YourLabs Community Documentation

Release 0.0.0

James Pic & Contributors

April 13, 2015

Contents

Contents:

CHAPTER 1

YourLabs Community Guidelines

This document explains how to maximize your chances of success when interacting with YourLabs community (and Open Source in general), particularely in the context of https://github.com/yourlabs

The purpose of this document is to address the unexpected growth of contributors (apparently we're almost a hundred now !) and to try to reduce the time needed to provide effective support and development for our Open Source apps.

1.1 Asking for help

Read StackOverflow's "How to ask" guide.

Questions about YourLabs apps should be asked on StackOverflow and be tagged at least with:

- django
- python
- [app-name], ie. django-autocomplete-light, django-cities-light, etc, etc ...

You may ping the yourlabs mailing list after posting your question.

1.2 GitHub issues and pull requests: keep it sharp !

Please, make only one issue per topic. If you want:

- some fix in some piece of code **and**,
- some fix in some other piece of code

Then please, make two distinct issues, one per topic. Else, chances are that it will be hard to stay on topic and that appropriate actions are effectively taken for any of the 2 issues.

Same goes for pull requests ... Which are even more dangerous. We're benevolents, we work on our apps at night and usually we're tired after our day of work. If you make it easy for us to make a mistake by merging a PR which addresses several issues, chances that you're making it easy to make a wrong release and affect all the community.

1.2.1 Reporting a bug

Read How to report a bug effectively by Simon Tatham.

Open an issue on GitHub and feel free to use this template:

```
OS Version:
Python version:
Django version:
App version:
Database version (if appliable):
Database driver version (if appliable):
Browser version (if appliable):

How to reproduce this bug:

-
-
-

Excepted result:

Actual result:
```

Bugs are usually fixed quite fast and published in a maintenance release. But it's faster with a pull request of course !

1.2.2 Bugfix pull request

Bugfix pull requests should:

- be focused on the specific issue, and not include anything un-related to the topic,
- hopefully, not break backward compatibility, which means unit tests (travis-ci runs them),
- hopefully, not break PEP8 standards, which means pep8 tests (travis-ci again)

Bugfix pull requests, once merged, cause an immediate maintenance release on PyPi and minor version bump.

1.2.3 Requesting a new feature

Now, this is a really touchy subject. **In general**, YourLabs apps which reach the 1.0.0 version have all the features which it was designed for and no new feature shall be included in the trunk codebase because James is horrified by the idea of having to deal with feature creep. And if you like the "style" of YourLabs apps, it's probably because of that attitude which is largely inspired by the suckless community. So if you'd like to contribute then you should try to get into that state of mind.

1.2.4 Pull requests for new feature

However, as we firmly believe in Open Source, we believe that we should share re-useable code. If it makes sense for a YourLabs app to carry your fine code then it is of course welcome, if it:

- doesn't break backward compatibility,
- is completely optionnal,
- is held in the app's `contrib` module, ie. `rules_light.contrib.your_awesome_feature`.

Do **not** assume that your feature will be accepted directly into the core (outside `contrib`) before it has been discussed and agreed on by a core developer.

You should **not** make a pull request for a new feature **before** it has been discussed on a simple GitHub issue.

By *discussed*, I mean that:

- requirements have been clarified and agreed on,
- then, documentation, have been clarified and agreed on,
- then, unit tests, have been clarified and agreed on.

We try to invent only if necessary, for example: if you want an additionnal interface for your feature to be able to plug in, then **try** to propose new django signals, rather than an API that comes out of nowhere.

1.3 Coding style

Coding style is mostly PEP8 and we require PEP257 docstrings with sphinx. You can see examples all over our code ;)

CHAPTER 2

Template for YourLabs core-dev to reply to unclear issues

```
We could not reproduce your issue. If it's still a problem for you, please
respond to this issue using the following template:


    OS Version:
    Python version:
    Django version:
    App version:
    Database version (if appliable):
    Database driver version (if appliable):
    Browser version (if appliable):

    Traceback (please paste the COMPLETE traceback here):



    Forms (only paste relevant section):



    Models (only paste relevant section):



    autocomplete_light_registry (only paste relevant section):



    javascript / html if relevant:



    How to reproduce this bug:

    -
    -
    -

    Excepted result:



    Actual result:
```

```
If you want a almost garanteed fast answer, then instead of pasting
Models/autocomplete_light_registry/javascript/html above, you could:

- fork the repo,
- reproduct your issue in a new, simple app in `autocomplete_light/example_apps`
- it should be reproducible in `test_project/manage.py runserver`.

See details in community documentation: http://docs.yourlabs.org

Thanks a lot for helping django-autocomplete-light better.

Sincerely

YourLabs core-dev team
```

CHAPTER 3

Release Guidelines

This document explains how to release an app maintained on https://github.com/yourlabs

3.1 `[ci skip]` in commit messages to save Travis-ci

Travis-ci gratefully runs our tests. If a commit doesn't change any code, it should contain `[ci skip]` in the commit message so that it doesn't trigger travis-ci build which takes quite a while - let's be good neighbours.

3.2 `tests/` needs new tests for new bugfix and feature

`tests/` should contain new unit tests for new features and regression tests for bugfixes

Ensure all tests pass on all platforms, use travis-ci for that purpose.

3.3 `docs/` needs love

Ensure all supported feature is documented in the `docs/source/` directory.

3.4 `CHANGELOG` should describe the release changes

Ensure that the `CHANGELOG` file is up to date, use `git log` to double check the consistency between `CHANGELOG` and git.

3.5 `AUTHORS` should be legally valid

Welcome to `AUTHORS`, contributors are listed in alphabetical order, use `git log` to double check the author list since last version bump.

3.6 Version bump: update the version number in the following files

3.6.1 docs/(source/)?conf.py

Update the version number in `docs/(source/)?conf.py`.

3.6.2 setup.py

Update the version number in `setup.py`.

3.6.3 cities_light/__init__.py

If releasing `django-cities-light`, update the `VERSION` variable in `cities_light/__init__.py`.

3.7 Pre-release mandatory tests with GitHub and Travis-ci

It is important that all tests jobs pass in Travis. If a job in the matrix fails then read the logs of the job and eventually restart the job - you'll need to login with your github account on travis for that.

For example, if a test job runs a test server + selenium in a thread then it'll be a bit resource hungry, the job might fail because "Timed out while waiting for window to load". This is a typical case where we'd like to restart a job on travis.

3.7.1 git tag

Tag the release with git tag, ie.:

```
git tag 2.0.4
```

3.7.2 git push origin <version>

Upload the tag, ie:

```
git push origin 2.0.4
```

3.7.3 Wait for Travis

Once all tests pass on travis then it's time to propagate on PyPi.

3.7.4 python setup.py sdist

Build the package:

```
python setup.py sdist
```

3.7.5 Check resulting build/

Check that the package looks ok, it should be in `build/`.

3.7.6 Test it in a virtualenv in `/tmp/testenv`

Might help finding final bugs.

3.7.7 Upload on PyPi `python setup.py sdist upload`

Upload the package on PyPi:

```
python setup.py sdist upload
```

3.8 Thanks for your contribution !

It's beyond my control: "l'outil appartient à celui qui le travaille" means "the tool belongs to whom works it".

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*