FoxPuppet Documentation

Release latest

Mozilla

Apr 02, 2019

Contents

1 Installation **3**
- 1.1 Requirements . . . 3
- 1.2 Install FoxPuppet . . . 3

2 User Guide **5**
- 2.1 Example . . . 5

3 Development **7**
- 3.1 Automated Testing . . . 7
- 3.2 Running Tests . . . 7
- 3.3 Keeping up with Firefox . . . 7

4 Release Notes **9**

5 Developer Interface **11**
- 5.1 FoxPuppet . . . 11
- 5.2 Windows . . . 11

Python Module Index **13**

FoxPuppet is a library for automating user interactions in Firefox using Selenium.

CHAPTER 1

Installation

1.1 Requirements

FoxPuppet will work with python 2.7, and 3.6

1.2 Install FoxPuppet

To install FoxPuppet using pip:

```
$ pip install FoxPuppet
```

To install from source:

```
$ python setup.py develop
```

CHAPTER 2

User Guide

FoxPuppet is a library for automating user interactions in Firefox using Selenium. This section gives some example use-cases of FoxPuppet.

2.1 Example

This is an example for setting up FoxPuppet and opening a private window:

```
from foxpuppet import FoxPuppet
from selenium.webdriver import Firefox


selenium = Firefox()
foxpuppet = FoxPuppet(selenium)

window = foxpuppet.browser.open_window(private=True)
selenium.quit()
```

This example opens first a non-private browser window, then proceeds to open a private browser window. Finally it quits via `selenium.quit()`.

> **Note** The initial browser window is automatically assigned to the `browser` attribute and is always available.

CHAPTER 3

Development

3.1 Automated Testing

All pull requests and merges are tested in Travis CI based on the `.travis.yml` file.

Usually, a link to your specific travis build appears in pull requests, but if not, you can find it on the pull requests page

The only way to trigger Travis CI to run again for a pull request, is to submit another change to the pull branch.

Test coverage is done by Coveralls.

3.2 Running Tests

You will need Tox installed to run the tests against the supported Python versions.

You also need to have Geckodriver installed so the tests can run Firefox.

```
$ pip install tox
$ tox
```

3.3 Keeping up with Firefox

As Firefox evolves, it will be necessary for FoxPuppet to keep up. This will likely present itself as a test failure introduces in Firefox nightly. FoxPuppet's tests are configures to run on Travis CI every day, which alerts us of regressions or more likely changes in Firefox that we need to adapt to.

When such failures occur, the following guide may help to determine the change that introduced the failure, and the fix that may be needed.

3.3.1 Replicate the failure

To run the tests against a specific Firefox version (you'll typically want Nightly) make sure the `PATH` environment variable is front-loaded with the path to the target binary. Note that you want the path that *contains* the binary, and not the path of the binary itself. Although this part should run fine in Python 3, it's worth targeting 2.7 using `-e py27` as the following steps currently require it. It's also a good idea to narrow the tests being run using `-k` as shown in this example:

```
$ export MOZ_HEADLESS=1
$ export PATH=/Applications/FirefoxNightly.app/Contents/MacOS:$PATH
$ tox -e py27 -- -k test_tracking_protection_shield
```

3.3.2 Determine the regression range

Run `mozregression` to determine the change that introduced the failure. Create a temporary virtual environment using Python 2.7 and install both `tox` and `mozregression`. You can then use `mozregression` providing the last time the test was known to pass, and the first time it was known to fail. This can be determined by the dates in the Travis CI results. Modify the following to suit your needs:

```
$ mozregression -b 2018-08-12 -g 2018-08-11 -c "tox -e py27 -- -k test_tracking_
↪protection_shield"
```

This will identify the builds within the range, and systematically download and execute the tests against the builds to identify the change that caused the failure. You don't need to worry about `PATH` as the FoxPuppet tests use the `MOZREGRESSION_BINARY` environment variable if it's set.

Assuming everything goes as expected, mozregression will report the last good revision, first bad revision, and provide a pushlog URL showing the change(s) that caused the regression. From this, it should be possible to determine if the failure is expected due to changes in Firefox, or if a bug has been introduced.

3.3.3 Supporting latest Firefox versions

If behaviour changes in Firefox and FoxPuppet needs to be updated, it is sometimes necessary to change the behaviour in FoxPuppet based on the Firefox version.

CHAPTER 4

Release Notes

CHAPTER 5

Developer Interface

This part of the documentation describes the interfaces for using FoxPuppet.

5.1 FoxPuppet

class foxpuppet.foxpuppet.**FoxPuppet**(*selenium*)
Set up the interface for interacting with the Firefox browser.

Parameters **selenium** – (WebDriver): Firefox WebDriver object.

5.2 Windows

This module contains all of the window types as well as the window manager.

5.2.1 Window Manager

class foxpuppet.windows.**WindowManager**(*selenium*)
A window manager that controls the creation of window objects.

Parameters **selenium** – (WebDriver): Firefox WebDriver object.

5.2.2 Browser Window

class foxpuppet.windows.browser.window.**BrowserWindow**(*selenium*, *handle*)
Representation of a browser window.

close()
Close the window.

document_element
Return the inner DOM window element.

Returns WebDriver element object for the DOM window element.

Return type WebElement

firefox_version
Major version of Firefox in use.

Returns Major component of the Firefox version.

Return type int

is_private
Property that checks if the specified window is private or not.

Returns True if this is a Private Browsing window.

Return type bool

navbar
Provide access to the Navigation Bar.

Returns FoxPuppet NavBar object.

Return type NavBar

notification
Provide access to the currently displayed notification.

Returns FoxPuppet BaseNotification object.

Return type BaseNotification

open_window(*private=False*)
Open a new browser window.

Parameters **private** (*bool*) – Optional parameter to open a private browsing window. Defaults to False.

Returns Opened window.

Return type *BrowserWindow*

switch_to()
Switch focus for Selenium commands to this window.

wait_for_notification(*notification_class=<class 'foxpuppet.windows.browser.notifications.base.BaseNotification'>*)
Wait for the specified notification to be displayed.

Parameters **notification_class** (BaseNotification, optional) – The notification class to wait for. If *None* is specified it will wait for any notification to be closed. Defaults to *BaseNotification*.

Returns Firefox notification.

Return type BaseNotification

Python Module Index

f

foxpuppet.foxpuppet, 11
foxpuppet.windows, 11
foxpuppet.windows.browser.window, 11

Index

B

BrowserWindow (*class in foxpuppet.windows.browser.window*), 11

C

close() (*foxpuppet.windows.browser.window.BrowserWindow method*), 11

D

document_element (*foxpuppet.windows.browser.window.BrowserWindow attribute*), 11

F

firefox_version (*foxpuppet.windows.browser.window.BrowserWindow attribute*), 12
FoxPuppet (*class in foxpuppet.foxpuppet*), 11
foxpuppet.foxpuppet (*module*), 11
foxpuppet.windows (*module*), 11
foxpuppet.windows.browser.window (*module*), 11

I

is_private (*foxpuppet.windows.browser.window.BrowserWindow attribute*), 12

N

navbar (*foxpuppet.windows.browser.window.BrowserWindow attribute*), 12
notification (*foxpuppet.windows.browser.window.BrowserWindow attribute*), 12

O

open_window() (*foxpuppet.windows.browser.window.BrowserWindow method*), 12

S

switch_to() (*foxpuppet.windows.browser.window.BrowserWindow method*), 12

W

wait_for_notification() (*foxpuppet.windows.browser.window.BrowserWindow method*), 12
WindowManager (*class in foxpuppet.windows*), 11