frog Documentation

Release 1.0

Brett Dixon

February 01, 2017

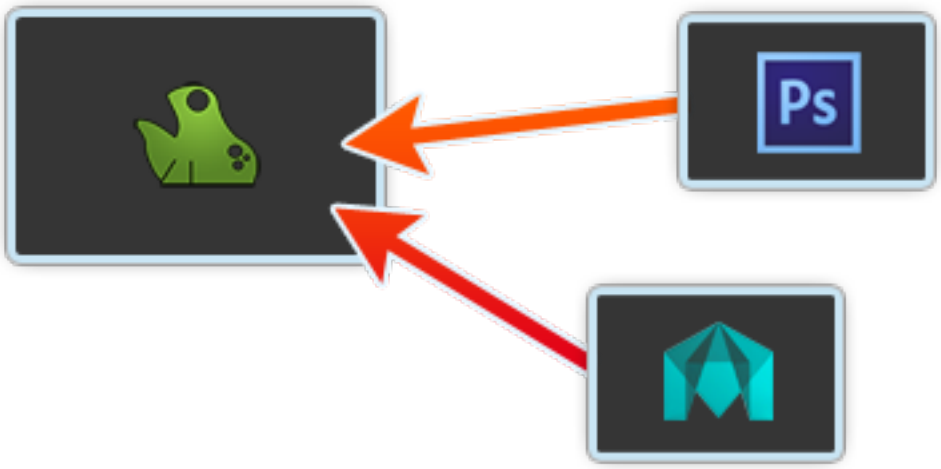
Contents

Frog Media Server is a server-client solution for sharing and maintaining large collections of images and videos

Quickstart

CHAPTER 1

Accessibility



First and foremost, Frog is easy to use and easy to add media files to. It has a straight forward API so integrating it into any other app or tool is trivial. Users can upload and view media instantly and the entire team benefits. For example, an artist could be working in Photoshop and simply hit a hotkey or panel button that would send a PNG and layered source PSD to Frog. The artist doesn't have to break his workflow just to share images with the rest of the team.

Since Frog is web based, everyone can get to it wherever they are. The viewer is done in HTML5 canvas which makes it even more accessible as no plugins are required.

CHAPTER 2

Visibility

One of the biggest problems I see in creative studios is that their media is usually hosted on a network drive somewhere and, if you're lucky, organized in folders. Searching is slow, browsing is near impossible, and sharing is a nightmare. A major benefit with Frog is just having everything visible at one time. Being able to browse all media in one place makes discovery a snap. There is also a sense of progress as users can see a history of the project's art over time.

CHAPTER 3

Discovery

To find images or videos, there is a search bar at the top just as there should be. You can do full text searches or filter by tags. Adding or removing tags to an image or a group of images is dead simple with a nice drag and drop interface anyone can use. Users can quickly search and filter a set of images, then send that link to others. For example, you could have a set of filters for a certain artist and a specific character. Share that link and you'll always have an up-to-date set of images meeting that criteria.

As mentioned before, when media is stored in a network folder, browsing is cumbersome. If Google has tought us anything, its that search is the only means of finding something when the number of items gets too large. No matter how organized a folder structure may be, search will always be faster and more accessible to users.

