django-flickr Documentation

Release 0.3.0

Jakub Zalewski

June 19, 2013

CONTENTS

Django-Flickr provides a mechanism to mirror user's Flickr photos into a Django project.

This project lets developers use the Flickr API but is not endorsed or certified by Flickr. It is advised to read Flickr APIs Terms of Use before building and deploying applications.

CHAPTER

ONE

PROJECT LINKS

Source

- builds: http://pypi.python.org/pypi/django-flickr/
- repository: https://bitbucket.org/zalew/django-flickr/

Meta

- docs: http://readthedocs.org/docs/django-flickr/en/latest/
- bugs, feature requests: https://bitbucket.org/zalew/django-flickr/issues
- roadmap, release log: https://bitbucket.org/zalew/django-flickr/wiki

CHAPTER

TWO

CONTENTS

2.1 Installation

2.1.1 Using PIP

The latest stable-ish build:

```
pip install django-flickr
```

Development version:

```
pip install hg+https://bitbucket.org/zalew/django-flickr
```

2.1.2 From source

```
hg clone https://bitbucket.org/zalew/django-flickr
cd django-flickr
./setup.py install
```

2.1.3 Dependencies

- Django 1.4
- Bunch - to ease up managing json data
- Taggit - for handling tags
- Taggit-templatetags - not obligatory really, but it's an awesome addition to taggit and the example view uses it
- Oauth2 - for auth

2.1.4 Configuration

Add 'flickr' to your INSTALLED_APPS and syncdb

Go to 'Your apps' on Flickr and generate an API key for your app. Put those data in your settings.py:

```
FLICKR_KEY = 'xxxxxxxxxxx'
FLICKR_SECRET = 'xxxxxxx'
FLICKR_PERMS = 'read'
```

and add flickr.urls to your urls.py

2.1.5 Authenticate your Flickr account

Run your django app, sign in. Go to /flickr/auth/ and click on the link. Authorize your app on flickr. If your app is public, you'll be redirected to complete the auth process, if not (f.ex. you set up some bogus link), just copy the GET frob variables to your /flickr/auth url. Currently it uses FlickrAuth (set to be deprecated some day probably), not Oauth. (Oauth support is implemented, but untested well. You can try it.)

http://blog.flickr.net/en/2012/01/13/start-the-new-year-fresh/

> The Flickr API fully supports oAuth, which is already used by hundreds of Flickr apps, but there are still some apps that use the old authentication system, called FlickrAuth. We are asking all developers to move over to the new standard bu July 31. There is a related blog post on this topic on our engineering blog code.flickr.com.

2.1.6 Upgrading

South is used for migrations. Project is in early stages so expect them.

2.2 Usage

2.2.1 Introduction

How it works

- Flickr API is hit only for syncing via management commands.
- You can choose if you want to sync only crucial photo information (1 API call per page of X photos) or fetch additional information (increases the number of API calls and execution time).
- The application works with data downloaded from Flickr and saved to the project's database. No API call is made when selecting and viewing data.
- Displayed photos are hotlinked from Flickr, but you can download the original photos and access/process them locally if you wish.

2.2.2 Management Commands

Sync photos database

```
./manage.py flickr_sync [options]

-u USER_ID, --user=USER_ID
                      Sync for a particular user. Default is 1 (in most
                      cases it's the admin and you're using it only for
                      yourself).
-i, --info            Fetch info for photos. It will take a long time to
                      sync as it needs to fetch Flickr data for every photo
                      separately.
-e, --exif            Fetch exif for photos. It will take a long time to
                      sync as it needs to fetch Flickr data for every photo
```

```
                      separately.
-s, --sizes           Fetch sizes details for photos. It is not needed,
                      sizes can be obtained dynanmically. It will take a
                      long time as it needs to fetch Flickr data for every
                      photo separately.
-g, --geo             Fetch geo data for photos. It will take a long time as
                      it needs to fetch Flickr data for every photo
                      separately.
-p, --photosets       Sync photosets. Photos must be synced first. If photo
                      from photoset not in our db, it will be ommited.
-c, --collections     Sync collections. Photos and sets must be synced
                      first.
--no-photos           Don't sync photos.
--update-photos       Update outdated photos. It will take a long time as
                      it needs to call Flickr api several times per photo.
--update-photos       Update tags in photos.
-d DAYS, --days=DAYS  Sync photos from the last n days.
--page=PAGE           Grab a specific portion of photos. To be used with
                      --per_page.
--per-page=PER_PAGE   How many photos per page should we grab? Set low value
                      (10-50) for daily/weekly updates so there is less to
                      parse, set high value (200-500) for initial sync and
                      big updates so we hit flickr less.
--ils                 Ignore last_sync.
--initial             It assumpts db flickr tables are empty and blindly
                      hits create().
-t, --test            Test/simulate. Don't write results to db.
```

