pytest-splinter Documentation

Release 1.8.5

Anatoly Bubenkov, Paylogic International and others

Oct 02, 2017

Contents

Contents

CHAPTER 1

Splinter plugin for the pytest runner

Install pytest-splinter

```
pip install pytest-splinter
```

Features

The plugin provides a set of fixtures to use splinter for browser testing with pytest

Fixtures

- **browser** Get the splinter's Browser. Fixture is underneath session scoped, so browser process is started once per test session, but the state of the browser will be clean (current page is `blank`, cookies clean).
- **session_browser** The same as `browser` except the lifetime. This fixture is session-scoped so will only be finalized at the end of the whole test session. Useful if you want to speedup your test suite paying with reduced test isolation.
- **browser_instance_getter** Function to create an instance of the browser. This fixture is required only if you need to have multiple instances of the Browser in a single test at the same time. Example of usage:

```
@pytest.fixture
def admin_browser(request, browser_instance_getter):
    """Admin browser fixture."""
    # browser_instance_getter function receives parent fixture -- our admin_browser
    return browser_instance_getter(request, admin_browser)

def test_2_browsers(browser, admin_browser):
    """Test using 2 browsers at the same time."""
```

```
    browser.visit('http://google.com')
    admin_browser.visit('http://admin.example.com')
```

- **splinter_selenium_implicit_wait** Implicit wait timeout to be passed to Selenium webdriver. Fixture gets the value from the command-line option splinter-implicit-wait (see below)
- **splinter_wait_time** Explicit wait timeout (for waiting for expicit condition via *wait_for_condition*). Fixture gets the value from the command-line option splinter-wait-time (see below)
- **splinter_selenium_speed** Speed for Selenium, if not 0 then it will sleep between each selenium command. Useful for debugging/demonstration. Fixture gets the value from the command-line option splinter-speed (see below)
- **splinter_selenium_socket_timeout** Socket timeout for communication between the webdriver and the browser. Fixture gets the value from the command-line option splinter-socket-timeout (see below)
- **splinter_webdriver** Splinter's webdriver name to use. Fixture gets the value from the command-line option splinter-webdriver (see below). To make pytest-splinter always use certain webdriver, override a fixture in your *conftest.py* file:

```
import pytest

@pytest.fixture(scope='session')
def splinter_webdriver():
    """Override splinter webdriver name."""
    return 'phantomjs'
```

- **splinter_remote_url** Splinter's webdriver remote url to use (optional). Fixture gets the value from the command-line option splinter-remote-url (see below). Will be used only if selected webdriver name is 'remote'.
- **splinter_session_scoped_browser** pytest-splinter should use single browser instance per test session. Fixture gets the value from the command-line option splinter-session-scoped-browser (see below)
- **splinter_file_download_dir** Directory, to which browser will automatically download the files it will experience during browsing. For example when you click on some download link. By default it's a temporary directory. Automatic downloading of files is only supported for firefox driver at the moment.
- **splinter_download_file_types** Comma-separated list of content types to automatically download. By default it's the all known system mime types (via mimetypes standard library).
- **splinter_browser_load_condition** Browser load condition, python function which should return True. If function returns False, it will be run several times, until timeout below reached.
- **splinter_browser_load_timeout** Browser load condition timeout in seconds, after this timeout the exception WaitUntilTimeout will be raised.
- **splinter_wait_time** Browser explicit wait timeout in seconds, after this timeout the exception WaitUntilTimeout will be raised.
- **splinter_firefox_profile_preferences** Firefox profile preferences, a dictionary which is passed to selenium webdriver's profile_preferences
- **splinter_firefox_profile_directory** Firefox profile directory to use as template for firefox profile created by selenium. By default, it's an empty directly inside pytest_splinter/profiles/firefox
- **splinter_driver_kwargs** Webdriver keyword arguments, a dictionary which is passed to selenium webdriver's constructor (after applying firefox preferences)
- **splinter_window_size** Size of the browser window on browser initialization. Tuple in form (<width>, <height>). Default is (1366, 768)

