PmWiki

toc top

PmWiki is a wiki-based system for collaborative creation and maintenance of websites.

PmWiki pages look and act like normal web pages, except they have an "Edit" link that makes it easy to modify existing pages and add new pages into the website, using basic editing rules. You do not need to know or use any HTML or CSS. Page editing can be left open to the public or restricted to small groups of authors.

Key PmWiki Features

Custom look-and-feel: A site administrator can quickly change the appearance and functions of a PmWiki site by using different skins and HTML templates. If you can't find an appropriate skin already made, you can easily modify one or create your own.

Access control: PmWiki password protection can be applied to an entire site, to groups of pages, or to individual pages. Password protection controls who can read pages, edit pages, and upload attachments. PmWiki's access control system is completely self-contained, but it can also work in conjunction with existing password databases, such as *.htaccess*, LDAP servers, and MySQL databases.

Customization and plugin architecture: One principle of the PmWikiPhilosophy is to only include essential features in the core engine, but make it easy for administrators to customize and add new markup. Hundreds of features are already available by using extensions (called "recipes") that are available from the PmWiki Cookbook.

PmWiki is written in PHP and distributed under the General Public License. It is designed to be simple to install, customize, and maintain for a variety of applications. This site is running pmwiki-2.2.99.

PmWiki is a registered trademark of Patrick R. Michaud. Since January 2009 PmWiki is actively maintained by Petko Yotov under the oversight of Dr Michaud.

PmWiki's home on the web is at www.pmwiki.org.

Last modified by Petko on February 03, 2013. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PmWiki

SideBar

toc top

Pm Wiki	Access Keys	Audiences	Auth User
Available Actions	Backup and Restore	Basic PmWiki editing rules	Basic Variables
Block Markup	Blocklist	Categories	Change Log
Conditional Markup	Contact us	Contributors	Creating New Pages
Custom Inter Map	Custom Markup	Custom Wiki Styles	Debug Variables
Deleting Pages	Design Notes	Documentation Index	Drafts
Edit Variables	FAQ	File Permissions	Fmt Page Name
Forms	Functions	Glossary	Group Customizations
GroupHeaders and GroupFooters	I18n Variables	Images	Include Other Pages
Initial Setup Tasks	PmWiki Installation	Inter Map	Internationalizations
Introduction	Layout Variables	Link Variables	Links
Local Customizations	Mailing Lists	Markup Expressions	Markup Master Index
Notify	Other Variables	Page Directives	Page File Format
Page History	Page List Templates	Page Lists	Page Text Variables
Page specific variables	Pagelist Variables	Passwords	Passwords Admin
Path Variables	Patrick Michaud	Per Group Customizations	Pm Wiki Philosophy
Ref Count	Release Notes	Requirements	Search
Security	Security Variables	Simultaneous Edits	Site Page Actions
Site Preferences	Skin Templates	Skins	Special Characters
Table directives	Tables	Text Formatting Rules	Troubleshooting
UTF-8	Upgrades	Upgrading From Pm Wiki 1	Upload Variables
Uploads	Uploads Administration	Url Approvals	Variables
Version	Web Feeds	Wiki Administrator	Wiki Farm Terminology
Wiki Farms	Wiki Group	Wiki Groups	Wiki Page
Wiki Sandbox	Wiki Structure	Wiki Style Examples	Wiki Styles
Wiki Trails	Wiki Wiki Web	Wiki Word	Wiki Words

Last modified by on July 11, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/SideBar

AccessKeys

toc top

Access keys (See also Wikipedia:access keys) are keyboard shortcuts for tasks that would otherwise require a mouse click. They are part of markup that may exist on any webpage. On PmWiki steps have been taken to make it easier to use access keys throughout a site, and to make it possible to adjust key assignments to accommodate different languages and preferences.

Using access keys in different operating systems and browsers

Access keys require you to hold down two or more keys.

- On Windows with Internet Explorer, press ALT + the access key.
- With Firefox, press SHIFT + ALT + the access key.
- On a Macintosh with Firefox, Omniweb, Internet Explorer, press Ctrl + the access key.
  - With Safari (Version 4.0.2) press Ctrl + Option + the access key.
- With Opera press Shift+Esc to enter (or exit) access-key mode.
- With Konqueror, press Ctrl to enter (or exit) access-key mode.
- With Chrome, press SHIFT + ALT + the access key

Exceptions exist for specific browsers, and specific versions. For example,

- Internet Explorer requires that the Enter key be pressed at the end of the sequence for versions 5 and up under Windows, but not under Macintosh (where access keys were not supported until after version 4.5).
- Firefox versions 1.5 and earlier simply use Alt, while Firefox version 2.0 uses Shift+Alt.

Note, in cases of conflicts between the keyboard shortcuts assigned by browsers and access keys assigned by links and other markup on webpages, many browsers, including Mozilla, Netscape and Internet Explorer, allow access keys to override the browser defaults and require a different sequence to continue using overridden browser assignments (typically, by pressing and releasing the Alt key, instead of holding it down).

Access key assignments in this PmWiki installation

The following is a list of the currently defined access keys for built-in actions. Remember that the letters identified below must be used together with the combination listed above (depending on your operating system and browser). Note that some actions do not have a corresponding access key by default.

Key Name	Key Value	Function
ak_view		view
ak_edit	e	edit
ak_history	h	history
ak_attach		attach
ak_print		print
ak_backlinks		backlinks
ak_logout		logout
ak_recentchanges	c	recent changes
ak_save	s	save or publish page
ak_saveedit	u	save and keep editing
ak_savedraft	d	save draft
ak_preview	p	preview page
ak_textedit	,	jump to edit textarea
ak_em		*emphasize* text
ak_strong		**strong** text

Note: If the 'Key Value' is the same as the 'Key Name', the access key is currently undefined.

When can these access keys be used

- Access keys ak_view, ak_edit, ak_history, ak_attach, ak_print, ak_backlinks, ak_logout and ak_recentchanges can be used all the time
- Access keys ak_save, ak_saveedit, ak_savedraft, ak_preview, ak_textedit can only be used in edit mode

Following table explains which button is activated by which access key. Note that the Cancel button has no access key.

Standard Edit mode	Draft Edit mode	Used Access Key
Save	Publish	ak_save
	Save draft	ak_savedraft
Save and edit	Save draft and edit	ak_saveedit
Preview	Preview	ak_preview

- Access keys ak_em and ak_strong work only in edit mode**and** when the GUIbuttons are enabled in local/config.php.

admins (intermediate)

Customizing access keys

PmWiki uses the same "phrase translation" methods for access key mappings as it does for internationalization. This makes it possible for administrators, skins, language translators, and visitors to all influence the way that specific keys are mapped to actions.

See SitePreferences and Site.Preferences for more information and a template.

Note that some skins (e.g., Lean) don't use the translation mechanism. In this case one must edit the template file itself in order to change the access keys.

By convention, the translation phrases for all of the access key actions start with the characters "ak_", so that the page variable "`$[ak_edit]`" is replaced by the access key for editing as defined by the current preferences, language, skin default, or site default.

Implementation of access keys

Access keys are implemented in html as optional parameters that can be added to links and many other types of markup.

