# Elodie Documentation

***Release 0.1.0***

**Jaisen Mathai**

**Dec 04, 2018**

# Contents

*~~ Your Personal EXIF-based Photo, Video and Audio Assistant ~~*


I work tirelessly to make sure your photos are always sorted and organized so you can focus on more important things. By photos I mean JPEG, DNG, NEF and common video and audio files.

You don't love me yet but you will.

I only do 3 things.

- Firstly I organize your existing collection of photos.
- Second I help make it easy for all the photos you haven't taken yet to flow into the exact location they belong.
- Third but not least I promise to do all this without a yucky proprietary database that some friends of mine use.

You can find out more information about me on GitHub.

CHAPTER 1

# API Documentation

This documentation is generated from the Python code.

**Modules**

- *elodie.media*
- *elodie.constants*
- *elodie.dependencies*
- *elodie.filesystem*
- *elodie.geolocation*
- *elodie.localstorage*
- *elodie.plist_parser*

## 1.1 elodie.media

The media module provides a base *Media* class for media objects that are tracked by Elodie. The Media class provides some base functionality used by all the media types, but isn't itself used to represent anything. Its sub-classes (*Audio*, *Photo*, and *Video*) are used to represent the actual files.

**class** elodie.media.media.**Media**(*source=None*)

The base class for all media objects.

**Parameters** **source** (*str*) – The fully qualified path to the video file.

**get_album**()

Get album from EXIF

**Returns** None or string

**get_camera_make**()
Get the camera make stored in EXIF.

**Returns** str

**get_camera_model**()
Get the camera make stored in EXIF.

**Returns** str

**get_coordinate**(*type='latitude'*)
Get latitude or longitude of media from EXIF

**Parameters** **type** (*str*) – Type of coordinate to get. Either "latitude" or "longitude".

**Returns** float or None if not present in EXIF or a non-photo file

**get_exiftool_attributes**()
Get attributes for the media object from exiftool.

**Returns** dict, or False if exiftool was not available.

**get_original_name**()
Get the original name stored in EXIF.

**Returns** str

**get_title**()
Get the title for a photo of video

**Returns** str or None if no title is set or not a valid media type

**reset_cache**()
Resets any internal cache

**set_album**(*album*)
Set album for a photo

**Parameters** **name** (*str*) – Name of album

**Returns** bool

**set_date_taken**(*time*)
Set the date/time a photo was taken.

**Parameters** **time** (*datetime*) – datetime object of when the photo was taken

**Returns** bool

**set_original_name**(*name=None*)
Sets the original name EXIF tag if not already set.

**Returns** True, False, None

**set_title**(*title*)
Set title for a photo.

**Parameters** **title** (*str*) – Title of the photo.

**Returns** bool

The audio module contains classes specifically for dealing with audio files. The *Audio* class inherits from the *Video* class.

**class** elodie.media.audio.**Audio**(*source=None*)
An audio object.

**Parameters** **source** (*str*) – The fully qualified path to the audio file.

**extensions = ('m4a',)**
Valid extensions for audio files.

The photo module contains the *Photo* class, which is used to track image objects (JPG, DNG, etc.).

**class** elodie.media.photo.**Photo**(*source=None*)
A photo object.

**Parameters** **source** (*str*) – The fully qualified path to the photo file

**extensions = ('arw', 'cr2', 'dng', 'gif', 'jpeg', 'jpg', 'nef', 'rw2')**
Valid extensions for photo files.

**get_date_taken**()
Get the date which the photo was taken.

The date value returned is defined by the min() of mtime and ctime.

**Returns** time object or None for non-photo files or 0 timestamp

**is_valid**()
Check the file extension against valid file extensions.

The list of valid file extensions come from self.extensions. This also checks whether the file is an image.

**Returns** bool

The video module contains the *Video* class, which represents video objects (AVI, MOV, etc.).

**class** elodie.media.video.**Video**(*source=None*)
A video object.

**Parameters** **source** (*str*) – The fully qualified path to the video file.

**extensions = ('avi', 'm4v', 'mov', 'mp4', 'mpg', 'mpeg', '3gp')**
Valid extensions for video files.

**get_date_taken**()
Get the date which the photo was taken.

The date value returned is defined by the min() of mtime and ctime.

**Returns** time object or None for non-photo files or 0 timestamp

## 1.2 elodie.constants

Settings used by Elodie.

elodie.constants.**accepted_language = 'en'**
Accepted language in responses from MapQuest

elodie.constants.**application_directory = '/home/docs/.elodie'**
Directory in which to store Elodie settings.

elodie.constants.**debug = False**
If True, debug messages will be printed.

elodie.constants.**exiftool_config = '/home/docs/checkouts/readthedocs.org/user_builds/elodie**
Path to Elodie's ExifTool config file.

elodie.constants.**hash_db = '/home/docs/.elodie/hash.json'**
File in which to store details about media Elodie has seen.

elodie.constants.**location_db = '/home/docs/.elodie/location.json'**
File in which to store geolocation details about media Elodie has seen.

elodie.constants.**script_directory = '/home/docs/checkouts/readthedocs.org/user_builds/elodi**
Elodie installation directory.

## 1.3 elodie.dependencies

Helpers for checking for an interacting with external dependencies. These are things that Elodie requires, but aren't installed automatically for the user.

elodie.dependencies.**EXIFTOOL_ERROR = u"It looks like you don't have exiftool installed, wh**
Error to print when exiftool can't be found.

elodie.dependencies.**get_exiftool**()
Get path to executable exiftool binary.

