ALL SELENIUM

Python Selenium all mouse actions using ActionChains

The Blog To Learn Selenium and Test Automation

Python Selenium all mouse actions using ActionChains

admin Python Selenium January 10, 2019 | 0

Page Contents

In this article, we are going to see all mouse actions available and supported by Python Selenium WebDriver as ActionChains object.

As discussed in our previous article, there might be many scenarios where we need to use mouse actions in automation testing. Some common uses case are given below.

- Mouse over on a menu item to bring hidden sub-menu
- Right click on a link or an element and do some operations
- Drag and drop items
- Double click on an item

What is ActionChains?

ActionChains are a ways provided by Selenium to automate low level interactions with website such as mouse movements, mouse button actions, key press, and context menu(right click menu) interactions. These special methods are useful for doing more complex actions like mouse over and drag and drop that are not possible by direct webdriver actions.

Using ActionChains object, we call methods that perform specific actions sequentially one by one. Actions called are pushed into a queue. When perform() method is called, the events are fired in the order they are queued up.

The ActionChains implementation is available in below path.

Python

```python
from selenium.webdriver.common.action_chains import ActionChains

actionchains = ActionChains(driver) # initialize ActionChain object
```

The WebDriver instance should be passed while creating ActionChains object.

ActionChains can be used in a chain pattern, adding all the methods in the same line.

Python

```python
menu = driver.find_element_by_id("menu")
submenu = driver.find_element_by_id("submenu1")
ActionChains(driver)
    .move_to_element(menu)
    .click(submenu)
    .perform()
```

Actions can be added to one by one to queue, then performed.

Python

```python
menu = driver.find_element_by_id("menu")
submenu = driver.find_element_by_id("submenu1")

actions = ActionChains(driver)
actions.move_to_element(menu)
actions.click(submenu)
actions.perform()
```

In both ways, the actions are executed or performed one after another in the order they are added.

All available ActionChains method

click(on_element=None)

Left clicks on a given element.

Parameters

on_element: The element to click on. If no element passed, clicks left mouse button on current mouse position.

click_and_hold(on_element=None)

Click and holds down the left mouse button on an element.

Parameters

on_element: The element to mouse down. If no element passed, clicks on current mouse position.

Performs a context-click (right click) on an element.

Parameters

on_element: The element to context-click or right click on. If no element passed, right clicks on current mouse position.

double_click(on_element=None)

Double-clicks on an element.

Parameters

on_element: The element to double-click on. If no element passed, double clicks on current mouse position.

drag_and_drop(source, target)

Holds down the left mouse button on the source element, then moves to the target element and releases the mouse button.

Parameters

source: The element to mouse down or the element to be dragged
target: The element to mouse up or target element to be dragged into

Example

Python

```python
from_element = driver.find_element_by_id("source")
to_element = driver.find_element_by_id("target")
ActionChains(driver)
    .drag_and_drop(from_element, to_element)
    .perform()
```

drag_and_drop_by_offset(source, xoffset, yoffset)

Holds down the left mouse button on the source element, then moves to the target offset and releases the mouse button.

Parameters

source: The element to mouse down.
xoffset: X offset to move to.
yoffset: Y offset to move to.

Sends a key press only, without releasing it. This is used mainly with modifier keys like Control, Alt and Shift.

Parameters

value: The modifier key to send. Keys class defines all values.
element: The element to send keys. If None, sends a key to current focused element.

Example, below script presses ctrl+c

Python

```python
ActionChains(driver)
    .key_down(Keys.CONTROL)
    .send_keys('c')
    .key_up(Keys.CONTROL)
    .perform()
```

key_up(value, element=None)

Releases a modifier key.

Parameters

value: The modifier key to send. Keys class defines all the values.
element: The element to send keys. If no element passed, sends a key to current focused element.

Example, below script presses ctrl+v

Python

```python
ActionChains(driver)
    .key_down(Keys.CONTROL)
    .send_keys('v')
    .key_up(Keys.CONTROL)
    .perform()
```

move_by_offset(xoffset, yoffset)

Moving the mouse to an offset from current mouse position.

