
django-browserstack-tools

Documentation

Release 0.1.1

QDQ media S.A.U.

November 12, 2016

1	django-browserstack-tools	3
1.1	Documentation	3
1.2	Quickstart	3
1.3	Features	3
2	Installation	5
3	Usage	7
4	Contributing	9
4.1	Types of Contributions	9
4.2	Get Started!	10
4.3	Pull Request Guidelines	10
4.4	Tips	11
5	Credits	13
5.1	Development Lead	13
5.2	Contributors	13
6	History	15
6.1	0.1.0 (2014-04-21)	15

Contents:

django-browserstack-tools

Unofficial Browserstack integration for Django

1.1 Documentation

The full documentation is at <https://django-browserstack-tools.readthedocs.org>.

1.2 Quickstart

Install django-browserstack-tools:

```
pip install django-browserstack-tools
```

Then use it in a project:

```
from browserstack_tools.testcases import BrowserStackLiveServerTestCase

class SeleniumTest(BrowserStackLiveServerTestCase):

    desired_capabilities = {'browser': 'Chrome',
                            'browser_version': '33.0',
                            'os': 'Windows',
                            'os_version': '8.1',
                            'resolution': '1680x1050'}

    def test_login(self):
        self.selenium.get('%s%s' % (self.live_server_url, reverse('home')))
        self.assertIn("Hello", self.selenium.title)
```

1.3 Features

- Create Selenium Test using BrowserStack in Django

Installation

At the command line:

```
$ easy_install django-browserstack-tools
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-browserstack-tools  
$ pip install django-browserstack-tools
```

Usage

To use django-browserstack-tools in a project:

```
import django-browserstack-tools
```

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at <https://github.com/dnmellen/django-browserstack-tools/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

4.1.4 Write Documentation

django-browserstack-tools could always use more documentation, whether as part of the official django-browserstack-tools docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/dnmellen/django-browserstack-tools/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *django-browserstack-tools* for local development.

1. Fork the *django-browserstack-tools* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-browserstack-tools.git
```

3. Install your local copy into a virtualenv. Assuming you have *virtualenvwrapper* installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-browserstack-tools
$ cd django-browserstack-tools/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass *flake8* and the tests, including testing other Python versions with *tox*:

```
$ flake8 browserstack_tools tests
$ python setup.py test
$ tox
```

To get *flake8* and *tox*, just *pip* install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in *README.rst*.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/dnmellen/django-browserstack-tools/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_browserstack_tools
```

Credits

5.1 Development Lead

- QDQ media S.A.U. <tecnologia@qdqmedia.com>

5.2 Contributors

None yet. Why not be the first?

History

6.1 0.1.0 (2014-04-21)

- First release on PyPI.