

MT2

The Wu-Tang of Thumbnailers
by Jesse Pavel
December 31, 2002

Contents

1	Introduction	1
1.1	Installation	1
1.2	Invocation	1
1.3	Resource File	5
2	Customization Files	5
2.1	Comment File	6
2.2	Group File	7
2.3	Date File	7
2.4	Hide File	8
2.5	List File	8
2.6	Include File	9
3	Output Plugins	9
3.1	Using Plugins	10
3.2	Installing New Plugins	10
3.3	Writing New Plugins	10
4	FAQ & Miscellany	11
5	Conclusion	11

1 Introduction

This manual describes the **mt2** (that is, **make-thumbnails 2**) program, which is designed to make it easy for one to create and maintain large numbers of pictures in HTML photo albums, along with descriptions, titles, grouping, and a lot of other features that I will eventually explain. I used to write that mt2 is not a program for Quiche-Eaters¹, but starting with version 1.2, it supports output plugins with which you can generate pages as fancy or colorful as you would like; however, its main focus is still on the meat and content of the albums.

First, I'll present an overview of what you need to run mt2, and how to actually invoke the program, along with all of the command-line arguments it accepts. Then, in the next section, I'll go into the files with which one can customize albums, and which are central to the really sweet features of the program. Finally, I'll describe how to use the output plugins (“themes” if you will) that come with mt2, and how you can write your own, if you're so inclined.

1.1 Installation

To run mt2, you'll need Python 2.2 or higher, and the *convert* program from the ImageMagick package; alternately, if you have the Python Imaging Library installed, mt2 will use it instead of starting an external program. Also, mt2 was written on Linux, and works with any Unix-like operating system (including OS X)—I've also recently tested it on Windows with PIL, and maybe with some tweaking it could run on the Mac as well. You can download python from <http://python.org/>, ImageMagick from <http://www.imagemagick.org/>, and the Python Imaging Library from <http://www.pythonware.com/products/pil/>.

Mt2 itself is distributed as two files, `makethumb2.py` and `mt2`, and a `plugins` directory: `mt2` is a small driver file that tries to determine what environment it is in, and then imports and invokes `makethumb2`. You should set the `MT2SCRIPTDIR` environment variable to the directory that contains `makethumb2.py` and the `plugins` subdirectory; and if the *convert* program is called something else than `convert` on your system, set `MT2CONVERT` to the name of the program you want to use. If you are on a Unix-like system, edit the `mt2` file and change the first line from “`#!/usr/bin/python2 -0`” to point correctly to your Python binary; if you are on Windows, you'll probably want to run the program with the command line “`python <path to mt2>/mt2`”.

1.2 Invocation

Here I will describe how to run the program, and go over the command-line arguments that it accepts. Once you have mt2 set up as above, you run it by typing `mt2`, along with any options you wish to pass it. In addition to command line arguments, mt2 can take configuration commands from a resource file that is specific to each directory thumbnailled. We'll cover these resource files after we present the command-line flags. A quick summary of all the options is taken directly from the output of mt2 itself:

¹Do a search for *Real Programmers Don't Use Pascal* on Google to understand.

Usage: mt2 [options]

Options:

```
-h or --help  show this message
-s <max thumbnail size> (100)
-d <thumbnail directory> ("thumbnails")
-a <base album name> ("00_album_images")
-ni <base imageless album name> = ("00_album_noimages")
-ani <name for both albums> ("00_album")
-l <list file name>
-i <include file name>
-g <group file name>
-hi <hide file name>
-C <comment file name>
-D <date file name>
-H <header file name>
-T <trailer file name>
-t <title string> ("Picture Album")
-hr <header string> ("

## 


```

By default, mt2 will generate albums of all the JPEG files in the current directory and stop. Sometimes that is what you want, but I know that other times you will want to make phat albums, and that is where all the options come in; and so, here is a detailing of each flag, along with some information about how the program works.

-h, -help This, if it weren't obvious, prints out the help text that you see above.

-s Normally, each thumbnail in the HTML album is confined to a 100x100 box, but you can change the size by using this flag with a new size, for instance **-s 40** to make tiny little thumbnails.