We wrap this since we call it in a few places and we do a fallback.

**Returns** str or None

elodie.dependencies.**verify_dependencies**()
Verify that external dependencies are installed.

Prints a message to stderr and returns False if any dependencies are missing.

**Returns** bool

## 1.4 elodie.filesystem

General file system methods.

**class** elodie.filesystem.**FileSystem**
A class for interacting with the file system.

**create_directory**(*directory_path*)
Create a directory if it does not already exist.

**Parameters** **directory_name** (*str*) – A fully qualified path of the to create.

**Returns** bool

**delete_directory_if_empty**(*directory_path*)
Delete a directory only if it's empty.

Instead of checking first using *len([name for name in os.listdir(directory_path)]) == 0*, we catch the OSError exception.

**Parameters** **directory_name** (*str*) – A fully qualified path of the directory to delete.

**get_all_files**(*path*, *extensions=None*)
Recursively get all files which match a path and extension.

**Parameters**

- **path string** (*str*) – Path to start recursive file listing
- **extensions** (*tuple(str)*) – File extensions to include (whitelist)

**Returns** generator

**get_current_directory**()
Get the current working directory.

**Returns** str

**get_file_name**(*media*)
Generate file name for a photo or video using its metadata.

We use an ISO8601-like format for the file name prefix. Instead of colons as the separator for hours, minutes and seconds we use a hyphen. https://en.wikipedia.org/wiki/ISO_8601#General_principles

**Parameters** **media** (*Photo* or *Video*) – A Photo or Video instance

**Returns** str or None for non-photo or non-videos

**get_folder_path**(*metadata*)
Given a media's metadata this function returns the folder path as a string.

**Parameters** **dict** (*metadata*) – Metadata dictionary.

**Returns** str

**get_folder_path_definition**()
Returns a list of folder definitions.

Each element in the list represents a folder. Fallback folders are supported and are nested lists. Return values take the following form. [

('date', '%Y-%m-%d'), [

('location', '%city'), ('album', ''), ('"Unknown Location", '')

]

]

**Returns** list

**parse_mask_for_location**(*mask*, *location_parts*, *place_name*)
Takes a mask for a location and interpolates the actual place names.

Given these parameters here are the outputs.

mask=%city location_parts=[('%city','%city','city')] place_name={'city': u'Sunnyvale'} output=Sunnyvale

mask=%city-%state location_parts=[('%city-','%city','city'), ('%state','%state','state')] place_name={'city': u'Sunnyvale', 'state': u'California'} output=Sunnyvale-California

mask=%country location_parts=[('%country','%country','country')] place_name={'default': u'Sunnyvale', 'city': u'Sunnyvale'} output=Sunnyvale

**Parameters**

- **mask** (*str*) – The location mask in the form of %city-%state, etc
- **location_parts** (*list*) – A list of tuples in the form of [('%city-', '%city', 'city'), ('%state', '%state', 'state')]
- **place_name** (*dict*) – A dictionary of place keywords and names like {'default': u'California', 'state': u'California'}

**Returns** str

**set_utime_from_metadata**(*metadata*, *file_path*)
Set the modification time on the file based on the file name.

## 1.5 elodie.geolocation

Look up geolocation information for media objects.

## 1.6 elodie.localstorage

Methods for interacting with information Elodie caches about stored media.

**class** elodie.localstorage.**Db**

A class for interacting with the JSON files created by Elodie.

**add_hash**(*key*, *value*, *write=False*)

Add a hash to the hash db.

**Parameters**

- **key** (*str*) –
- **value** (*str*) –
- **write** (*bool*) – If true, write the hash db to disk.

**add_location**(*latitude*, *longitude*, *place*, *write=False*)

Add a location to the database.

**Parameters**

- **latitude** (*float*) – Latitude of the location.
- **longitude** (*float*) – Longitude of the location.
- **place** (*str*) – Name for the location.
- **write** (*bool*) – If true, write the location db to disk.

**all**()

Generator to get all entries from self.hash_db

:returns tuple(string)

**backup_hash_db**()

Backs up the hash db.

**check_hash**(*key*)

Check whether a hash is present for the given key.

**Parameters** **key** (*str*) –

**Returns** bool

**checksum**(*file_path*, *blocksize=65536*)

Create a hash value for the given file.

See http://stackoverflow.com/a/3431835/1318758.

**Parameters**

- **file_path** (*str*) – Path to the file to create a hash for.
- **blocksize** (*int*) – Read blocks of this size from the file when creating the hash.

**Returns** str or None

**get_hash**(*key*)
Get the hash value for a given key.

**Parameters** **key** (*str*) –

**Returns** str or None

**get_location_coordinates**(*name*)
Get the latitude and longitude for a location.

**Parameters** **name** (*str*) – Name of the location.

**Returns** tuple(float), or None if the location wasn't in the database.

**get_location_name**(*latitude*, *longitude*, *threshold_m*)
Find a name for a location in the database.

**Parameters**

- **latitude** (*float*) – Latitude of the location.
- **longitude** (*float*) – Longitude of the location.
- **threshold_m** (*int*) – Location in the database must be this close to the given latitude and longitude.

**Returns** str, or None if a matching location couldn't be found.

**update_hash_db**()
Write the hash db to disk.

**update_location_db**()
Write the location db to disk.

## 1.7 elodie.plist_parser

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## e

# Index

## M



## P



## R



## S



## U



## V