pytmux Documentation

Release 0.1.0

Wraithan

July 16, 2015

Contents

pytmux is a light wrapper around tmux. You define a session using a json file and it tries to create it. For more information see below:

Contents:

CHAPTER 1

Install

1.1 From PyPI

Installation is as simple as:

```
pip install pytmux
```

1.2 From GitHub

Or if you would like to install the latest from GitHub:

```
pip install https://github.com/wraithan/pytmux/archive/master.tar.gz
```

1.3 From Source

Finally you can install from source by doing:

```
git clone https://github.com/wraithan/pytmux
cd pytmux
python setup.py install
```

CHAPTER 2

Usage

pytmux tries to keep to the principle of least astonishment, much like Python. If a command or configuration option isn't immediately obvious, please open an issue and we can talk about how to improve the situation.

2.1 Commands

2.1.1 list

To list all of the available configs use:

```
pytmux list
```

2.1.2 edit

To create or edit a config, use:

```
pytmux edit <name>
```

If you would like use a config as the base for a new config:

```
pytmux edit <name> --copy <other_name>
```

2.1.3 run

To start or switch to a session use:

```
pytmux run
```

2.1.4 doctor

If you are having trouble with one of your configs, use:

```
pytmux doctor
```

2.2 Configuration

Lets start with a base configuration, then we can walk through what each part means.

```
{
    "name": "example",
    "directory": "~/devel/example",
    "windows": [
        {
            "name": "editor",
            "command": "emacs"
        },
        {
            "name": "some shell"
        },
        {
            "command": "tail -f some.log"
        },
        {}
    ]
}
```

The first property is `name` which is the name of the session. This should be a short descriptive name of the what you'll be using the session for. You'll want to make sure it is unique so can run it and not conflict with other running sessions.

Next you have the `directory` property which is what directory to use for starting each window. It is optional, if it isn't provided then the current directory that pytmux is run from is used.

After that you have `windows` which is the list of windows you want instantiated. Both of the properties (`name` and `command`) are optional. If you specify a `name` it will name the window that, without a `name` you'll get a window named using `automatic-rename` in tmux (which uses lets the program set the title). If you specify a `command` then that will be run in the window using `send-keys`, without the `command` it will open your default shell. Without either, you'll get a default shell, in a window that uses `automatic-rename`.

CHAPTER 3

How to Contribute

All development is done via GitHub using issues and pull requests. If you are doing something major (new feature, backwards incompatible change, reworking lots of code) please open an issue and discuss it before starting. If you are doing something minor (docs update, small bug fix, etc) feel free to just dive in and send a pull request.

3.1 Code Standards

All code should pass `pep8` and `pyflakes`. A simple tool for checking is flake8. You should be able to run:

```
flake8 .
```

At the root of the repository and have 0 errors or warnings. That said, everyone slips now and then, and one or two isn't a big deal and I'll just fix them myself. But if your whole patch fails then I'll likely reject it until you clean up your code.

3.2 Reporting Issues

You can report issues and feature via the issue tracker. If you have an issue please give full steps to reproduce (such as a config file that has the problem). If you have a feature you'd like, please provide as much information as possible (such as what flags in tmux the feature uses or how exactly you'd like it to work.)

CHAPTER 4

Indices and tables

- genindex
- modindex
- search