django-socket-server Documentation

Release 0.0.4

Ashley Wilson

Mar 24, 2017

Contents

Contents:

CHAPTER 1

django-socket-server

Django Socket Server

Quickstart

1. Install *django-socket-server*:

```
pip install django-socket-server
```

2. Add *socket_server* to *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'socket_server',
    ...
)
```

Create a *sockets.py* in an application of your project.

django-socket-server will discover the socket files that are in applications installed against Django.

An example *sockets.py* looks like this:

```
from socket_server.namespace import EventNamespace


class Namespace(EventNamespace):

    def client_connected(self, client):
        super(Namespace, self).client_connected(client)

        print 'Send ping'
        self.emit_to(client, 'ping')
```

```
    def register_callbacks(self):
        return {
            'pong': self.pong
        }

    def pong(self, client, **kwargs):
        print 'Received pong event'
```

Messages are sent and received in JSON, and always contain an *event* key. This key is then mapped to callbacks, added inside *register_callbacks*.

You can specify a namespace name using the name property like so:

```
class Namespace(EventNamespace):
    name = 'pingpong'
```

If you do not specify a name, the app name will be used by default.

Start Socket Server

Use the management command provided to start the socket server: *python manage.py start_socket*.

You may pass an optional *–port* to override the default port of *3000*.

Client connection

The above example would expose the following: *ws://localhost:3000/pingpong*

Documentation

The full documentation is at https://django-socket-server.readthedocs.io.

Links

- Autobahn Python
- Twisted

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-socket-server
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-socket-server
$ pip install django-socket-server
```

CHAPTER 3

Usage

To use django-socket-server in a project:

1. Add *socket_server* to *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'socket_server',
    ...
)
```

CHAPTER 4

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/CptLemming/django-socket-server/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/CptLemming/django-socket-server/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 :)

Get Started!

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

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

```
$ git clone git@github.com:your_name_here/django-socket-server.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-socket-server
$ cd django-socket-server/
$ 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 socket_server 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.

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/CptLemming/django-socket-server/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

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

CHAPTER 5

Credits

Development Lead

- Ashley Wilson <scifilem@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

0.0.1 (2015-01-01)

- First release on PyPI.

0.0.2 (2015-01-30)

- Added python client classes

0.0.3 (2015-01-30)

- Catch server shutdown and pass to namespaces

0.0.4 (2015-02-01)

- Remove debugging and add “room” event errors