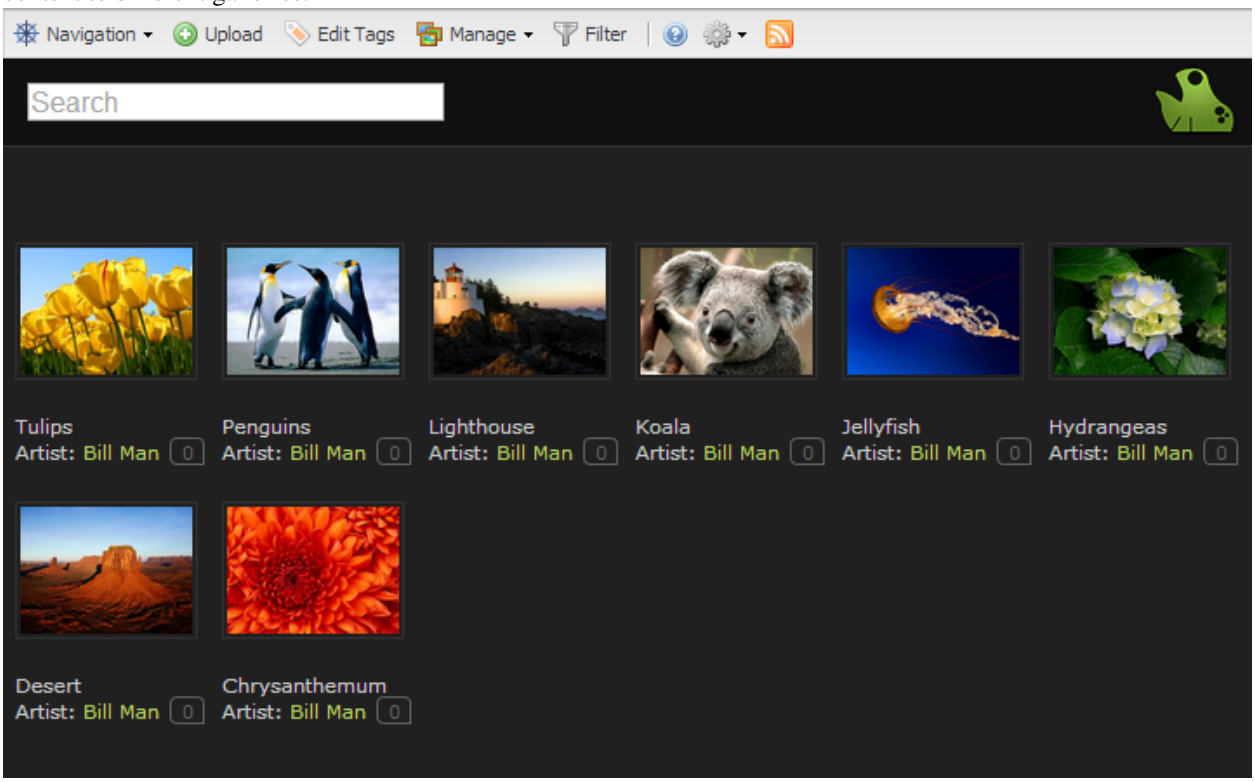
CHAPTER 4

User Guide

The user guide is for those actually using an instllation of Frog. It goes over the UI and how to interact and add content.

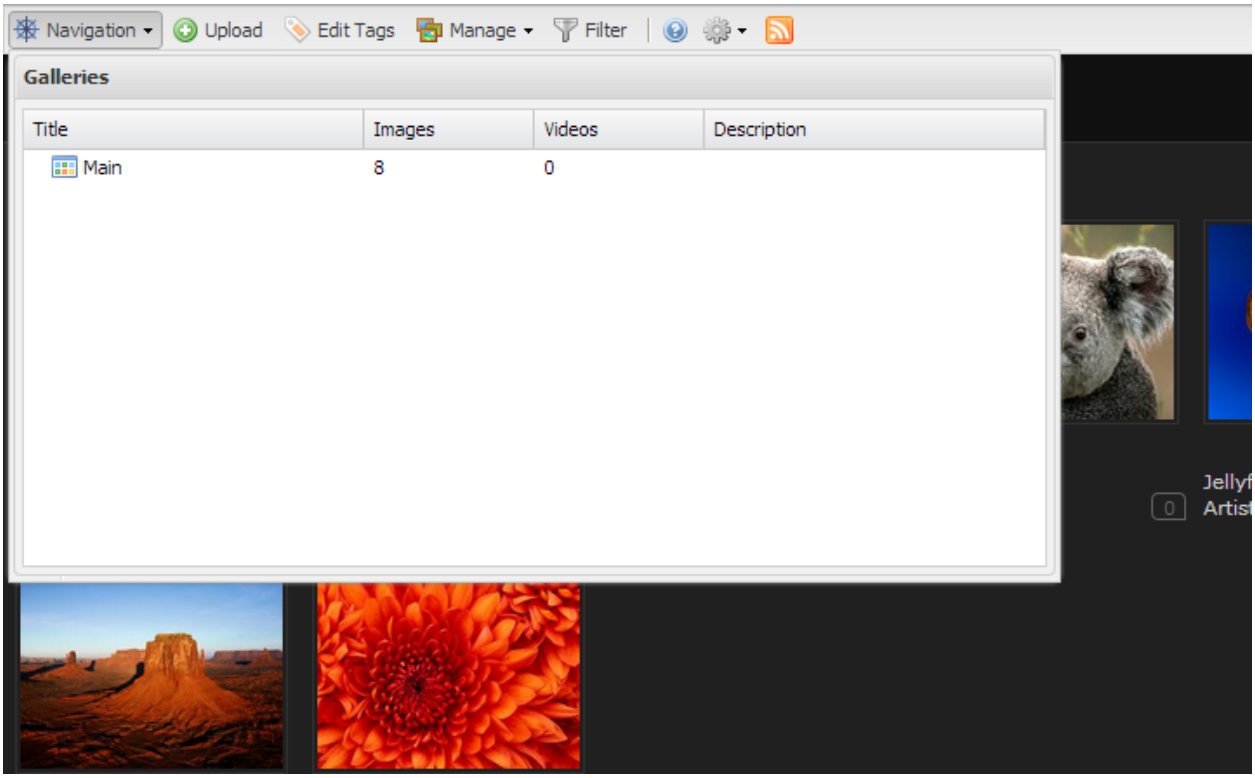
4.1 User Interface

Frog has one main UI feature at the top of the browser and has dialogs for adding and removing tags and copying content to different galleries.



