**How To Use Selenium WebDriver In Python To Make Web Automation Testing Example**

[Richard Trump](https://www.code-learner.com/author/code-learner/) March 30, 2019

Selenium is mainly used for automated testing, it support multiple browsers, it can simulate browser for web page loading and for solving JavaScript rendering problems in crawlers. This example will show you how to use selenium webdriver in Python.

**1. Declare Web Browser Object.**

1. # import webdriver class from selenium package.
2. **from** selenium **import** webdriver
3. #Webdriver can be considered as the driver of the web browser, to semiulate web browser we must use webdriver, webdriver support a variety of browsers, here we use google Chrome as an example.
4. browser = webdriver.Chrome()

**2. Browse Web Page And Get Page Html Source Code.**

1. # import webdriver class.
2. **from** selenium **import** webdriver
3. # create google chrome browser webdriver.
4. browser = webdriver.Chrome()
5. web\_page\_url = 'https://www.google.com'
6. # use above webdriver to browse google home page.
7. browser.get(web\_page\_url)
8. # browser.page\_source will return all web page html source code.
9. page\_source\_html = browser.page\_source
10. # print page source html code.
11. **print**(page\_source\_html)
12. # close and quit google chrome.
13. browser.close()

**3. Find Single Web Element.**

WebDriver provide below methods to locate single web element.

1. **find\_element\_by\_name**
2. **find\_element\_by\_id**
3. **find\_element\_by\_xpath**
4. **find\_element\_by\_link\_text**
5. **find\_element\_by\_partial\_link\_text**
6. **find\_element\_by\_tag\_name**
7. **find\_element\_by\_class\_name**
8. **find\_element\_by\_css\_selector**
9. **find\_element(by\_condition, value)** : This method can implement all above method. For example,  **find\_element\_by\_name** can be implemented by **find\_element(BY.NAME, ‘tom’)** etc.
10. **from** selenium **import** webdriver
11. browser = webdriver.Chrome()
12. url = 'https://www.taobao.com'
13. browser.get(url)
14. # find web element by id.
15. input\_first = browser.find\_element\_by\_id('q')
16. # find web element by css selector.
17. input\_second = browser.find\_element\_by\_css\_selector('#q')
18. # find web element by xpath
19. input\_third = browser.find\_element\_by\_xpath('//\*[@id="q"]')
20. # print out above web element
21. **print**(input\_first,input\_second,input\_third)
22. # close and quite google chrome web browser
23. browser.close()

**4. Find Multiple Web Elements.**

1. input\_first = browser.find\_elements\_by\_id('q')
2. input\_first = browser.find\_elements\_by\_name('tom')

**5. Use Selenium WebDriver To Send Keyword & Click Button.**

Below code will input keyword in search box and implement search action in google chrome automatically.

1. # import webdriver and time class.
2. **from** selenium **import** webdriver
3. **import** time
4. # create google chrome web browser.
5. browser = webdriver.Chrome()
6. # browse a web page url.
7. browser.get('https://www.taobao.com')
8. # find the search input text box web element.
9. input = browser.find\_element\_by\_id('q')
10. # send keyword into above search text box.
11. input.send\_keys('iPhone')
12. # sleep 5 seconds.
13. time.sleep(5)
14. # clear the search input text box.
15. input.clear()
16. # input another search keyword in search box.
17. input.send\_keys('iPad')
18. # find the search submit button.
19. button = browser.find\_element\_by\_class\_name('btn-search')
20. # click above button to search.
21. button.click()

**6. Use Selenium WebDriver To Implement Drag & Drop Action.**

1. # import webdriver class.
2. **from** selenium **import** webdriver
3. # import ActionChains class.
4. **from** selenium.webdriver **import** ActionChains
5. # create google chrome web browser.
6. browser = webdriver.Chrome()
7. # browse below url in google chrome.
8. url = 'http://www.runoob.com/try/try.php?filename=jqueryui-api-droppable'
9. browser.get(url)
10. # switch to iframe.
11. browser.switch\_to.frame('iframeResult')
12. # find the drag & drop source object.
13. source = browser.find\_element\_by\_css\_selector('#draggable')
14. # find the drag & drop target object.
15. target = browser.find\_element\_by\_css\_selector('#droppable')
16. # create an ActionChains object.
17. actions = ActionChains(browser)
18. # connect drag & drop action source and target object.
19. actions.drag\_and\_drop(source, target)
20. # implement drag & drop action.
21. actions.perform()#执行动作

**7. Run JavaScript In Selenium WebDriver.**

Below code will invoke javascript in selenium webdriver to scroll the scroll bar. This kind of action do not supported by selenium webdriver by default, so we need to use javascript to implement it.

1. **import** webdriver **class**
2. **from** selenium **import** webdriver
3. # create a google chrome web browser.
4. browser = webdriver.Chrome()
5. # browse below url.
6. browser.get('https://www.zhihu.com/explore')
7. # execute javascript which will scroll the page to the bottom.
8. browser.execute\_script('window.scrollTo(0, document.body.scrollHeight)')
9. # execute javascript to popup an alert dialog.
10. browser.execute\_script('alert("To Bottom")')

**8. Get Web Element’s Attribute.**

Call web\_element.get\_attribute(attr\_name) method.

