proj Documentation

Release 0.1.0

Lars Yencken

October 15, 2014

Contents

Contents:

CHAPTER 1

proj

A command-line folder manager for many projects.

- Free software: BSD license
- Documentation: https://proj.readthedocs.org.

`proj` assumes the following working setup:

- You have a directory of active projects that you're working on (e.g. ~/Projects)
- You have a directory of inactive projects, your archive (e.g. ~/Archive)

Given this setup, `proj` helps you add and remove projects from your archive, and keeps your archive organised in `<year>/<quarter>` subfolders, based on when each project was last worked on.

1.1 Installation

Install the package with pip:

```
pip install git+git://github.com/larsyencken/proj
```

Then, tell `proj` where your archive directory is, by adding a line to your `.bashrc` or `.zshrc` file:

```
export PROJ_ARCHIVE=~/Archive
```

1.2 Usage

Use proj to get rid of clutter in your main directory of projects by archiving ones that aren't being worked on. Proj will detect when you last made a change and file it accordingly.

```
$ ls
cocktails-that-are-blue   news-for-llamas   old-crusty-project
$ proj archive old-crusty-project
old-crusty-project -> /Users/lars/Archive/2012/q3/old-crusty-project
$ ls
cocktails-that-are-blue   news-for-llamas
$ proj list
2012/q3/old-crusty-project
```

Now we've archived this project, but we can restore it at any time.

```
$ proj restore old-crusty-project
/Users/lars/Archive/2012/q3/old-crusty-project -> old-crusty-project
$ ls
cocktails-that-are-blue   news-for-llamas   old-crusty-project
```

1.3 Features

1.3.1 0.1.0

- `proj archive`: archive a project to an appropriate directory
- `proj restore`: restore a project from the archive
- `proj list`: search the archive for a project

CHAPTER 2

Installation

At the command line:

```
$ easy_install proj
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv proj
$ pip install proj
```

CHAPTER 3

Usage

To use proj in a project:

```
import proj
```

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/larsyencken/proj/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

proj could always use more documentation, whether as part of the official proj 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/larsyencken/proj/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 *proj* for local development.

1. Fork the *proj* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/proj.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 proj
   $ cd proj/
   $ 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 proj 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, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/larsyencken/proj/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_proj
```

CHAPTER 5

Credits

5.1 Development Lead

- Lars Yencken <lars@yencken.org>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

0.1.0 (2014-01-11)

- First release on PyPI.

CHAPTER 8

Indices and tables

- *genindex*
- *modindex*
- *search*