- d** Mt2 by default creates a subdirectory named *thumbnails* for each directory that you want to thumbnail. You can change the name using this flag.
- a** The HTML albums are named by a combination of a prefix and a number. For instance, if one has three albums in a directory, they will by default have the names *00_album_images-00.html*, *00_album_images-01.html*, and *00_album_images-02.html*. To change the “00_album_images” part of these names, use the **-a** flag with the new name.
- ni** This flag works exactly as the **-a** flag, except that it affects the text-only albums, which are by default named with the prefix *00_album_noimages*.
- ani** With this flag, you can set the base name for both the image album and the text only album at the same time. If you use *-ani base*, the albums will have the prefixes *base_images* and *base_noimages*, respectively.
- l** Mt2, instead of reading filenames itself from the directory, can take them from a “list file”, a procedure that I will discuss in detail in the next section. This flag allows one to give the name of a list file on the command line.
- i** Using an “include file” allows you to keep the names of the pictures you wish to process in a file, instead of constructing an elaborate glob to include them all. See the paragraph on include files in the next section for more information.
- g** Group files, which can be given with this flag, are a way of telling Mt2 that one wants certain files to appear next to each other in the albums, though they might not otherwise do that with the given sort order—they too will be discussed in the next section.
- hi** “Hide files” will be discussed later, but briefly, allow one to tell mt2 not to thumbnail certain files. They are specified using this flag.
- C** Each picture can have a comment printed under its name in the album, and these comments are stored in a “comment file”, which can be given with this flag. We’ll talk about comment files in detail in the next section.
- D** You can override the date displayed in the thumbnail albums for each picture with specific dates of your own, using a “date file”, indicated by this flag—in the next section, I give more detail about this type of file.
- H** One might want a potentially complicated header to be inserted into each album, and with this flag one can specify an HTML file to fill this role.
- T** This parallels the **-H** flag, except the trailer file will be inserted at the end of each album.
- hr** If the header that you want at the top of each album is fairly simple, you can just pass it on the command line using this flag. For instance, **-hr '<h1>Big Ugly Photos</h1>'**.

- tr** This flag is just like **-hr**, except that it works for the trailer.
- n** If you have many images in a directory, mt2 splits them among a number of albums to help with navigation and load times. By default, it puts 30 images per album, but you can change that using this flag.
- G** Normally, mt2 thumbnails all the JPEG files in the current directory, using the wildcard pattern of *.jpg,*.JPG,*.jpeg,*.JPEG. You can specify a different pattern with this flag; for instance, to thumbnail all GIFs instead, use the argument **-G '*.gif,*.GIF'** (note the single quotes around the wildcards).
- b** When processing directories recursively, mt2 includes the `".."` (back) directory in the list of folders; but when operating non-recursively, it doesn't include it—this option forces the inclusion of `".."`, regardless of the mode.
- f** If there is a directory for which you do not want thumbnails generated, you can create a file named `.nothumbnail` in it; normally, if mt2 detects such a file, it will abandon its attempt to create albums; however, if you want it to thumbnail the directory anyhow, use this flag.
- ft** When mt2 is working recursively (see the **-R** switch below), it first checks if each directory seems to have changed since it last thumbnailled it; if it hasn't changed, mt2 skips it. This flag causes mt2 to create new albums regardless of whether it thinks the old albums are up to date.
- R** Mt2 usually works on just the current directory, but by using this flag, you can tell mt2 to operate not only on the current directory, but also on all its subdirectories. There are two types of directories mt2 skips, though: directories that contain a `.nothumbnail` file, and directories that are named the same as the thumbnail directory, so that it doesn't accidentally make thumbnails of thumbnails.
- np** After creating an album, mt2 usually checks if there are any extra thumbnail images lying around in the thumbnail directory, and if so deletes them. This flag prevents it from pruning these extraneous images.
- fp** Some configuration options cause pruning to be disabled; by specifying **-fp**, you can tell mt2 to force pruning.
- nd** In the albums, next to the picture names, mt2 usually displays the date that the picture was taken (or uploaded). Sometimes you want to disable the display of dates in the albums, and this flag does that.
- Ii** For each directory that mt2 thumbnails, it can create a symbolic link from `index.html` to the first album file, so that when someone looks at the directory from their browser, they are served the image album instead. This will work only on Unix and its kin.
- Ini** This is the same as **-Ii**, except the link will be made from `index.html` to the text-only album.

