cookiecutter-modern-pypackage

Release 1.2.0

Federico Jaureguialzo

Jan 17, 2021

CONTENTS

CHAPTER

ONE

GETTING STARTED

1.1 Cookiecutter Modern PyPackage

Cookiecutter template for a modern Python package.

- GitHub repo: https://github.com/fedejaure/cookiecutter-modern-pypackage.git
- Documentation: https://cookiecutter-modern-pypackage.readthedocs.io
- Free software: MIT license

1.1.1 Features

- Dependency tracking using Poetry
- Testing setup with Pytest
- Github Actions ready for Continuous Integration testing
- Linting provided by Flake8 with Flakehell
- Docstring linting provided by Darglint using the Google Python Style Guide
- Static type checking by Mypy
- Formatting provided by Black and Isort
- Checks dependencies for known security vulnerabilities with Safety
- Git hooks managed by pre-commit.
- All development tasks (lint, format, test, etc) wrapped up in a python CLI by invoke
- Multiple Python environments testing provided by Nox
- Documentation provided by Sphinx ready for generation with, for example, Read the Docs
- Command line interface using Click (optional)
- Automated dependency updates with Dependabot
- Coverage reports on Codecov
- Automated releases to PyPI and TestPyPI

1.1.2 Quickstart

Install the latest Cookiecutter if you haven't installed it yet (this requires Cookiecutter 1.4.0 or higher):

```
pip install -U cookiecutter
```

Generate a Python package project:

```
cookiecutter gh:fedejaure/cookiecutter-modern-pypackage --checkout v1.2.0
```

Then:

- Create a repo and put it there.
- Install the dev requirements into a virtualenv. (`poetry install`)
- Install pre-commit hooks. (`poetry run inv install-hooks`)
- Configure Codecov repository settings. (Codecov App, `CODECOV_TOKEN`)
- Add the repo to your Read the Docs account + turn on the Read the Docs service hook.
- Configure PyPI and TestPyPI tokens. (`PYPI_TOKEN`, `TEST_PYPI_TOKEN`)
- Release your package by pushing a new tag.

For more details, see the tutorial.

1.1.3 Credits

This cookiecutter was built for learning purpose and inspired by:

- audreyr/cookiecutter-pypackage: Cookiecutter template for a Python package.
- briggySmalls/cookiecutter-pypackage: A fork from audreyr/cookiecutter-pypackage using Poetry for package management, with linting, formatting and more.
- hypermodern-python: Hypermodern Python article series.

1.2 Tutorial

Note: Did you find any of these instructions confusing? Edit this file and submit a pull request with your improvements!

To start with, you will need a GitHub account and an account on PyPI. Create these before you get started on this tutorial. If you are new to Git and GitHub, you should probably spend a few minutes on some of the tutorials at the top of the page at GitHub Help.

1.2.1 Step 1: Install Cookiecutter

Install cookiecutter:

```
$ pip install cookiecutter
```

We'll also need poetry so install that too.

1.2.2 Step 2: Generate Your Package

Now it's time to generate your Python package.

Use cookiecutter, pointing it at the cookiecutter-pypackage repo:

```
$ cookiecutter gh:fedejaure/cookiecutter-modern-pypackage --checkout v1.2.0
```

You'll be asked to enter a bunch of values to set the package up. If you don't know what to enter, stick with the defaults.

1.2.3 Step 3: Create a GitHub Repo

Go to your GitHub account and create a new repo named `mypackage`, where `mypackage` matches the `[project_name]` from your answers to running cookiecutter.

You will find one folder named after the `[project_name]`. Move into this folder, and then setup git to use your GitHub repo and upload the code:

```
$ cd mypackage
mypackage $ git init .
mypackage $ git add .
mypackage $ git commit -m "Initial skeleton."
mypackage $ git remote add origin git@github.com:myusername/mypackage.git
mypackage $ git push -u origin master
```

Where `myusername` and `mypackage` are adjusted for your username and package name.

You'll need a ssh key to push the repo. You can Generate a key or Add an existing one.

1.2.4 Step 4: Install Dev Requirements

You should still be in the folder containing the `pyproject.toml` file.

Install the new project's local development requirements inside a virtual environment using poetry:

```
$ poetry install
$ poetry run inv install-hooks
```

1.2.5 Step 5: Set Up Codecov

Codecov provides highly integrated tools to group, merge, archive, and compare coverage reports.

