StaleElementReferenceException on Python Selenium

Asked 5 years, 3 months ago Active 22 days ago Viewed 60k times

40

18

I am getting the following error while using Selenium in python:

```
selenium.common.exceptions.StaleElementReferenceException: Message: u'stale element
reference: element is not attached to the page document\n
```

Interestingly enough, the error pops up at different times in the for loop. Sometimes it gets through eg. 4 iterations and other times eg. 7.

Some of the relevant code being run is:

```
for i in range(0, 22):
    u = driver.find_elements_by_id("data")
    text = u[0].get_attribute("innerHTML")
    driver.find_elements_by_class_name("aclassname")[0].click()
```

What does this error mean and what is something I can try to fix this?

python html selenium-webdriver dom

edited Jun 6 '19 at 2:02
ivanleoncz
3,418 ● 3 ● 35 ● 39

asked Nov 18 '14 at 20:28
bill999
2,081 ● 4 ● 36 ● 70

The error means "i cant find the element" more or less. The way to fix this is to slow it down somehow. maybe through implicit or explicit waits. – TehTris Nov 18 '14 at 20:39

OK. What is an implicit wait? – bill999 Nov 18 '14 at 20:40

Or, better yet, how would I do an explicit wait until the `aclassname` element is loaded? – bill999 Nov 18 '14 at 20:49

1 This is already answered here: stackoverflow.com/q/24775988/3124333 – SiKing Nov 18 '14 at 21:36

@SiKing no answers solves the problem. As we get StaleElementReferenceException after 2-3 iterations. Can you give me exact solution – Vishav Gupta Feb 12 at 10:55

7 Answers

22

It means the element is no longer in the DOM, or it changed.

You are inside a for loop. Think what happens during the iterations:

1. something changes when you click on the element (last line)
2. So the page is changing
3. You enter the next iteration. Now are trying to find a new element (your first line inside the loop).
4. You found the element
5. It finishes changing
6. You try to use it by getting an attribute
7. Bam! The element is old. You got it in step 4, but it finished changing on step 5

The following code may help you find the element:

```
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import StaleElementReferenceException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions

my_element_id = 'something123'
ignored_exceptions=(NoSuchElementException,StaleElementReferenceException,)
your_element = WebDriverWait(your_driver,
some_timeout,ignored_exceptions=ignored_exceptions)\
                        .until(expected_conditions.presence_of_element_located((By.ID,
my_element_id)))
```

edited Feb 12 at 14:23

answered Jul 4 '17 at 22:34

Emilio M.
839 ● 7 ● 18

1 Good discussion in general. Steps 4-6 might be incredibly subtle, too... A page I was scraping had a dropdown on it that, at some point in load, was being replaced by an exact copy of itself, so there was a race condition in selecting an `<option>`, clicking it, and submitting the form that resulted in one of several possible errors. – kevlarr Jan 24 '18 at 21:59

I have struggled for days with a stale element reference exception until I stumbled across your answer. After implementing ignored_exceptions, my code works and obtains all the requested data every time. Thanks! – Life is complex Jun 29 '19 at 20:14

@Emilio M. this answers doesn't work me. It gives Timeout exception. Please help if there is another solution – Vishav Gupta Feb 12 at 11:16

@EmilioM. I have tried other selectors also but problem remains same. – Vishav Gupta Feb 13 at 5:59

19

Selenium Support `Explicit` and `Implicit` Waits. If you think waiting for certain amount of time is enough for your page to be loaded, use:

```
driver.implicitly_wait(secs)
```

but if you want to wait for a special event (e.g. waiting for a particular element to be loaded) you can do something like:

```
from selenium.webdriver.support.ui import WebDriverWait
...
...
def find(driver):
    element = driver.find_elements_by_id("data")
    if element:
        return element
    else:
        return False
element = WebDriverWait(driver, secs).until(find)
```

answered Apr 19 '15 at 0:31

Nima Soroush
8,111 ● 3 ● 42 ● 50

4

Beyond the answers here, if you are using ActionChains, and the page has changed, be sure to reinstantiate your ActionChains object (dont reuse an old one), otherwise your ActionChain will be using a stale DOM. I.e. do this;

```
action_chain = ActionChains(driver)
action_chain.double_click(driver.find_element_by_xpath("//tr[2]/p")).perform()
```