- sn, -sa, -sd** The pictures in the albums can be sorted by name (**-sn**), by ascending date from oldest to newest (**-sa**), or by descending date from newest to oldest (**-sd**). The default is by ascending date.
- r** In addition to taking configuration options from the command line, mt2 can read a resource file to get options, and with this flag, you can specify an alternate file from which to read customizations. Mt2 by default searches for a file named *.mt2rc*, and if that is not present, then *.make-thumbnailsrc*, for historical purposes.
- lp** This switch will have mt2 examine its plugin directory and list the plugins that it finds there.
- op** Mt2 uses an output plugin to actually write the HTML (though theoretically the plugin could write whatever format of file it wishes, perhaps a PDF) albums, and one can choose which to use with this switch. The default is *plainhtml*. See the section on plugins for more information about this.
- pc** You can pass configuration options to the output plugin, in addition to any options that the plugin reads from its personal files, using this switch. As the argument to **-pc**, pass a string of the format “*key:value,key:value,...*”, where the keys and values will have meaning dependent on the plugin you select. For example, with the *darkhtml* plugin, to be discussed later, you can use this switch in the line, **-pc** “*columns:3,data_url:http://host.com/darkhtml/*”.

1.3 Resource File

In each directory, you can keep a ‘resource file’ so that you don’t have to type all of your options on the command line every time you invoke mt2. Also, there is a global resource file, *~/mt2/mt2rc*, that will be read before processing any directory-specific resource files. The directives that you can use in the resource file correspond to command line options, so I’ll just present a table, Table 1, showing which directives correspond to which options. The format of the resource file is a series of *key=value* pairs, one per line; here is an example file:

```
sample .mt2rc
sort-order=date_ascending
header-string=<h1>Hot Beach Babes</h1>
glob-pattern=beach*.jpg
```

2 Customization Files

The files that one can use to customize the order in which mt2 displays images, and the comments or headings that it writes among them, are the crux of generating interesting

Table 1: Resource file directives.

Directive	Command Line Switch	Example
sort-order	-sn (name) -sa (date_ascending) -sd (date_descending)	sort-order=date_ascending
album-name	-a	album-name=00_vacationpics
no-image-album	-ni	no-image-album=00_vacationpics_textonly
both-album-names	-ani	both-album-names=june_pictures
thumbnail-directory	-t	thumbnail-directory=thumbs
list-file	-l	list-file=picturelist.txt
include-file	-i	include-file=vacation-pics.txt
group-file	-g	group-file=groups.lst
hide-file	-hi	hide-file=hidden-pics.txt
header-file	-H	header-file=intro.html
trailer-file	-T	trailer-file=credits.html
title-string	-t	title-string=Florida Pictures
header-string	-hr	header-string=My Phat Photos
trailer-string	-tr	trailer-string=<h2>Taken by Magnus</h2>
comment-file	-C	comment-file=comments.txt
date-file	-D	date-file=special-dates.txt
glob-pattern	-G	glob-pattern=*.jpg,*.gif,*.tiff
force-back	-b	force-back=true
max-size	-s	max-size=150
num-images	-n	num-images=20
date	-nd	date=no
prune	-np, -fp	prune=yes
output-plugin	-op	output-plugin=darkhtml
plugin-config	-pc	plugin-config=columns:5

albums. In this part, I discuss the *comment file*, *group file*, *date file*, *hide file*, *list file*, and *include file*. For all the examples, I assume that we are dealing with an album that contains five images, IMG-1.JPG...IMG-5.JPG; additionally, we assume that they were created in reverse-alphabetical order, that is, IMG-5.JPG was created before IMG-4.JPG, and so on.