Example: `<a href="http://example.com" accesskey="x">Example</a>` would create a link to example.com that could be triggered by clicking on the linked word "example" or using the access key Akey+x. That same action key link could be created in PmWiki markup by typing `%accesskey="x"%[[http://example.com|Example]]%%`, like this: Example. Try it and see if it works. Note that this AKey+x access key only works this way on this page, because it is simply a shortcut for accessing the link that exists only on this page.

The list of access key assignments in default PmWiki installations generally work throughout a site because links have been created in PmWiki skins and editing screens that incorporate access key parameters using the access key translation phrases. One location where those links can be viewed is Site.PageActions. That page contains the links that the default PmWiki skin, and many other skins, use to generate links such as "View" "Edit" and "History" that appear on most pages (other than editing screens). Each of the links in that page also has an `%accesskey=$[ak_xxx]%` declaration in front of it, which enables a specific access key for that link.

How can I change the keyboard shortcuts for editing and saving a page?

See Customizing access keys.

Last modified by mfwolff on March 31, 2016. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/AccessKeys

Audiences

toc top

This page contains Patrick Michaud's comments regarding the "audiences" for which PmWiki was designed. As such, many people are reluctant to modify the page, because it is a statement of his opinions and describes some of the thought that went into creating PmWiki. (And we all thank him for that!)

Patrick's comments

I think of PmWiki in terms of two audiences:

- *Authors* are the people who generate web content using PmWiki, and
- *wiki administrators* are the folks who install, configure, and maintain a PmWiki installation on a web server.

In some senses it could be claimed that as the primary developer of PmWiki I should only have wiki administrators as my target audience, and that authors are the target audience for the administrators. But what really makes PmWiki useful to wiki administrators is that I've put a lot of consideration into creating a tool that is usable by authors, so I have to keep the needs of both audiences in mind as I'm designing and adding new features to PmWiki.

Within the authoring audience I see that there are "naive authors" and "experienced authors".

"Naive authors" are the folks who use wiki to generate content but may know next-to-nothing about HTML, much less style sheets or PHP or the like. Naive authors are easily discouraged from generating web content if they have to wade through markup text that has lots of funny and cryptic symbols in them. So, if we want a site with lots of contributors, we have to be very careful not to do things that will cause this group to exclude themselves from participating.

"Experienced authors" are the folks who know a lot about HTML and could write their content as HTML, but have chosen to use wiki because of its other useful features (ease of linking, collaboration, ease of updates, revision histories, etc.) or because they want to collaborate with naive authors. Experienced authors usually don't have any problem with documents with lots of ugly markup in them; after all, they already know HTML. Experienced authors are sometimes frustrated with wiki because it doesn't have markup that would let them do something they know they can do in HTML (e.g., tables, stylesheets, colored text, etc.). And, they sometimes have difficulty understanding why naive authors would turn away from documents that have lots of markup sequences in them.

For the wiki administrator audience--the folks who install and may want to customize PmWiki--their backgrounds and goals are often quite diverse. PmWiki is designed so that it can be installed and be useful with minimal HTML/PHP knowledge, but it doesn't restrict people who know HTML/PHP from doing some fairly complex things. For one, PmWiki allows a site administrator to build-in markup sequences and features customized to his/her needs (and the needs of his/her audiences).

The separate needs of these audiences are behind most of the PmWikiPhilosophies. The people who develop PmWiki software must continually keep naive authors in mind as new features are requested and proposed by expert authors and Wiki Administrators. Sometimes it may seem to these latter groups that it's okay to implement the complex features because "naive authors don't have to use them", but the truth is that if complex/ugly markup sequences are available then they will eventually be used by someone, and once used they become a barrier to the naive authors. So, if I see that a feature could become a barrier to a naive author I don't include it in the base implementation of PmWiki, but instead find ways to let Wiki Administrators include it as a local customization.

AuthUser

AuthUser is PmWiki's identity-based authorization system that allows access to pages to be controlled through the use of usernames and passwords. AuthUser can be used in addition to the password-based scheme that is PmWiki's default configuration.

AuthUser is a very flexible system for managing access control on pages, but flexibility can also bring complexity and increased maintenance overhead to the wiki administrator. This is why PmWiki defaults to the simpler password-based system. For some thoughts about the relative merits of the two approaches, see PmWiki:ThoughtsOnAccessControl.

See also: Cookbook:Quick Start for AuthUser.

Activating AuthUser

To activate PmWiki's identity-based system, add the following line to*local/config.php*:

```
include_once("$FarmD/scripts/authuser.php");
```

Ensure that you have set a site wide admin password, otherwise you will not be able to edit SiteAdmin.AuthUser.

> Note: Older versions of PmWiki (before 2.2.0-beta58) use *Site.AuthUser*.

PmWiki caches some group and page authorization levels when a page is accessed. For this reason, it is better to include `authuser.php` quite early in config.php, notably

- after any recipe which inserts some custom writable PageStore class (MySQL, SQLite, Compressed PageStore or other)
- and after any internationalization (UTF-8 and XLPage).

(If you don't use a custom PageStore class and i18n, include `authuser.php` first thing in `config.php`.)

All other recipes should be included after these.

Creating user accounts

Most of AuthUser's configuration is performed via the SiteAdmin.AuthUser page. To change the AuthUser configuration, simply edit this page like any other wiki page (you'll typically need to use the site's admin password for this).

To create a login account, simply add lines to SiteAdmin.AuthUser that look like:

```
username: (:encrypt password:)
```

For example, to create a login account for "alice" with a password of "wonderland", enter:

```
alice: (:encrypt wonderland:)
```

When the page is saved, the "`(:encrypt wonderland:)`" part of the text will be replaced by an encrypted form of the password "wonderland". This encryption is done so that someone looking at the SiteAdmin.AuthUser page cannot easily determine the passwords stored in the page.

To change or reset an account's password, simply replace the encrypted string with another `(:encrypt:)` directive.

The password cannot contain spaces, tabs, new lines, columns ":" and equals "="; on some systems it should contain at least 4 characters. Usernames and passwords are case sensitive, eg. "User" is not the same as "user".

Controlling access to pages by login

Pages and groups can be protected based on login account by using "passwords" of the form `id:username` in the password fields of `?action=attr` (see PmWiki.Passwords). For example, to restrict a page to being edited by Alice, one would set the password to "`id:alice`".

It's possible to use multiple "id:" declarations and passwords in the `?action=attr` form, thus the following setting would allow access to Alice, Carol, and anyone who knows the password "quick":

```
quick id:alice,carol
```

To allow access to anyone who has successfully logged in, use "`id:*`".

One can also perform site-wide restrictions based on identity in the `$DefaultPasswords` array: e.g.

```
# require valid login before viewing pages
 $DefaultPasswords['read'] = 'id:*';
# Alice and carol may edit
 $DefaultPasswords['edit'] = 'id:alice,carol';
# All admins and Fred may edit
 $DefaultPasswords['edit'] = array('@admins', 'id:Fred');
```

You can change the `$DefaultPasswords` array in local customization files such as:

- local/config.php (for entire wiki)
- farmconfig.php (for entire wikifarm)

Organizing accounts into groups

AuthUser also makes it possible to group login accounts together into authorization groups, indicated by a leading "@" sign. As with login accounts, group memberships are maintained by editing the SiteAdmin.AuthUser page. Group memberships can be specified by either listing the groups for a login account (person belongs to groups) or the login accounts for a group (group includes people). You can repeat or mix-and-match the two kinds as desired:

```
@writers: alice, bob
carol: @writers, @editors
@admins: alice, dave
```

Then, to restrict page access to a particular group, simply use "`@group`" as the "password" in `?action=attr` or the `$DefaultPasswords` array, similar to the way that "`id:username`" is used to restrict access to specific login accounts.