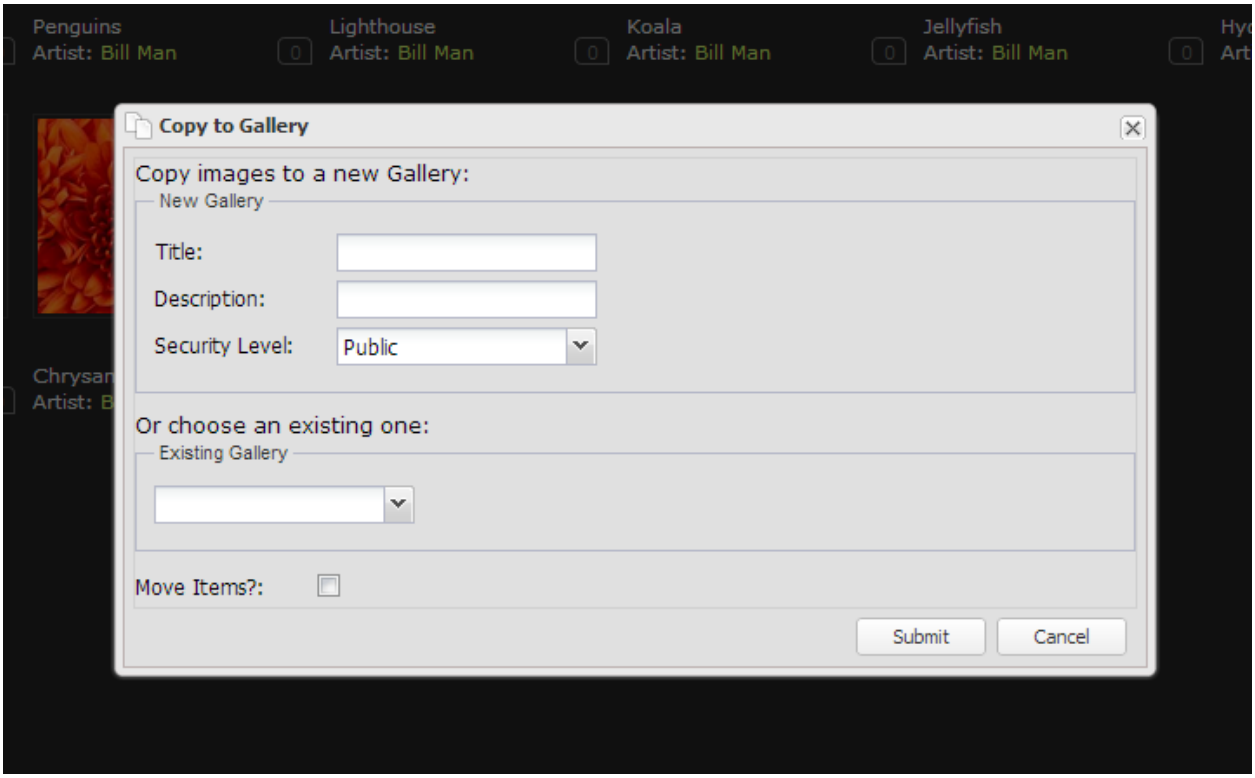
4.1.1 Navigation

The navigation window will show a list of galleries and any sub galleries they contain. Next to each item there is an image and video count along with the description for the gallery. You will only be able to see Galleries you have perssion to see or own.



