Preparing Your Installation

chapter 1

in this chapter

- ☑ Site architecture
- ☑ Maintenance and regular tasks
- ☑ Streamlining your installation
- ☑ Security

Stop worrying. You've chosen Movable Type (MT). It's a remarkably stable, easy to prepare and maintain piece of software. Indeed, part of its charm is that straight out of the box, with nothing added to it, and none of the hacks that make up the rest of this book applied, MT pretty much takes care of itself. Nearly three years of heavy use, plus two major code revisions, and the experience of the weblogging world's finest developers have produced a package that can look after itself.

We're not going to go into detail about how you install Movable Type. Frankly, we think you're able to follow instructions, and Six Apart's documentation is very good in this respect. In addition, the publishers of this book have another, *Movable Type Bible, Desktop Edition*, by Rogers Cadenhead, which deals with the nuts and bolts of installation very admirably.

Instead, we're going to jump straight into the more interesting stuff. First up: Site Architecture.

Site Architecture

Weblogs, by their very frequently updated natures, grow very quickly. It is not uncommon to have sites of more than a thousand pages, and many are 10 times that. Add in comments and TrackBacks, images, feeds, and perhaps some audio and video too, and you'll start to find that a server can get a little messy. Furthermore, everything on the site itself has a URL, and it is common practice for readers to play with the URLs to move around. How many times have you looked around someone's blog archives by changing the URL a little to see what you get?

Therefore, you need to plan a site architecture that will both keep things in order and make for sensible and future-proof URLs that encourage exploration.

Much of the following is based more on art than science.

Images

I like to place all of my images in a separate directory, `/images`. This keeps them organized but also available for any interesting scripting projects I might like to do in the future. To do this consistently, you need to remember that MT's image upload interface will need an extra bit of typing, as shown in Figure 1-1.

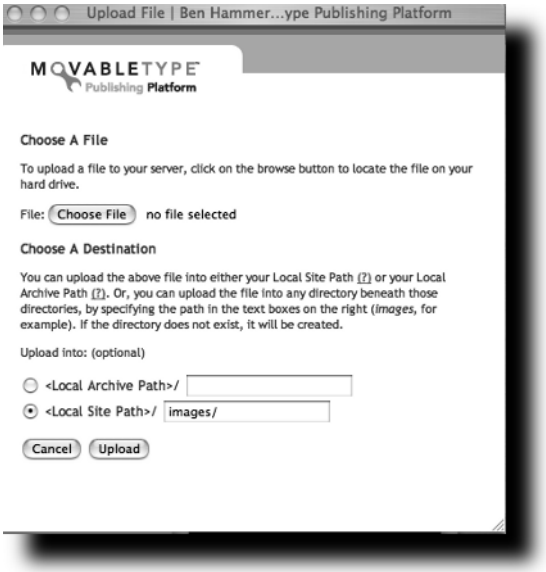

FIGURE 1-1: Using the Upload File dialog box

You will need to do the same in any desktop blogging tool you may be using as well.

Note that you can't move all of the images to this directory. Without an option, MT will automatically place, and replace if it's removed, a file called `nav-commenters.gif` into the root directory of every blog. It's the tiny little person with a speech bubble icon that the default templates use to indicate the commenting link (see Figure 1-2). At the time of writing, you can't stop this file from being replaced.

Archives

With respect to the post archives, things have moved on since versions 1 and 2 of MT. Since 3.0 Movable Type, creating archives occurs in an extremely sensible URL structure (namely, for the individual entry page):

```
Archive_Path/YYYY/MM/DD/Dirified_Entry_Title.html
```

