writeme Documentation

Release 1.0

Gabe Meringolo, Matt Favoino, Darren Haynes, Chris Closser, Che

Dec 22, 2017

Contents

Write-me generates concise, effective, and specific READMEs based on your codebase.

Contents:

CHAPTER 1

Introduction

1.1 Description

Write-me is a PyPI package that searches through a user's codebase, and uses filepaths, docstrings, web framework specifications, installed requirements, and other user-added data to generate a README.md file.

This package was written as a project by students at Code Fellows, Seattle in December 2017.

Project Repository: write-me

1.2 Change Log

Write-me is currently on its first version: write-me-1.0. Future breaking and non-breaking changes in functionality will be documented here.

CHAPTER 2

Installation

2.1 Prerequisites

To use write-me, you must have:

- Python
- Pip

To ensure that you have these packages installed, execute the following on your commandline:

`$ pip --version` and `$ python --version`

The output of both should be something like 'Python 3.6.3'. If running the `which` command returns nothing to the commandline, the package is not installed. In that case, download the packages using the instructions from the links above.

2.2 Installing write-me

Now that you have Python and pip installed, you can proceed to installing write-me! As write-me is a PyPI package, it is pip installable. On the commandline, execute the following:

```
$ pip install write-me
```

If the installation is a success, the package will collect and install, and will end with the line:

```
$ Successfully installed write-me-1.0
```

CHAPTER 3

User Guide

3.1 Description

Generate a README, with the following sections:

- Project Overview (Description, Version, Highlight Features ^, Authors ^)
- Documentation ~ ^
- Getting Started (Prerequisites, Dependencies, Installation, Local Serving ^)
- Testing (Running Tests, Testing Modules)
- URLs (URL Modules)
- Modules ~
- Development Tools
- Contributions ~
- License
- Acknowledgements ^

  KEY:

  - ^ Section cannot be fully automated. User editing necessary after README creation.
  - ~ Section generated only with use of verbose `-v` flag.

3.2 Commandline Tools

Note: The `genreadme` command must be executed from the root of the application: the highest possible level of the directory.

3.2.1 genreadme

The `genreadme` command is the core functionality of write-me. Upon execution at the root level of a directory, it generates the a README for the current project. This command can be use as a standalone tool (without flagged options selected) or with up to two "options" flags.

If the command is used without an additional flag, it generates a simple README *without* web framework-specific sections. The "URLs" and "Local Serving" sections are eliminated. This option is best for non-web application respositories, such as: data science projects, data structures, and PyPI packages.

3.2.2 -d, --django

The `-d` or `--django` flag is used to specify that the current repositiory is a web application created using the Django web framework. With this flag, the README is generated with Django-specific serving instructions, and references Django's default port 8000 for local hosting.

See also:

https://www.djangoproject.com/

3.2.3 -p, --pyramid

The `-p` or `--pyramid` flag is used to specify that the current repositiory is a web application created using the Pyramid web framework. With this flag, the README is generated with Pyramid-specific serving instructions, and references Pyramid's default port 6543 for local hosting.

See also:

https://trypyramid.com/

3.2.4 -f, --flask

The `-f` or `--flask` flag is used to specify that the current repositiory is a web application created using the Flask web framework. With this flag, the README is generated with Flask-specific serving instructions, and references Flask's default port 5000 for local hosting.

See also:

http://flask.pocoo.org/

3.2.5 -v, --verbose

The `-v` or `--verbose` flag is used to create a longer and more verbose version of the default generated README. If used without any other web framework flag, this tag only adds the following section:

- Documentation (links to other documentation sources)
- Contributions (provides contact information for open source contributors)

When used *with* one of the web framework flags, it adds the two sections above, as well as the following section:

- Modules (lists all Python modules used in the project)

3.2.6 -h, --help

The `-h` or `--help` flag is used to show all write-me commandline options at the Terminal. When used, it displays a short description of the `-d`, `-p`, `-f`, and `-v` flags, and exits out of the write-me commandline tool.

CHAPTER 4

Tutorial

4.1 Tutorial #1: Simple Project

The most simple usage of write-me is using the `genreadme` command to make a README for a simple project that is not a web application. Tutorial #1 walks through using write-me in this scenario using an example project: a `data-structures` repository containing data structures written in python and javascript, as well as configuration data.

4.1.1 Installing write-me

The first step is to navigate to the root of the demo repository: the upmost level of `data-structures`.

```
data-structures [master *]$ ls
.cache                  .gitignore              ENV                     __pycache__
.coverage               .tox                    LICENSE                 js-src
.git                    Data_structures.egg-info README.md              src
data-structures [master *]$
```

The first step is to install the write-me PyPI package using pip. Once pip is installed, execute the following line:

```
$ pip install write-me
```

For further instructions regarding , see the "Installation" section of this documentation site.

```
data-structures [master *]$ pip install write-me
Collecting write-me
  Using cached write_me-0.5-py3-none-any.whl
Requirement already satisfied: markdown-generator in /Library/Frameworks/Python.framework/Versions/3.6/lib/pyt
hon3.6/site-packages (from write-me)
Installing collected packages: write-me
Successfully installed write-me-0.5
```

4.1.2 Using genreadme

Now that write-me is installed successfully, execute the `genreadme` command. If the repo root already contains a README, the terminal window will prompt the user to confirm that the program can overwrite the old README, saving it as README.md.old.

