
django-cropduster Documentation

Release 4.8.18

The Atlantic

Aug 02, 2018

Contents

1	Compatibility	3
1.1	Contents	3
1.2	License	10
1.3	Indices and tables	10

django-cropduster is a project that makes a form field available that uses the [Jcrop jQuery plugin](#). It is a drop-in replacement for Django's `ImageField` and allows users to generate multiple crops from images, using predefined sizes and aspect ratios. django-cropduster was created by developers at [The Atlantic](#).

django-cropduster is compatible with python 2.7 and 3.4, and Django versions 1.4 - 1.8.

1.1 Contents

1.1.1 Quick start guide

Django version 1.4–1.8 needs to be installed to use django-cropduster. Installing cropduster should install its dependencies, django-generic-plus, Pillow, and python-xmp-toolkit.

Installation

```
pip install django-cropduster
```

Go to <https://github.com/theatlantic/django-cropduster> if you need to download a package or clone/fork the repository.

Setup

Open `settings.py` and add `cropduster` to your `INSTALLED_APPS`

```
INSTALLED_APPS = (  
    # ...  
    'cropduster',  
)
```

Add URL-patterns:

```
urlpatterns = patterns('',  
    # ...
```

(continues on next page)

(continued from previous page)

```
)
    url(r'^cropduster/', include('cropduster.urls')),
)
```

Collect the static files:

```
$ python manage.py collectstatic
```

Example Usage

Model field

`CropDusterField` takes the same arguments as Django's `ImageField`, as well as the additional keyword arguments `sizes`. The sizes should either be a list of `cropduster.models.Size` objects, or a callable that returns a list of `Size` objects.

```
from cropduster.models import CropDusterField, Size

class ExampleModel(models.Model):

    image = CropDusterField(upload_to="some/path", sizes=[
        Size("main", w=1024, h=768, label="Main", auto=[
            Size("square", w=1000, h=1000),
            Size("main@2x", w=2048, h=1536, required=False),
        ]),
        Size("thumb", w=400, label="Thumbnail"),
        Size("freeform", label="Free-form")])

    second_image = CropDusterField(upload_to="some/path",
        field_identifier="second",
        sizes=[Size("100x100", w=100, h=100)])
```

Given the above model, the user will be prompted to make three crops after uploading an image for field `image`: The first “main” crop would result in a 1024x768 image. It would also generate a 1000x1000 square image (which will be an optimal recropping based on the crop box the user created at the 4/3 aspect ratio) and, optionally, a “retina” crop (“main@2x”) if the source image and user crop are large enough. The second “thumbnail” cropped image would have a width of 400 pixels and a variable height. The third “freeform” crop would permit the user to select any size crop whatsoever.

The field `second_image` passes the keyword argument `field_identifier` to `CropDusterField`. If there is only one `CropDusterField` on a given model then the `field_identifier` argument is unnecessary (it defaults to “”). But if there is more than one `CropDusterField`, `field_identifier` is a required field for the second, third, etc. fields. This is because it allows for a unique generic foreign key lookup to the `cropduster` image database table.

Admin Integration

Adding the `cropduster` widget to the django admin requires no extra work. Simply ensure that the field is included in the `ModelAdmin` class.

Template usage

To get a dictionary containing information about an image within a template, use the `get_crop` templatetag:

```
{% load cropduster_tags %}

{% get_crop obj.image 'large' as img %}

{% if img %}
<figure>
  
  {% if img.attribution %}
  <figcaption>
    {{ img.caption }} (credit: {{ img.attribution }})
  </figcaption>
  {% endif %}
</figure>
{% endif %}
```

Testing

To run the unit tests:

```
DJANGO_SELENIUM_TESTS=1 python manage.py test cropduster
```

1.1.2 Customization

Available Settings

CROPDUSTER_JPEG_QUALITY The value of the `quality` keyword argument passed to Pillow's `save()` method for JPEG files. Can be either a numeric value or a callable which gets the image's width and height as arguments and should return a numeric value.

CROPDUSTER_PREVIEW_WIDTH, CROPDUSTER_PREVIEW_HEIGHT The maximum width and height, respectively, of the preview image shown in the cropduster upload dialog.

CROPDUSTER_GIFSICLE_PATH The full path to gifsicle binary. If this setting is not defined it will search for it in the `PATH`.