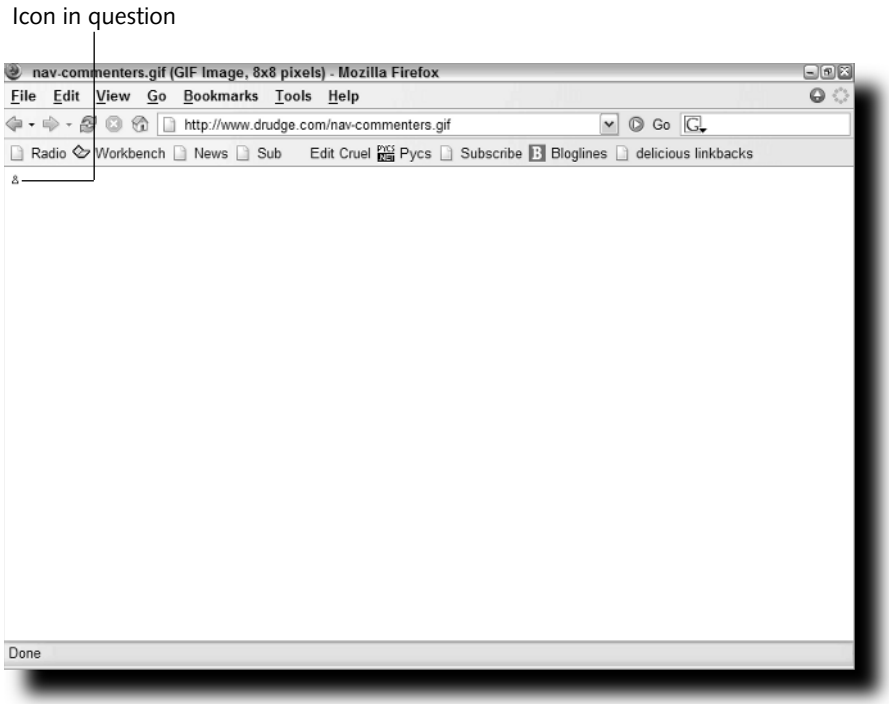

FIGURE 1-2: The Nav-Commenters icon

This is very sensible for two reasons. First, it produces URLs that are independent of the content management system's (CMS) own variables. It might sound strange in a book touting the usefulness of Movable Type, but there's always a possibility that you will change your CMS in the future. Having as neutral a file structure as possible will prove invaluable. Second, the logical structure of the URLs means that people can move around your site from their browser's address bar. Consider the logical positions of all of the types of archive indexes:

- Yearly archives: `Archive_Path/YYYY/index.html`
- Monthly archive: `Archive_Path/YYYY/MM/index.html`
- Complete archive index: `Archive_Path/index.html`

It makes sense to do it like this, as this is exactly how a slightly curious reader will try to look around your site — by deleting bits from the URL and seeing what she finds. There is one exception: currently, the Weekly indexes default to `Archive_Path/week_YYYY_MM_DD.html`, which I do not like. Rather, I would change it to

```
Archive_Path/YYYY/MM/DD-DD.html
```

by adding the following line in the Archive File Template box in the Archive Files configuration page:

```
<$MTArchiveDate format="%Y/%m/%d"$>-<$MTArchiveDateEnd
format="%d"$>.html
```

All this done, you end up with a filesystem that looks like the one shown in Figure 1-3.

