
imagestore Documentation

Release 0

Pavel Zhukov

Oct 07, 2017

Contents

1	Contents:	1
1.1	Installation	1
1.2	Available settings	2
1.3	Extending Imagestore	3
1.4	Example project	3
1.5	Django CMS Integration	3
1.6	Watermarking	4
2	Indices and tables	5

Installation

- Install with pip or easy install (All dependencies will be installed automatically, however if you use Python 3 you may need to install specific versions of `sorl-thumbnail` and `django-autocomplete-light`):

```
pip install imagestore
```

- If you use django before version 1.7 we recommend to install south for smooth migrations:

```
pip install south
```

- Symlink or copy `imagestore/static/imagestore.css` to your `MEDIA_ROOT`, or write your own style (staticfiles supported as well).
- Add `imagestore`, `django-tagging` and `sorl.thumbnail` to your `INSTALLED_APPS`. your `INSTALLED_APPS` should look like:

```
INSTALLED_APPS = (  
    ....  
    'imagestore',  
    'sorl.thumbnail',  
    'tagging',  
)
```

- Add `imagestore.urls` to your urls with `namespace='imagestore'`:

```
urlpatterns = patterns('',  
    .....  
    (r'^gallery/', include('imagestore.urls', namespace='imagestore')),  
    .....  
)
```

- Set `IMAGESTORE_SHOW_USER` to `False`, if you don't want to show information from user profile or have no user profile.

- Run:

```
./manage.py migrate
```

- Add jquery and jqueryui load to your template to use tagging autocomplete and/or prettyphoto
- If you want to use prettyPhoto put `prettyPhoto` to your media directory and include `imagesotore/prettyphoto.html` to your template

Available settings

IMAGESTORE_UPLOAD_TO (“**imagestore/**”) Path for uploading images

IMAGESTORE_IMAGES_ON_PAGE (**20**) Number of images in one page (album/user/tag view)

IMAGESTORE_ON_PAGE (**20**) Number of albums on page (index view)

IMAGESTORE_SELF_MANAGE (**True**) If true, imagestore install handler on launch, that grant add/change/delete permissions for Album and Image models for every created user (with this permissions users can create personal galleries, if you don’t want it set this settings to False).

IMAGESTORE_TEMPLATE (“**base.html**”) Here you can set template that imagestore templates will inherit. Imagestore templates expect next blocks in basic template:

- head (inside <head> tag for scripts and styles inserting)
- title (inside <title> tag)
- breadcrumb
- content (main content)
- content-related (this block used for tag-cloud, user info and create/edit links)

IMAGESTORE_SHOW_USER (**True**) Show user info (such as avatar, link to profile and other stuff) Default template expects that profile has avatar ImageField and `get_absolute_url` method You can customize view it by overriding `imagestore/user_info.html` template

Notice, that since imagestore version 2.7.4, which supports custom user model, in `imagestore/user_info.html` passes `user` variable with current logged in user.

IMAGE_MODEL (“**imagestore.models.Image**”) Class for storing images. See [extending imagestore](#) for details.

ALBUM_MODEL (“**imagestore.models.Album**”) Class for storing albums. See [extending imagestore](#) for details.

IMAGESTORE_IMAGE_FORM (“**imagestore.forms.ImageForm**”) Form for uploading images. See [extending imagestore](#) for details.

IMAGESTORE_ALBUM_FORM (“**imagestore.forms.AlbumForm**”) Form for creating albums. See [extending imagestore](#) for details.

IMAGESTORE_LOAD_CSS (“**True**”) Load CSS file ‘static/imagestore.css’ in imagestore templates. If you want to use custom theme - disable this settings.

IMAGESTORE_UPLOAD_ALBUM_PROCESSOR (“**imagestore.models.upload.process_zipfile**”) Function for processing uploaded zip archives from admin interface. Function gets `AlbumUpload` model instance and should process file from `zip_file` field to upload images. For example, you can override this setting to provide function, which do nothing, and process file lately

IMAGESTORE_BRIEF_TO_ALT_TEMPLATE (“**{0}_{1}**”) There is template tag `imagestore_alt` which automatically generates images alt attribute based on image title or, if title is empty, on album brief field and (optional) loop counter. Setting determines alt attribute format when brief (`{0}`) and counter (`{1}`) are used.

Extending Imagestore

You can extend imagestore by customizing Image and Album classes as well as forms for their creation. Basic abstract classes that require to extend exists in `models.bases.image` and `models.bases.album`.

After you create your classes, tell imagestore to use them by setting next settings:

- `IMAGESTORE_IMAGE_MODEL`
- `IMAGESTORE_ALBUM_MODEL`
- `IMAGESTORE_IMAGE_FORM`
- `IMAGESTORE_ALBUM_FORM`

You should set model-related settings to `app_label.model_name` string. For example:

```
IMAGESTORE_IMAGE_MODEL = 'mystoreapp.MyImage'
IMAGESTORE_ALBUM_MODEL = 'mystoreapp.MyAlbum'
```

You should set form-related settings as string with python path to required class. For example:

```
IMAGESTORE_IMAGE_FORM = 'mystoreapp.forms.MyImageForm'
IMAGESTORE_ALBUM_FORM = 'mystoreapp.forms.MyAlbumForm'
```

Internally imagestore uses [django-swappable-models](#) reusable app. So you can read their docs to know how to use it correctly.

Warning

- It is required to add `app_label` to your Image and Album models.
- Migrations with swappable models tested only with django migrations. Use it with south with caution.

Migrating from old versions (before 2.9.0)

- Before imagestore v.2.9.0 you have to set model-related settings to full python path to model class. Now it should be in `app_label.model_class` form.
- As now imagestore uses `django-swappable-models` app for swapping Album and Image models you should use swapper's methods for importing or referencing to imagestore models. For more info look at [django-swappable-models docs](#)

Example project

Package contains preconfigured django project with installed imagestore. You can find it in `test` directory inside project root, or get from [repository](#)

Django CMS Integration

Imagestore can show an album as a plugin in django-cms, and can be used as a django-cms app.

To use plugins, just add `imagestore.imagestore_cms` to your `INSTALLED_APPS`

If you want to use imagestore as a django-cms application

- Set `IMAGESTORE_SHOW_USER` to `False`
- Because django-cms build connect apps without namespace settings you need to tell django where to search imagestore namespace, you can do it by adding django-cms urls with the 'imagestore' namespace:

```
url(r'^$', include('cms.urls')),  
url(r'^$', include('cms.urls', namespace='imagestore'))
```

Watermarking

Use [watermarker](#) sorl integration to add watermark to your images

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`