key\_press

key\_press\_radio.click()

enter.click()

ActionChains(driver).send\_keys('a').perform()

print result.get\_attribute('value')

driver.quit()

结果：

key downed charCode=[0] keyCode=[17] CTRLkey upped charCode=[0] keyCode=[16] NONEkey pressed charCode=[97] keyCode=[0] NONE

示例2：

示例网址<http://sahitest.com/demo/label.htm>

代码：

# -\*- coding: utf-8 -\*-

from selenium import webdriver

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

from selenium.webdriver.common.keys import Keys

from time import sleep

driver = webdriver.Firefox()

driver.implicitly\_wait(10)

driver.maximize\_window()

driver.get('http://sahitest.com/demo/label.htm')

input1 = driver.find\_elements\_by\_tag\_name('input')[3]

input2 = driver.find\_elements\_by\_tag\_name('input')[4]

action = ActionChains(driver)

input1.click()

action.send\_keys('Test Keys').perform()

action.key\_down(Keys.CONTROL).send\_keys('a').key\_up(Keys.CONTROL).perform() # ctrl+a

action.key\_down(Keys.CONTROL).send\_keys('c').key\_up(Keys.CONTROL).perform() # ctrl+c

action.key\_down(Keys.CONTROL, input2).send\_keys('v').key\_up(Keys.CONTROL).perform() # ctrl+v

print input1.get\_attribute('value')

print input2.get\_attribute('value')

driver.quit()

结果：

Test Keys

Test Keys

复制粘贴用WebElement< input >.send\_keys()也能实现，大家可以试一下，也可以用更底层的方法，同时也是os弹框的处理办法之一的win32api，有兴趣也可以试试SendKeys、[keybd\_event](http://blog.csdn.net/yizhou2010/article/details/6178115)