Django Daguerre Documentation

Release 1.0.1

Stephen Burrows, Harris Lapiroff

September 04, 2014

Contents

Figure 1: Louis Daguerre, Father of Photography

Django Daguerre makes it easy to adjust images on-the-fly without slowing down your templates and without needing to generate everything ahead of time with a cron job. You don't need to make any changes to your models; it **Just Works**.

```
{% load daguerre %}
<img src="{% adjust my_model.image width=200 height=400 %}" />

{% adjust_bulk my_queryset "method.image" width=200 height=400 as adjusted_list %}
{% for my_model, image in adjusted_list %}
  <img src="{{ image }}" />
{% endfor %}
```

Code http://github.com/littleweaver/django-daguerre

Docs http://readthedocs.org/docs/django-daguerre/

Build status

CHAPTER 1

Contents

1.1 Installation and Setup

1.1.1 Requirements

- Python 2.6+
- PIL 1.1.7 (Or Pillow)

1.1.2 Installation

You can install the latest version of Daguerre using `pip`:

```
pip install django-daguerre
```

You can clone the repository yourself at https://github.com/littleweaver/django-daguerre.

1.1.3 Setup

Ensure that `'daguerre'` is in your project's `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    'daguerre',
    ...
)
```

Add the following or similar anywhere in your URLconf:

```
urlpatterns = patterns('',
    url(r'^daguerre/', include('daguerre.urls')),
    ...
)
```

1.2 Using daguerre

1.2.1 {% adjust %}

The easiest way to use Daguerre is through the `{% adjust %}` template tag:

```
{% load daguerre %}
<img src="{% adjust my_model.image width=128 height=256 %}" />
```

daguerre works directly with any ImageField (or storage path). There is no magic. You don't need to change your models. It Just Works.

Let's be lazy

So the {% adjust %} tag renders as a URL, which gets an adjusted image, right? Well, yes, but in a very lazy fashion. It actually renders a URL to an adjustment view, which runs the adjustment (if necessary), and then redirects the user to the actual adjusted image's URL.

The upshot is that no matter how many {% adjust %} tags you have on a page, it will render as quickly when the thumbnails already exist as it will when the thumbnails still need to be created. The thumbnails will then be filled in as the user starts to request them.

Note: The adjustment view has some light security in place to make sure that users can't run arbitrary image resizes on your servers.

Different adjustments

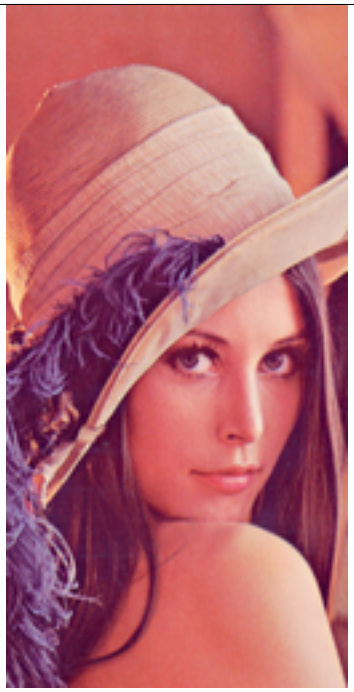
The {% adjust %} tag currently supports three different adjustments: **fit, fill, and crop.** These can be passed in as an additional parameter to the tag:

```
<img src="{% adjust my_model.image width=128 height=256 adjustment="fit" %}" />
```

Take this picture:

Let's use {% adjust %} with width 128 (25%) and height 256 (50%), with each of the three adjustments.

Figure 1.1: Full size: 512x512

"fit"	"fill" (default)	"crop"
		
Fits the entire image into the given dimensions without distorting it.	Fills the entire space given by the dimensions by cropping to the same width/height ratio and then scaling.	Crops the image to the given dimensions without any resizing.

Note: If you have defined `Areas` for an image in the admin, those will be protected as much as possible (according to their priority) when using the crop or fill adjustments. Otherwise, any cropping will be done evenly from opposing sides.

Getting adjusted width and height