Once the command has been successfully run, the terminal will display the following message, containing an indication of success and user TODOs — content necessary for a robust README that can not be gathered from the reposory data.

```
data-structures [master *]$ genreadme

    Do you want to overwrite your present README file?
    Don't worry, if you overwrite your present README it will be backed up to README.md.old
    Yes or no?
    Y
README generated.
```

4.1.3 README generation confirmation

The README has now been successfully created. On that same repo level, the user can now see their newly generated `README.md`, as well as their old version in `README.md.old`.

```
data-structures [master *]$ ls
.cache                  .gitignore              ENV                     __pycache__
.coverage               .tox                    LICENSE                 js-src
.git                    Data_structures.egg-info README.md              src
data-structures [master *]$ pip install write-me
Collecting write-me
  Using cached write_me-0.5-py3-none-any.whl
Requirement already satisfied: markdown-generator in /Library/Frameworks/Python.framework/Versions/3.6/lib/pyt
hon3.6/site-packages (from write-me)
Installing collected packages: write-me
Successfully installed write-me-0.5
data-structures [master *]$ genreadme

    Do you want to overwrite your present README file?
    Don't worry, if you overwrite your present README it will be backed up to README.md.old
    Yes or no?
    Y
README generated.
data-structures [master !?*]$ ls
.cache                  .tox                    README.md               src
.coverage               Data_structures.egg-info README.md.old
.git                    ENV                     __pycache__
.gitignore              LICENSE                 js-src
data-structures [master !?*]$
```

4.2 Tutorial #2: Web Framework

The slightly more advanced usage of write-me is using the same command, `genreadme` to make a README for a project that uses a web framework. The current version of write-me supports web framework implementation with Django, Pyramid, and Flask. Tutorial #2 walks through using write-me in this scenario using an example project: a `django-imager` repository containing a Django webapp used for hosting images.

4.2.1 Installing write-me

The first step is to navigate to the root of the demo repository: the upmost level of `django-imager`.

```
(ENV) django-imager [master]$ ls
.git                ENV                 django-imager.conf  simple_nginx_config
.gitignore          LICENSE             imagersite
.gitkeep            README.md           requirements.pip
(ENV) django-imager [master]$
```

The first step is to install the write-me PyPI package using pip. In this example, it is being installed within a virtual environment called ENV. Once pip is installed, execute the following line:

```
$ pip install write-me
```

For further instructions regarding , see the "Installation" section of this documentation site.

```
(ENV) django-imager [master]$ pip install write-me
Collecting write-me
  Using cached write_me-0.5-py3-none-any.whl
Collecting markdown-generator (from write-me)
  Using cached markdown_generator-0.1.3.tar.gz
Installing collected packages: markdown-generator, write-me
  Running setup.py install for markdown-generator ... done
Successfully installed markdown-generator-0.1.3 write-me-0.5
(ENV) django-imager [master]$
```

4.2.2 Using genreadme -d

Now that write-me is installed successfully, execute the `genreadme -d` or `genreadme --django` command. (They are equivalent.) If you are using Pyramid or Flask instead of Django, just substitute `-d` for `-p` or `-f`. If the repo root already contains a README, the terminal window will prompt the user to confirm that the program can overwrite the old README, saving it as README.md.old.

Once the command has been successfully run, the terminal will display the following message, containing an indication of success and user TODOs — content necessary for a robust README that can not be gathered from the reposory data.

```
(ENV) django-imager [master]$ genreadme -d

    Do you want to overwrite your present README file?
    Don't worry, if you overwrite your present README it will be backed up to READM
E.md.old
    Yes or no?
    Y
README generated.
```

4.2.3 README generation confirmation

The README has now been successfully created. On that same repo level, the user can now see their newly generated `README.md`, as well as their old version in `README.md.old`.

```
(ENV) django-imager [master]$ ls
.git                ENV                 django-imager.conf  simple_nginx_config
.gitignore          LICENSE             imagersite
.gitkeep            README.md           requirements.pip
(ENV) django-imager [master]$ pip install write-me
Collecting write-me
  Using cached write_me-0.5-py3-none-any.whl
Collecting markdown-generator (from write-me)
  Using cached markdown_generator-0.1.3.tar.gz
Installing collected packages: markdown-generator, write-me
  Running setup.py install for markdown-generator ... done
Successfully installed markdown-generator-0.1.3 write-me-0.5
(ENV) django-imager [master]$ genreadme -d

    Do you want to overwrite your present README file?
    Don't worry, if you overwrite your present README it will be backed up to READM
E.md.old
    Yes or no?
    Y
README generated.
(ENV) django-imager [master !?]$ ls
.git                ENV                 README.md.old       requirements.pip
.gitignore          LICENSE             django-imager.conf  simple_nginx_config
.gitkeep            README.md           imagersite
(ENV) django-imager [master !?]$
```