Django-rgallery Documentation

Release 0.0.1

Óscar M. Lage

April 10, 2015

Contents

It's just another django gallery application. Get ready to store your photos and organize them by tags and/or folders.

CHAPTER 1

Contents

1.1 Quickstart

1.1.1 Installation

- Install Django-rgallery with your favorite Python package manager, `pip` will install the dependencies too (compressor, sorl.thumbnail, taggit, etc...):

  ```
  pip install django-rgallery
  ```

- Add `'compressor'`, `'sorl.thumbnail'`, `'taggit'` and `'rgallery'` to your `INSTALLED_APPS` setting:

  ```
  INSTALLED_APPS = (
      # rgallery-dependencies
      'compressor',
      'taggit',
      'sorl.thumbnail',
      'rgallery',
  )
  ```

- Add a proper url for the application in your project urls.py:

  ```
  url(r'^gallery/', include('rgallery.urls',
                            namespace='rgallery',
                            app_name='rgallery')),
  ```

  You probably wants to add this urls too just in case you're using `debug_toolbar` and serving media elements in a development enviroment:

  ```
  if settings.DEBUG:
      import debug_toolbar
      urlpatterns += patterns(
          '',
          url(r'^__debug__/', include(debug_toolbar.urls)),
          url(r'^media/(?P<path>.*)$', 'django.views.static.serve',
              {'document_root': settings.MEDIA_ROOT, }),
      )
  ```

- Execute a `syncdb` command in your django project:

  ```
  ./manage.py syncdb
  ```

- See the list of *Settings* to modify Django Compressor's default behaviour and make adjustments for your website.

1.1.2 Dependencies

Other required packages

In case you're installing Django-rgallery differently (e.g. directly from the repo), make sure to attach it in develop mode:

```
python setup.py develop
```

This ensures that you install all the required dependencies.

Template dependencies

Django-rgallery templates have some functionality that depends on software like jquery or dropzonejs. You have to install and link them from your django project `site_base.html` template. The static files dependencies are the following:

- jquery
- bootstrap
- fontawesome
- mediaelementjs
- colorbox
- dropzonejs
- select2
- jquery-cookie

At this point, a good recommendation to work with static files as packages for install, update new versions... is bower. You can just install `bower` in your system, create a `bower.json` a `.bowerrc` files like the following ones and type a:

```
bower install
```

bower.json

```
{
  "name": "my-project",
  "version": "1.0.0",
  "dependencies": {
    "jquery": "~1.9.1",
    "bootstrap": "~3.0.3",
    "font-awesome": "~4.0.3",
    "mediaelement": "~2.13.1",
    "jquery-colorbox": "~1.4.26",
    "dropzone": "3.10.2",
    "select2": "3.5.2",
    "jquery-cookie": "~1.4.1"
  }
}
```

.bowerrc

In the `.bowerrc` file we can configure where to store the bower installation, in this case `project/static/vendors` will do the job:

```
{
    "directory" : "project/static/vendors"
}
```

site_base.html

We have to put all the css/js referencies and links in our project `site_base.html`, let me show you a sample:

```
{% load staticfiles %}
<link rel="stylesheet" href="{% static 'vendors/font-awesome/css/font-awesome.min.css' %}">
<link rel="stylesheet" href="{% static 'vendors/mediaelement/build/mediaelementplayer.min.css' %}">
<link rel="stylesheet" href="{% static 'vendors/jquery-colorbox/example3/colorbox.css' %}">
<link rel="stylesheet" href="{% static 'vendors/dropzone/downloads/css/dropzone.css' %}">
<link rel="stylesheet" href="{% static 'vendors/select2/select2.css' %}">
<link rel="stylesheet" href="{% static 'vendors/select2/select2-bootstrap.css' %}">
....
<script type="text/javascript" src="{% static 'vendors/jquery/jquery.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/transition.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/alert.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/modal.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/dropdown.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/scrollspy.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/tab.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/tooltip.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/popover.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/button.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/collapse.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/carousel.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/bootstrap/js/affix.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/jquery-colorbox/jquery.colorbox.js' %}"></scri
<script type="text/javascript" src="{% static 'vendors/mediaelement/build/mediaelement-and-player.mi
<script type="text/javascript" src="{% static 'vendors/dropzone/downloads/dropzone.min.js' %}"></scri
<script type="text/javascript" src="{% static 'vendors/select2/select2.min.js' %}"></script>
<script type="text/javascript" src="{% static 'vendors/jquery-cookie/jquery.cookie.js' %}"></script>
```

Optionally you can add your own custom css/less/js files:

```
<link rel="stylesheet" href="{% static 'less/custom.less' %}" type="text/less">
...
<script type="text/javascript" src="{% static 'js/custom.js' %}"></script>
```

1.2 Settings

Django-rgallery has a number of settings that control its behavior. They've been given sensible defaults.

