cookiecutter-namespace-template Documentation

Release 0.3.0

Veit Schiele

Apr 16, 2024

CONTENTS

CHAPTER

ONE

GETTING STARTED

1.1 Cookiecutter Namespace Template

Cookiecutter Namespace Template for a Python package.

1.1.1 Features

- Testing setup with `unittest` or `pytest`
- Tox testing: Setup to easily test for Python 3.7, 3.8, 3.9, 3.10 and 3.11.
- Sphinx docs: Documentation ready for generation with, for example, ReadTheDocs
- bump2version: Pre-configured version bumping with a single command
- If the cookiecutter-namespace-template project template has been changed, you can apply these changes with

```
$ cruft update
```

- Optional auto-release to PyPI when you push a new tag to main (optional)
- Optional command line interface using Typer or Click

If you really want to create a new package with Python 2, in spite of the Python 2.7 countdown and the Sunsetting Python 2 support, then use `cookiecutter-namespace-template` <0.2.

1.1.2 Quickstart

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

```
$ python -m pip install -U cruft
```

2. Generate a Python package project:

```
$ python -m cruft create https://github.com/veit/cookiecutter-namespace-template.git
```

3. Create a repo and put it there.
4. Register your project with PyPI.
5. Add the repo to your ReadTheDocs account and turn on the ReadTheDocs service hook.
6. If you want to add the pyup badge to your README file

1. create a new account at pyup.io or log into your existing account
2. click on the green *Add Repo* button
3. click *Pin* to add the repo

7. Release your package by pushing a new tag to main.

Pull requests

If you have differences in your preferred setup, I encourage you to fork this to create your own version. I also accept pull requests on this, if they're small, atomic, and if they make my own packaging experience better.

1.2 Tutorial

1. Install cruft

   First, you need to create a virtualenv for the package project. Use your favorite method, or create a virtualenv for your new package like this:

   ```
   python3 -m venv ~/.virtualenvs/my.package
   ```

   Here, `my.package` is the name of the package that you'll create.

Then install cruft:

```
$ cd ~/.virtualenvs/my.package
$ source bin/activate
$ python -m pip install cruft
```

1. Generate Your Package

   Now it's time to generate your Python package.

   Use cruft, pointing it at the cookiecutter-namespace-template repo:

   ```
   $ cruft create https://github.com/veit/cookiecutter-namespace-template.git
   ```

   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.

2. Create a Git Repo

   Go to your Git account and create a new repo named `my.package`, where `my.package` matches the `[namespace.package]` from your answers to running cookiecutter.

   Note: If your venv folder is within your project folder, be sure to add the venv folder name to your `.gitignore` file.

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

   ```
   $ cd my.package
   $ git init .
   $ git add .
   ```

(continues on next page)

(continued from previous page)

```
$ git commit -m "Initial commit"
$ git remote add origin git@example.org:MYUSERNAME/MY.PYCKAGE.git
$ git push -u origin main
```

Where *MYUSERNAME* and *MY.PACKAGE* 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.

3. Install dev requirements

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

   Install the new project's local development requirements:

   ```
   $ python -m pip install -e '.[dev]'
   ```

4. Release on PyPI

   Here's a *release checklist* you can use.

   See also:

   - Packaging Python Projects
   - Python Packaging User Guide

1.3 Prompts

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

1.3.1 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 to to create the namespace and the package name. So spaces and special characters should be avoided.

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

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

`release_date`
: The date of the first release.

pypi_username
Your Python Package Index account username.

year
The year of the initial package copyright in the license file.

version
The starting version number of the package.

1.3.2 Options

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

command_line_interface
Whether to create a *console script* using Typer or Click.

Console script entry point will match the `project_slug`. Options: `['Typer', 'Click', "No command-line interface"]`

1.4 PyPI Release Checklist

1.4.1 For Every Release

1. Update `HISTORY.rst`
2. Commit the changes:

```
$ git add HISTORY.rst
$ git commit -m "Changelog for upcoming release 0.1.1."
```

3. Update version number (can also be patch or major)

```
$ bump2version minor
```

4. Install the package again for local development, but with the new version number:

```
$ python -m pip install -e '.[dev]'
```

5. Run the tests:

```
$ tox
```

6. Push the commit:

```
$ git push
```

7. Push the tags, creating the new release on both GitHub and PyPI:

```
$ git push --tags
```

8. Check the PyPI listing page to make sure that the README, release notes, and roadmap display properly. If not, try one of these:
    1. Copy and paste the RestructuredText into http://rst.ninjs.org/ to find out what broke the formatting.

9. Edit the release on GitHub (e.g. https://github.com/veit/cookiecutter-namespace-template/releases). Paste the release notes into the release's release page, and come up with a title for the release.

CHAPTER

TWO

OPTIONS

2.1 Console Script Setup

Optionally, your package can include a console script using Typer, Click or argparse.

2.1.1 How it works

If the `command_line_interface` option is set to `['Typer']`, `['click']` or `['argparse']` during setup, cookiecutter will add a file `cli.py` in the `project_slug` subdirectory.

2.1.2 Usage

To use the console script in development:

```
$ python -m 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

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search