Or better yet dont use an instantiation;

```
ActionChains(driver).double_click(driver.find_element_by_xpath("//tr[2]/p")).perform()
```

answered Aug 30 '19 at 14:41

n00b
2,600 ● 3 ● 22 ● 49

4

```
>>>Stale Exceptions can be handled using **StaleElementReferenceException** to continue
to execute the for loop.

from selenium.common import exceptions

# and customize your code of for loop as:

for i in range(0, 22):
   try:
        u = driver.find_elements_by_id("data")
        text = u[0].get_attribute("innerHTML")
        driver.find_elements_by_class_name("aclassname")[0].click()
   except exceptions.StaleElementReferenceException,e:
        print(e)
        pass
```

Note: Python 3+ : replace **exceptions.StaleElementReferenceException,e -> exceptions.StaleElementReferenceException as e**

edited Feb 7 at 13:20

answered Oct 26 '18 at 4:29

Vishal

476 ● 5 ● 9

3

When webpage got refreshed or switched back from different window or form and trying to access an element user will get staleelementexception.

Use webdriverwait in try --except block with for loop: EX :

Code in which I got staleelementexception :

```
driver.find_element_by_id(tc.value).click()
```

driver.find_element_by_xpath("xpath").click()

Fix :

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

driver.find_element_by_id(tc.value).click()
for i in range(4):
   try:
        run_test = WebDriverWait(driver, 120).until( \
        EC.presence_of_element_located((By.XPATH, "xpath")))
        run_test.click()
        break
   except StaleElementReferenceException as e:
        raise e
```

 1 ● 1  39 ● 1

1

I Would like to add one more solution here which is worked for me.

I was trying to access the button in the top menu panel on my webpage after refreshing the content area on the same page, Which gave me the following error,

```
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.StaleElementReferenceException: Message: The element
reference of <span id="dk1-combobox" class="dk-selected combinationText visibleElements
"> is stale; either the element is no longer attached to the DOM, it is not in the
current frame context, or the document has been refreshed
```

Then I started to search for the solution to click the stale element on the web page. After two days of thinking and googling, I got a solution.

To access the stale element on the page, first, we need to focus the mouse over the particular section and perform click option

```
EditReport1 = driver.find_element_by_id('internalTab1')
ActionChains(driver).move_to_element(EditReport1).click(EditReport1).perform()
```

move_to_element will move the mouse over the stale element which we need to access once we got our control on the element the click operation is successfully performed.

This is worked for me. If anyone finds it working please comment your's as well which will help some other in future.

Thank you

answered Jul 21 '19 at 7:36

Jeya Suriya Muthumari

1,356 ● 1 ● 14 ● 36

I tried it but this work for only for maximum three iterations. Please give some other solution as StaleElementReferenceException is very annoying – Vishav Gupta Feb 12 at 10:52

0

This is the python solution for this problem:

```
def clickAndCatchStaleRefException(locator):



    driver = sel2._current_browser()
    result = False
    attempts = 0
```

```
find_element_by_xpath
    # function will throw an error.
    # But if you pass an xpath directly you don't need this


    while attempts < 2:
        try:
            driver.find_element_by_xpath(locator).click()
            result = True
            break
        except EC as e:
            raise e
        finally:
            attempts += 1
    return result
```

edited Nov 17 '18 at 22:22

Anna
1,378 ● 1 ● 15 ● 26

answered Oct 3 '17 at 3:44

Rashid
19 ● 3

5 This is not a good solution. WebDriver has selenium.webdriver.support.wait.WebDriverWait and selenium.webdriver.support.expected_conditions to solve this problem properly, as specified in the other answer. Retrying to get an element may work, but it's not the correct way of doing it. – Emilio M. Oct 10 '17 at 20:00

I agree, but it doesn't work all the time. For example if you work with an application using jquery, some elements could be replaced unexpectedly and StaleElementReferenceException will be raised even after WebDriverWait is complete. In these particular cases, in never heard about a better solution than trying until you get the expected result. (Note : i am not using selenium to perform automated tests, but to scrap odds corporate's apps) – Sylvan LE DEUNFF Dec 12 '18 at 15:26

Sylvan, If you check my answer above, it handles the specific case you are mentioning. That way is the best way to do it. You still wait, but use the standard tools Selenium provides instead of custom code. It's not only cleaner, but also safer, since it's a proven solution that works for everyone. – Emilio M. Mar 24 '19 at 21:24