Excluding individuals from password groups

Group password memberships are maintained by editing the SiteAdmin.AuthUser page. To specify a password group that allows access to anyone who is authenticated, you can specify:

```
@wholeoffice: *
```

If you need to keep "Fred" out of this password group :

```
@wholeoffice: *,-Fred
```

To allow all users except Fred to change page attributes, for example, you can add to config.php :

```
 $DefaultPasswords['attr'] = array('id:*,-Fred');
```

Getting account names and passwords from external sources

The AuthUser script has the capability of obtaining username/password pairs from places other than the SiteAdmin.AuthUser page, such as passwd-formatted files (usually called '.htpasswd' on Apache servers), LDAP servers, or even the *local/config.php* file.

Passwd-formatted files (.htpasswd/.htgroup)

Passwd-formatted files, commonly called *.htpasswd* files in Apache, are text files where each line contains a username and an encrypted password separated by a colon. A typical *.htpasswd* file might look like:

```
alice:vK99sgDV1an6I
carol:Q1kSeNcTfwqjs
```

To get AuthUser to obtain usernames and passwords from a *.htaccess* file, add the following line to SiteAdmin.AuthUser, replacing "/path/to/.htpasswd" with the filesystem path of the *.htpasswd* file:

```
htpasswd: /path/to/.htpasswd
```

Creation and maintenance of the *.htpasswd* file can be performed using a text editor, or any number of other third-party tools available for maintaining *.htpasswd* files. The Apache web server typically includes an *htpasswd* command for creating accounts in .htpasswd:

```
$ htpasswd /path/to/.htpasswd alice
New password:
Re-type new password:
Adding password for user alice
$
```

Similarly, one can use *.htgroup* formatted files to specify group memberships. Each line has the name of a group (without the "@"), followed by a colon, followed by a space separated list of usernames in the group.

```
writers: carol
editors: alice carol bob
admins: alice dave
```

Note that the groups are still "@writers", "@editors", and "@admins" in PmWiki even though the file doesn't specify the @ signs. To get AuthUser to load these groups, use a line in SiteAdmin.AuthUser like:

```
htgroup: /path/to/.htgroup
```

Configuration via *local/config.php*

AuthUser configuration settings can also be made from the*local/config.php* file in addition to the SiteAdmin.AuthUser page. Such settings are placed in the $AuthUser array, and *must be set prior to including the*authuser.php *script*. Some examples:

```
# set a password for alice
$AuthUser['alice'] = pmcrypt('wonderland');
# set a password for carol
$AuthUser['carol'] = '$1$CknC8zAs$dC8z2vu3UvnIXMfOcGDON0';
# define the @editors group
$AuthUser['@editors'] = array('alice', 'carol', 'bob');
# Use local/.htpasswd for usernames/passwords
$AuthUser['htpasswd'] = 'local/.htpasswd';
# Use local/.htgroup for group memberships
$AuthUser['htgroup'] = 'local/.htgroup';
```

Configuration via LDAP

Authentication can be performed via an external LDAP server -- simply set an entry for "ldap" in either SiteAdmin.AuthUser or the *local/config.php* file.

```
# use ldap.airius.com for authentication
$AuthUser['ldap'] = 'ldap://ldap.airius.com/ou=People,o=Airius?cn?sub';
```

Make sure to include AuthUser below the entry for the ldap server:

```
# Want to use AuthUser so we can use ldap for passwords
include_once(" $FarmD/scripts/authuser.php");
```

And remember to assign the Security Variables for edit and history (or whatever):

```
#Security Variables set login for edit & history page
# to let anyone edit that has an ldap entry:
 $HandleAuth['diff'] = 'edit';
 $DefaultPasswords['edit'] = 'id:*';
 $Author =  $AuthId;
```

LDAP authentication in AuthUser closely follows the model used by Apache 2.0's mod_auth_ldap module; see especially the documentation for AuthLDAPUrl for a description of the url format.

For servers that don't allow anonymous binds, AuthUser provides `$AuthLDAPBindDN` and `$AuthLDAPBindPassword` variables to specify the binding to be used for searching.

See also Cookbook:AuthUser via Microsoft LDAP

Setting the Author Name

By default, PmWiki will use a login name in the Author field of the edit form, but allows the author to change this value prior to saving. To force the login name to always be used as the author name, use the following sequence in config.php to activate AuthUser:

```
include_once(" $FarmD/scripts/authuser.php");
 $Author =  $AuthId; # after include_once()
```

To allow more flexibility, but still enable changes to be linked to the authorized user, one can give the author name a prefix of the `$AuthId` instead:
```
include_once("$FarmD/scripts/author.php");
include_once("$FarmD/scripts/authuser.php");
if ($Author) {
 if (strstr($Author, '-') != false) {
     $Author = "$AuthId-" . preg_replace('/^[^-]*-/', '', $Author);
 } else if ($Author != $AuthId) {
     $Author = $AuthId . '-' . $Author;
 } else {
     $Author = $AuthId;
 }
    } else {
 $Author = $AuthId;
    }
    $AuthorLink = "[[~$Author]]";
```
The above will allow the user to put in the author name of their choice, but that will always be replaced by that name prefixed with " `$AuthId`-". The reason why $AuthorLink needs to be set is that, if it isn't, the RecentChanges page will have the wrong link in it.

Removing the "Author" edit field

To force users to edit with their AuthID instead of having a field they can place any name in. This enables administration to keep track of who is doing what better. This line also links the Author name to their Profile.
Go to Site.EditForm, remove the line
`$[Author]: (:input e_author:)`
or replace it with
`$[Author]: [[Profiles/{$Author}]]`

Authorization, Sessions, and WikiFarms

PmWiki uses PHP sessions to keep track of any user authorization information. By default PHP is configured so that all interactions with the same server (as identified by the server's domain name) are treated as part of the same session.

What this means for PmWiki is that if there are multiple wikis running within the same domain name, PHP will treat a login to one wiki as being valid for all wikis in the same domain. The easiest fix is to tell each wiki to have use a different "session cookie". Near the top of a wiki's *local/config.php* file, before calling authuser or other recipes, add a line like:

```
        session_name('XYZSESSID');
```

The XYZSESSID can be any unique name (letters only is safest).

See Also

- PmWiki.Passwords
- PmWiki.PasswordsAdmin
- Cookbook:AuthUser for tips and tricks
- SiteAdmin.AuthUser

Can I specify authorization group memberships from with *local/config.php*?

Yes -- put the group definition into the $AuthUser array (in config.php):

```
        $AuthUser['@editors'] = array('alice', 'carol', 'bob');
```

Can I have multiple admin groups?

Yes, define the groups with `array('@admins', '@moderators');` like this:

```
$DefaultPasswords['admin'] = array( pmcrypt('masterpass'), # global password
  '@admins', '@moderators', # +users in these groups
  'id:Fred', 'id:Barney');  # +users Fred and Barney
```

I'm running multiple wikis under the same domain name, and logins from one wiki are appearing on other wikis. Shouldn't they be independent?