2.1 Comment File

Mt2 has the ability to write comments underneath the name of a picture in an album. Using a *comment file* is the easiest way to tell the program what comment goes with which picture. To give a picture a comment, you write it's name at the beginning of a line, put a colon after it, and then on subsequent lines you write whatever HTML comment you wish, closing it with a period on a line by itself. For example, if we want to comment two of our pictures, we could create a file named `comments.txt`, given below, and then pass it to mt2

with either the `-C` option, or using the *comment-file* resource directive.

sample comments.txt

```
IMG-4.JPG:
This is Martha and I as we were hunting for pickles in the farm
outside Edinburg in 1998.  What <b>big</b> pickles!
.
IMG-3.JPG:
How did that Rottweiler get onto the field!  Run Martha, run!
.
```

2.2 Group File

Let us say that in our album, we want IMG-3.JPG to be shown directly underneath IMG-5.JPG, and for both of the pictures to be grouped with a heading explaining them. None of the sort orders we have available (name, date) will result in those two pictures being adjacent—so we use a group file, and pass it to mt2 using the `-g` option, or the *group-file* resource. To construct a group, write the word “group” at the beginning of a line in the file; if you want a comment, put a color after it, and write the comment as you would in the comment file. Then, list the pictures to be placed in that group, in the order in which you want them to appear in the album, closing the list with the word “endgroup”. Note that you can have more than one group per group-file; just list them one after another. Here is an example:

sample groupfile.txt

```
group:
Martha being chased by a huge dog!
.
IMG-5.JPG
IMG-3.JPG
endgroup
```

After all of your pictures are sorted by whatever criterion you selected, mt2 will read the group file and rearrange the photos to make sure the pictures you indicated are adjacent. In addition, if the group has a comment (as above), it will write before the first picture a heading with the comment, and put a line under the last picture; however, if you don’t want a bottom delimiter for your commented group, put the phrase `--no delimit--` somewhere in your comment—it will be removed before the comment is written to the HTML file, so it can appear anywhere. Also, if the phrase `--trailer--` appears in the comment, it too will be removed, and the comment will appear at the end of the group of pictures. You can use a commented, undelimited group to put a title over or under a specific picture without resorting to a list file.

2.3 Date File

Normally, when mt2 attaches dates to the thumbnail images, it uses the time the picture was last modified to calculate the displayed date; however, there are times when you want

the date shown to be different from the modification time—for instance, if you scan a picture of your trip to Las Vegas from thirty years ago, you probably want the date next to the picture to say 3/6/1972, rather than the date you scanned it. For this purpose you can use a date file, specified by the *-D* flag or *date-file* directive, and the format of which is like a comment file, except that the “picture names” in this case are dates as you want them displayed, and the “comments” are lists of file names, one per line, for which that date should be displayed. Here is a brief example.

sample datefile.txt

```
3/6/1972:
IMG-2.JPG
IMG-4.JPG
.
12/03/2002:
IMG-3.JPG
.
```

There are two notes of which you should be aware: a special date named *none*, and a special filename called *default*. If you mark a file using the above mechanism as having a date of “none”, no date will be displayed with that file, even if dates are turned on; also, if you include the filename “default” within the list for a certain date, that date will be displayed for all files that don’t appear in any other date list. Here is how those two items could be used.

datefile-special.txt

```
none:
IMG-3.JPG
IMG-5.JPG
.
March 23, 2002:
default
.
```

2.4 Hide File

There may be certain pictures or directories (“folders” in the album) that you do not want shown, for instance that one your friend took of you and the sheep with the beautiful eyes, so to make sure they are not shown, create a *hide file*: this is simply a file with a list of the photos and directories you want to leave out of the album, one per line. Pass this file to *mt2* using the *-hi* flag or the *hide-file* directive in the resources.

2.5 List File

The list file is somewhat of a relic, a holdout from the elder days before comment, group, and hide files existed, when we were all *happy* using only list files, and grateful even for that.

Basically, you can use a list file to completely control which pictures are shown, the order in which they are shown, and any comments you want to include (though the comment file is still heeded, if you have one). The downside is that if you add any new pictures to the directory, you must also add them to the list file, which gets to be a pain in the ass after you have a lot of pictures, and don't remember which ones are already in there. Mt2 will generate an album for only those pictures in your list file. Additionally, you can use the special picture name "none", to leave a blank space in the album, with a comment if you'd like. So, for instance, if our list file is

```
sample listfile.txt
none:
These are all the pictures from my European vacation.<br>
All three of them.
.
IMG-2.JPG
IMG-4.JPG:
The funkiest picture ever.
.
IMG-1.JPG
```

mt2 will generate an album containing only IMG-2.JPG, IMG-4.JPG, and IMG-1.JPG, will comment the appropriate ones, and will write the comment associated with "none" at the place it appears in the file, here at the top.