```
{% load daguerre %}
{% adjust my_model.image width=128 height=128 adjustment="fit" as image %}
<img src="{{ image }}" width={{ image.width }} height={{ image.height }} />
```

The object being set to the `image` context variable is an `AdjustmentInfoDict` instance. In addition to rendering as the URL for an image, this object provides access to some other useful pieces of information—in particular, the width and height that the adjusted image *will have*, based on the width and height of the original image and the parameters given to the tag. This can help you avoid changes to page flow as adjusted images load.

Named crops (advanced)

If you are defining `Areas` in the admin, you can refer to these by name to pre-crop images **before** applying the adjustment you've selected. For example:

```
{% load daguerre %}
<img src="{% adjust my_model.image width=128 height=128 adjustment="fit" crop="face" %}" />
```

This would first crop the image to the "face" `Area` (if available) and then fit that cropped image into a 128x128 box.

Note: If a named crop is being used, `Areas` will be ignored even if you're using a fill or crop adjustment. (This may

change in the future.)

1.2.2 {% adjust_bulk %}

If you are using a large number of similar adjustments in one template - say, looping over a queryset and adjusting the same attribute each time - you can save yourself queries by using `{% adjust_bulk %}`.

```
{% load daguerre %}
{% adjust_bulk my_queryset "method.image" width=200 height=400 as adjusted_list %}
{% for my_model, image in adjusted_list %}
  <img src="{{ image }}" />
{% endfor %}
```

The syntax is similar to `{% adjust %}`, except that:

- `as <varname>` is required.
- an iterable (`my_queryset`) and an lookup to be performed on each item in the iterable (`"method.image"`) are provided in place of an image file or storage path.
- `{% adjust_bulk %}` **doesn't support named crops**.

1.2.3 Editing Areas

Daguerre provides a widget which can be used with any `ImageField` to edit `Areas` for that image file. Using this widget with a `ModelAdmin` is as simple as defining appropriate formfield_overrides.

```
from daguerre.widgets import AreaWidget

class YourModelAdmin(admin.ModelAdmin):
    formfield_overrides = {
        models.ImageField: {'widget': AreaWidget},
    }
    ...
```

1.3 Management commands

1.3.1 ./manage.py clean_daguerre

Cleans out extra data stored by daguerre:

- `AdjustedImages` and `Areas` that reference storage paths which no longer exist.
- Duplicate `AdjustedImages`.
- Adjusted image files which don't have an associated `AdjustedImage`.

1.3.2 ./manage.py preadjust [--remove]

Looks for a `DAGUERRE_PREADJUSTMENTS` setting using the following syntax:

```
DAGUERRE_PREADJUSTMENTS = {
    ('myapp.MyModel', 'template.style.lookup'): (
        {'adjustment': 'fit',
         'width': 800,
         'height': 500},
        {'adjustment': 'fill',
         'width': 300,
         'height': 216},
        ...
    ),
}
```

In this dictionary, the values are lists of adjustment keyword arguments. The keys are tuples where the first value is either an `'<applabel>.<ModelName>'` string, a model class, or a queryset, and the second value is a template-style lookup which can be applied to each item of the given model type in order to get an `ImageFieldFile` or a storage path.

The command will collect storage paths based on the keys and make `AdjustedImages` for each one based on the list of kwargs, if such an `AdjustedImage` doesn't already exist.

If `--remove` is specified, the command will delete all `AdjustedImage` instances which do not match one of the model/lookup/kwargs combinations specified in `DAGUERRE_PREADJUSTMENTS`.

1.4 Project Details

1.4.1 Authors

Django Daguerre is primarily the work of Stephen Burrows, with contributions from Harris Lapiroff. It was originally written for Oberlin College in Oberlin, Ohio.

1.4.2 Contributing

Django Daguerre is a still-nascent project and greatly appreciates any contributions—code, tests, documentation, bug reports, feature requests, & al.

Development is done through the official GitHub repository. You may fork the repository and issue pull requests. You may also add bugs and feature requests under Issues.

1.4.3 News

August 22, 2012

This is the first official release of **Django Daguerre**, version 0.1!

CHAPTER 2

API Docs

2.1 Adjustments

Daguerre provides a variety of adjustments to use when processing images.

2.1.1 Built-In Adjustments

When used with the template tag, these adjustments should be referred to by their lowercase name:

```
{% adjust adjustment="fit" width=300 %}
```

See *Using daguerre* for examples.

2.2 Models

CHAPTER 3

Indices and tables

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

d