ImageKit Documentation

Release 1.1.0

Justin Driscoll, Bryan Veloso, Greg Newman, Chris Drackett & Ma

February 13, 2012

CONTENTS

ImageKit is a Django app that helps you to add variations of uploaded images to your models. These variations are called "specs" and can include things like different sizes (e.g. thumbnails) and black and white versions.

CHAPTER

ONE

INSTALLATION

1. Install PIL or Pillow. If you're using 'ImageField's in Django, you should have already done this.
2. `pip install django-imagekit` (or clone the source and put the imagekit module on your path)
3. Add `'imagekit'` to your `INSTALLED_APPS` list in your project's settings.py

Note: If you've never seen Pillow before, it considers itself a more-frequently updated "friendly" fork of PIL that's compatible with setuptools. As such, it shares the same namespace as PIL does and is a drop-in replacement.

CHAPTER

TWO

ADDING SPECS TO A MODEL

Much like `django.db.models.ImageField`, Specs are defined as properties of a model class:

```
from django.db import models
from imagekit.models import ImageSpec

class Photo(models.Model):
    original_image = models.ImageField(upload_to='photos')
    formatted_image = ImageSpec(image_field='original_image', format='JPEG',
            options={'quality': 90})
```

Accessing the spec through a model instance will create the image and return an ImageFile-like object (just like with a normal `django.db.models.ImageField`):

```
photo = Photo.objects.all()[0]
photo.original_image.url # > '/media/photos/birthday.tiff'
photo.formatted_image.url # > '/media/cache/photos/birthday_formatted_image.jpeg'
```

Check out `imagekit.models.ImageSpec` for more information.

CHAPTER

THREE

PROCESSORS

The real power of ImageKit comes from processors. Processors take an image, do something to it, and return the result. By providing a list of processors to your spec, you can expose different versions of the original image:

```
from django.db import models
from imagekit.models import ImageSpec
from imagekit.processors import resize, Adjust

class Photo(models.Model):
    original_image = models.ImageField(upload_to='photos')
    thumbnail = ImageSpec([Adjust(contrast=1.2, sharpness=1.1),
            resize.Crop(50, 50)], image_field='original_image',
            format='JPEG', options={'quality': 90})
```

The `thumbnail` property will now return a cropped image:

```
photo = Photo.objects.all()[0]
photo.thumbnail.url # > '/media/cache/photos/birthday_thumbnail.jpeg'
photo.thumbnail.width # > 50
photo.original_image.width # > 1000
```

The original image is not modified; `thumbnail` is a new file that is the result of running the `imagekit.processors.resize.Crop` processor on the original.

The `imagekit.processors` module contains processors for many common image manipulations, like resizing, rotating, and color adjustments. However, if they aren't up to the task, you can create your own. All you have to do is implement a `process()` method:

```
class Watermark(object):
    def process(self, image):
        # Code for adding the watermark goes here.
        return image

class Photo(models.Model):
    original_image = models.ImageField(upload_to='photos')
    watermarked_image = ImageSpec([Watermark()], image_field='original_image',
            format='JPEG', options={'quality': 90})
```

CHAPTER

FOUR

ADMIN

ImageKit also contains a class named `imagekit.admin.AdminThumbnail` for displaying specs (or even regular ImageFields) in the Django admin change list. AdminThumbnail is used as a property on Django admin classes:

```
from django.contrib import admin
from imagekit.admin import AdminThumbnail
from .models import Photo


class PhotoAdmin(admin.ModelAdmin):
    list_display = ('__str__', 'admin_thumbnail')
    admin_thumbnail = AdminThumbnail(image_field='thumbnail')


admin.site.register(Photo, PhotoAdmin)
```

AdminThumbnail can even use a custom template. For more information, see `imagekit.admin.AdminThumbnail`.

CHAPTER

FIVE

COMMANDS

CHAPTER

SIX

AUTHORS

ImageKit was originally written by Justin Driscoll.

The field-based API was written by the bright minds at HZDG.

6.1 Maintainers

- Bryan Veloso
- Matthew Tretter
- Chris Drackett
- Greg Newman

6.2 Contributors

- Josh Ourisman
- Jonathan Slenders
- Eric Eldredge
- Chris McKenzie
- Markus Kaiserswerth
- Ryan Bagwell
- Alexander Bohn

CHAPTER

SEVEN

DIGGING DEEPER

7.1 API Reference

7.1.1 `models` Module

7.1.2 `processors` Module

members

7.1.3 `admin` Module

class `imagekit.admin.`**`AdminThumbnail`**(*image_field*, *template=None*)

A convenience utility for adding thumbnails to Django's admin change list.

Parameters

- **image_field** – The name of the ImageField or ImageSpec on the model to use for the thumbnail.
- **template** – The template with which to render the thumbnail

7.2 Changelog

7.2.1 v1.1.0

- A `SmartCrop` resize processor was added. This allows an image to be cropped based on the amount of entropy in the target image's histogram.
- The `quality` argument was removed in favor of an `options` dictionary. This is a more general solution which grants access to PIL's format-specific options (including "quality", "progressive", and "optimize" for JPEGs).
- The `TrimColor` processor was renamed to `TrimBorderColor`.
- The private `_Resize` class has been removed.

7.2.2 v1.0.3

- `ImageSpec._create()` was renamed `ImageSpec.generate()` and is now available in the public API.
- Added an `AutoConvert` processor to encapsulate the transparency handling logic.
- Refactored transparency handling to be smarter, handling a lot more of the situations in which one would convert to or from formats that support transparency.
- Fixed PIL zeroing out files when write mode is enabled.

7.2.3 v1.0.2

- Added this changelog.
- Enhanced extension detection, format detection, and conversion between the two. This eliminates the reliance on an image being loaded into memory beforehand in order to detect said image's extension.
- Fixed a regression from the 0.4.x series in which ImageKit was unable to convert a PNG file in `P` or "palette" mode to JPEG.

7.2.4 v1.0.1

- Minor fixes related to the rendering of `README.rst` as a reStructured text file.
- Fixed the included admin template not being found when ImageKit was and the packaging of the included admin templates.

7.2.5 v1.0

- Initial release of the *new* field-based ImageKit API.

CHAPTER

EIGHT

INDICES AND TABLES

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

PYTHON MODULE INDEX