2.6 Include File

Sometimes you want an album to contain only a subset of the pictures in a directory, and if those pictures have diverse names, creating a glob to match all of them would be cumbersome—in this case you can use an *include file*, which is simple a list of filenames, one per line, to be processed. Unlike a list file, an include file does not affect the way the pictures are sorted, nor does it allow you to attach comments to pictures. When you use an include file, pruning is turned off, since we assume that there are many other pictures in the directory.

3 Output Plugins

The format in which mt2 writes its album files is completely controlled by an 'output plugin', which is a small and simple² program that takes picture information from the main program, such as the photo name, thumbnail location, et cetera, and does something with that information, usually writing an HTML output.

²Though not necessarily so—output plugins can be arbitrarily complex, and could be so varied as to write PostScript files or publish albums directly to a networked site.

3.1 Using Plugins

To find out what plugins you have installed, run `mt2 -lp`; once you know which output plugin you want to use, you can pass it to `mt2` using the `-op` option or the `output-plugin` resource directive. The default plugin is *plainhtml*, which has no support or configuration files, and thus should be able to run in all circumstances. Currently, `mt2` also comes with a number of other plugins which use auxiliary graphics files and can be customized with their own configuration files—see the next section for information on how to set them up. Also, for most of the plugins, there are associated plugins that produce an HTML page for each photo.

3.2 Installing New Plugins

If you download a third-party plugin, it should come with its own installation instructions, but here I'll show how to set up the *darkhtml* plugin that comes with `mt2`, because that process should be fairly typical of most other plugins also. Before we begin, please note that there are two directories that will be searched for plugins: the global plugin directory, which is located with the `mt2` program itself; and a user-specific plugin directory, located at `~/mt2/plugins`.

Darkhtml consists of program and configuration files—`darkhtml.py` and `darkhtml.cfg` respectively—which should be placed in the plugin directory, and a `darkhtml_data` subdirectory that contains the graphics files that *darkhtml* references. You should either move or link the `darkhtml_data` directory so that it is available on the Web, and then set the `data_url` key in the `darkhtml.cfg` file to reflect the location of the data directory. For instance, if my `/home/jpavel/www` directory appears on the Web as `http://www.pathwayjr.com/jpavel/`, I can move `darkhtml_data` into my `www` directory, and then set `data_url` to `http://www.pathwayjr.com/jpavel/darkhtml_data` in the configuration file. The number of columns in which *darkhtml* writes its albums is configurable as well.

After all of this, run `mt2` with the `-op darkhtml` switch, and it should work.

3.3 Writing New Plugins

To write a new plugin, create a file in the plugins directory named *plugin-name.py*, where *plugin-name* should be replaced by whatever you want to call your plugin. Plugins are written in Python, so you'll have to know a little bit at least: on the Python website, there is a good tutorial that should fill in everything you need to know. The best way to learn the details of writing an output plugin is to look at `plainhtml.py`, which describes each method that you'd have to implement; a good way to get started is to copy `plainhtml.py` to your new filename, and start editing that. For an example of a plugin that uses external data and parses its own configuration file, examine `darkhtml.py`. There is a utility class, `_htmlgenerator.py`, that can retrofit another plugin with the ability to write an HTML file for each picture, with navigation buttons; look at `dropshadowpages.py` for an example of how to easily create a plugin that uses the `_htmlgenerator`.

4 FAQ & Miscellany

I call this a FAQ even though the questions aren't frequently asked; but they were good ones, so here they go.

Q. It would be nice to denote the groups in the comment file also, so I don't have to type in all the image filenames again in a separate file. How can I do this?

- *Dave Hsu (davhsu@uclink4.berkeley.edu)*

A. In the comment file, just use the group/endgroup directives as you would in a group file, and then pass that one file to mt2 as both the group-file and the comment-file: when mt2 is parsing it as a comment file, it will ignore the group and endgroup 'filenames' (unless you have pictures named "group" and "endgroup") and when it is parsing it as a group file, it ignores the picture comments.

5 Conclusion

I'm not sure what else I can say. Wherever you are running mt2, you probably have access to its source code, so you can tinker with it to your heart's content, making it even more elite. Or elite at all. But really, go read that document about Quiche-Eaters and Real Programmers: it's a riot. Well, until the time comes for mt3, take care.

— *Jesse*

jpavel@alum.mit.edu