django-photologue Documentation

Release 2.7

Justin Driscoll/Richard Barran

October 19, 2014

Contents

The Photologue documentation is being transferred from a Google Code wiki to a Sphinx-generated setup (if you're reading this at readthedocs.org, then you're looking at these docs).

This documentation is accurate and up-to-date (hopefully!); the old docs on the Google Code wiki are extensive but have not been updated in a long time.

Contents:

CHAPTER 1

Installation

1.1 Introduction

The easiest way to install Photologue is with pip:

```
pip install django-photologue
```

You can also install the development version which is on Github:

```
git clone git://github.com/jdriscoll/django-photologue.git
cd django-photologue
python setup.py install
```

This code should work ok - like Django itself, we try to keep the master branch bug-free.

1.1.1 Python 3

Photologue works with Python 3 (3.3 or later) and Django >= 1.5. Like Django itself, support for Python 3 can be described as "should work, but needs more time on production sites to prove itself". Use it, but make sure that all features work!

1.2 Dependencies

- Django.
- Pillow.
- South.

These 3 apps will be installed automatically if they are not already there.

Note:

- Pillow can be tricky to install; sometimes it will install smoothly out of the box, sometimes you can spend hours figuring it out - installation issues vary from platform to platform, and from one OS release to the next, so listing them here would not be practical. Google is your friend, and it's worth noting that Pillow is a fork of PIL, so googling 'PIL installation <your platform>' can also help.
- You should not have installed both PIL and Pillow; this can cause strange bugs. Please uninstall PIL before you install Pillow.

- In some situations, you might not be able to use Pillow at all (e.g. if another package has a dependency on PIL). Photologue has a clumsy answer for this: write a temporary file `/tmp/PHOTOLOGUE_NO_PILLOW`, then install Photologue. This will tell Photologue to install without Pillow. It *should* work, but it hasn't been tested!

Note:

- Photologue has the same support policy as Django (last 2 versions released).

Photologue also uses the Django admin app, so enable it if you have not already done so.

1.3 Configure Your Django Settings

1. Add 'photologue' to your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = (
     # ...other installed applications,
     'photologue',
     'south',
)
```

2. Confirm that your MEDIA_ROOT and MEDIA_URL settings are correct (Photologue will store uploaded files in a folder called 'photologue' under your `MEDIA_ROOT`).

1.4 Add the urls

Add photologue to your projects urls.py file:

```
urlpatterns += patterns('',
    ...
    (r'^photologue/', include('photologue.urls')),
)
```

1.5 Sync Your Database

Use South to setup the new tables:

```
python manage.py migrate photologue
```

If you are installing Photologue for the first time, this will set up some default PhotoSizes to get you started - you are free to change them of course!

1.6 Instant Photo Gallery

Photologue comes with basic templates for galleries and photos. You can of course override them, or completely replace them. Note that all Photologue templates inherit from `photologue/root.html`, which itself just inherits from a site-wide `base.html` - you can change this to use a different base template.

1.7 Sitemap

Photologue can be used in your site's sitemap.xml to generate a list of all the Gallery and Photo pages.

To use, add the following to the sitemap definition section of your project's urls.py:

```
...
from photologue.sitemaps import GallerySitemap, PhotoSitemap

sitemaps = {...
            'photologue_galleries': GallerySitemap,
            'photologue_photos': PhotoSitemap,
            ...
            }
etc...
```

Note: There is also a PhotologueSitemap class which combines the above 2 classes, but it will be removed in Photologue 3.0.

CHAPTER 2

Customising and extending

2.1 Extending templates

Photologue comes with a set of basic templates to get you started quickly - you can of course replace them with your own. That said, it is possible to extend the basic templates in your own project and override various blocks, for example to add css classes. Often this will be enough.

The trick to extending the templates is not special to Photologue, it's used in other projects such as Oscar.

First, set up your template configuration as so:

```
TEMPLATE_LOADERS = (
    'django.template.loaders.filesystem.Loader',
    'django.template.loaders.app_directories.Loader',
)

from photologue import PHOTOLOGUE_APP_DIR
TEMPLATE_DIRS = (
    ...other template folders...,
    PHOTOLOGUE_APP_DIR
)
```

The `PHOTOLOGUE_APP_DIR` points to the directory above Photologue's normal templates directory. This means that `path/to/photologue/template.html` can also be reached via `templates/path/to/photologue/template.html`.

For example, to customise `photologue/gallery_list.html`, you can have an implementation like:

```
# Create your own photologue/gallery_list.html
{% extends "templates/photologue/gallery_list.html" %}