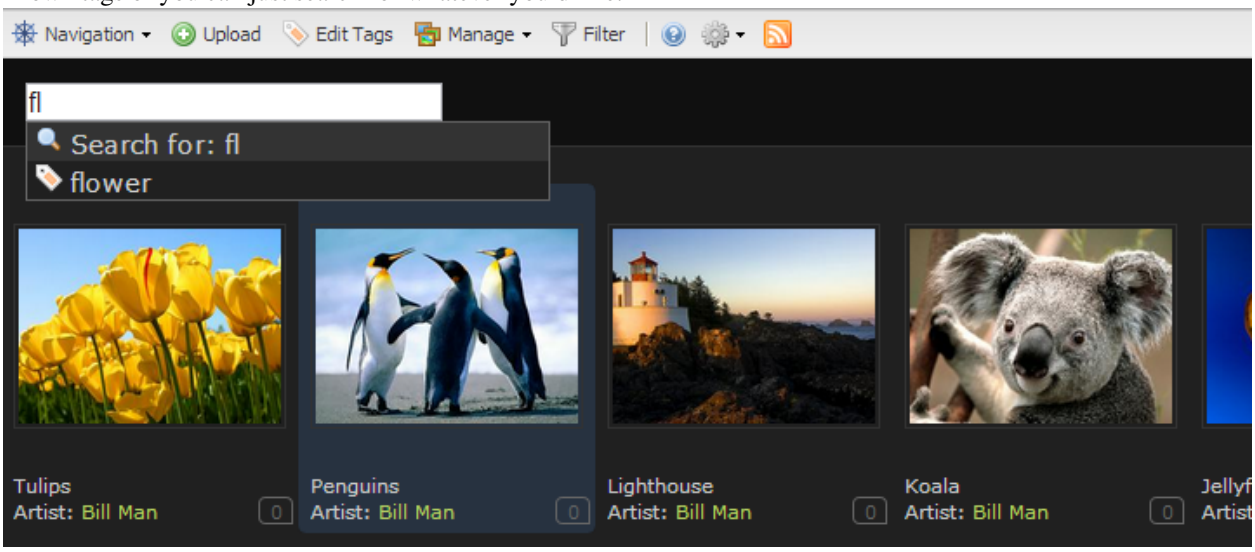
4.1.2 Copy to Gallery

This will be how you create new galleries and copy or move content between them. If you're creating a new gallery, simply fill in the field in the top section and all selected pieces will be added to the new gallery. If you want to copy or move pieces, choose the gallery from the bottom part of the dialog. Use the Move Items checkbox to move the pieces rather than copy them.



4.2 Search & Filter

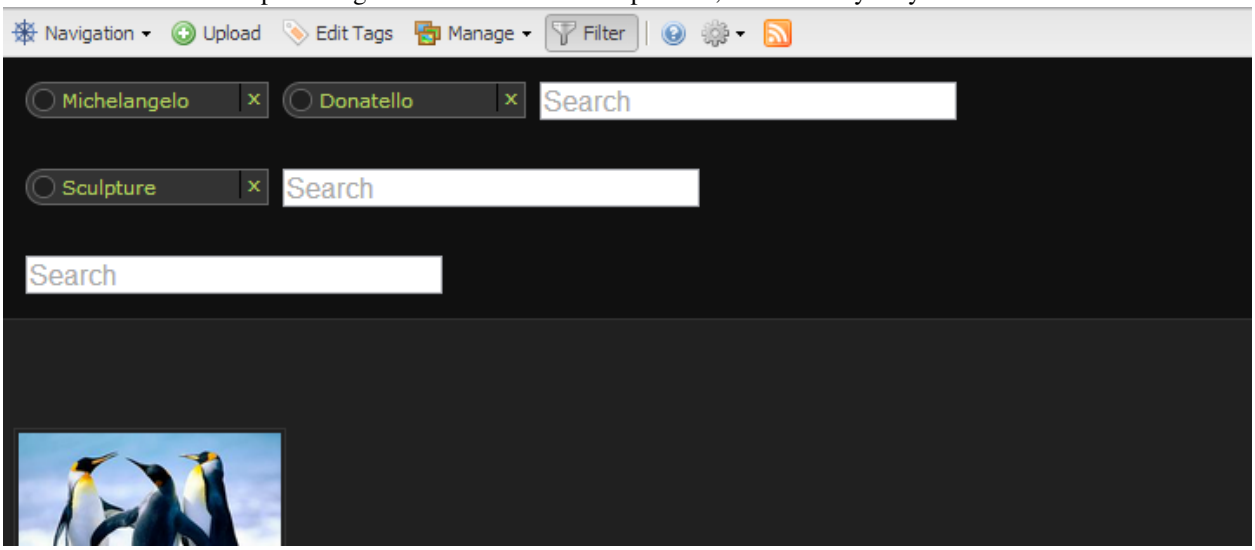
Searching and filtering is done by typing in your query into the search box at the top of Frog. It will autocomplete on known tags or you can just search for whatever you'd like.



4.2.1 Advanced Search

By clicking the Filter button on the toolbar at the top, you'll be able to do some more complicated searches. How this works is everything in each row is OR'ed together and the rows are AND'ed together. For example, say you wanted to

find all images from Donetello or Michelangelo that was tagged with sculpture. You would put the two artists names in one row and the "sculpture" tag in the next. Sounds complicated, but it's really easy to use.

