
splinter Documentation

Release 0.7.5

andrews medina

November 02, 2016

| | | |
|----------|--|-----------|
| 1 | Sample code | 3 |
| 2 | Features | 5 |
| 3 | Getting started | 7 |
| 4 | Basic browsing and interactions | 9 |
| 5 | JavaScript support | 11 |
| 6 | Walking on... | 13 |
| 7 | Drivers | 15 |
| 7.1 | Browser based drivers | 15 |
| 7.2 | Headless drivers | 15 |
| 7.3 | Remote driver | 15 |
| 8 | Get in touch and contribute | 17 |

Splinter is an open source tool for testing web applications using Python. It lets you automate browser actions, such as visiting URLs and interacting with their items.

Sample code

```
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")
```

Note: if you don't provide any driver to Browser function, firefox will be used.

Features

- simple api
- multi webdrivers (chrome webdriver, firefox webdriver, phantomjs webdriver, zopetestbrowser, remote webdriver)
- css and xpath selectors
- support to iframe and alert
- execute javascript
- works with ajax and async javascript

what's new in splinter?

Getting started

- Why use Splinter
- Installation
- Quick tutorial

Basic browsing and interactions

- Browser and navigation
- Finding elements
- Mouse interactions
- Interacting with elements and forms
- Verify the presence of texts and elements in a page, with matchers
- Cookies manipulation

JavaScript support

- Executing JavaScript

Walking on...

- Dealing with HTTP status code and exceptions
- Interacting with iframes, alerts and prompts
- Full API documentation

7.1 Browser based drivers

The following drivers open a browser to run your actions:

- `Chrome WebDriver`
- `Firefox WebDriver`
- `Remote WebDriver`

7.2 Headless drivers

The following drivers don't open a browser to run your actions (but has its own dependencies, check the specific docs for each driver):

- `Phantomjs WebDriver`
- `zope.testbrowser`
- `django client`
- `flask client`

7.3 Remote driver

The remote driver uses Selenium Remote to control a web browser on a remote machine.

- `Remote WebDriver`

Get in touch and contribute

- Community
- Contribute
- Writing new drivers
- Setting up your splinter development environment