Download photos

```
./manage.py flickr_download [options]

-u USER_ID, --user=USER_ID
                      Sync for a particular user. Default is 1 (in most
                      cases it's the admin and you're using it only for
                      yourself).
-a, --all             By default downloads only photos which have not been
                      downloaded (default behavior). Use this option to
                      (re)download all.
-p, --public          Only public photos.
-s, --size            Specify size for download (by default original for pro
                      accounts and large for non-pro).
-r, --reset           Clear downloads db table. Does not affect your files.
```

Photos are downloaded under your MEDIA folder. Default settings you can override:

```
# default settings
FLICKR_DOWNLOAD_DIRBASE = 'flickr' # under MEDIA_ROOT
FLICKR_DOWNLOAD_DIRFORMAT = '%Y/%Y-%m' # Photo.date_posted
# photos with date_posted January 2009 will land in /media/flickr/2009/2009-01/

# example custom settings
FLICKR_DOWNLOAD_DIRBASE = 'downloads'
FLICKR_DOWNLOAD_DIRFORMAT = '%Y/%m/%d'
# photos with date_posted 23 January 2009 will land in /media/downloads/2009/01/23/ etc.
```

2.2.3 Models

```
from flickr.models import FlickrUser, Photo, PhotoSet, Collection
```

Basics

Every model (except FlickrUser) is based on FlickrModel

```
class FlickrModel(models.Model):
    flickr_id = models.CharField(unique=True, db_index=True, max_length=50)
    user = models.ForeignKey(FlickrUser)
    show = models.BooleanField(default=True) #show the photo on your page?
    last_sync = models.DateTimeField(auto_now=True, auto_now_add=True)

    class Meta:
        abstract = True
```

Every object belongs to a FlickrUser which is mapped to a Django User

```
class FlickrUser(models.Model):
    user = models.OneToOneField(User)
    flickr_id = models.CharField(max_length=50, null=True, blank=True)
    nsid = models.CharField(max_length=32, null=True, blank=True)
    # ---- / more fields / -----
    token = models.CharField(max_length=128, null=True, blank=True) # authed
    perms = models.CharField(max_length=32, null=True, blank=True) # flickr permissions
    last_sync = models.DateTimeField(auto_now=True, auto_now_add=True)
```

Photo

Selecting photos

```
# all photos
Photo.objects.all()

# only public photos
Photo.objects.public()

# only the ones with show=True
#(default True, you can hide photos from viewing on your website by setting it False
Photo.objects.visible()
```

Photo properties

Accessing properties of each photo is independent of the way you used to sync them (check options in *Management Commands*), although some attributes may not be available if you didn't sync your photos with the corresponding options. The syntax is always the same:

```
(photo_object).(size_label).(property)

size_label       - square: Square (75 x 75 pixels)
                 - largesquare: Large Square (150 x 150 pixels)
                 - thumb: Thumbnail (100 px on longest side)
                 - small: Small (240 px on longest side)
```

```
                  - small320: Small 320 (320 px on longest side)
                  - medium: Medium (500 px on longest side)
                  - medium640: Medium 640 (640 px on longest side)
                  - medium800: Medium 800 (800 px on longest side)
                  - large: Large (1024 px on longest side)
                  - large1600: Large 1600 (1600 px on longest side)
                  - large2048: Large 2048 (2048 px on longest side)
                  - ori: Original (original size)

property          - source: url to image source.
                  - url: url to web page.
                  - width: width in pixels.
                  - height: height in pixels.
```

Photo source and **photo url web page** are either retrieved from the synced data in the database (if `--sizes` option was used) or dynamically generated (according to Flickr docs, so this will always return a valid url for all **web sizes** (see issue #20).

```
p = Photo.objects.get(id=123)
p.large.source            # Image source url for large size.
p.square.source           # source url for square image (75x75)...
```

Special sizes large 1600 and large 2048 are only available if used `--sizes` option while syncing and **original** will only be available for flickr pro accounts.

```
p.ori.url                 # Url to web page for ori image.
p.large2048.url           # Url to web page for ori image.
```