Log into your account at Codecov. If you don't have one, create one and log into it.

Click on *Add new repository*. Choose the desired one. Then follow the instructions to setup the *CODECOV_TOKEN* on the github secrets.

Install the Codecov github App.

Now your coverage reports will be generated when a new PR is created.

1.2.6 Step 6: Set Up Read the Docs

Read the Docs hosts documentation for the open source community. Think of it as Continuous Documentation.

Log into your account at Read the Docs . If you don't have one, create one and log into it.

If you are not at your dashboard, choose the pull-down next to your username in the upper right, and select "My Projects". Choose the button to Import the repository and follow the directions.

Now your documentation will get rebuilt when you make documentation changes to your package.

1.2.7 Step 7: Release on PyPI and TestPyPI

The Python Package Index or PyPI is the official third-party software repository for the Python programming language. Python developers intend it to be a comprehensive catalog of all open source Python packages.

TestPyPI is a separate instance of the Python Package Index (PyPI) that allows you to try out the distribution tools and process without worrying about affecting the real index.

Log into your account at PyPI and TestPyPI. Go to Account Settings and generate an API tokens.

Go to the repository settings on GitHub, and add tow secrets named *PYPI_TOKEN* and *TEST_PYPI_TOKEN* with the tokens that you just generated.

Release your package by pushing a new tag.

1.2.8 Having problems?

Visit our Issues page and create a new Issue. Be sure to give as much information as possible.

1.3 Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

1.3.1 Unreleased

1.3.2 1.2.0 - 2021-01-17

Added

- pyproject `documentation` entry.
- pyproject `tool.poetry.urls` section.

Fixed

- Readme links.

Changed

- sphinx from `^3.4.0` to `^3.4.3`.
- safety from `^1.10.0` to `^1.10.3`.
- flake8-blind-except from `^0.1.1` to `^0.2.0`.
- flake8-annotations from `^2.1.0` to `^2.5.0`.
- isort from `^5.6.4` to `^5.7.0`.
- invoke from `^1.4.1` to `^1.5.0`.
- flakehell from `^0.7.1` to `^0.9.0`.
- parametrize cli tests.

1.3.3 1.1.3 - 2020-12-23

Changed

- sphinx from `^3.3.0` to `^3.4.0`.
- recommonmark from `0.6.0` to `0.7.1`
- watchdog from `^0.10.2` to `^1.0.2`.
- pre-commit from `^2.8.2` to `^2.9.3`.
- flakehell from `^0.7.0` to `^0.7.1`.
- safety from `^1.9.0` to `^1.10.0`.
- darglint from `^1.3.0` to `^1.5.8`.
- flake8-bugbear from `^20.1.4` to `^20.11.1`.
- actions/setup-python from `v2.1.4` to `v2.2.1`.
- pytest from `^6.1.2` to `^6.2.1`.

1.3.4 1.1.2 - 2020-11-07

Changed

- flakehell from `^0.6.1` to `^0.7.0`.
- create-release action from `v1` to `v1.1.4`.
- checkout action from `v2` to `v2.3.4`.
- setup-python action from `v2` to `v2.1.4`.
- sphinx from `^3.2.1` to `^3.3.0`.
- pre-commit from `^2.7.1` to `^2.8.2`.
- pytest from `^6.1.1` to `^6.1.2`.

Fixes

- mypy nox session requirements.

1.3.5 1.1.1 - 2020-10-18

Fixes

- `docs/conf.py` imports.
- coverage config.

1.3.6 1.1.0 - 2020-10-17

Changed

- to `src` structure.
- `poject_name` validation.

Added

- `project_title`.

1.3.7 1.0.1 - 2020-10-15

Fixed

- unnecessary `validation_depth` on `mindsers/changelog-reader-action`.

1.3.8 1.0.0 - 2020-10-15

Added

- License section on the docs.
- Codecov integration.
- PyPI and TestPyPI steps on the release workflow.
- Python `3.9` support.

Changed

- github actions ready to configure activity types.
- isort from `^5.5.4` to `^5.6.4`.
- bump2version from `master` to `^1.0.1`.
- mypy from `^0.782` to `^0.790`.
- coverage from `^5.1` to `^5.3`.
- pytest-cov from `^2.8.1` to `^2.10.1`.
- pytest from `^5.4.2` to `^6.1.1`.
- flake8 from `^3.7.9` to `^3.8.4`.