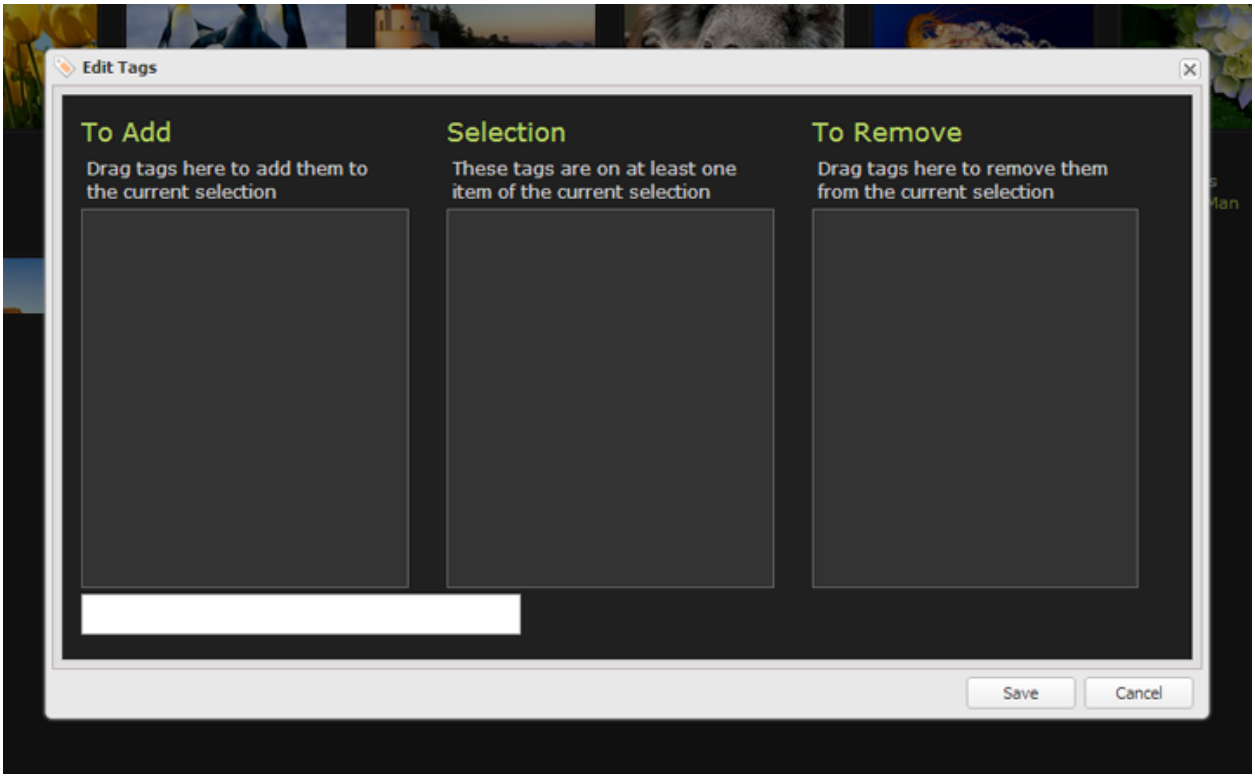


4.3 Tags

Tagging has been made pretty easy in Frog. You can select images and click the Edit Tags button or hit the TAB key to bring up the dialog. Here you'll have three columns:

- Tags to add to all selected items
- Tags that are on at least one of the selected items
- Tags to remove from all selected items

You can type in the name of a new tag or it will autocomplete for existing ones. You can also drag the tags to the different columns for quick addition or removal.



4.4 Adding Content

Adding content to Frog is as simple as dragging stuff from your desktop onto the site. It will upload and do any conversions neccessary and add them to the current Gallery. You can also click the Upload button on the toolbar at the top to bring up a familiar file selection dialog.

CHAPTER 5

Admin Guide

5.1 Installation

To install frog and many other python packages, you'll want to start with distribute and pip. Pip makes installing python packages a snap.

5.1.1 Distribute & Pip

Installing Requests is simple with pip:

```
$ pip install django-frog
```

5.1.2 Get source on GitHub

You can always clone the repository on GitHub.

You can either clone the public repository:

```
git clone git://github.com/theiviaxx/Frog.git
```

Download the tarball:

```
$ curl -OL https://github.com/theiviaxx/Frog.git
```

Or, download the zipball:

```
$ curl -OL https://github.com/theiviaxx/Frog/zipball/master
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages easily:

```
$ python setup.py install
```

5.2 Quickstart

This guide is written for windows to get up and running within a few minutes, but is not at all ready for a production server. If you're on linux, you should be able to follow the instructions below with minimal changes to get up and running.

5.2.1 Requirements

- Install Python 2.6 or later
- Install Pillow
- *Install Frog*

5.2.2 Start a project

First lets open a command line and:

```
> cd c:\\users\\brett
> python c:\\python27\\Scripts\\django-admin.py startproject dev
> cd dev
> mkdir static
> mkdir static\\frog
> copy C:\\python27\\lib\\site-packages\\frog\\static\\* static\\frog
```

5.2.3 Settings

Now we need to modify the default settings to include Frog and add some other stuff. The easiest thing to do is just copy and paste the settings file below and make the path changes to fit your machine. Edit the *c:\Users\brett\dev\dev\settings.py* file:

```
# Django settings for dev project.
import os
APPROOT = os.path.abspath(os.path.join(os.path.dirname(__file__), '..'))