1.1.3 How it Works

GenericForeignFileField

Nearly all of the functionality in `cropduster` comes from its django model field, `CropDusterField`. A great deal of functionality, in turn, comes from the `GenericForeignFileField` in the package `django-generic-plus`. Put in simplest terms, `django-generic-plus` allows one to create django model fields that are a hybrid of a `FileField` and a reverse generic foreign key (similar to Django's `GenericRelation`, except that the relationship is one-to-one rather than one-to-many). In some respects these fields act the same as a `FileField` (or, in the case of `django-cropduster`, an `ImageField`), and when they are accessed from a model they have the same API as a `FieldFile`. But, as part of their hybrid status, `GenericForeignFileField` fields also have functionality that allows relating a file to one or more fields in another model. In the case of `django-cropduster`, this model is `cropduster.models.Image`. An example might be edifying. Let's begin with a simple model:

```
class Author(models.Model):
    name = models.CharField(max_length=255)
    headshot = CropDusterField(upload_to='img/authors', sizes=[Size("main")])
```

Assuming that we are dealing with an Author created in the Django admin, one would access the `cropduster.Image` instance using `Author.headshot.related_object`:

```
>>> author = Author.objects.get(pk=1)
>>> author.headshot
<CropDusterImageFieldFile: img/authors/mark-twain/original.jpg>
>>> author.headshot.path
"/www/project/media/img/authors/mark-twain/original.jpg"
>>> author.headshot.related_object
<Image: /media/img/authors/mark-twain/original.jpg>
```

The accessor at `author.headshot.related_object` is basically equivalent to running the following python code:

```
try:
    Image.objects.get(
        content_type=ContentType.objects.get_for_model(author),
        object_id=author.pk,
        field_identifier='')
except Image.DoesNotExist:
    return None
```

Creating an instance with a `cropduster` field outside of the Django admin requires the creation of an instance of `cropduster.Image` and a call to the `generate_thumbs` method:

```
from cropduster.models import Image

author = Author.objects.create(
    name="Mark Twain",
    headshot="img/authors/mark-twain/original.jpg")
author.save()

image = Image.objects.create(
    content_object=author,
    field_identifier='',
    image=author.headshot.name)

author.headshot.generate_thumbs()
```

Note: Cropduster requires that images follow a certain path structure. Let's continue with the example above. Using the built-in Django `ImageField`, uploading the file `mark-twain.jpg` would place it in `img/authors/mark-twain.jpg` (relative to the `MEDIA_ROOT`). Because `cropduster` needs a place to put its thumbnails, it puts all images in a directory and saves the original image to `original.%(ext)s` in that folder. So the `cropduster`-compatible path for `img/authors/mark-twain.jpg` would be `img/authors/mark-twain/original.jpg`. When a file is uploaded via the Django admin this file structure is created seamlessly, but it must be kept in mind when importing an image into `cropduster` from outside of the admin.

1.1.4 Changelog

4.11.13 (Aug 2, 2018)

- Fix Django 1.11 that prevented updating images in standalone mode
- Fix bug that threw `exempi` exceptions when uploaded images had iPhone face-recognition region metadata

4.11.12 (Jul 3, 2018)

- Fix Django 1.11 bug where newly uploaded images weren't named correctly.

4.11.11 (Jun 6, 2018)

- Support Django 2.0 and Django 2.1 alpha

4.11.10 (Jun 6, 2018)

- Fix Django 1.11 bug that prevented save of existing images

4.11.9 (Mar 28, 2018)

- Add `skip_existing` kwarg to `generate_thumbs()` method

4.11.0 (Mar 12, 2017)

- Add support for Django 1.10, drop support for Django < 1.8

4.10.0 (July 26, 2015)

- New: Add `Image.alt_text` field (requires a migration), which also gets returned now in the `{% get_crop %}` templatetag.
- Removed: `exact_size` argument for `get_crop` templatetag. Looking up exact sizes in the database and including the caption/attribution/alt_text is now the default behavior.

4.9.0 (May 13, 2016)

- Fixed: upload and crop views now require admin login

4.8.49 (Apr 14, 2016)

- Fix bugs with `regenerate_thumbs()` when `permissive=True`

4.8.41 (Dec 16, 2015)

- New: Django 1.9 support

4.8.39 (Oct 28, 2015)

- Fixed: bug in `best_fit` calculation where scaling could cause the image dimensions to drop below mins.

4.8.38 (Oct 22, 2015)

- Fixed: Bug where `for_concrete_model` might not be set correctly.

4.8.37 (Sep 28, 2015)

- New: Add ability to retain xmp metadata (if `CROPDUSTER_RETAIN_METADATA = True`)

4.8.36 (Sep 17, 2015)

- Improved: optimized cropduster inline formset with `prefetch_related` on thumbs

4.8.35 (Sep 3, 2015)

- Fixed: Initial migrations in Django 1.8.

4.8.34 (Aug 30, 2015)

- Fixed: The `python-xmp-toolkit` package is now optional.

4.8.32 (Jul 27, 2015)

- Improved: Drag resizing of non-corner handlers in `jCrop` scales in a more sensible way.

4.8.31 (Jul 26, 2015)

- Fixed: Center initial crop when min/max aspect ratio is specified

4.8.30 (Jul 22, 2015)

- Fixed: A bug in updates when CropDusterField is defined on a parent model

4.8.28 (Jul 16, 2015)

- Fixed: CropDusterField kwargs `min_w`, `min_h`, `max_w`, and `max_h` now work as expected.