This is caused by the way that PHP treats sessions. See PmWiki.AuthUser#sessions for more details.

Is there any way to record the time of the last login for each user when using AuthUser? I need a way to look for stale accounts.

See Cookbook:UserLastAction.

Though every setting seems correct, authentication against LDAP is not working. There is nothing in ldap log, what's wrong?

Be sure ldap php module is installed (on debian apt-get install php(4|5)-ldap ; apache(2)ctl graceful)

The login form asks for username and password, but only password matters.

Username can be left blank and it still signs in under the account. Is this intentional and if so, can I change it so that the username and password must both be entered? - X 1/18/07 Never mind I think this has something to do with using the admin password. I created a test account and it's working ok.

Make sure you are not entering the admin password when testing the account because, if the password is equal to the admin password, it will authenticate directly through the config.php file and skip any other system.

Do note that even with AuthUser activated you can still log in with a blank username and only entering the password. In that case any password you enter will be "accepted" but only passwords which authenticate in the given context will actually give you any authorization rights. Using this capability AuthUser comfortably coexists with the default password-based system.

If you want to require both username and password, then you need to set an admin id**before** including authuser.php:

```
## Define usernames and passwords.
$AuthUser['carol'] = '$1$CknC8zAs$dC8z2vu3UvnIXMfOcGDON0';

## Enable authentication based on username.
include_once('scripts/authuser.php');

# $DefaultPasswords['admin'] = pmcrypt('secret');
$DefaultPasswords['admin'] = 'id:carol';
```

A username and password will then be required before login is successful.

Is there any way to hide IP addresses once someone has logged in so that registered users can keep their IP addresses invisible to everyone except administrators? - X 1/18/07

Yes, see solution provided at PITS:00400.

Is there a way that people could self-register through AuthUser?

You can see HtpasswdForm or UserAdmin for recipes providing this feature.

I would like it that after I have AuthUser turned and a user is authenticated to get on my site, that if I have a password put on a particular page or group that they don't get the AuthUser form to show up (username and password), but only the typical field for password?

See this thread of the mailing list

AvailableActions

Page actions are applied to wiki pages, as a query string appended to the URL. Security can be applied to all default actions, and script actions with one exception, but not diag actions, through the use of passwords.

Also documented are all other URL queries.

NOTE: All actions will be disabled if the following is set:

```
$EnableActions = 0;
include('pmwiki.php');
```

This will initialize PmWiki (along with any configuration/customizations that are being made, e.g. from local/config.php), but won't actually perform any actions. The caller can then call the desired action or other functions as desired. This is available from Version 2.2.0-beta22 on up.

PmWiki Actions

See also site page actions.

?action=**attr**
: displays dialog for setting/changing password of the specified page or group of pages, see passwords, see also `$EnablePostAttrClearSession` if you do not want to have the session cleared after validating changeGeneral use of passwords and login

?action=**browse**

display the specified page (default action if no `?action=` is present)

?action=**crypt**
: displays a form for generating hashed passwords out of clear text for usage in your config.php

?action=**diff**
: show a change history of the specified page, see page history History of previous edits to a page

?action=**download**&upname=*file.ext*
: retrieve the page's attachment named *file.ext*, see `$EnableDirectDownload`

?action=**edit**
: edit the specified page, see basic editing PmWiki's basic edit syntax

?action=**login**
: prompt visitor for username/password, by default using Site.AuthForm

?action=**logout**
: remove author, password, and login information

?action=**print**
: display the specified page using the skin specified by `$ActionSkin`['print']

?action=**refcount**
: bring up the reference count form, which allows the user to generate a list of links (all, missing, existing or orphaned) in or from specified groups. See Ref Count Link references counts on pages . Part of the core distribution but must be enabled by the administrator.

?action=**search**
: displays searchbox on current page, see search Targeting and customising search results

?action=**search**&q=*searchterm*
: performs search with *searchterm* and displays results on current page

?action=**search**&q=link=*pagename*
: performs backlinks search with *pagename* and displays results on current page

?action=**source**
: show page source

?action=**atom**
?action=**rdf**
?action=**rss**
?action=**dc**
: If web feeds are enabled, returns a syndication feed based on the contents of the page or other options provided by the url, see web feeds Web feed notification of changes

?action=**upload**
: display a form to upload an attachment for the current group, see uploads

---

Query string parameters

?**from**=*page name*
: use when a page is redirected

?**n**=*page name*
: display page

?**setprefs**=*SomeGroup.CustomPreferences*
sets cookie to custom preferences page. See site preferences Customisable browser setting preferences: Access keys, edit form

---

Actions enabled by `$EnableDiag`

The following actions are available only if you set `$EnableDiag = 1;` in your configuration file. They can be used for debugging and should not be set in a production environment.

?action=**ruleset**
displays a list of all markups in 4 columns:

- column 1 = markup-name (1. parameter of markup())
- column 2 = when will rule apply (2. parameter of markup())
- column 3 = PmWiki's internal sort key (derived from #2)
- column 4 = Debug backtrace information for potentially incompatible rules (filename, line number, pattern)

(see Custom Markup Using the Markup() function for custom wiki syntax; migration to PHP 5.5).
To see more than what `?action=ruleset` gives you, apply the Cookbook:MarkupRulesetDebugging recipe: it can also show the pattern and the replacement strings.

- doesn't make use of PmWiki's authorization mechanisms.

?action=**phpinfo**
displays the output of phpinfo() and exits. No page will be processed

- doesn't make use of PmWiki's authorization mechanisms.

?action=**diag**
displays a dump of all global vars and exits. No page will be processed

- doesn't make use of PmWiki's authorization mechanisms.

Actions enabled by PmWiki Scripts

?action=**analyze**
see Site Analyzer and Analyze Results

?action=**approvesites**
see Url approvals Require approval of Url links

- doesn't make use of PmWiki's authorization mechanisms.

Actions enabled by Cookbook recipes

(more information about Custom Actions)

?action=**admin**
see Cookbook:UserAuth2

?action=**backup**
see Cookbook:BackupPages

?action=**clearsky**
see Cookbook:SearchCloud

?action=**cm-dependencies**
see Cookbook:CodeMirror

?action=**comment**
see Cookbook:CommentBox

?action=**comments**
see Cookbook:Comments

?action=**comment-rss**

see Cookbook:CommentDb

?action=**convert**
see Cookbook:ROEPatterns

?action=**converttable**
Cookbook:ConvertTable

?action=**copy**
see Cookbook:MovePage

?action=**csv**
see CSVAction

?action=**downloaddeleted**
?action=**delattach**
?action=**deldelattach**
?action=**fileinfo**
?action=**thumbnail**
?action=**undelattach**
Cookbook:Attachtable

?action=**delete**
see Cookbook:DeleteAction

?action=**discuss**
see Cookbook:DiscussionTab

?action=**downloadman**
see Cookbook:DownloadManager

?action=**expirediff**
see Cookbook:ExpireDiff

?action=**import**
see Cookbook:ImportText

?action=**lang**
see Cookbook:MultiLanguageViews
?action=**setlang**
see Cookbook:MultiLanguageViews