1. **from** selenium **import** webdriver
2. browser = webdriver.Chrome()
3. url = 'https://www.zhihu.com/explore'
4. browser.get(url)
5. # get website logo.
6. logo = browser.find\_element\_by\_id('zh-top-link-logo')
7. # get logo class attribute value.
8. logo\_attr\_class = logo.get\_attribute('class')
9. # print above class attribute value.
10. **print**(logo\_attr\_class)
11. browser.close()

**9. Get Web Element’s Text, Id, Location, Tag Name and Size Value.**

Call web\_element’s text, id, location, tag\_name and size property.

1. **from** selenium **import** webdriver
2. browser = webdriver.Chrome()
3. url = 'https://www.zhihu.com/explore'
4. browser.get(url)
5. # find input text box
6. input = browser.find\_element\_by\_class\_name('zu-top-add-question')
7. # get input text box text.
8. input\_text = input.text
9. # print out the text.
10. **print**(input\_text)
11. # print input text box id, location, tag\_name and size
12. **print**(input.id)
13. **print**(input.location)
14. **print**(input.tag\_name)
15. **print**(input.size)
16. browser.close()

**10. Switch Frame In Selenium.**

1. **from** selenium **import** webdriver
2. ......
3. browser = webdriver.Chrome()
4. ......
5. browser.switch\_to.frame('frame\_name')
6. ......

**11. Implicit Wait.**

When an implicit wait is used to execute the selenium test, if WebDriver does not find the element in the DOM, it will continue to wait and throw **cannot find the element exception** when the specified time timeout.

1. **from** selenium **import** webdriver
2. browser = webdriver.Chrome()
3. # wait 10 seconds for loading web page, if not loaded in 10 seconds, it will throw an exception.
4. browser.implicitly\_wait(10)
5. browser.get('https://www.yahoo.com')

**12. Explicit Wait.**

Specify a wait condition, and a maximum wait time, the program will determine whether the condition is met in the wait time, if it met, it will return, if it do not met, it will continue to wait, and an exception will be thrown after the wait time timeout.

1. # import webdriver class.
2. **from** selenium **import** webdriver
3. # import By class
4. **from** selenium.webdriver.common.by **import** By
5. # explicit wait need import WebDriverWait class.
6. **from** selenium.webdriver.support.ui **import** WebDriverWait
7. # the expected\_conditions class is used to specify the wait condition.
8. **from** selenium.webdriver.support **import** expected\_conditions **as** EC
9. # create google chrome web browser and browse a web page.
10. browser = webdriver.Chrome()
11. browser.get('https://www.taobao.com/')
12. # create the WebDriverWait class, the timeout value is 10 seconds.
13. wait = WebDriverWait(browser, 10)
14. # wait until to find the input text box which id value is 'q'.
15. input = wait.until(EC.presence\_of\_element\_located((By.ID, 'q')))
16. # wait until the button is clickable.
17. button = wait.until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, '.btn-search')))
18. **print**(input, button)

**13. Implement Web Browser Forward & Back.**

1. **import** time
2. **from** selenium **import** webdriver
3. # create the web browser.
4. browser = webdriver.Chrome()
5. # browse three web page url one by one.
6. browser.get(url\_1)
7. browser.get(url\_2)
8. browser.get(url\_3)
9. # now go back to url\_2.
10. browser.back()
11. time.sleep(1)
12. # now forward to url\_3
13. browser.forward()
14. browser.close()

**14. Manage Web Browser Cookie.**

1. **from** selenium **import** webdriver
2. browser = webdriver.Chrome()
3. browser.get('https://www.zhihu.com/explore')
4. # get current web browser cookies and print.
5. **print**(browser.get\_cookies())
6. # add a new cookie to web browser.
7. browser.add\_cookie({'name': 'name', 'domain': 'www.zhihu.com', 'value': 'germey'})
8. **print**(browser.get\_cookies())
9. # delete all web browser cookies
10. browser.delete\_all\_cookies()
11. **print**(browser.get\_cookies())

**15. Open New & Switch Between Browser Windows.**

1. **import** time
2. **from** selenium **import** webdriver
3. browser = webdriver.Chrome()
4. # browse url\_1 in current window.
5. browser.get(url\_1)
6. # open a new window by execute javascript.
7. browser.execute\_script('window.open()')
8. # print all window handlers.
9. **print**(browser.window\_handles)
10. # switch to the newly opened window.
11. browser.switch\_to\_window(browser.window\_handles[1])
12. # browse url\_2 in new opened window.
13. browser.get(url\_2)
14. time.sleep(1)
15. # switch back to the original browser window
16. browser.switch\_to\_window(browser.window\_handles[0])
17. # browse url\_3 in the original web browser window.
18. browser.get(url\_3)

**16. Exception Management.**

1. **from** selenium **import** webdriver
2. # import TimeoutException, NoSuchElementException class.
3. **from** selenium.common.exceptions **import** TimeoutException, NoSuchElementException
4. browser = webdriver.Chrome()
5. # below source code will demo TimeoutException.
6. **try**:
7. browser.get('https://www.google.com')
8. **except** TimeoutException:
9. **print**('Time Out')
10. # below source code will demo NoSuchElementException.
11. **try**:
12. browser.find\_element\_by\_id('hello')
13. **except** NoSuchElementException:
14. **print**('No Element')
15. **finally**:
16. browser.close()