... we are now extending the built-in gallery_list.html and we can override
the content blocks that we want to customise ...
```

2.2 Settings

Photologue has several settings to customise behaviour; at present this part of the documentation is unfortunately incomplete.

2.2.1 PHOTOLOGUE_USE_CKEDITOR

Default: `False`

If you have already installed django-ckeditor then you can use to edit the TextArea fields of Gallery and Photo in the admin. Simply set the setting to `True`.

2.2.2 PHOTOLOGUE_GALLERY_PAGINATE_BY

Default: `20`

Number of galleries to display per page for GalleryListView.

2.2.3 PHOTOLOGUE_PHOTO_PAGINATE_BY

Default: `20`

Number of photos to display per page for PhotoListView.

2.2.4 PHOTOLOGUE_GALLERY_LATEST_LIMIT

Default: `None`

Default limit for gallery.latest

2.2.5 PHOTOLOGUE_GALLERY_SAMPLE_SIZE

Default: `5`

Number of random images from the gallery to display.

2.2.6 PHOTOLOGUE_IMAGE_FIELD_MAX_LENGTH

Default: `100`

max_length setting for the ImageModel ImageField

2.2.7 PHOTOLOGUE_SAMPLE_IMAGE_PATH

Default: `os.path.join(os.path.dirname(__file__), 'res', 'sample.jpg'))`

Path to sample image

2.2.8 PHOTOLOGUE_MAXBLOCK

Default: `256 * 2 ** 10`

Modify image file buffer size.

2.2.9 PHOTOLOGUE_DIR

Default: `'photologue'`

The relative path from your `MEDIA_ROOT` setting where Photologue will save image files. If your `MEDIA_ROOT` is set to "/home/user/media", photologue will upload your images to "/home/user/media/photologue"

2.2.10 PHOTOLOGUE_PATH

Default: `None`

Look for user function to define file paths. Specifies a "callable" that takes a model instance and the original uploaded filename and returns a relative path from your `MEDIA_ROOT` that the file will be saved. This function can be set directly.

For example you could use the following code in a util module:

```
# myapp/utils.py:

import os

def get_image_path(instance, filename):
    return os.path.join('path', 'to', 'my', 'files', filename)
```

Then set in settings:

```
# settings.py:

from utils import get_image_path

PHOTOLOGUE_PATH = get_image_path
```

Or instead, pass a string path:

```
# settings.py:

PHOTOLOGUE_PATH = 'myapp.utils.get_image_path'
```

2.3 Third-party contributions

Photologue has a 'contrib' folder that includes some useful tweaks to the base project. At the moment, we have just one contribution:

2.3.1 Bootstrap templates

Replaces the normal templates with a new set that work well with Bootstrap.

To use these, edit your `TEMPLATE_DIRS` setting:

```
from photologue import PHOTOLOGUE_APP_DIR
TEMPLATE_DIRS = (
    ...
    os.path.join(PHOTOLOGUE_APP_DIR, 'contrib/bootstrap/templates'),
    ... other folders containing Photologue templates should come after...
)
```

The templates are incomplete - for example, we are missing templates for date-filtered galleries and photos. Pull requests are welcome!

CHAPTER 3

Contributing to Photologue

Contributions are always very welcome. Even if you have never contributed to an open-source project before - please do not hesitate to offer help. Fixes for typos in the documentation, extra unit tests, etc... are welcome. And look in the issues list for anything tagged "easy_win".

3.1 Example project

Photologue includes an example project to get you quickly ready for contributing to the project - do not hesitate to use it!

3.2 Workflow

Photologue is hosted on Github, so if you have not already done so, read the excellent Github help pages. We try to keep the workflow as simple as possible; most pull requests are merged straight into the master branch.

Features that will take a while to develop might warrant a separate branch in the project; at present only the ImageKit integration project is run on a separate branch.

3.3 Coding style

No surprises here - just try to follow the conventions used by Django itself.

3.4 Unit tests

Including unit tests with your contributions will earn you bonus points, maybe even a beer. So write plenty of tests.

3.5 Documentation

Keeping the documentation up-to-date is very important - so if your code changes how Photologue works, or adds a new feature, please check that the documentation is still accurate, and update it if required.

We use Sphinx to prepare the documentation; please refer to the excellent docs on that site for help.

Note: The CHANGELOG is part of the documentation, so if your patch needs the end user to do something - e.g. run a South migration - don't forget to update it!

3.6 Translations

Photologue manages string translations with Transifex. Contributions are very welcome, either by editing the translations directly on the Transifex site, or by submitting pull requests with updated .po files.

3.7 New features

In the wiki there is a wishlist of new features already planned for Photologue - you are welcome to suggest other useful improvements.

If you're interested in developing a new feature, it is recommended that you first discuss it on the mailing list or open a new ticket in Github, in order to avoid working on a feature that will not get accepted as it is judged to not fit in with the goals of Photologue.

3.7.1 A bit of history

Photologue was started by Justin Driscoll in 2007. He quickly built it into a powerful photo gallery and image processing application, and it became successful.

Justin then moved onto other projects, and no longer had the time required to maintain Photologue - there was only one commit between August 2009 and August 2012, and approximately 70 open tickets on the Google Code project page.

At this point Richard Barran took over as maintainer of the project. First priority was to improve the infrastructure of the project: moving to Github, adding South, Sphinx for documentation, Transifex for translations, Travis for continuous integration, zest.releaser.

The codebase has not changed much so far - and it needs quite a bit of TLC (Tender Loving Care), and new features are waiting to be added. This is where you step in...

3.8 And finally...

Please remember that the maintainer looks after Photologue in his spare time - so it might be a few weeks before your pull request gets looked at... and the pull requests that are nicely formatted, with code, tests and docs included, will always get reviewed first ;-)

CHAPTER 4

Indices and tables

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

p

P