1.2.1 Base settings

`django.conf.settings.`**`GALLERY_DESCRIPTION`**

Example `'This is my new gallery'`

String that describes your gallery

`django.conf.settings.`**GALLERY_KEYWORDS**

Example `'photos, moblog, photography, photoblog, shots'`

String with comma separated keywords that describes your gallery

`django.conf.settings.`**DROPBOX_APP_KEY**

Example `d232fg3f234d32`

Just in case you want to use dropbox as backend, we need the APP_KEY that dropbox gives to you when you create your app there.

`django.conf.settings.`**DROPBOX_APP_SECRET**

Example `fn5njgkl2n5gnb5`

Just in case you want to use dropbox as backend, we need the APP_SECRET that dropbox gives to you when you create your app there.

`django.conf.settings.`**DROPBOX_ACCESS_TYPE**

Example `app_folder` or `dropbox`

If the access is `dropbox` you need to specify the path, if it's `app_folder` it is ok with `/`

`django.conf.settings.`**DROPBOX_ACCESS_PATH**

Example `/` or `custom_path/`

The path where the photos are stored in Dropbox, to export from there to our rgallery instance.

`django.conf.settings.`**RGALLLERY_THUMBS**

Example `[60, 200, 750, 1000]`

The list of sizes you want to make thumbnail, it mainly depends on the use of the photo you will make in template.

`django.conf.settings.`**FFPROBE**

Example `/opt/local/bin/ffprobe`

The path to `ffprobe` command in your system.

`django.conf.settings.`**FFMPEG**

Example `/opt/local/bin/ffmpeg`

The path to `ffmpeg` command in your system.

`django.conf.settings.`**FFMPEG_VCODEC_THUMB**

Example `png`

The photo (from video) format.

`django.conf.settings.`**FFMPEG_THUMB_SIZE**

Example `444x250`

The photo (from video) thumb size.

`django.conf.settings.`**FFMPEG_THUMB_SIZE_INVERSE**

Example `250x444`

The photo (from video) thumb inversed size.

1.2.2 Settings example

```
########## DJANGO-RGALLERY
GALLERY_DESCRIPTION = 'Moblog, day by day photos, shots usually from mobile'
GALLERY_KEYWORDS = 'photo, fotoblog, daily photos, moblog, photoblog, shots'
DROPBOX_APP_KEY = '123d32d23dwfqwf'
DROPBOX_APP_SECRET = 'b567brb45n45n'
DROPBOX_ACCESS_TYPE = 'app_folder'
DROPBOX_ACCESS_PATH = '/'
RGALLLERY_THUMBS = [60, 200, 750, 1000]
FFPROBE = '/opt/local/bin/ffprobe'
FFMPEG = '/opt/local/bin/ffmpeg'
FFMPEG_VCODEC_THUMB = 'png'
FFMPEG_THUMB_SIZE = '444x250'
FFMPEG_THUMB_SIZE_INVERSE = '250x444'
########## DJANGO-RGALLERY
```

1.3 Backends

In some cases it's useful to use a kind of backend as dropbox, but in other cases something like a local dir where the photos are stored in or a form in the web could be great sources too.

1.3.1 Dropbox

The dropbox backend connects to the dropbox api to download and save the photos from a proper folder to Django-rgallery. You can activate it by enabling custom dropbox *Settings* and running the `mediasync` custom command (*Commands*) with the proper `--storage=dropbox` option:

```
python manage.py mediasync --storage=dropbox
```