?action=**move**
see Cookbook:MovePage

?action=**PageUrl**
see Cookbook:CommentBoxPlus

?action=**pageindex**
see Cookbook:ListCategories

?action=**pdf**
see Cookbook:GeneratePDF or Cookbook:PmWiki2PDF

?action=**postupload2**
see Cookbook:UploadForm

?action=**publish**
see Cookbook:PublishPDF

?action=**purgeqns**
see Cookbook:ASCIIMath

?action=**pwchange**
see Cookbook:UserAuth2

?action=**imgtpl**
(the *imgtpl* action is called automatically and should not be called by a link in a wiki page)
?action=**createthumb**
(the *createthumb* action is called automatically and should not be called by a link in a wiki page)
?action=**mini**
(this action is called automatically and should not be called by a link in a wiki page)

?action=**purgethumbs**
: see Cookbook:ThumbList
: see Cookbook:Mini

?action=**recipecheck**
: see Cookbook:RecipeCheck

?action=**regen**
: see Cookbook:PageRegenerate

?action=**reindex**
: see Cookbook:Reindex

?action=**rename**
?action=**links**
: see Cookbook:RenamePage

?action=**share**
?action=**unshare**
: see Cookbook:SharedPages

?action=**sitemapaddgroups**
?action=**sitemapupdate**
: see Cookbook:Sitemapper

?action=**totalcounter**
: see Cookbook:TotalCounter

?action=**trash**
?action=**untrash**
: see Cookbook:Trash

?action=**webadmin**
: see Cookbook:WebAdmin

?action=**zap**
: see Cookbook:ZAP

Query string parameters enabled by Cookbook recipes

?**color**=*colorscheme*
:?**setcolor**=*colorscheme*
?**skintheme**=*theme*
?**setskintheme**=*theme*
: see Cookbook:ChoiceColorChanger {Cookbook/ChoiceColorChanger $:Summary}

?**skin**=*skinname*
?**setskin**=*skinname*
: see SkinChange

Custom actions

- See CustomActions.

Backup and Restore

This page has some background information on making backups and explains some basic *nix backup and restore procedures.

Introduction

Your wiki installation contains some unique data in the following directories:

```
local/         Local configuration scripts
cookbook/      Recipes obtained from the Cookbook
pub/           Publicly accessible files
wiki.d/        Wiki pages
uploads/        Uploaded files (attachments)
```

A good backup plan will include periodically archiving these directories — or at bare minimum`local/` and `wiki.d/`. Good practice dictates keeping your backup archives on a separate machine.

Simple Backup and Restore (*nix)

When it comes to backup, simpler is better. Since the pmwiki distribution is very small (about 1/4 megabyte), it's simplest to just archive the distribution files along with the data.

Making a Backup Archive

The following *nix command, executed from the parent directory of your wiki's directory, will put a complete backup archive of your site in your home directory.

```
tar -zcvf  ~/wiki-backup-`date +%Y%m`.tar.gz  wiki/
```

Restoring the Backup Archive

Simple Method

Your site can be restored and running in under 30 seconds with

```
tar -zxvf ~/wiki-backup-200512.tar.gz
find wiki/uploads/ -type d |xargs chmod 777
find wiki/wiki.d/ -type d |xargs chmod 777
```

A Slightly-More-Secure Method

The simple restore commands above will give you world-writable files and directories. You can avoid world-writable permissions by letting PmWiki create directories with the proper attributes (ownership and permissions) for you.

Start with

```
tar -zxvf ~/wiki-backup-200512.tar.gz
rm -rf wiki/wiki.d
rm -rf uploads
chmod 2777 wiki/
```

Now upload a file in each group that had uploads. If your site doesn't have uploads, just visit your site once so the wiki.d/ directory will be created.

Finish your installation with

```
chmod 755 wiki/
tar -zxvf ~/wiki-backup-200512.tar.gz
```

Details

The commands on this page assume your site is in a directory called "wiki/". The test backup was made in December, 2005 so it's named accordingly.

Your site will only have an uploads/ directory if uploads are enabled.

The backup command uses a date stamp (YYYYMM) in the filename. If you automate the command via cron you'll wind up with monthly snapshots of your site. You can get a daily snapshot by appending %d to the date command (`` `date +%Y%m%d` `` will get you YYYYMMDD). Be wary of space limitations if you have a large uploads/ directory.

See Also

- A thread [gmane.org] on the pmwiki-users mailing list.
- A Backup Pages recipe in the cookbook.

Miscellaneous

Backup via FTP

Download and install a ftp client like Filezilla

1. Using the ftp client connect to the server where you host pmWiki using
   1. the IP address (ex: 123.234.56.67) or the ftp name (ex: ftp.myhost.com)
   2. supply your account name (ex: mylogin) and password (ex: myp4ssw0rd)
2. Move to your pmWiki directory (ex:`/usr/mylogin/web/wiki/` or `/tahi/public_html/pmwiki`)
3. Select the folder you want to backup as explained before (probably either only the data or the whole wiki directory)

   - for data you will want to backup both the directories
      - `wiki.d` for user page data
      - `pmwikiuploads` (or `uploads`) for your attachments (uploads)
   - for system you will want, at a minimum, to backup both the directories
      - `local` for configuration data
      - `pub` for local CSS and skins customisations
4. Download them to a local folder
5. Use 7zip or a similar software to build an archive of this backup

You can also very easily sync your FTP directories with your hard disc via this command line:

```
    wget -nv -np -m ftp://user:password@ftp.yourhost.net/
```

Download Wget for Windows (other systems normally have it installed).

Alternatively, you can also mirror your FTP directories with lftp:

```
    lftp -u your_user_name,your_password -e "mirror --verbose /wiki.d /path/to/local/folder" ftp://your_host
```

(this will mirror only the /wiki.d folder, replace with / to mirror everything)

Using rsync

See Cookbook:BackupWithRsync and Cookbook:TwoWayMirroringWithRsync.

Last modified by Petko on December 28, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/BackupAndRestore

Basic PmWiki editing rules

toc top

The pages on this site are wiki-based pages, which means that pages can be created and edited by multiple authors. To edit a page, click the **Edit** link that exists **somewhere** on the page, usually in the header or footer. Some pages may be password-protected, depending on the system's security policies, but many systems allow open editing of pages.

PmWiki is not WYSIWYG - When editing a page, you see the *markup text* that describes the content of the page. The basic rules for page markup are simple:

1. Use a blank line to start a new paragraph more.
2. To make a list, start each line with # for numbered (ordered) lists or * for bulleted (unordered) lists more.
3. To make a heading, start a line with two or more ! marks; !! is a subheading, and !!! is a sub-subheading more.
4. To emphasize text, enclose it in 2 or 3 *single quotes*; `''text''` for italics or `'''text'''` for bold more.
5. To make a link to another page, enclose the page's name in double brackets; for example `[[basic editing]]` links to this page.
6. To make a link to another site, type its address, such as [[http://example.com/]]. Email links must have "mailto:" before such as mailto:xyz@example.com

If you want to experiment with editing a page, try it on the Wiki Sandbox. You can edit the Wiki Sandbox without affecting anything important on this site. On talk pages and discussions, it's courteous to **sign** your contribution; using ~~~ effectively 'signs' the name that you provide in the Author field on the Page Edit form.