- **splinter_screenshot_dir** pytest-splinter browser screenshot directory. This fixture gets the value from the command-line option *splinter-screenshot-dir* (see below).
- **splinter_make_screenshot_on_failure** Should pytest-splinter take browser screenshots on test failure? This fixture gets the value from the command-line option *splinter-make-screenshot-on-failure* (see below).
- **splinter_screenshot_encoding** Encoding of the *html screenshot* on test failure. UTF-8 by default.
- **splinter_screenshot_getter_html** Function to get browser html screenshot. By default, it saves *browser.html* with given path and *splinter_screenshot_encoding* encoding.
- **splinter_screenshot_getter_png** Function to get browser image (png) screenshot. By default, it calls *browser.save_sceenshot* with given path.
- **splinter_driver_executable** Filesystem path of the webdriver executable. This fixture gets the value from the command-line option *splinter-webdriver-executable* (see below).
- **splinter_browser_class** Class to use for browser instance. Defaults to *pytest_splinter.plugin.Browser*.
- **splinter_clean_cookies_urls** List of additional urls to clean cookies on. By default, during the preparation of the browser for the test, pytest-splinter only cleans cookies for the last visited url from previous test, as it's not possible to clean all cookies from all domains at once via webdriver protocol, by design. This limitation can be worked around if you know the list of urls, the domains for which you need to clean cookies (for example https://facebook.com). If so, you can override this fixture and put those urls there, and pytest-splinter will visit each of them and will clean the cookies for each domain.

Command-line options

- ***–splinter-implicit-wait*** Selenium webdriver implicit wait. Seconds (default: 5).
- ***–splinter-speed*** selenium webdriver speed (from command to command). Seconds (default: 0).
- ***–splinter-socket-timeout*** Selenium webdriver socket timeout for for communication between the webdriver and the browser. Seconds (default: 120).
- ***–splinter-webdriver*** Webdriver name to use. (default: firefox). Options:
  - firefox
  - remote
  - chrome
  - phantomjs

  For more details refer to the documentation for splinter and selenium.
- ***–splinter-remote-url*** Webdriver remote url to use. (default: None). Will be used only if selected webdriver name is 'remote'.

  For more details refer to the documentation for splinter and selenium.
- ***–splinter-session-scoped-browser*** pytest-splinter should use a single browser instance per test session. Choices are 'true' or 'false' (default: 'true').
- ***–splinter-make-screenshot-on-failure*** pytest-splinter should take browser screenshots on test failure. Choices are 'true' or 'false' (default: 'true').
- ***–splinter-screenshot-dir*** pytest-splinter browser screenshot directory. Defaults to the current directory.
- ***–splinter-webdriver-executable*** Filesystem path of the webdriver executable. Used by phantomjs and chrome drivers. Defaults to the None in which case the shell PATH variable setting determines the location of the executable.

Browser fixture

As mentioned above, browser is a fixture made by creating splinter's Browser object, but with some overrides.

- **visit** Added possibility to wait for condition on each browser visit by having a fixture.
- **wait_for_condition** Method copying selenium's wait_for_condition, with difference that condition is in python, so there you can do whatever you want, and not only execute javascript via browser.evaluate_script.

Several browsers for your test

You can have several browsers in one test.

```
import pytest

@pytest.fixture
def admin_browser(browser_instance_getter):
    return browser_instance_getter(admin_browser)

def test_with_several_browsers(browser, admin_browser):
    browser.visit('http://example.com')
    admin_browser.visit('about:blank')
    assert browser.url == 'http://example.com'
```

Automatic screenshots on test failure

When your functional test fails, it's important to know the reason. This becomes hard when tests are being run on the continuos integration server, where you cannot debug (using –pdb). To simplify things, a special behaviour of the browser fixture is available, which takes a screenshot on test failure and puts it in a folder with the a naming convention compatible to the jenkins plugin. The html content of the browser page is also stored, this can be useful for debugging the html source.

Creating screenshots is fully compatible with pytest-xdist plugin and will transfer the screenshots from the slave nodes through the communication channel automatically.

If a test (using the browser fixture) fails, you should get a screenshot files in the following path:

```
<splinter-screenshot-dir>/my.dotted.name.test.package/test_name-browser.png
<splinter-screenshot-dir>/my.dotted.name.test.package/test_name-browser.html
```

The *splinter-screenshot-dir* for storing the screenshot is generated by a fixture and can be provided through a command line argument, as described above at the configuration options section.

Taking screenshots on test failure is enabled by default. It can be controlled through the *splinter_make_screenshot_on_failure* fixture, where return *False* skips it. You can also disable it via a command line argument:

```
pytest tests/functional --splinter-make-screenshot-on-failure=false
```

In case taking a screenshot fails, a pytest warning will be issued, which can be viewed using the *-rw* argument for *pytest*.

Python3 support

Python3 is supported, check if you have recent version of splinter as it was added recently.

Example

test_your_test.py:

```
def test_some_browser_stuff(browser):
    """Test using real browser."""
    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('btnK')
    # Interact with elements
    button.click()
    assert browser.is_text_present('splinter.cobrateam.info'), 'splinter.cobrateam.
↪info wasn't found... We need to'
    ' improve our SEO techniques'
```

Contact

If you have questions, bug reports, suggestions, etc. please create an issue on the GitHub project page.

