django-filer-gallery Documentation

Release 1.0 pre-alpha

Ãyvind Saltvik

October 15, 2016

Contents

This document refers to version 1.0 pre-alpha

CHAPTER 1

Getting Started

1.1 Dependencies

- Django >= 1.2.7
- django-filer >= 0.8.3
- easy-thumbnails >= 1.0-alpha-16
- django-staticfiles or django.contrib.staticfiles

1.2 Installation

Install *django-filer-gallery* from github and *easy-thumbnails* using pip:

```
pip install -e http://github.com/fivethreeo/django-filer-gallery.git#egg=django-filer-gallery
pip install easy-thumbnails==1.0-alpha-16
```

Add the required apps to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    'filer_gallery',
    'easy_thumbnails',
    'filer',
    'django.contrib.staticfiles',
    # 'staticfiles'
    ...
)
```

and then run `syncdb` or `migrate` if you're using South.

CHAPTER 2

Contributing to django-filer-gallery

2.1 Contributing to django-filer-gallery

Like every open-source project, django-filer-gallery is always looking for motivated individuals to contribute to it's source code. However, to ensure the highest code quality and keep the repository nice and tidy, everybody has to follow a few rules (nothing major, I promise :))

2.1.1 In a nutshell

Here's what the contribution process looks like, in a bullet-points fashion, and only for the stuff we host on GitHub:

1. django-filer-gallery is hosted on GitHub, at fork
2. The best method to contribute back is to create an account there, then fork the app. You can use this fork as if it was your own project, and should push your changes to it.
3. When you feel your code is good enough for inclusion, "send a pull request", by using the nice GitHub web interface.

2.1.2 Contributing Code

Getting the source code

- Code *must* be tested. Your pull request should include unit-tests (that cover the piece of code you're submitting, obviously)
- Documentation should reflect your changes if relevant. There is nothing worse than invalid documentation.
- Usually, if unit tests are written, pass, and your change is relevant, then it'll be merged.

Since we're hosted on GitHub, django-filer-gallery uses git as a version control system.

The GitHub help is very well written and will get you started on using git and GitHub in a jiffy. It is an invaluable resource for newbies and old timers alike.

Syntax and conventions

We try to conform to PEP8 as much as possible. A few highlights:

- Indentation should be exactly 4 spaces. Not 2, not 6, not 8. **4**. Also, tabs are evil.
- We try (loosely) to keep the line length at 79 characters. Generally the rule is "it should look good in a terminal-base editor" (eg vim), but we try not be [Godwin's law] about it.

Process

This is how you fix a bug or add a feature:

1. fork on GitHub.
2. Checkout your fork.
3. Hack hack hack, test test test, commit commit commit, test again.
4. Push to your fork.
5. Open a pull request.

Tests

Having a wide and comprehensive library of unit-tests and integration tests is of exceeding importance. Contributing tests is widely regarded as a very prestigious contribution (you're making everybody's future work much easier by doing so). Good karma for you. Cookie points. Maybe even a beer if we meet in person :)

Generally tests should be:

- Unitary (as much as possible). I.E. should test as much as possible only one function/method/class. That's the very definition of unit tests. Integration tests are interesting too obviously, but require more time to maintain since they have a higher probability of breaking.
- Short running. No hard numbers here, but if your one test doubles the time it takes for everybody to run them, it's probably an indication that you're doing it wrong.

In a similar way to code, pull requests will be reviewed before pulling (obviously), and we encourage discussion via code review (everybody learns something this way) or IRC discussions.

Running the tests

To run the tests simply execute `runtests.sh` from your shell. To make sure you have the correct environment you should also provide the `--rebuild-env` flag, but since that makes running the test suite slower, it's disabled by default. You can also see all flags using `--help`.

2.1.3 Contributing Documentation

Perhaps considered "boring" by hard-core coders, documentation is sometimes even more important than code! This is what brings fresh blood to a project, and serves as a reference for old timers. On top of this, documentation is the one area where less technical people can help most - you just need to write a semi-decent English. People need to understand you. We don't care about style or correctness.

Documentation should be:

- We use Sphinx/restructuredText. So obviously this is the format you should use :) File extensions should be .rst.
- Written in English. We can discuss how it would bring more people to the project to have a Klingon translation or anything, but that's a problem we will ask ourselves when we already have a good documentation in English.
- Accessible. You should assume the reader to be moderately familiar with Python and Django, but not anything else. Link to documentation of libraries you use, for example, even if they are "obvious" to you (South is the first example that comes to mind - it's obvious to any Django programmer, but not to any newbie at all). A brief description of what it does is also welcome.

Pulling of documentation is pretty fast and painless. Usually somebody goes over your text and merges it, since there are no "breaks" and that GitHub parses rst files automagically it's really convenient to work with.

Also, contributing to the documentation will earn you great respect from the core developers. You get good karma just like a test contributor, but you get double cookie points. Seriously. You rock.

Section style

We use Python documentation conventions fo section marking:

- # with overline, for parts
- * with overline, for chapters
- =, for sections
- -, for subsections
- ^, for subsubsections
- ", for paragraphs

2.1.4 Translations

For translators we have a Transifex account where you can translate the .po files and don't need to install git or mercurial to be able to contribute. All changes there will be automatically sent to the project.

2.2 Indices and tables

- genindex
- modindex
- search