Cookiecutter for Birdhouse Documentation

Release 0.5.0

Birdhouse

Feb 07, 2022

CONTENTS:

A Cookiecutter template for a Birdhouse bird package

Cookiecutter is a command-line utility to create projects from templates. This *cookiecutter-birdhouse* template creates a barebone PyWPS server adhering to Birdhouse conventions. It comes complete with a framework for installation, configuration, deployment, documentation and tests. It even includes a `Dockerfile` for containerization! Create your project then get started writing new WPS processes in minutes.

You may at any time update your project using the latest cookiecutter template using Cruft.

- GitHub repo: https://github.com/bird-house/cookiecutter-birdhouse/
- Documentation: http://cookiecutter-birdhouse.readthedocs.io/en/latest/
- Free software: BSD license

Warning: This is the cookiecutter template for PyWPS *without* the Buildout deployment. The template for the Buildout deployment is on branch 0.2.x.

CHAPTER

ONE

FEATURES

- Ready-made PyWPS server (a bird)
- Pre-configured `.travis.yml` for Travis-CI automated deployment and testing
- Pre-configured `.codacy.yml` for automated Codacy code review
- A `Dockerfile` and `docker-compose.yml` for containerization
- Preconfigured Sphinx documentation that can be hosted on ReadTheDocs
- A `Makefile` to install the code, start, stop and poll the server and more

CHAPTER

TWO

INSTALLATION

Prior to installing cookiecutter-birdhouse, the cookiecutter and cruft packages must be installed in your environment. This is achieved via the following commands:

```
$ conda install -c conda-forge cookiecutter
$ pip install cruft
```

With cookiecutter and cruft installed, the cookiecutter-birdhouse template can be installed with:

```
$ cruft create https://github.com/bird-house/cookiecutter-birdhouse.git
```

Once cookiecutter clones the template, you will be asked a series of questions related to your project:

```
full_name [Full Name]:
email [your@email]:
github_username [bird-house]:
project_name [Babybird]:
project_slug [babybird]:
project_repo_name [babybird]:
project_readthedocs_name [babybird]:
project_short_description [A Web Processing Service for Climate Data Analysis.]:
version [0.1.0]:
Select open_source_license:
1 - Apache Software License 2.0
2 - MIT license
3 - BSD license
4 - ISC license
5 - GNU General Public License v3
Choose from 1, 2, 3, 4, 5 [1]:
http_port [5000]:
```

The answer to all those questions are recorded in the `.cruft.json` file in your generated bird.

CHAPTER

THREE

USAGE

After answering the questions asked during installation, a *bird* Python package will be created in your current working directory. This package will contain a configurable PyWPS service with some initial test processes.

Then:

- Create a repo and put it there.
- Add the repo to your Travis-CI account.
- Add the repo to your ReadTheDocs account + turn on the ReadTheDocs service hook.

For more details, see the cookiecutter-pypackage tutorial.

See the babybird example of a generated bird.

To keep the generated bird up-to-date with the cookiecutter template:

```
$ cruft update  # uses configurations in the .cruft.json file
```

Cruft can be configured to ignore template changes to certain files, see https://timothycrosley.github.io/cruft/#updating-a-project. Potential files to ignore:

- demonstration files, because they are meant to be erased
- environment files and list of processes, list of tutorial notebooks since they naturally are different between each bird

See cruft_skip example.

To link already generated project that was not initially generated using `cruft create`:

```
$ cruft link https://github.com/bird-house/cookiecutter-birdhouse
```

This will create the `.cruft.json` file so subsequently `cruft update` can be used. You will need to answer the same questions as `cruft create` above.

Note that after `cruft link`, the `commit` field in the `.cruft.json` file will initially be wrong if you selected the default value. To ensure a proper subsequent `cruft update`, you need to edit the `.cruft.json` file and put the proper last commit of the cookiecutter used in that `commit` field. See cruft_link example.

CHAPTER

FOUR

DEVELOPMENT

If you want to extend the cookiecutter template then prepare your development environment as follows:

```
# clone repo
$ git clone git@github.com:bird-house/cookiecutter-birdhouse.git

# change into repo
$ cd cookiecutter-birdhouse

# create conda environment
$ conda env create -f environment.yml

# activate conda environment
$ source activate cookiecutter-birdhouse

# run tests
$ make test

# bake a new bird with default settings
$ make bake

# the new "baked" bird is created in the cookies folder
$ ls -l cookies/
babybird

# well ... you know what to do with a bird :)

# finally you may clean it all up
$ make clean
```

CHAPTER

FIVE

BUMP A NEW VERSION

Make a new version of this Cookiecutter in the following steps:

- Make sure everything is commit to GitHub.
- Update `CHANGES.rst` with the next version.
- Dry Run: `bumpversion --dry-run --verbose --new-version 0.3.1 patch`
- Do it: `bumpversion --new-version 0.3.1 patch`
- … or: `bumpversion --new-version 0.4.0 minor`
- Push it: `git push --tags`

See the bumpversion documentation for details.