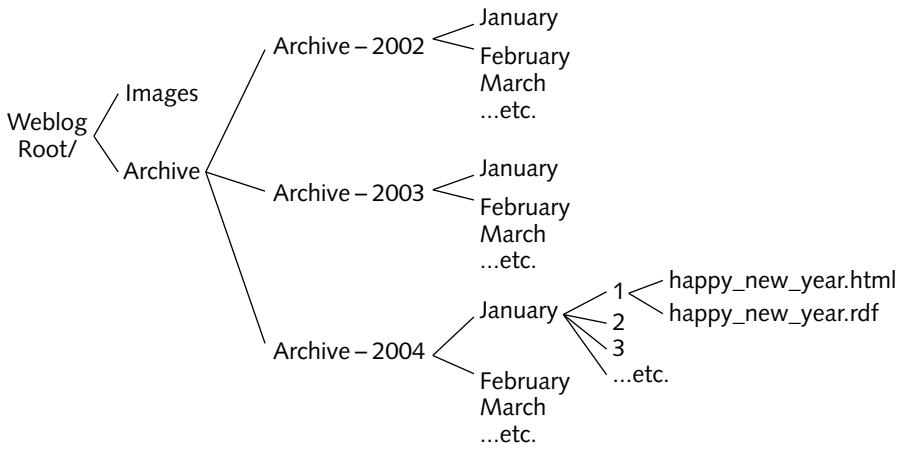

Figure 1-3: Exploring the Archive filesystem

Note that the space for the individual archive is taken by two different files: the standard HTML page and an RSS file for the entry and its comments. As new formats appear, they can fit into the architecture very easily in this same manner.

Maintenance and Regular Tasks

As a professionally produced piece of software running on and with provably reliable platforms, MT really doesn't need any regular maintenance. There aren't any temporary files to remove, or automatically generated crud to delete. However, there are some preventive measures you should take.

Since MT3.1, Movable Type has shipped with a plugin pack containing Sebastian Delmont's TypeMover plugin. This plugin enables you to back up your entire weblog data, including preferences and templates. You are very much advised to install and use this regularly.

Sadly, there appears to be no way to automate the downloading of the backups, so you have to do it manually, but it's very straightforward. The same plugin, incidentally, is very useful if you want to build an MT-based site on a local machine and then move it en masse to a public server. I find this makes templates a whole lot snappier to develop.

Streamlining Your Installation

Right out of the box, Movable Type is already pretty fast. With 3.1's introduction of dynamically built pages, performance has increased a great deal. Even so, and especially if you are not using the dynamic build option, there are a few changes you can make from the default. First, look at what you begin with. A clean installation of MT saves the following in the root directory of the blog:

- Main Index
- Archives Index
- RSS 1.0 Index
- RSS 2.0 Index
- Atom Index
- Stylesheet
- RSD file

Despite the loud brayings of the content syndication communities, you really do not need to be producing both RSS and Atom feeds. Personally, I prefer to produce only one, RSS 1.0, and then use external services to convert it to RSS 2.0 or Atom for the people who really care. (Technically speaking, I do it in this order because RSS 1.0 is the most complicated and data-rich format and so downgrades nicely. I couldn't really go from 2.0 to 1.0, especially when you consider the additional information you can place within the feed after you have visited Chapter 3.) Services such as that found at `www.feedburner.com` are good for this. Either way, you can delete all but one of the feeds straight away.

In addition, turn off the rebuilding of the Stylesheet and RSD files. These do not need to be rebuilt unless you change your design or upgrade your installation, respectively.

Posting Frequency

Consider how often you post to your site and adjust the number of day's posts on the front page to suit. If you're posting multiple times a day, this should be set pretty low. If you're posting only once a month, make it high. The risk is that you will have either an enormously large front page, or, should you not post for a while and then have a comment cause a rebuild, a completely empty one. Neither is good — you should pay attention to this if you're going on holiday, for example. I have been caught out with a setup of "Display 10 days' worth of posts," when on day 11 someone left a comment on an old entry. The front page rebuilt and was left empty for a week. In the spider-filled ecosystem of the web, a week's worth of an empty front page can cause terrible malady, not the least of which is a loss of stumble-upon readership.

If you are very committed to a minimalist filesystem, you can delete the RSD file altogether. The file makes the setting up of offline editing tools a few seconds faster, but if you remember the path to your `mt-xmlrpc.cgi` file and your blog ID, you actually don't need it. Mine is history.

Relying on Third-Party Services

Consider moving all web-services-based things out of your templates and into other files pulled in by server side includes. If you are using MT, specifically MT-PerlScript, to pull data from another site while you rebuild your indexes, you will slow the process down considerably. It also helps, in these sorts of scripts or plugins, to use as much caching as possible. A bad day of network congestion or a slow response time from the remote server might even kill your rebuild process. The more recent plugins, such as Tim Appnel's MT-Feed, take this into account, and plugin developers should give serious thought to any potential slowing effects their plugin might have on page rebuilds.