Examples of common markups

The tables below demonstrate many of the common markups used to format pages. The left column shows what to write to achieve the effect, the right column shows the effect of the markup. More details are available from the text formatting rules and other documentation pages. An exhaustive list of default markup is available as the markup master index.

Paragraphs and line breaks

What to type	What it looks like

`Consecutive lines`
`will be merged together`
`as part of the same paragraph.`
 	
`One or more empty lines will start a new`
`paragraph.`
	Consecutive lines will be merged together as part of the same paragraph.
	
	One or more empty lines will start a new paragraph.

`Two backslashes at the end of a line \\`
`force a line break.`
 	
`Or use this markup: [[<<]] to force a`
`break.`
	Two backslashes at the end of a line
	force a line break.
	
	Or use this markup:
	to force a break.

Further reading:

- text formatting rules for more information on linebreaks, indented or hanging paragraphs.

- wiki styles for centered or right justified paragraphs and "floating" text (boxes), borders and much more.

Lists

Start each line with # for numbered (ordered) lists or * for bulleted (unordered) lists:

`* Bullet list`
`* Another item`
`** More asterisks produce sub-items`
`** etc.`
	• Bullet list
	• Another item
	    ○ More asterisks produce sub-items
	    ○ etc.

`# Numbered lists`
`# Another item`
`## more hashes produce sub-items`
	1. Numbered lists
	2. Another item
	    1. more hashes produce sub-items

`# List types`
`# can be mixed`
`** numbered list with unordered sub-list`
	1. List types
	2. can be mixed
	    ○ numbered list with unordered sub-list

Learn more about lists (including definition lists) and list styles.

Headings

Headings are useful for creating a "well-structured" page. They're not just for making big text.

What to type	What it looks like

`!! Major Subheading`
`!!! Minor Subheading`
`!!!! And More`
`!!!!! Subheadings`
	Major Subheading
	Minor Subheading
	And More
	Subheadings

Text Emphasis

To emphasize, enclose text in apostrophes (single-quote marks), not double-quotes.

What to type	What it looks like

`''Emphasize'' (italics),`
`'''strong''' (bold),`
`'''''very strong''''' (bold italics).`
	Emphasize (italics), **strong** (bold), ***very strong*** (bold italics).

Links

To make a link to another page, enclose the page's name in double square brackets.

What to type	What it looks like
`Practice editing in the [[wiki sandbox]]`	Practice editing in the wiki sandbox

Note that words are automatically *capitalized* in page titles. The link above links to the page WikiSandbox.

Text after a pipe (|) is used as the link text:

`Practice editing in the`
`[[WikiSandbox | practice area]].`
	Practice editing in the practice area.

Endings become part of the link text, parentheses hide parts of the link name:

`[[wiki sandbox]]es.`
 	
`[[(wiki) sandbox]].`
	wiki sandboxes.
	
	sandbox.

When linking to a page in a different WikiGroup, provide the group name, followed by a separator, and then the page name:

`[[Main.Wiki Sandbox]] shows group + name`	Main.Wiki Sandbox shows group + name

`[[Main/Wiki Sandbox]] shows only name`	Wiki Sandbox shows only name

Links to external sites

`bare url:  http://www.pmwiki.org`	bare url: http://www.pmwiki.org
`link text: [[http://www.pmwiki.org | PmWiki home]]`	link text: PmWiki home

Links as reference to external sites

`bare url: http://www.pmwiki.org`	bare url: http://www.pmwiki.org
`link text: [[http://www.pmwiki.org | #]]`	link text: [1]

Colons make InterMap (also called InterWiki) links to other wikis:

`What's an [[Wikipedia:aardvark]], anyway?`	What's an Wikipedia:aardvark, anyway?

Links to nonexistent pages are displayed specially, to invite others to create the page.

PmWiki supports more link types and a lot of display options, see Links to learn more.

Preformatted text

Preformatted text is displayed using a monospace font and not generating linebreaks except where explicitly indicated in the markup.

Note that very long lines of preformatted text can cause the whole page to be wide.

For preformatted text with markup (e.g. emphasis) being processed, start each line with a space:

```
 Lines that begin with a space
 are formatted exactly  as     typed
 in a '''fixed-width'''   font.
```

```
Lines that begin with a space
are formatted exactly  as     typed
in a **fixed-width**   font.
```

If you don't want Wiki markup to be processed, use [@ @]. Can also be used inline.

```
[@
Text escaped this way has
the HTML ''code'' style
@]
```

```
Text escaped this way has
the HTML ''code'' style
```

Escape sequence

If you don't want Wiki markup to be processed, but lines reformatted use`[= =]`. Can also be used inline.

```
[=
markup is ''not'' processed
but lines are reformatted
=]
```

markup is "not" processed but lines are reformatted

Horizontal line

```
Four or more dashes at
the beginning of a line
----
produce a "horizontal rule"
```

Four or more dashes at the beginning of a line

---

produce a "horizontal rule"

Tables

Simple tables use double pipe characters to separate cells:

```
|| border=1
||! head 1 ||! head 2 ||! head 3 ||
|| cell 1  ||  cell 2 ||  cell 3 ||
```

head 1	head 2	head 3
cell 1	cell 2	cell 3

See simple tables and advanced tables to learn more about the rich feature set of PmWiki tables.

Images

See Images

Character formatting

What to type	**What it looks like**

`* @@Monospaced text@@`
`* Text with '^superscripts^'`
`* Text with '_subscripts_'`
`* deleted {-strikethrough-} text`
`* inserted {+underline+} text`
`* [+big+], [++bigger++] text`
`* [-small-], [--smaller--] text`
	• `Monospaced text`
	• Text with ^{superscripts}
	• Text with $_{subscripts}$
	• deleted ~~strikethrough~~ text
	• inserted underline text
	• big, bigger text
	• small, smaller text

Use WikiStyles to change the text color .

Page titles

The `(:title:)` directive sets the page's title to something other than its page name.

`The name of this page is "{$Name}", and its title is "{$Title}".`	The name of this page is "BasicEditing", and its title is "Basic PmWiki editing rules".

Page Description

- The `(:Description Page summary here:)` directive sets the page description. The description is used by search engines, and can be displayed in search results and in page lists.

`(:Description PmWiki's basic edit syntax:)`
`The summary description of this page is {$Description}.`
	The summary description of this page is PmWiki's basic edit syntax.

I'm new to PmWiki, where can I find some basic help for getting started?

The Basic Editing page is a good start. From there, you can just follow the navigational links at the top or the bottom of the page (they are called Wiki Trails) to the next pages, or to the Documentation Index page, which provides an outline style index of essential documentation pages, organized from basic to advanced.

How do I include special characters such as Copyright (©) and Trademark (® or ™) on my wiki pages?

See special characters on how to insert special characters that don't appear on your keyboard.

How can I preserve line-breaks from the source text?

PmWiki normally treats consecutive lines of text as being a paragraph, and merges and wraps lines together on output. This is consistent with most other wiki packages. An author can use the `(:linebreaks:)` directive to cause the following lines of markup text in the page to be kept as separate lines in the output. Or a wiki administrator can set in *config.php* `$HTMLPNewline = '<br/>';` to force literal new lines for the whole site.