Parameters

xoffset: X offset to move to, as a positive or negative integer.
yoffset: Y offset to move to, as a positive or negative integer.

Moving the mouse to the middle of an element. This action helps us to deal with dropdown menu that appears when the user moves the mouse over an element or when the user clicks on an element.

Parameters

to_element: The WebElement to move to.

Example

Python

```python
menu = driver.find_element_by_id("allmenu")
ActionChains(driver)
    .move_to_element(menu)
    .perform()
# wait until sub menu appears on the screen
WebDriverWait(self.webdriver, 5)
    .until(EC.visibility_of_element_located((By.ID, "home")))
home_menu = driver.find_element_by_id("home")
home_menu.click()
```

move_to_element_with_offset(to_element, xoffset, yoffset)

Move the mouse by an offset of the specified element.
Offsets are relative to the top-left corner of the element.

Parameters

to_element: The WebElement to move to.
xoffset: X offset to move to.
yoffset: Y offset to move to.

pause(seconds)

Pause all actions for the specified duration (in seconds).

perform()

Performs all stored actions.

release(on_element=None)

Releasing a held mouse button on an element.

Parameters

on_element: The element to mouse up. If None, releases on current mouse position.

reset_actions()

To clear all actions already stored locally and/or on the remote end.

send_keys(keys_to_send)

Sends keys to current focused element.

Parameters

keys_to_send: The keys to send.

send_keys_to_element(element, keys_to_send)

Sends keys to an element.

Parameters

element: The element to send keys.
keys_to_send: The keys to send.

For both send_keys and send_keys_to_element method arguments, Modifier keys constants can be found in the 'Keys' class.

Common Scenarios

Open link in new Tab

User has to press Control key and click on the link to open it in new browser tab. Same operation can be scripted in Selenium WebDriver as follows

Python

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains

driver = webdriver.Firefox(executable_path="path to firefox webdriver")
driver.get('https://www.google.com')
element = driver.find_element_by_link_text('Privacy')
ActionChains(driver)
    .key_down(Keys.CONTROL)
    .click(element)
    .key_up(Keys.CONTROL)
    .perform()
driver.quit()
```

In case of Mac os

Python

```python
ActionChains(driver) \
    .key_down(Keys.COMMAND) \
    .click(element) \
    .key_up(Keys.COMMAND) \
    .perform()
```

User has to press Shift key and click on the link to open it in new browser window. Same operation can be scripted in Selenium WebDriver as follows

Python

```python
ActionChains(driver) \
    .key_down(Keys.SHIFT) \
    .click(element) \
    .key_up(Keys.SHIFT) \
    .perform()
```

Handle Mouse overs

Python

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains

driver = webdriver.Firefox(executable_path="path to driver")
driver.get('https://www.w3schools.com/howto/howto_css_dropdown.asp')
element = driver.find_element_by_class_name('dropbtn')
ActionChains(driver) \
        .move_to_element(element) \
        .perform()
element = driver.find_element_by_link_text('Link 1')
element.click()
driver.quit()
```

Use pause() method if you like to wait before each operations. For example, if it takes few seconds to display menu items.

Hope this article helps you. Please let us know your thoughts in comments section.

Related Posts

Mouse Over actions using Python Selenium WebDriver

Capture screenshot of an Element using Python Selenium WebDriver

how to handle Stale Element Reference Exception in Python Selenium

ACTION CHAINS PYTHON SELENIUM

« Previous: TestNG Annotations 101

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment

Search ...

RECENT POSTS

Python Selenium all mouse actions using ActionChains

TestNG Annotations 101

How to debug tests running on Docker containers

ARCHIVES

January 2019

December 2018

November 2018

October 2018

September 2018

August 2018

June 2018

May 2018

April 2018

March 2018

February 2018

January 2018

December 2017

November 2017

CATEGORIES

- General
- Java Selenium
- Python
- Python Selenium
- Test Automation
- Testing
- TestNG
- Tools