Photo **width** and **height** will only be available if `flickr_sync` was called with the `--sizes` option.

```
p.ori.height              # Height of the original photo
p.medium640.width         # Width for medium 640 size.
```

Some useful features

```
p = Photo.objects.get(id=123)
p.get_next() # next photo in order like on Flickr
p.get_prev() # previous photo

# link to the Flickr page. Works with every supported object: FlickrUser, Photo, Photoset, Collection
p.flickr_page_url
```

Photoset

```
photoset = Photoset.objects.get(id=123)
photo = Photo.objects.get(id=456)
photo.get_next_in_photoset(photoset)
photo.get_previous_in_photoset(photoset)
photoset.cover() # returns the cover Photo
```

Collection

```
c = Collection.objects.get(id=123)
c.parent # if collection is nested
c.sets.all() # sets in this collection
c.icon # the collage picture you see on Flickr
```

2.2.4 Templates

```
{% load flickr_tags %}
```

Small photo linking to flickr page

```
{% flickr_photo photo "small" 1 %}
```

Large photo without link

```
{% flickr_photo photo "large" %}
```

Photos

```
{% load flickr_tags %}

<h1>Django-Flickr (Demo Page)</h1>

<h2>Photos</h2>
     <ul class="flickr photos">
     {% for photo in photo_list %}
     <li>{% flickr_photo photo "small" 1 %}
             <h3><a href="{{ photo.get_absolute_url }}">{{ photo.title }}</a></h3>
             <p>{{ photo.description }}</p>
             <dl>
                     <dt>taken</dt><dd>{{ photo.date_taken|date:"d.m.Y" }}</dd>
                     <dt>tags</dt><dd>
                             <ul class="tags">
                                     {% for tag in photo.tags.all %}
                                             <li>#{{ tag }}</li>
                                     {% endfor %}
                                     </ul>
                     </dd>
             </dl>
     </li>
     {% endfor %}
     </ul>
     {% include "flickr/pagination.html" %}
```

Photosets

```
{% load flickr_tags %}

<h2>Photosets</h2>

     <ul class="flickr sets">
     {% for set in photosets %}
     {% if set.cover %}
     <li><a href="{{ set.get_absolute_url }}">{% flickr_photo set.cover "thumb" %}</a>
```

```
                <h3><a href="{{ set.get_absolute_url }}">{{ set.title }}</a></h3>
        </li>
        {% endif %}
        {% endfor %}
        </ul>
```

Tags

```
{% load taggit_extras %}

<h2>Tags</h2>

        {% get_taglist as tags for 'flickr' %}
        <ul class="tags">
        {% for tag in tags %}
                <li>#{{tag}} ({{tag.num_times}})</li>
        {% endfor %}
        </ul>
```

2.2.5 Flickr API

Using the API is easy. Here's an example authorized call to flickr.people.getPhotos.

```
from flickr.api import FlickrApi

FLICKR_KEY = getattr(settings, 'FLICKR_KEY', None)
FLICKR_SECRET = getattr(settings, 'FLICKR_SECRET', None)
PERMS = getattr(settings, 'FLICKR_PERMS', None)

api = FlickrApi(FLICKR_KEY, FLICKR_SECRET)
api.get('flickr.people.getPhotos')

# Returns JSON by default. If you want XML:
api.get('people.getPhotos', format='xml') # yep, also works without 'flickr.'
```

Currently supports only read methods with GET. Writing with POST soon to be implemented.

2.3 How to contribute

2.3.1 Reporting issues and submitting proposals

Found a bug? Let us know.

Got an idea for a feature? Let's discuss it.

This is the place for it: https://bitbucket.org/zalew/django-flickr/issues

2.3.2 Committing code

Non-breaking changes are welcome, just explicitely explain (in your commit message and/or pull request) what you did and why - it increases the chance your improvement gets merged upstream. Provide tests, make sure all tests pass, the build works, etc.

If you developped or want to develop a new feature, just show the code and/or submit a proposal. Explain the idea behind it and use cases.

As always, the best place for open discussion is: https://bitbucket.org/zalew/django-flickr/issues

CHAPTER

THREE

CONTRIBUTORS

- Jakub Zalewski
- Javier García Sogo

CHAPTER

FOUR

BUGS

The project is in early stages, expect bugs. Please report any issues.