Can I just enter HTML directly?

By default (and by design), PmWiki does not support the use of HTML elements in the editable markup for wiki pages. There are a number of reasons for this described in the PmWiki Philosophy and Audiences. Enabling HTML markup within wiki pages in a collaborative environment may exclude some potential authors from being able to edit pages, and pose a number of display and security issues. However, a site administrator can use the Cookbook:Enable HTML recipe to enable the use of HTML markup directly in pages.

Where can I find more documentation?

See the documentation index and the markup master index pages.

Last modified by on December 29, 2014. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/BasicEditing

BasicVariables

toc top

Where the variables are available as wiki markup they are shown as *Variable value "{$VariableName}"*.

`$AsSpacedFunction`
The name of the function used to convert WikiWords into normal, spaced strings. Defaults to 'AsSpaced'.
`$AsSpacedFunction = 'MyAsSpaced';`

`$Author`
Set to the current reader, who is potentially an author (see discussion). See also `$EnablePostAuthorRequired`.

Variable value "Petko"

`$AuthorGroup`

The WikiGroup for user profiles. Defaults to 'Profiles'. This variable is implicit in the markup `[[~AuthorName]]`

```
$AuthorGroup = 'Users';
```

`$AuthId`

For sites using user-based authorization, tracks the "reader" or login name.

```
SessionAuth($pagename);
if( isset($AuthId) ) { // this person has been authenticated
```

`$AuthPw`

Request for documentation, meanwhile see here.

```
SessionAuth($pagename);
if( isset($AuthPw) ) { // this person has entered a password
```

`$BaseName` Variable value "PmWiki.BasicVariables"

`$BaseNamePatterns`

Allows population of the `{$BaseName}` PageVariable. The key to the hash is the pattern to be replaced and the value is the replacement string.

```
# If {$FullName} is 'Group.Page-Draft' then {$BaseName} is 'Group.Page'
$BaseNamePatterns['/-Draft$/'] = '';
# If {$FullName} is 'Comments-Group.Page' then {$BaseName} is 'Group.Page'
$BaseNamePatterns['/^Comments-/'] = '';
```

`$CategoryGroup`

The WikiGroup used for categories. Defaults to 'Category'. (See Categories). This variable is implicit in the markup `[[!CategoryName]]`

`$CookiePrefix`

A string prefix to be prepended to cookies set from PmWiki scripts. It defaults to '', but can be set to a different value to avoid conflicts with similar-named cookies from other applications, or to allow multiple wikis from the same domain to store separate cookies.

`$CookiePrefix = 'pmwiki_';` # set cookie prefix to 'pmwiki_'

If you have a WikiFarm, use the following in each field's*config.php* to get a unique prefix for each field in the farm, thus isolating each field's cookies.

```
$CookiePrefix = substr($tmp = md5(__FILE__), 0, 5).'_';
```

`$DefaultGroup` Variable value "Main"

WikiGroup used by default (on startup) when no group is specified in the URL.

`$DefaultName` Variable value "HomePage"

Name of the default HomePage of each WikiGroup. Used when the group doesn't have a page with the same name as the group.

Note that the behavior will differ based on whether the page exists or not. Probably you want to alter `$PagePathFmt` in addition to `$DefaultName` if you really want it to take effect.

Note: See comment below under `$DefaultPage` re the order how this must be defined within your (farm)config scripts - this must be set prior to any call of ResolvePageName().

`$DefaultPage`

Startup page when PmWiki is called without a specified page, normally *`$DefaultGroup`*. *`$DefaultName`*.

Note: for `$DefaultGroup`, `$DefaultName` and `$DefaultPage` variables to work, they should be defined in the beginning of (farm)config.php, before any call to the function ResolvePageName(). This means, before any script from PmWiki and before any recipe that might be using this function. This also means it cannot be set in a per-page or per-group customization script - ResolvePageName() is called before these are loaded.

Please note that this variable is intended to be set in (farm)config.php, not in individual groups. Trying to use different `$DefaultName`, `$DefaultPage` or `$PagePathFmt` settings in different groups will cause cross-group linking anomalies.

`$EnableLocalConfig`

Allows/disables local/config.php customizations (usually for a farm's wikis). Can be set to zero in local/farmconfig.php to prevent the farm's wikis' local/config.php from being loaded.

```
$EnableLocalConfig = 0; # disable PmWiki's local/config
```

`$EnableStdConfig`

Disables scripts/stdconfig.php and a large part of the core functionalities provided by the scripts in the pmwiki/scripts directory and outlined in the core documentation, unless included by your own local configuration (notably core markup rules, page history, skins, uploads). Allows you to completely reshape the way PmWiki behaves, if you need to.

```
$EnableStdConfig = 0; # disable many standard features
```

`$EnablePGCust`

Allows/disables per-page and per-group customizations. Can be set to zero in any local customization file to prevent remaining page/group customizations from being loaded.

```
$EnablePGCust=0; # turn off per-page/group configs
```

`$EnableRedirect`

When enabled (default), causes page redirects to automatically be performed by the browser. Setting `$EnableRedirect` to zero causes PmWiki to pause and issue a "Redirect to *link*" message instead. This is sometimes useful when debugging recipes to be able to see the results of actions before page redirections occur. Not to be confused with `$EnableRedirectQuiet`.

`$EnableWikiWords`
: Enable WikiWord processing.

`$EnableWSPre`
: Enables a markup rule that causes lines with leading spaces to be treated as sections of preformatted text. If set to a value greater than 1, indicates the minimum number of leading spaces required for this treatment.

```
$EnableWSPre = 1; # leading spaces are preformatted text
$EnableWSPre = 0; # leading spaces are normal lines of text
$EnableWSPre = 4; # 4+ spaces are preformatted text
```

`$FTimeFmt`
: Can be used to override the default date format used by the "ftime" function. The default `$FTimeFmt` is `$TimeFmt`. (See Markup Expressions)

`$GroupPattern`
: The regular expression pattern used for valid WikiGroup name specifications. Defaults to allowing any group name beginning with an uppercase letter, but can be set to limit the valid group names (see Cookbook:LimitWikiGroups).

```
# limit groups to Site, SiteAdmin, PmWiki, and MyGroup
$GroupPattern = '(?:Site|SiteAdmin|PmWiki|MyGroup)';
#for case-sensitive group names, note the ?-i switch:
$GroupPattern = '(?-i:Site|SiteAdmin|PmWiki|MyGroup)';
```

`$LinkWikiWords`
: If set, then bare WikiWords in a page are automatically linked to pages of the same name. Note that this value can also be affected by the `(:linkwikiwords:)` and `(:nolinkwikiwords:)` directives.

```
$LinkWikiWords = 1; # turn on WikiWord links
$LinkWikiWords = 0; # disable WikiWord links
```

Note, this setting requires WikiWords to be enabled, see `$EnableWikiWords`.

`$LogoutRedirectFmt`
: Identifies the page to which the visitor should be sent after an `?action=logout`. Defaults to the current page.

```
$LogoutRedirectFmt = 'Site.Logout'; # ?action=logout target
```

`$LogoutCookies`
: An array of cookie names to be removed when `?action=logout` is invoked.