DEBUG = True
TEMPLATE_DEBUG = DEBUG

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(APPROOT, 'frog.sqlite'),
    }
}

SITE_ID = 1

MEDIA_URL = 'http://127.0.0.1:8000/static/'
MEDIA_ROOT = os.path.join(APPROOT, 'static') + '\\'

ADMIN_MEDIA_PREFIX = '/static/admin/'
SECRET_KEY = 'secret'

TEMPLATE_LOADERS = (
    'django.template.loaders.filesystem.Loader',
    'django.template.loaders.app_directories.Loader',
)

TEMPLATE_CONTEXT_PROCESSORS = (
    "django.contrib.auth.context_processors.auth",
    "django.core.context_processors.debug",
    "django.core.context_processors.i18n",
```

```
    "django.core.context_processors.media",
    "django.core.context_processors.static",
    "django.core.context_processors.tz",
    "django.contrib.messages.context_processors.messages",
    "frog.context_processors.media",
)

MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
)

ROOT_URLCONF = 'dev.urls'

INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.admin',
    'django.contrib.comments',
    'frog',
)

LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console':{
            'level':'DEBUG',
            'class':'logging.StreamHandler',
        }
    },
    'loggers': {
        'django.request': {
            'handlers': ['console'],
            'level': 'ERROR',
            'propagate': True,
        },
    }
}

AUTHENTICATION_BACKENDS = (
    'frog.auth.SimpleAuthBackend',
)

FROG_FFMPEG = 'c:/ffmpeg/ffmpeg.exe'
FROG_SITE_URL = 'http://127.0.0.1:8000'
```

5.2.4 URLs

Next we have to add the Frog URLs so edit the *c:\Users\brett\dev\dev\urls.py* file:

```
from django.conf.urls import patterns, include, url
from django.conf import settings

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns(
    '',
    url(r'^static/(?P<path>.*)$', 'django.views.static.serve', {
        'document_root': settings.MEDIA_ROOT,
        'show_indexes': True
        }),
    url(r'^media/(?P<path>.*)$', 'django.views.static.serve', {
        'document_root': settings.MEDIA_ROOT,
        'show_indexes': True
        }),
    url(r'^frog/', include('frog.urls')),

    url(r'^admin/', include(admin.site.urls)),
)
```

5.2.5 Start the Server

At the command line type in

```
> python manage.py syncdb
> python manage.py runserver
```

Now go to http://127.0.0.1:8000/frog

5.3 Server Setup

This page will guide you through setting up a production server for a proper deployment. There are many giudes out the to get a full Python/Django stack going so this is not meant to be definitive. The point here is to get a webserver running that can handle the traffic you require. This guide assumes a server running Ubuntu 12.04, though as long as you can install or build the dependencies, and distro would work. I haven't set up a production server using IIS, but I'm sure there are guides to getting python and Django running on it.

5.3.1 The Parts

At the time of this writing, a common flavor is to use nginx and gunicorn. Nginx will be your main server and will serve static content such as JavaScript and images, then pass on any requests to gunicorn that need to be handled by Django. This makes for fast handling of the large images Frog is meant to display and a lightweight system to maintain down the road. As for a database, I'll be using MySQL, but I believe Postgres, or any other DB supported by Django's ORM, would work as well.

- nginx
- gunicorn
- MySQL

5.3.2 Install Packages

Since Ubuntu 12.04 comes with python 2.7, we'll just use that. We will need to install PIP and distribute though. I'll assume we'll be in */var/www*.

PIP

```
$ curl http://python-distribute.org/distribute_setup.py | python
$ curl https://raw.github.com/pypa/pip/master/contrib/get-pip.py | python
```

virtualenv

This is optional. It will keep your python packages in a isolated dir and it helps to keep things organized if the server will be doing other things with python. If you don't care, feel free to skip this and move on to **:ref:'Install Packages'_**.

```
$ pip install virtualenv
$ virtualenv dev
$ cd dev
$ source bin/activate
```

Now we need to install our python packages. Installing Frog will add what it needs.

```
$ pip install django-frog
$ pip install pillow
```

Make a folder for static files

```
$ mkdir /var/www/static
$ ln -s /usr/local/lib/python27/site-packages/frog/static /var/www/static/frog
$ mkdir /var/www/dev/migrations
$ touch /var/www/dev/migrations/__init__.py
```

5.3.3 Install MySQL

Ubuntu comes with MySQL 5, so no need to install anything further. We just need the python bindings:

```
$ apt-get install python-mysqldb
```

Next we just need to create a database and add a user. For this tutorial, I'll use 'frog' as the database and 'django' as the user:

```
> mysql -u root -p
mysql> create database frog;
mysql> create user 'django'@'localhost' identified by 'django';
mysql> grant all on frog.* to 'django'@'localhost' identified by 'django';
```

5.3.4 Install Nginx

The idea here is that nginx will sit in front of everything. Nginx is a lightweight and fast webserver, though for this tutorial, we'll just be using it to serve static files. So Nginx will redirect traffic to gunicorn when it's a django request and just server the static files when not. Gunicorn will listen on port :8000 and Nginx on port :80.