4.8.26 (Jul 12, 2015)

- Fixed: AttributeError in Django 1.6+ when using custom cropduster formfield
- Fixed: Updated django-generic-plus to fix an issue with multiple CropDusterFields spanning model inheritance.

4.8.25 (Jul 11, 2015)

- Fixed: Orphaned thumbs were being created when cropping images with multiple sizes (issue #41)

4.8.23 (Jun 15, 2015)

- Fixed: Off-by-one rounding bug in `Size.fit_to_crop()`

4.8.22 (Jun 12, 2015)

- Improved: Show help text about minimum image on upload dialog, when applicable.

4.8.19 (Jun 9, 2015)

- Improved: Animated GIFs are now processed by gifsicle if available
- New: Added actual documentation
- New: Add setting `CROPDUSTER_JPEG_QUALITY`; can be numeric or a callable

4.8.18 (Jun 5, 2015)

- Fixed: Non-South migrations in Django 1.7 and 1.8 were broken.
- Improved: Appearance of the cropduster widget in the Django admin without Grappelli.

4.8.17 (May 31, 2015)

- New: Grappelli is no longer required to use django-cropduster.
- Fixed: Python 3 bug in `cropduster.models.Thumb.to_dict()`.

4.8.16 (May 29, 2015)

- New: Django 1.8 compatibility.

4.8.15 (May 5, 2015)

- Fixed: bug where blank `Image.path` prevents image upload.

4.8.14 (Apr 28, 2015)

- Improved: Image dimensions are no longer recalculated on every save.

4.8.13 (Apr 21, 2015)

- Improved: Added cachebusting to `get_crop` templatetag.

4.8.10 (Apr 12, 2015)

- New: Add `required` keyword argument to `Size`, allowing for crops which are only generated if the image and crop dimensions are large enough.

4.8.8 (Apr 10, 2015)

- Improved: Use bicubic downsampling when generating crops with Pillow version $\geq 2.7.0$.
- Improved: Retain ICC color profile when saving image, if Pillow has JPEG ICC support.

4.8.7 (Mar 18, 2015)

- Fixed: `field_identifier` now defaults to empty string, not `None`.
- Fixed: Bug that caused small JPEG crops to be saved at poor quality.

4.8.4 (Mar 5, 2015)

- New: Give cropduster a logo.

4.8.3 (Feb 23, 2015)

- New: Make default JPEG quality vary based on the size of the image; add `get_jpeg_quality` setting that allows for overriding the default JPEG quality.

4.8.0 (Feb 12, 2015)

- New: Django 1.7 compatibility
- New: Add `field_identifier` keyword argument to `CropDusterField`, which allows for multiple `CropDusterField` fields on a single model.
- New: Add unit tests, including Selenium tests.

4.7.6 (Jan 21, 2015)

- Fix: Bug in `CropDusterImageFieldFile.generate_thumbs` method

4.7.5 (Jan 21, 2015)

- New: Add `CropDusterImageFieldFile.generate_thumbs` method, which generates and updates crops for a `CropDusterField`.

4.7.4 (Dec 17, 2014)

- Improved: Height of CKEditor dialog for smaller monitors.
- Improved: Add convenience `@property` helpers: `Thumb.image_file`, `Thumb.url`, `Thumb.path`, and `Image.url`.
- Improved: Use filters passed to `limit_choices_to` keyword argument in `ReverseForeignRelation`.

4.7.3 (Nov 25, 2014)

- Fixed: Regression from 4.7.2 where `get_crop` templatetag did not always return an image.

4.7.1 (Oct 16, 2014)

- Improved: `Image.caption` field no longer has a maximum length.

4.6.4 (Jul 10, 2014)

- Fixed: Querysets of the form `Image.objects.filter(thumbs__x=...)`.
- Improved: Disable “Upload” button before a file has been chosen.
- Fixed: Error in CKEditor widget triggered by user clicking the “OK” button without uploading an image.

4.6.3 (Jul 9, 2014)

- Fixed: Python 3 regression that raised `ValueError` when the form received an empty string for the `thumbs` field.
- Improved: Style and functionality of the delete checkbox.

4.6.2 (Jul 9, 2014)

- Fixed: Deleting a cropduster image did not clear the file field on the generic-related instance, which caused cropduster to subsequently render file widgets in legacy mode.

4.6.1 (Jul 8, 2014)

- Fixed: Bug that prevented CKEditor plugin from downloading external images already existing in WYSIWYG.

4.6.0 (Jul 8, 2014)

- Python 3 compatibility
- Django 1.6 compatibility
- Removed: Dependency on `jsonutils`.
- Improved: Support `python-xmp-toolkit 2.0.0+`.

1.2 License

The django code is licensed under the [Simplified BSD License](#). View the `LICENSE` file under the root directory for complete license and copyright information.

The Jerop jQuery library included is used under the [MIT License](#).

1.3 Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)