Fixed

- missing pre-commit requirement.
- get release version on the release workflow.

1.3.9 0.2.1 - 2020-10-05

Changed

- changelog-reader-action from v1.1.0 to v2.
- sphinx from 3.0.4 to 3.2.1.
- flakehell from 0.3.6 to 0.6.1.
- black from 19.10b0 to 20.8b1.
- xdoctest from 0.12.0 to 0.15.0.
- mypy from 0.770 to 0.782

Fixed

- read the docs dependencies.

1.3.10 0.2.0 - 2020-10-04

Added

- Dependabot configuration.
- Safety session to nox.
- Safety step to the test workflow.

Changed

- flake8 version to `^3.7.9`.
- isort version to `^5.5.4`.
- poetry export without hashes on the noxfiles.

Removed

- Pyup.io integration.
- seed-isort-config from the pre-commit-config.

Fixed

- docs/readme.md symbolic link to README.md.
- docs/changelog.md symbolic link to CHANGELOG.md.
- missing badges.

1.3.11 0.1.4 - 2020-09-07

Changed

- Python actions to the v2.

Removed

- Unnecessary python steps on the release workflow.

Fixed

- bump2version version.

1.3.12 0.1.3 - 2020-08-13

Fixed

- isort support for pyproject.toml
- docs conf code style.

Removed

- sphinx-autodoc-typehints from the dev requirements.

1.3.13 0.1.2 - 2020-06-14

Fixed

- Read the docs build config.

Removed

- Pytype from the dev requirements.

1.3.14 0.1.1 - 2020-06-14

Added

- New option `serve` to the invoke docs task.

Changed

- Improve docs tutorial section.
- Improve docs index section.

Fixed

- README spelling.
- Ivoke pytype task typo.

1.3.15 0.1.0 - 2020-06-11

Added

- First release.

CHAPTER

TWO

BASICS

2.1 Prompts

When you create a package, you are prompted to enter these values.

2.1.1 Templated Values

The following appear in various parts of your generated project.

full_name Your full name.

email Your email address.

github_username Your GitHub username.

project_name The name of your new Python package project. This is used in the package name and the Github repository name, so use - insteed of spaces.

project_slug The namespace of your Python package. This should be Python import-friendly. Typically, it is the slugified version of project_name.

project_title The title of your new Python project. This is used in documentation, so spaces and any characters are fine here.

project_short_description A 1-sentence description of what your Python package does.

version The starting version number of the package.

2.1.2 Options

The following package configuration options set up different features for your project.

open_source_license Whether to add a license file. Options: ["MIT", "BSD", "ISC", "Apache Software License 2.0", "GNU General Public License v3", "Not open source"s]

command_line_interface Whether to create a console script using Click. Console script entry point will match the project_name. Options: ["Click", "No command-line interface"]

2.2 Invoke

The generated project is ready to run some useful tasks like formatting, linting, testing.

To do this we use pyinvoke to wrap up the required commands.

Execute *inv[oke] –list* to see the list of available commands.

```
$ poerty shell
$ inv[oke] --list
Available tasks:

  clean           Run all clean sub-tasks.
  clean-build     Clean up files from package building.
  clean-docs      Clean up files from documentation builds.
  clean-python    Clean up python file artifacts.
  clean-tests     Clean up files from testing.
  coverage        Create coverage report.
  docs            Build documentation.
  flake8          Run flake8.
  format          Format code.
  hooks           Run pre-commit hooks.
  install-hooks   Install pre-commit hooks.
  lint            Run all linting.
  mypy            Run mypy.
  safety          Run safety.
  tests           Run tests.
  version         Bump version.
```

CHAPTER

THREE

ADVANCED FEATURES

3.1 Console Script Setup

Optionally, your package can include a console script using Click (Python 3.6+).

3.1.1 How It Works

If the 'command_line_interface' option is set to ['click'] during setup, cookiecutter will add a file 'cli.py' in the project_slug subdirectory. An entry point is added to pyproject.toml that points to the main function in cli.py.

3.1.2 Usage

To use the console script in development:

```
pip install -e projectdir
```

'projectdir' should be the top level project directory with the pyproject.toml file

The script will be generated with output for no arguments and –help.

--help show help menu and exit

3.1.3 More Details

You can read more about Click at: http://click.pocoo.org/

3.2 MIT License