License

This software is licensed under the MIT license

See License file

CHAPTER 2

Authors

Anatoly Bubenkov original idea and implementation, new features and improvements

These people have contributed to *pytest-splinter*, in alphabetical order:

- Andreas Pelme
- Andrey Makhnach
- Aymeric Augustin
- Daniel Hahler
- Ionel Cristian Mărieș
- Laurence Rowe
- Marco Buccini
- Oleg Pidsadnyi
- Peter Lauri
- Suresh V
- Tomáš Ehrlich

CHAPTER 3

Changelog

1.8.5

- Fixed issue with xdist #94 (bubenkoff)

1.8.3

- Profile does not work with geckodriver+remote webdriver #90) (pelme)

1.8.2

- Fixed missing *switch_to* method (some selenium *expected_conditions* are broken without it, see #93)

1.8.1

- Ensure node's *splinter_failure* always exists (bubenkoff, pelme)
- Correctly handle skipped tests (bubenkoff, schtibe)

1.8.0

- Limit retry behavior for *prepare_browser* (bubenkoff)
- Workaround for cleaning cookies (Edge browser) (bubenkoff)

1.7.8

- Make it possible to override the default value for –splinter-wait-time (magnus-staberg)

1.7.7

- Make it possible to override the default *–splinter-webdriver* (pelme)
- Fix screenshots for function scoped fixtues (pelme)

1.7.6

- Support pytest 3 (bubenkoff)
- Less opionated override of splinter's visit (bubenkoff)

1.7.5

- escape screenshot paths for path separators (bubenkoff)

1.7.4

- use tmpdir_factory to get session scoped tmpdir (RonnyPfannschmidt, bubenkoff)

1.7.3

- fixed Firefox freezing when opening a missing codec extension (olegpidsadnyi)

1.7.2

- fixed taking a screenshot with pytest>=2.9.0 (olegpidsadnyi)

1.7.1

- pytest warnings fixed (firebirdberlin)
- remove firefox firstrun script (aaugustin, bubenkoff)

1.7.0

- add possibility to clean cookies on given domains during the browser cleanup, document cookies cleanup (bubenkoff)

1.6.6

- screenshot encoding made flexible (bubenkoff)

1.6.2

- pass timeout to restored connection (bubenkoff)

1.6.0

- added html screenshot (bubenkoff, blueyed)

1.5.3

- remote webdriver fixes (bubenkoff)

1.5.2

- respect splinter_make_screenshot_on_failure (bubenkoff)

1.5.1

- use native selenium socket timeout feature (pelme)

1.5.0

- pytest tmpdir_factory support (bubenkoff)
- depend on splinter 0.7.3, remove the previous status_code monkey patch (pelme)
- add option *–splinter-wait-time* to specify splinter explicit wait timeout (pelme)

1.4.6

- ensure base tempdir exists (bubenkoff)

1.4.0

- introduce splinter_browser_class fixture (bubenkoff, ecesena)

1.3.8

- correctly handle zope.testbrowser splinter driver (bubenkoff)

1.3.7

- pass *splinter_selenium_implicit_wait* as *wait_time* to splinter Browser (lrowe)

1.3.6

- properly respect webdriver executable command line option (bubenkoff, bh)

1.3.5

- add option –splinter-webdriver-executable for phantomjs and chrome (sureshvv)

1.3.4

- make `browser_instance_getter` session scoped, add `session_browser` fixture (bubenkoff, sureshvv)

1.3.3

- make `mouse_over` comparible with more use-cases (bubenkoff)

1.3.1

- properly handle driver switch during the test run (bubenkoff)
- respect splinter_session_scoped_browser fixture (bubenkoff)

1.2.10

- handle exceptions during screenshot saving (blueyed, bubenkoff)
- documentation improvements (blueyed)

1.2.9

- status_code is back in a lazy way (bubenkoff)

1.2.7

- Fix automatic download of pdf content type (bubenkoff)

1.2.4

- fix failing the test run if pytest-xdist is not installed, as it's completely optional dependency (bubenkoff, slafs)

1.2.3

- improve exception handing when preparing the browser instance (bubenkoff)
- require pytest (bubenkoff)

1.2.0

- automatic screenshot capture on test failure (bubenkoff)
- improvements to the browser preparation procedure (bubenkoff)
- boolean config options made more clear (bubenkoff)

1.1.1

- restore browser parameters on each test run instead of once for browser start (bubenkoff)

1.1.0

- added possibility to have multiple browser instances for single test (amakhnach, bubenkoff)

1.0.4

- Fixed browser fixture to support splinter_browser_load_condition and splinter_browser_load_timeout by default. (markon)

1.0.3

- unicode fixes to setup.py (bubenkoff, valberg)

1.0.2

- wait_for_condition now receives pytest_bdd.plugin.Browser object, not selenium webdriver one (bubenkoff)

1.0.1

- Refactoring and cleanup (bubenkoff)

1.0.0

- Initial public release