```
$ sudo apt-get install nginx
```

Edit */etc/nginx/nginx.conf*. You'll only need to edit the server section of the default conf:

```
...
http {
  ...
  log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';
  ...
  server {
      listen       80;
      server_name  dev;
      access_log   logs/www/dev.log  main;

      client_max_body_size 512M;

      # serve django admin files
      # This location may differ based on your setup
      location /media  {
        alias /usr/lib/python2.7/site-packages/django/contrib/admin;
        expires 30d;
      }

      # serve static files
      location /static  {
        alias /var/www/static;
        expires 30d;
      }

      # pass requests for dynamic content
      location / {
        proxy_pass_header Server;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_connect_timeout 10;
        proxy_read_timeout 10;
        proxy_pass http://127.0.0.1:8000/;
      }
  }
  ...
}
...
```

5.3.5 Install Gunicorn

Gunicorn will run all of the django/python stuff behind nginx. We'll need to install it and then write a service script so we can start/stop/restart it quickly. Install gunicorn:

```
$ pip install gunicorn
```

Edit */var/www/dev/gunicorn.sh*

```
#!/bin/bash
set -e
LOGFILE=/var/log/gunicorn/frog.log
LOGDIR=$(dirname $LOGFILE)
NUM_WORKERS=3
# user/group to run as
```

```
USER=your_unix_user
GROUP=your_unix_group
cd /var/www/dev
test -d $LOGDIR || mkdir -p $LOGDIR
exec gunicorn dev.wsgi:application \
--user=$USER --group=$GROUP --log-level=debug \
--log-file=$LOGFILE 2>>$LOGFILE
```

Make it executable

```
chmod ug+x /var/www/dev/gunicorn.sh
```

Next create an Upstart service, /etc/init/gunicorn.conf

```
description "Frog Django instance"
start on runlevel [2345]
stop on runlevel [06]
respawn
respawn limit 10 5
exec /var/www/dev/gunicorn.sh
```

Make a symlink to init.d

```
sudo ln -s /lib/init/upstart-job /etc/init.d/gunicorn
```

5.3.6 Setup Django

Now we need to configure Django to include Frog and handle our requests. Edit */var/www/dev/dev/settings.py* and make the following changes:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'frog',
        'USER': 'django',
        'PASSWORD': 'django',
        'HOST': '',
        'PORT': '',
    }
}
MEDIA_ROOT = '/var/www/static/'
MEDIA_URL = 'http://127.0.0.1/static/'
TEMPLATE_CONTEXT_PROCESSORS = (
 "django.contrib.auth.context_processors.auth",
 "django.core.context_processors.debug",
 "django.core.context_processors.i18n",
 "django.core.context_processors.media",
 "django.core.context_processors.static",
 "django.core.context_processors.tz",
 "django.contrib.messages.context_processors.messages",
 "frog.context_processors.media",
)
# Needed for redirect
LOGIN_URL = '/frog'

INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.comments',
    'haystack',
    'django.contrib.admin',
    'frog',
    'south',
)
SOUTH_MIGRATION_MODULES = {
    'frog': 'migrations',
}

AUTHENTICATION_BACKENDS = ('frog.auth.SimpleAuthBackend',)
SESSION_COOKIE_AGE = 31556926 # 1 year
```

Now add the Frog URLs to */var/www/dev/dev/urls.py*

```
...
url(r'^frog/', include('frog.urls')),
...
```

5.3.7 Start It

With everything in place, we just need to start it up and cross our fingers

```
service gunicorn start
service nginx start
```

Go to http://server and see if there's a happy Django page

5.4 Existing Django Project

If you already have a working Django project and you'd like to add Frog to it, please follow these steps

5.4.1 Migrations

Frog uses south to handle its model migrations. This probably wont come up often, if ever, but should you need to adjust the models, South will be invaluable. Under your project root:

```
$ mkdir migrations
$ touch migrations/__init__.py
```

Now all Frog's model migrations will be stored here instead of in the site-packages folder.

5.4.2 Install Frog

First use pip to install Frog, it will depend on Django 1.5+ so it may upgrade your installation. Make sure you are prepared for that! Also, the paths below are for example only, please make any changes to fit your system.

```
$ pip install django-frog
$ ln -s /usr/local/lib/python27/site-packages/frog/static /var/www/static/frog
```

5.4.3 Settings

Edit your project setting.py file

```
INSTALLED_APPS = (
    ...
    'haystack', ## -- Haystack must come before admin
    'django.contrib.admin',
    'frog',
    ...
    'south',
)
SITE_URL = 'http://servername'
## -- Used for creating new users
DOMAIN = 'yourdomain.com'
## -- Path to ffmpeg for processing video
FFMPEG = 'path/to/ffmpeg'

SOUTH_MIGRATION_MODULES = {
    'frog': 'migrations',
}

TEMPLATE_CONTEXT_PROCESSORS = (
 "django.contrib.auth.context_processors.auth",
 "django.core.context_processors.debug",
 "django.core.context_processors.i18n",
 "django.core.context_processors.media",
 "django.core.context_processors.static",
 "django.core.context_processors.tz",
 "django.contrib.messages.context_processors.messages",
 "frog.context_processors.media",
)
```

Note that the following middlewares need to be enabled:

- django.contrib.sessions.middleware.SessionMiddleware
- django.middleware.csrf.CsrfViewMiddleware
- django.contrib.auth.middleware.AuthenticationMiddleware
- django.contrib.messages.middleware.MessageMiddleware

Also add the Frog URLs to your projects urls.py

```
...
url(r'^frog/', include('frog.urls')),
...
```

5.4.4 Database

Do an initial migration and create the tables to the Frog models

```
$ python manage.py schemamigration frog --initial
$ python manage.py migrate frog
```

That's it, you should be able to go to http://server/frog

5.5 Troubleshooting

CHAPTER 6

Developer Guide

6.1 Introduction

Frog is a simple server/client setup and most of the work is done on the client. The server is simply a REST API for the client(s) to work with. There are only a couple URLs which will render HTML, the rest will return a standardized JSON serialized response object.

6.1.1 Response

```
{
    "isSuccess": true,     // Did the request succeed
    "isError": false,      // Did the request fail
    "message": "",         // A string relevent to the request result
    "values": [],          // A list of objects.  This will always contain the object in "value"
    "value": {}            // An object.  If the request was for a list, this will be the first
                              member of the list
}
```

6.2 Views

6.2.1 Gallery

Galleries are simply collections of pieces. They are displayed in order of creation and can share pieces between them. Galleries can have sub galleries that can old specific sets of images, then can be promoted to the parent Gallery.

Gallery API

```
GET     /        Lists the galleries currently visible by the current user
POST    /        Creates a gallery object
GET     /id      Gallery object if visible by the current user
PUT     /id      Adds image or video objects to the gallery
DELETE  /id      Removes image or video objects from the gallery
GET     /filter  Returns a filtered list of image and video objects
```

6.2.2 Piece

A piece is either an Image or a Video and is made abstract so you can add more asset types if you'd like. A piece represents one thumbnail in Frog.

Piece API

```
GET     /image/id  Returns a rendered page displaying the requested image
GET     /video/id  Returns a rendered page displaying the requested video
POST    /image/id  Add tags to an image object
POST    /video/id  Add tags to an video object
DELETE  /image/id  Flags the image as deleted in the database
DELETE  /video/id  Flags the video as deleted in the database
```

6.2.3 Tag

All objects are tagged. As your data set grows, hierarchial navigation becomes combersome and inefficient. Users will almost always resort to search of some sort.

Tag API

```
GET     /         Lists all tags
POST    /         Creates a Tag object
PUT     /         Adds tags to guids
DELETE  /         Removes tags from guids
GET     /search   Search tag list
GET     /manage   Renders a form for adding/removing tags
POST    /manage   Adds and removes tags from guids and commits data
```

6.2.4 Comment

Frog implements a basic comment system for each Piece.

Comment API

```
GET     /         Returns a rendered list of comments
GET     /id       Returns a serialized comment
POST    /id       Creates a comment for an object
PUT     /id       Updates the content of the comment
```

6.3 Frog Settings

Note: If you are installing Frog 1.0 from a previous version, please make sure you change any previous setting to the new ones below.

FROG_IMAGE_SIZE_CAP
: The maximum image size to allow for uploads. Anything larger will be scaled proportionally to fit into a square of this value.

  Default is 5120

FROG_IMAGE_SMALL_SIZE
: This value determines what is considered a "small" version of the image

Default is 600

FROG_THUMB_SIZE
The value to scale thumbnails down to

Default is 256

FROG_UNIQUE_ID
A string to a module function that will be called to determine a unique value for the image. This will be used to find and overwrite existing images. The function should take two arguments, a path and user. The *path* arg will have as much of the filename as possible and the *user* arg will be a standard Django User instance. The default is

```
def uniqueid(path, user)
    username = 'Anonymous' if user.is_anonymous() else user.username
    return '%s_%s' % (username, os.path.split(path)[1])
```

FROG_PATH
A path relative to our MEDIA_ROOT setting where Frog will store it's image and video files

Default is ''

FROG_FFMPEG
Absolute path to ffmpeg

FROG_SCRUB_DURATION
If the video is less than this value, in seconds, then it will be converted so it can be scrubbed frame by frame. This is slower and takes longer but works well for animations.

Default is 60

FROG_FFMPEG_ARGS
The argument string sent to ffmpeg to convert a video to a web compatible format. For advanced use only.

Default is *-vcodec libx264 -b:v 2500k -acodec libvo_aacenc -b:a 56k -ac 2 -y*

FROG_SCRUB_FFMPEG_ARGS
The argument string sent to ffmpeg to convert a video to a scrubbable, web compatible format. For advanced use only.

Default is *-vcodec libx264 -b:v 2500k -x264opts keyint=1:min-keyint=8 -acodec libvo_aacenc -b:a 56k -ac 2 -y*

FROG_SITE_URL
The absolute URL for the host. This is used in RSS feeds and comments to give absolute URLs to content

f

F