cookiecutter-python

Release

Apr 08, 2017

Contents

Cookiecutter template for python projects. This template will setup basic structure to start a new `python project`. It can also be used over an `existing project` to generate, for example, your documentation with sphinx.

- Source code on GitLab

CHAPTER 1

Getting Started

Tutorial

This tutorial will get you started in using this template to create a python project.

Step 1: Install Cookiecutter

You can use your system python to start your python project, or use a `virtualenv`. This tutorial will use the `virtualenv` method.

First, you need to create and activate your virtualenv for your python project. Use your favorite method, or create a virtualenv for your project like this:

```
virtualenv -p python2.7 ~/.virtualenvs/your_python_project
```

Here, `your_python_project` is the name of the python project that you'll create.

Note: You can use this template on an existing project !

Activate your environment:

```
source ~/.virtualenvs/serverdoc/bin/activate
```

You can got back anytime to the system python by typing:

```
deactivate
```

On Windows, activate it like this. You may find that using a Command Prompt window works better than gitbash.

```
> \path\to\env\Scripts\activate
```

On both platform, install cookiecutter with pip:

```
pip install cookiecutter
```

Step 2: Generate your project

Now it's time to generate your python project.

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

```
cookiecutter https://gitlab.com/ericdevost/cookiecutter-python.git
```

Note: If you want to use cookiecutter-python over an existing project, you can add the `-f` flag to the above command, meaning to not fail if the project exist.

You'll be asked to enter values to set your project up. If you don't know what to enter, stick with the defaults. You can find a detailed documentation on the values in the *Prompts* section of this documentation.

Step 3: Build your python project locally

Once your project had been set up with cookiecutter, you can install the needed dependencies for the documentation.

```
$ python -m pip install -r requirements/_docs.txt
```

You are now ready to build your documentation locally with one simple command:

```
$ python setup.py docs
```

Your newly built documentation will be accessible in the `build/docs/html` folder.

Step 4: Write your doc

Make any changes, corrections or additions to your documentation. You can build it anytime by typing `python setup.py docs` and see the changes in your browser. Once you are satisfied, it's time to publish your documentation

Step 5: Create a GitLab Repo

Go to your GitLab account and create a new repo i. Name it after `your project name`, where `your project` matches the `project_name` you entered when you ran cookiecutter.

Step 6: Activate your GitLab repo

On your computer, enter your newly created `project folder`, where `project folder` is the `project_name` you entered when you ran cookiecutter, then activate your repository:

```
cd ``project_name``
git init .
git add .
git commit -m "Initial skeleton."
```

```
git remote add origin your-gitla-repo
git push -u origin master
```

Step 8: Build you doc on Read the docs

You can create an account on Read the docs to build your documentation hosted on GitLab.

Troubleshooting

Note: Can you help improve this file? Edit this file and submit a pull request with your improvements!

Windows Issues

- Some people have reported issues using git bash; try using the Command Terminal instead.
- Virtual environments can sometimes be tricky on Windows. If you have Python 3.5 or above installed (recommended), this should get you a virtualenv named `myenv` created inside the current folder:

```
> c:\Python35\python -m venv myenv
```

Or:

```
> c:\Python35\python c:\Python35\Tools\Scripts\pyvenv.py myenv
```

- Some people have reported that they have to re-activate their virtualenv whenever they change directory, so you should remember the path to the virtualenv in case you need it.

module.demonstration_cookie

setup module

Setup script

class `setup.`**`BakedDocumentation`**(*dist*, ***kw*)
Bases: `setuptools.Command`

`description` = **'create documentation distribution from baked project'**

`finalize_options`()

`initialize_options`()

`run`()
Run command.

`user_options` = []

class `setup.`**`Clean`**(*dist*, ***kw*)
Bases: `setuptools.Command`

Custom clean command to tidy up the project root.

finalize_options()

initialize_options()

run()

user_options = []

class setup.**Cookiecutter**(*dist*, ***kw*)
Bases: setuptools.Command

Bake the cookies

description = 'bake the cookiecutter'

finalize_options()
Post-process options.

initialize_options()
Set default values for options.

run()
Run command.

user_options = [('no-input', None, 'Do not prompt for parameters and only use cookiecutter.json file content', ' [defau

class setup.**Documentation**(*dist*, ***kw*)
Bases: setuptools.Command

Bake the cookies

description = 'create documentation distribution'

finalize_options()

initialize_options()
Set default values for options.

run()
Run command.

targets = {'dirhtml': {'comment': 'The HTML pages are in {build_dir!s}.'}, 'latex': {'comment': 'The LaTeX files are

user_options = [('builder=', None, 'documentation output format [default: html]'), ('paper=', None, 'paper format [d

class setup.**Venv**(*dist*, ***kw*)
Bases: setuptools.Command

Setup venvs for development or production

description = 'create a virtualenv pre-installed with dependencies'

finalize_options()
Post-process options.

initialize_options()
Set default values for options.

run()
Run command.

user_options = [('deps=', None, 'path to requirements.txt')]

class setup.**WriteVersionFile**(*dist*, ***kw*)
Bases: setuptools.Command

description = 'Write a Java properties file for use by Jenkins'

finalize_options()
Post-process options.

initialize_options()
Set default values for options.

run()
Run command.

user_options = []

setup.**get_distribution_info**()

setup.**main**()

setup.**working_directory**(**args*, ***kwds*)
A context manager which changes the working directory to the given path, and then changes it back to its previous value on exit.

tests package

Contributing

Contributions to the `cookiecutter-python` template project is highly welcome and encouraged ! You can contribute in many ways.

Report issues

You can report any issues with the project in the source code on GitLab

CHAPTER 2

Basics

Prompts

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

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 documentation, so spaces and any characters are fine here.

project_slug 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.

version The starting version number of the package.

pypi_username Your Python Package Index account username.

Options

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

TODO

CHAPTER 3

Advanced Features

Console Script Setup

Optionally, your package can include a console script

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 setup.py that points to the main function in cli.py.

Usage

To use the console script in development:

```
pip install -e projectdir
```

`projectdir` should be the top level project directory with the setup.py file

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

--help show help menu and exit

Known Issues

Installing the project in a development environment using:

```
python setup.py develop
```

will not set up the entry point correctly. This is a known issue with Click. The following will work as expected:

```
pip install mypackage --editable
```

With 'mypackage' adjusted to the specific project.

More Details

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

What does this cookie taste like?

Todo

Write more about features of the baked project

Documentation

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index