django-subcommand2 Documentation

Release 0.1.1

Ashley Wilson

July 11, 2016

Contents

1 django-subcommand2 **3**
- 1.1 Documentation . . . 3
- 1.2 Install . . . 3
- 1.3 Usage . . . 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 (2016-01-26) . . . 15
- 6.2 0.1.1 (2016-01-26) . . . 15

Contents:

CHAPTER 1

django-subcommand2

1.1 Documentation

The full documentation is at https://django-subcommand2.readthedocs.io.

1.2 Install

Install django-subcommand:

```
pip install django-subcommand2
```

1.3 Usage

```
# myapp.management.commands.parent_command.py
from subcommand.base import SubcommandCommand

from .subcommands.sub import MySubcommand


class Command(SubcommandCommand):
    help = 'My Parent Command'

    subcommands = {
        'sub': MySubcommand,  # python manage.py parent_command sub
    }


# myapp.management.commands.subcommands.sub.py
from django.core.management.base import BaseCommand


class MySubcommand(BaseCommand):
    help = 'My Sub Command'
```

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-subcommand2
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-subcommand2
$ pip install django-subcommand2
```

CHAPTER 3

Usage

To use django-subcommand2 in a project:

```
# myapp.management.commands.parent_command.py
from subcommand.base import SubcommandCommand

from .subcommands.sub import MySubcommand


class Command(SubcommandCommand):
    help = 'My Parent Command'

    subcommands = {
        'sub': MySubcommand,  # python manage.py parent_command sub
    }


# myapp.management.commands.subcommands.sub.py
from django.core.management.base import BaseCommand


class MySubcommand(BaseCommand):
    help = 'My Sub Command'
```

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/CptLemming/django-subcommand2/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-subcommand2 could always use more documentation, whether as part of the official django-subcommand2 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/CptLemming/django-subcommand2/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-subcommand2* for local development.

1. Fork the *django-subcommand2* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-subcommand2.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-subcommand2
$ cd django-subcommand2/
$ 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 subcommand 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/CptLemming/django-subcommand2/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_subcommand
```

CHAPTER 5

Credits

5.1 Development Lead

- Ashley Wilson <scifilem@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2016-01-26)

- First release on PyPI.

6.2 0.1.1 (2016-01-26)

- Update badges.