Cuteshop Documentation

Release 0.3.1

Tzu-ping Chung

September 09, 2016

Contents

Contents:

CHAPTER 1

Cuteshop

Package manager for Qt projects.

- Free software: MIT license
- Documentation: https://cuteshop.readthedocs.org.

1.1 What is Cuteshop?

1.1.1 Short Version

It's like CocoaPods, but for Qt (qmake-based) projects, if you know what that means.

1.1.2 Long Version

Cuteshop manages library dependencies for Qt (qmake-based) projects.

You describe your depedencies in a file called `Shopfile`. Cuteshop analyzes it, resolves all the library dependencies for you, and generate boilerplate qmake configurations for you to use.

CHAPTER 2

Installation

It is recommended you use pip to install. If you're using Python 2.7.9 or 3.4, it should be already present, otherwise you can find installing instructions on their manual page.

After you acquired pip, run the folloing in the command line:

```
$ pip install cuteshop
```

Depending on your environment setup, you may need `sudo` to install Cuteshop to your system path. We strongly recommend against it, however, and encourage you to use virtualenv or Python 3.4's new `venv` module to manage it instead. Refer to their respective documentation for more information.

CHAPTER 3

Usage

Cuteshop is mainly designed to be used as an executable. A script called `cuteshop` should already be installed to your `PATH`. To use it, you need to create a `Shopfile` in your project directory:

```
packages:
    - hoedown
```

This is essentially YAML data, with all packages you wish to install listed under that `packages` key.

Now run `cuteshop` inside the project directory (the same that contains `Shopfile`). All dependencies will be fetched into subdirectory `3rdparty`. Cureshop automatically generates a `subdir` project for you; you can then add it as a subproject to build with your main sources. All libraries will be built statically and put into `3rdparty/lib`, while headers go into `3rdparty/include`.

CHAPTER 4

Frequently Askedd Questions

4.1 What should I ignore/include in source control?

Since Cuteshop manages the dependencies for you, it is easy to rebuild things when you move between environments. We recommend you to check-in `Shopfile`, but ignore the whole `3rdparty` directory in your version control system.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/uranusjr/cuteshop/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

Cuteshop could always use more documentation, whether as part of the official Cuteshop docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/uranusjr/cuteshop/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *cuteshop* for local development.

1. Fork the *cuteshop* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/cuteshop.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv cuteshop
$ cd cuteshop/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 cuteshop tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.3, and 3.4. Check https://travis-ci.org/uranusjr/cuteshop/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ nosetests tests.test_cuteshop
```

CHAPTER 6

Credits

6.1 Contributors

- Tzu-ping Chung <uranusjr@gmail.com>

CHAPTER 7

History

7.1 0.3.1 (2016-01-05)

- Git download plan now raises an exception on error.
- Fix error caused by missing spec-source option.

7.2 0.3.0 (2016-01-04)

- Add option for additional spec sources.
- Add support for OTHER_FILES.
- Minor wording fixes.

7.3 0.2.1 (2015-09-20)

- Add support for RESOURCES.
- Add QtHandlebarsJS spec.

7.4 0.2.0 (2015-07-07)

- Add specs for QtSignal and QtYAML.
- Fix implementation for usage on Windows.
- More spec parameters made available.
- More bug fixes.

7.5 0.1.0 (2014-12-16)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search