From Editor to PyPi Documentation

Release 0.0.1

Gabriel Falcão

Jun 03, 2018

Agenda:

Danger: This documentation is a work-in-progress. Please watch the project on github to receive updates on its repository.

CHAPTER 1

Programme

1. Dependencies
   (a) github account
   (b) readthedocs account
   (c) pypi account
   (d) python 3.6
   (e) cookiecutter
   (f) if working in teams use a simple editor with no frills (VSCode is great)
2. Write a python module that serves as HTTP client for a public API (httpbin.org)
3. Create a setup.py
   (a) declare dependencies
   (b) name, description and version
   (c) long description as rst docstring
   (d) add trove classifiers PEP 0301
   (e) pick an open source license
      i. MIT if you don't care
      ii. GPL for viral recognition
4. Publishing a package
5. Adding command-line entrypoints
6. Distributing package data
7. Define extra dependencies (such as requests[security])
8. Writing tests
   (a) Tooling: Nose (and rednose vs. pinocchio), Sure, HTTPretty, Coverage

   (b) Unit, Functional and Integration
   (c) Collecting coverage
   (d) Run tests on Travis
9. Writing docs
   (a) Generate API reference from code
   (b) Syntax for function or method params and return values
   (c) Use intersphinx to reference other libraries documented with sphinx
      i. Did you know about `objects.inv` ?
   (d) Publish docs with ReadTheDocs

CHAPTER 2

Step-by-step guide

2.1 1. Create service accounts

We will depend on services are free for open source projects.

Feel free to skip any services in which you already have an account.

Below you can find direct links for account creation pages:

1. Make a github account
2. Make a PyPi account
3. Make a Read-The-Docs account
4. Make a Travis account (requires a github account)

2.2 2. Install virtualenv

```
pip install virtualenv
```

See also:

If you are feeling adventurous, use pipenv a great tool for python development workflow including managing virtualenvs and keeping track of installed packages (Pipfile)

2.3 3. Install cookiecutter

To speed-up our productivity we will use cookiecutter.

```
pip install cookiecutter
```

2.4 4. Create your package with cookiecutter

Warning: you will need to have git installed for this step.

The template is available on github at gabrielfalcao/cookiecutter-from-editor-to-pypi

Cookiecutter is smart enough to install directly from github, for that just run the following command from your `~/projects` folder or wherever you prefer to keep personal projects.

```
cookiecutter gh:gabrielfalcao/cookiecutter-from-editor-to-pypi
```

Important: **Answer carefully** to the interactive questions. Try your best to avoid typos in this step as it might impact many files of your newly-created project.

2.5 5. Enter your virtualenv

This gives you a sandbox python environment where it's safe to install dependencies without compromising the integrity of your system packages.

In your terminal, go to your project root path and run:

```
source __virtualenv__/bin/activate
```

2.6 6. Run tests

Considering that you succeeded on the step *5. Enter your virtualenv* just run:

```
make unit
make functional
```

2.7 7. Build your pre-generated documentation

Considering that you succeeded on the step *5. Enter your virtualenv* just run:

```
source __virtualenv__/bin/activate
make docs
```

2.8 8. Create a github repository and push your code

Head to https://github.com/new and create a repository that matches your pypi package name.

The cookiecutter template already created a git repository for you, so you can simply **go to your project root path and add the git remote ‘‘origin‘‘ as instructed by github.**

CHAPTER 3

Configuring PyPi for publishing packages

Note: You will need a PyPi account for publishing your packages. Refer to the *1. Create service accounts* page in the *Step-by-step guide* for more information

Head to pypi.org/account/register and create an account.

Make sure to create a strong password, include special characters.

CHAPTER 4

Write your PyPi credentials to `.pypirc`

Edit the file `~/.pypirc` and add your credentials.

Below is a snippet to make things easier.

```
[distutils]
index-servers =
    pypi

[pypi]
username:YOUR_USERNAME
password:YOUR_PASSWORD
```

CHAPTER 5

Creating a `setup.py`

5.1 `README.rst` and `long_description`

The `setup(long_description=...)` field of `setup.py` contains valid reStructuredText This is "juice" of your package documentation on pypi.

Reading this dinamically from a README.rst is usually a good idea and makes your project have an uniform introduction on both pypi and github.

5.2 Declare Trove Classifiers

Trove is an open-source project that attemps to classify a large package repository.

The existing classifiers are a derived from FreshMeat and SourceForge classifiers.

You can learn more at PEP 301.

CHAPTER 6

Next Steps

6.1 Improve your build

- Consider using flake8 to lint-check your project.
- Consider using tox to test your package against multiple python versions.

6.2 Host a private PyPi

pip is pretty flexible when it comes to finding python packages, it will even scrape HTML pages and look for `<a>` tags whose `href=""` attribute match the package name and archived in a supported format (e.g.: tar.gz, zip, etc)

You can configure the `~/.pypirc` with several profiles and several different pypi servers.

Here are some open source pypi servers and related tools:

- Warehouse - The official pypi server software that powers pypi.org
- Basket - A personal favorite, useful for creating a local pypi mirror of packages to be used when offline or working remote.
- pypiserver - a minimal pypi server, no pretty UI.
- devpi-server - devpi is a universal python module packaging, testing and release tool. It comes with the command `devpi-server` which runs a pypi server.
- pure apache mod_rewrite

6.3 Get better at writing docs

- https://pythonhosted.org/an_example_pypi_project/sphinx.html

6.4 Explore other tools

- Many developers prefer py.test over nosetests, you should check it out.

6.5 Please give feedback

Create issues and/or pull-requests to this workshop at github: https://github.com/gabrielfalcao/from-your-editor-to-pypi/issues

- Suggestions
- Bug (or typo) fixes
- Or whatever could be improved about the workshop, really.