django-shells Documentation

Release 0.1.0

Tzu-ping Chung

December 25, 2015

Contents

Contents:

CHAPTER 1

django-shells

Better shells for your *manage.py*.

1.1 Documentation

The full documentation is at https://django-shells.readthedocs.org.

1.2 Quickstart

Install django-shells:

```
pip install django-shells
```

Then add `'shells'` to your `INSTALLED_APPS`.

1.3 Features

- TODO

1.4 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-pypackage

CHAPTER 2

Installation

At the command line:

```
$ pip install django-shells
```

CHAPTER 3

Usage

To use django-shells, add `'shells'` in your `INSTALLED_APPS`. This replaces the `shell` and `dbshell` management commands with a customised version.

3.1 The `shell` Command

The new `shell` command adds two choices as your Python shell: `ptpython` and `ptipython`. Both requires you to install Jonathan Slender's ptpython, and the latter also requires IPython.

The interpreter is chosen automatically based on what your environment has. All command line options are identical to the built-in `shell` command, except that the `--interface` (and the `-i` shorthand) supports two additional values `ptpython` and `ptipython`.

3.2 The `dbshell` Command

Two additional database clients are added: `pgcli` for PostgreSQL, and `mycli` for MySQL. Both require you to install a Python package with the same name.

The client is chosen automatically based on your database settings, and what your environment provides. You can also use the `--plain` and `--interface` (shorthanded `-i`) options to specify one explicitly.

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/uranusjr/django-shells/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-shells could always use more documentation, whether as part of the official django-shells 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/uranusjr/django-shells/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-shells* for local development.

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

```
$ git clone git@github.com:your_name_here/django-shells.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-shells
$ cd django-shells/
$ 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:

```
$ flake8 shells
```

To get flake8, just pip install it 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 2.7, 3.3 onwards, and for PyPy.

CHAPTER 5

Credits

5.1 Contributors

- Tzu-ping Chung <uranusjr@gmail.com>

CHAPTER 6

History

6.1 0.1.0 (2015-11-27)

- First release on PyPI.
- Support for ptpython, ptipython, pgcli, and mycli implemented.