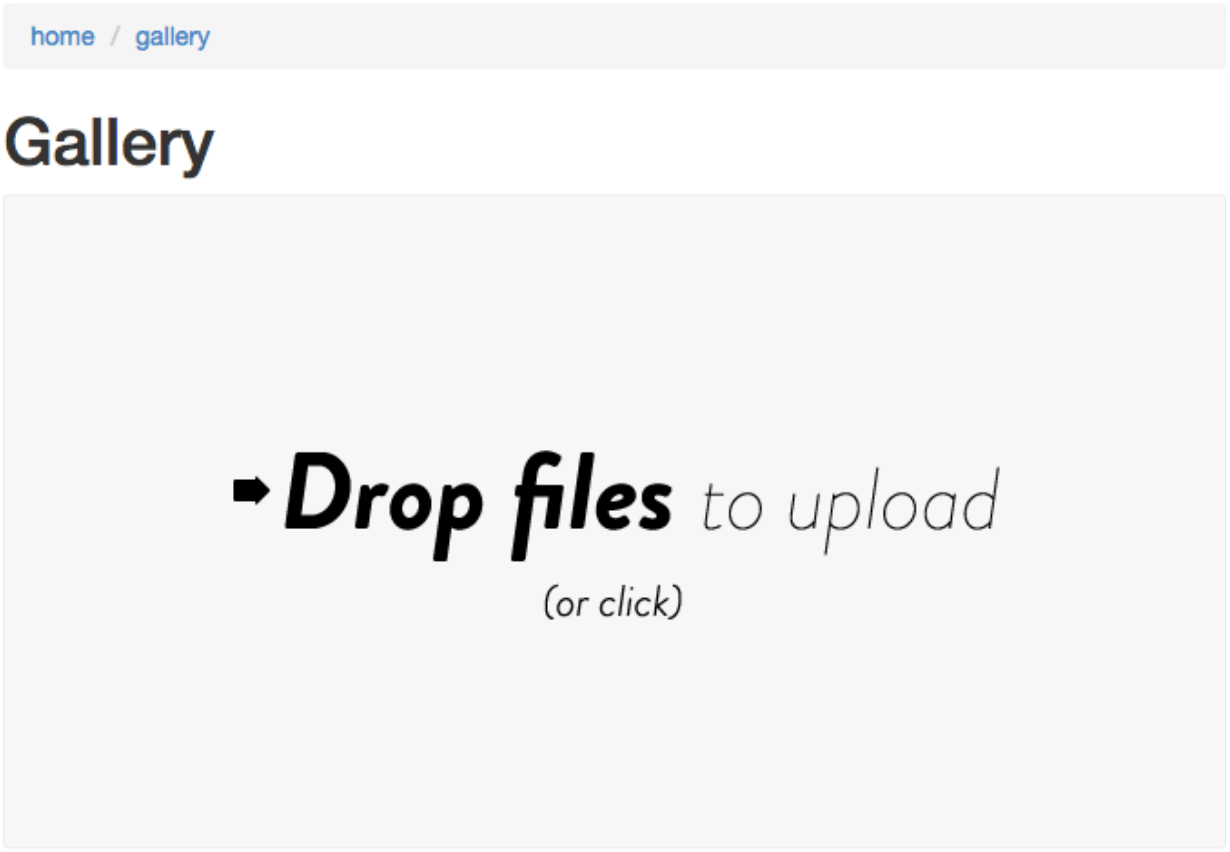
1.3.2 File

The file backend connects to a local folder and download all the photos there, saving them into Django-rgallery. You can activate it by running custom `mediasync` command (*Commands*) with the proper `--storage=file` and `--source=/path/to/photos` options:

```
python manage.py mediasync --storage=file --source=/path/to/photos
```

1.3.3 Form

The form backend is used to save photos from the DropzoneJS input. If you're a site administrator it will appears in your Django-gallery index page:



1.4 Commands

Django-rgallery has some custom commands to ease tasks as sync the backends or make proper thumbs.

1.4.1 mediasync

The `mediasync` custom command allows you to synchronize from a backend or storage option to Django-rgallery. It's the main key of this project, tries to download and parse photos from a source bucket, saving it on the database and converting the photos to fit the web. This command has some interesting arguments:

`--storage` Either `dropbox` or `file`. The `dropbox` one tries to connect with the dropbox api credentials and retrieve the photos from the app_folder, please check *Settings* for more information:

```
./manage.py mediasync --storage=dropbox
./manage.py mediasync --storage=file --source=/path/to/photos
```

`--source` `/path/to/photos`, only valid when `--storage=file`. This argument say to Django-rgallery where to retrieve the photos locally, it should be a local folder:

```
./manage.py mediasync --storage=file --source=/path/to/photos
```

`--tags` `comma,separated,tags`. If you want to add some photos with a custom tag you can specify it with this argument:

```
./manage.py mediasync --storage=file --source=/photos --tags=tag1,tag2
```

--thumbs Either `yes` or `no` (default). If you want to create thumbs of the new photos you can specify it with this argument:

```
./manage.py mediasync --storage=file --source=/photos --thumbs=yes
```

1.4.2 mkthumb

The `mkthumb` command allows you to create missing thumbs. Probably sometimes you get some photo with no thumb caused to a bad upload, connection issues or whatever. You can rebuild thumbs with this command:

```
./manage.py mkthumb
```

1.5 Contributing

Like every open-source project, Django-rgallery is always looking for motivated individuals to contribute to it's source code.

1.5.1 Community

People interested in developing for the Django-rgallery should head over to #codigo23 on the **'freenode'_** IRC network for help and to discuss the development.

You may also be interested in following @oscarmlage on Twitter.

Note: This very document is based on the contributing docs of the django-compressor project. Many thanks for allowing us to steal it!

1.6 Changelog

1.7 TODO

- Static files + bower (here or in project, i.e. fontawesome for icons).
- Javascript in template (removeSelection(), ajaxChangeStatus()...).
- Copy metadata when photo is added via form.
- Test that photo uploads properly now that we're using Pillow instead PIL.
- Create custom commands documentation
- Add a button to allow select/deselect all the photos in the page.
- Check cache (#expire_view_cache("app_gallery-gallery")).
- Change THUMBNAIL_DEBUG = False to True (in project) and see why it's not properly uploading the photo.
- Activate folders via config.
- Create documentation about installing rgallery in a django project from scratch.

D

F

G

R