# Example

from splinter import Browser

with Browser() as browser:

# Visit URL

url = "http://www.google.com"

browser.visit(url)

browser.fill('q', 'splinter - python acceptance testing for web applications')

# Find and click the 'search' button

button = browser.find\_by\_name('btnG')

# Interact with elements

button.click()

if browser.is\_text\_present('splinter.readthedocs.io'):

print("Yes, the official website was found!")

else:

print("No, it wasn't found... We need to improve our SEO techniques")

# browser type

browser = Browser('chrome')

browser = Browser('firefox')

browser = Browser('zope.testbrowser')

# Managing Windows

browser.windows # all open windows

browser.windows[0] # the first window

browser.windows["window\_name"] # the window\_name window

browser.windows.current # the current window

browser.windows.current = browser.windows[3] # set current window to window 3

# splinter api不提供但是可以通过其他来搞定的，比如通过driver来设置window的大小。

browser.driver.set\_window\_size(1600, 1000)

window = browser.windows[0]

window.is\_current # boolean - whether window is current active window

window.is\_current = True # set this window to be current window

window.next # the next window

window.prev # the previous window

window.close() # close this window

window.close\_others() # close all windows except this one

# Reload/back/forward a page

browser.reload()

browser.back()

browser.forward()

# get page tile /page content /url

browser.title

browser.html

browser.url

# change Browser User-Agent

b = Browser(user\_agent="Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en)")

# Finding elements，returns a list with the found elements

browser.find\_by\_css('h1')

browser.find\_by\_xpath('//h1')

browser.find\_by\_tag('h1')

browser.find\_by\_name('name')

browser.find\_by\_text('Hello World!')

browser.find\_by\_id('firstheader')

browser.find\_by\_value('query')

# get element

first\_found = browser.find\_by\_name('name').first

last\_found = browser.find\_by\_name('name').last

second\_found = browser.find\_by\_name('name')[1]

# Get value of an element

browser.find\_by\_css('h1').first.value

# Clicking links,return the first link

browser.click\_link\_by\_href('http://www.the\_site.com/my\_link')

browser.click\_link\_by\_partial\_href('my\_link')

browser.click\_link\_by\_text('my link')

browser.click\_link\_by\_partial\_text('part of link text')

browser.click\_link\_by\_id('link\_id')

# element is visible or invisible

browser.find\_by\_css('h1').first.visible

# Verifying if element has a className

browser.find\_by\_css('.content').first.has\_class('content')

# click button

browser.find\_by\_name('send').first.click()

browser.find\_link\_by\_text('my link').first.click()

# Mouse

browser.find\_by\_tag('h1').mouse\_over()

browser.find\_by\_tag('h1').mouse\_out()

browser.find\_by\_tag('h1').click()

browser.find\_by\_tag('h1').double\_click()

browser.find\_by\_tag('h1').right\_click()

# Mouse drag and drop

draggable = browser.find\_by\_tag('h1')

target = browser.find\_by\_css('.container')

draggable.drag\_and\_drop(target)

# Interacting with forms

browser.fill('query', 'my name')

browser.attach\_file('file', '/path/to/file/somefile.jpg')

browser.choose('some-radio', 'radio-value')

browser.check('some-check')

browser.uncheck('some-check')

browser.select('uf', 'rj')

# screenshot

browser.driver.save\_screenshot('your\_screenshot.png')

# 看不太懂

# trigger JavaScript events, like KeyDown or KeyUp, you can use the type method.

browser.type('type', 'typing text')

'''

If you pass the argument slowly=True to the type method you can interact with the page on every key pressed. Useful for

'''

# testing field's auto completion (the browser will wait until next iteration to type the subsequent key).

for key in browser.type('type', 'typing slowly', slowly=True):

pass # make some assertion here with the key object :)

# You can also use type and fill methods in an element:

browser.find\_by\_name('name').type('Steve Jobs', slowly=True)

browser.find\_by\_css('.city').fill('San Francisco')

# Dealing with HTTP status code and exceptions

browser.visit('http://cobrateam.info')

browser.status\_code.is\_success() # True

browser.status\_code == 200 # True

browser.status\_code.code # 200

# try:

# browser.visit('http://cobrateam.info/i-want-cookies')

# except HttpResponseError, e:

# print "Oops, I failed with the status code %s and reason %s" % (e.status\_code, e.reason)

# test

# Cookies manipulation

browser.cookies.add({'whatever': 'and ever'}) # add a cookie

browser.cookies.all() # retrieve all cookies

browser.cookies.delete('mwahahahaha') # deletes the cookie 'mwahahahaha'

browser.cookies.delete('whatever', 'wherever') # deletes two cookies

browser.cookies.delete() # deletes all cookies

# Frames, alerts and prompts

# Using iframes，You can use the get\_iframe method and the with statement to interact with iframes. You can pass the

# iframe's name, id, or index to get\_ifram

with browser.get\_iframe('iframemodal') as iframe:

iframe.do\_stuff()

# Chrome support for alerts and prompts is new in Splinter 0.4.Only webdrivers (Firefox and Chrome) has support for

# alerts and prompts.

alert = browser.get\_alert()

alert.text

alert.accept()

alert.dismiss()

prompt = browser.get\_alert()

prompt.text

prompt.fill\_with('text')

prompt.accept()

prompt.dismiss()

# use the with statement to interacte with both alerts and prompts

with browser.get\_alert() as alert:

alert.do\_stuff()

# Executing javascript

browser.execute\_script("$('body').empty()")

browser.evaluate\_script("4+4") == 8

# Matchers

browser = Browser()

browser.visit('https://splinter.readthedocs.io/')

browser.is\_text\_present('splinter') # True

browser.is\_text\_present('splinter', wait\_time=10) # True, using wait\_time

browser.is\_not\_present('text not present') # True

browser.is\_element\_present\_by\_css('h1')

browser.is\_element\_present\_by\_xpath('//h1')

browser.is\_element\_present\_by\_tag('h1')

browser.is\_element\_present\_by\_name('name')

browser.is\_element\_present\_by\_text('Hello World!')

browser.is\_element\_not\_present\_by\_id('firstheader')

browser.is\_element\_not\_present\_by\_value('query')

browser.is\_element\_present\_by\_value('query', wait\_time=10)