Ping and Comment Time-Outs

This slowing effect is particularly noticeable with comments and TrackBacks. MT installations with slow rebuilds will find that their readers leave the same comment two or three times, believing the first attempt to have failed when the browser timed out. TrackBacks, too, can time out, meaning that the remote server doesn't know it was successful. Automatic trackbacking then tries again the next time the remote site is itself rebuilt. By improving the chances of the rebuilds happening quickly, you will stop these repeated attempts.

For TrackBacks, you can edit `mt.cfg` to increase the time-out interval to allow other sites to be slow when you TrackBack to them. Simply uncomment the following line:

```
# PingTimeout 20
```

The number is the time, in seconds, with the default being 15. But 20 is better, and 30 just right.

Temp Files

Movable Type produces and saves temporary files to disk during page rebuilds. You can turn this off, which speeds up rebuilds considerably, albeit at the expense of server memory. If you believe your machine is big enough to deal with it (and it most probably is, to be honest), edit `mt.cfg` and find this line:

```
# NoTempFiles 1
```

Uncomment it, like so:

```
NoTempFiles 1
```

Save the config file again. Obviously, this will have no effect at all on post-version 3.1 dynamically produced pages.

Security

By now, you should have read the install documents and deleted `mt-load.cgi` and the mt-upgrade scripts and removed the Melody/Nelson identity. For security purposes, you should take a couple of other steps as well.

Installing MT-Blacklist

Next, you must install MT-Blacklist. It comes within the plugin pack included with MT3.1, and its workings are covered elsewhere in this book. MT-Blacklist will keep the vast majority of comment spammers at bay, and it works well to stop multiple copies posted by visitors to flood a site. As I write this chapter, this past week my own site has been hit by over 1,000 attempted comment spams, all of them stopped by MT-Blacklist. It is extremely necessary to install it.

Then, as the installation instructions suggest, but right at the bottom where it tends to be overlooked, you should protect your `mt.cfg` file by adding the following to the `.htaccess` file within the directory `mt.cfg` is found:

```
<Files mt.cfg>
    <Limit GET>
    deny from all
    </Limit>
</Files>
```

This will prevent anyone from looking at your settings from the open web.

SuExec

The most extreme method of securing your installation is to use Apache enabled with SuExec. SuExec enables CGI scripts to run under an individual's user ID, meaning that you don't need to set the folder permissions to 777 as before. By eschewing this, you lock the directories down.

Currently, Apache does not have SuExec enabled by default: You need to enable it yourself or, more likely, ask your system administrator or hosting company to do it for you (this explanation is beyond the scope of this book). The truly interested can look here: `http://httpd.apache.org/docs-2.0/suexec.html`.

Once SuExec is up and running, you need to tell MT to take advantage of it. This means changing `mt.cfg`. Backup your system first, and then scroll through the file for these lines:

```
# DBUmask 0022
# HTMLUmask 0022
# UploadUmask 0022
# DirUmask 0022
```

Uncomment them to the following:

```
DBUmask 0022
HTMLUmask 0022
UploadUmask 0022
DirUmask 0022
```

Then find this section:

```
# HTMLPerms 0777
# UploadPerms 0777
```

Again, uncomment the lines like so:

```
HTMLPerms 0644
UploadPerms 0644
```

These changes will enable MT to work within the secure constrains of SuExec, and you won't have to make your folders world-writable.

Summary

The experience of using a content management system such as Movable Type is a "Eureka!" moment for most web publishers. There's no better way to create new content and edit existing pages than an effective CMS, which makes the old way of editing pages by hand in a text editor seem vastly inferior.

Movable Type removes the need for hand-editing toil on weblogs and other rapidly changing websites. In the hands of a Movable Type hacker, the software can be extended into specialized areas or used to develop new kinds of sites entirely.