This page briefly describes PmWiki's settings for file and directory permissions in a typical Unix environment.

Simple installation (out of the box)

First, let's look at PmWiki without any cookbook scripts loaded. PmWiki needs to be able to write into the

- *wiki.d/* directory to be able to save pages
- *uploads/* directory to save uploads.

Those are the *only* directories that need to be writable by the webserver. It doesn't matter to PmWiki who owns or creates those directories, as long as it has write permission to them.

Everything else should be owned by the account holder, and readable by the webserver account (but normally not writable by the webserver account).

That's it -- everything else depends on the specific PHP configuration and running environment, which is detailed below (and which is why there isn't a definitive answer that applies to every situation). But the above two rules are absolute and answer 95% of the questions about directory permissions.

On a Unix host the webserver typically runs with a userid and groupid that is different from the account holder. Usually the webserver account is something like "nobody", "apache", "www", or "httpd". Thus, in a standard installation, the account holder manually creates the wiki.d/ and uploads/ directories, and sets the permissions on the directories to be world-writable in order for PmWiki (running as the webserver account) to be able to create files there.

```
$ pwd
/home/pmichaud/public_html/pmwiki
$ mkdir uploads
$ mkdir wiki.d
$ chmod 777 uploads wiki.d
$ ls -ld . uploads wiki.d
drwxr-xr-x   12 pmichaud pmichaud     1024 Feb 10 11:51 .
drwxrwxrwx    8 pmichaud pmichaud     1024 Jan 23 11:58 uploads
drwxrwxrwx    2 pmichaud pmichaud    54272 Feb 10 15:29 wiki.d
```

Avoiding world-write directories

However, lots of people don't like having those world-writable (rwx) permissions on the directories. The only practical way to eliminate the world write permissions is if we can get the webserver and account holder to be the owner and group of the directories and the files within them. Since Unix typically doesn't allow non-superusers to change ownerships of files or directories that already exist, we have to make sure they are created with the correct ownerships in the first place.

To get the directories to be owned by the webserver account, we let PmWiki take care of creating them. This means we temporarily grant write permission to the parent, and then execute PmWiki to allow it to create the directories. However, we also want the newly created directories to have the same group as the account holder, so the account holder can remove or manipulate files in the directories. Therefore, we use Unix's *setgid* capability (2777 or 'rws' permissions) to cause all newly created files to inherit the same group as the parent.

To avoid world-write directories, use the following instructions **instead of** the instructions above. If you already have created the *wiki.d/* and *uploads/* directories, use chown and chmod to match the following results.

```
$ pwd
/home/pmichaud/public_html/pmwiki
$ chmod 2777 .
$ ls -ld .
drwxrwsrwx  10 pmichaud pmichaud 4096 May 28 09:55 .
# <-- execute pmwiki.php script from web browser -->
$ ls -ld . uploads wiki.d
drwxrwsrwx  10 pmichaud pmichaud 4096 May 28 09:55 .
drwxrwsr-x   2 nobody   pmichaud 4096 May 28 09:55 uploads
drwxrwsr-x   2 nobody   pmichaud 4096 May 28 09:55 wiki.d
$ chmod 755 .
drwxr-xr-x  10 pmichaud pmichaud 4096 May 28 09:55 .
drwxrwsr-x   2 nobody   pmichaud 4096 May 28 09:55 uploads
drwxrwsr-x   2 nobody   pmichaud 4096 May 28 09:55 wiki.d
```

Now the two directories are owned by 'nobody', which means the webserver can write to them. We don't have world-writable permissions on the directories, and the account holder (pmichaud) still has write permissions to the files and directories by virtue of the group ownership and permissions. The setgid bit also ensures that any files or subdirectories created within *uploads/* or *wiki.d/* will belong to the same (pmichaud) group.

Safe mode

HOWEVER, if a site is running in PHP's "safe_mode", then the "let PmWiki create the directories" solution doesn't work, as PHP will only create files in directories that are owned by the same user that owns the pmwiki.php script itself. Thus, PmWiki (apache) *cannot* create the directories in this case, or safe_mode will complain when PmWiki attempts to write a file into those directories. The *only* way for things to work in safe_mode is to manually create the needed directories and set their permissions to 777, as outlined at the beginning of this section.

PHP running as script owner

There are some webservers and PHP installations that are configured to run a PHP script with the same identity as the owner of the script. This is often called "suexec PHP" or even just "suPHP". In this case, since the PmWiki script ends up running with the same identity as the account holder, then everything "just works" out of the box without doing anything manually. PmWiki creates any directories and files as needed, each owned by the account holder, and permissions aren't generally an issue at all.

Cookbook scripts

Okay, now let's look at cookbook scripts. If a cookbook script has files that it wants to make available to browsers, such files should generally be placed somewhere within the 'pub/' hierarchy and referenced via '`$PubDirUrl`'.

If a cookbook recipe needs to *write* files to disk, then the same rules apply to that directory as for the wiki.d/ and uploads/ directories above, with the exact ownerships and permissions depending on the webserver and PHP configuration. In general the cookbook recipe should do the same as PmWiki, and just call PmWiki's mkdirp($dir) function. PmWiki will then take care of creating the directory (if it can) or prompting for its creation as appropriate.

For example, if cookbook recipe 'frobot' wants to distribute a .css file, then that file should go somewhere like pub/css/frobot.css or pub/frobot/frobot.css. The directories and files in this case should be created and owned by the account owner, since the cookbook recipe doesn't need to create or modify any of the files when it runs.

As an alternate example, the Cookbook:MimeTeX recipe wants to be able to create cached images for the math markup, and those images need to be available to the browser. Thus, MimeTeX uses a pub/cache/ directory, which should be created in whatever manner was used to create the wiki.d/ and uploads/ directories (i.e., according to the webserver and PHP configuration). Again, Cookbook:MimeTeX just solves this by calling mkdirp("pub/cache"), and letting that function create the directory or prompt the administrator for the appropriate action based upon the server settings encountered.

See also

- Cookbook:Directory and file permissions