`$NamePattern`
: The regular expression pattern used for valid page names. Defaults to allowing pages beginning with an uppercase letter or digit, followed by sequences of alphanumeric characters, hyphens, and underscores.

`$pagename`
: A variable to access information about the current page. Accessible via `$pagename = ResolvePageName($pagename);` To use inside a function, remember to declare

```
    global $pagename;
```

See + for more information, including when it's possible to use this variable. Once you have `$pagename`, page variables become accessible:

```
    $page = PageVar($pagename, '$FullName'); # =$pagename
    $group = PageVar($pagename, '$Group');
    $name = PageVar($pagename, '$Name');
```

`$PagePathFmt`
: An array controlling how the default group home-page name will be determined.
Please note that this variable is intended to be set in (farm)config.php, not in individual groups. Trying to use different `$DefaultName`, `$DefaultPage` or `$PagePathFmt` settings in different groups will cause cross-group linking anomalies.
Default Setting:

```
    $PagePathFmt = array('{$Group}.$1','$1.$1','$1.{$DefaultName}');
```

Setting to use if you wish `$DefaultName` to actually be the name of your group home-pages:

```
    $PagePathFmt = array('{$Group}.$1','$1.{$DefaultName}','$1.$1');
```

Do note that if the Groupname.Groupname page does exist but Groupname.Defaultname does not exist, then Groupname.Groupname will still take precedence. You may remove the '$1.$1' entirely to **require** Groupname.Defaultname to be the group homepage - that would look like this:

```
    $PagePathFmt = array('{$Group}.$1','$1.{$DefaultName}');
```

`$SiteGroup`
: Default group for storing configuration and utility pages such as Site.Search, Site.EditForm, Site.PageNotFound, etc.

Variable value "Site"

`$SiteAdminGroup`

Default group for locked administrative pages such as SiteAdmin.AuthList, SiteAdmin.AuthUser, SiteAdmin.ApprovedUrls, etc, defaults to 'SiteAdmin'.

`$Skin`
: The name of the directory containing the skin (theme) files, default "pmwiki". See Skins.

`$SpaceWikiWords`
: If set, then WikiWords in pages are automatically spaced according to `$AsSpacedFunction`. Note that this value can also be affected by the `(:spacewikiwords:)` and `(:nospacewikiwords:)` directives.

```
$SpaceWikiWords = 1; # turn on WikiWord spacing
$SpaceWikiWords = 0; # turn off WikiWord spacing
```

`$TimeFmt`
: The format to use for dates and times, in strftime() format. The default value is `'%B %d, %Y at %I:%M %p'`, which gives dates of the form "September 8, 2005 at 10:57 PM".

```
$TimeFmt = '%B %d, %Y'; # dates as "September 8, 2005"
$TimeFmt = '%Y-%m-%d'; # dates as "2005-09-08"
```

`$Version`
: A string representing the release version of PmWiki.

Variable value "pmwiki-2.2.99"

`$VersionNum`
: A number representing the release version of PmWiki, with the major and minor release components padded with zeroes to produce three digits. Thus, release "pmwiki-2.1.40" will have `$VersionNum` set to 2001040.

Variable value "2002099"

The first digit is a 2, the next three digits are the major release number, and the last three digits are the minor release number. Beta releases use 900-999 for the minor release number. Thus:

```
2.1.0          2001000
2.1.1          2001001
...
2.1.27         2001027
2.2.0-beta1    2001901
2.2.0-beta2    2001902
...
2.2.0-beta18   2001918
...
2.2.0          2002000
```

`$WikiWordPattern`
: The pattern that describes a WikiWord.

`$EnableRelativePageVars`
: This setting controls how Page variables in included pages are understood by PmWiki.

```
$EnableRelativePageVars = 1; # PmWiki current default
```

In this case `{$Name}` displays the name of the physical page where it written. If `{$Name}` is in an included page, it will display the name of the included page. (This is currently PmWiki's default.)

```
$EnableRelativePageVars = 0; # revert to previous default
```

In this case `{$Name}` displays the name of the currently browsed page. Even if `{$Name}` is in an included page, it will display the name of the browsed page. This was PmWiki's default in versions 2.2.8 and earlier, and changed in 2.2.9, but you can revert it back with this line in config.php.

`{*$Name}` with an asterisk always displays the name of the currently browsed page, regardless of `$EnableRelativePageVars`.

Categories: PmWiki Developer

Last modified by Petko on June 26, 2017.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/BasicVariables

toc top

BlockMarkup

toc top

"Block markup" is a term used in the sources of PmWiki indicating all markups resulting in HTML block elements[1] or in other words multiple paragraphs and other content.

- Forms
- paragraphs
- indent/outdent
- lists
- list items
- headings
- divisions and semantic HTML5 elements
- images
- pre
- tables

WikiStyles can be applied to blocks, else you don't need to bother about "blockmarkup" as a PmWiki user.

Division blocks

Division <div> HTML blocks are inserted with the `(:div:)...(:divend:)` markup. You can have the HTML `id=` and `class=` attributes like `(:div id=id1 class="class1 class2":)`. A `(:div:)` markup automatically closes a previously open such tag. To have nested tags, you need to number the tag, and the matching tag end:

```
(:div:)
Outer block
(:div2:)
Inner block
(:div2end:)
(:divend:)
```

Semantic HTML5 elements

Since version 2.2.75, PmWiki allows the inclusion of a few semantic HTML5 elements. Note that an opening semantic markup automatically closes any previously opened tag of the same type, but does not verify or tidy the structure for you, so make sure you use closing tags when needed.

`(:article:)...(:articleend:)`
Inserts an <article> tag. You can have the HTML `id=` and `class=` attributes like `(:article id=id1 class="class1 class2":)`. An `(:article:)` markup automatically closes a previously open such tag. To have nested tags, you need to number the tag, and the matching tag end:

```
(:article:)
Outer article
(:article2:)
Inner article
(:article2end:)
(:articleend:)
```

`(:section:)...(:sectionend:)`
Inserts a <section> tag. You can have the HTML `id=` and `class=` attributes like `(:section id=id1 class="class1 class2":)`. A `(:section:)` markup automatically closes a previously open such tag. To have nested tags, you need to number the tag, and the matching tag end, like the `(:article:)` markup.

`(:header:)...(:headerend:)`
Inserts a <header> tag. You can have the HTML `id=` and `class=` attributes like `(:header id=id1 class="class1 class2":)`. A `(:header:)` markup automatically closes a previously open such tag, and it is not possible to nest such tags.

`(:footer:)...(:footerend:)`
Inserts a <footer> tag. You can have the HTML `id=` and `class=` attributes like `(:footer id=id1 class="class1 class2":)`. A `(:footer:)` markup automatically closes a previously open such tag, and it is not possible to nest such tags.

`(:aside:)...(:asideend:)`
Inserts an <aside> tag. You can have the HTML `id=` and `class=` attributes like `(:aside id=id1 class="class1 class2":)`. An `(:aside:)` markup automatically closes a previously open such tag, and it is not possible to nest such tags.

`(:address:)...(:addressend:)`
Inserts an <address> tag. You can have the HTML `id=` and `class=` attributes like `(:address id=id1 class="class1 class2":)`. An `(:address:)` markup automatically closes a previously open such tag, and it is not possible to nest such tags.

`(:nav:)...(:navend:)`
Inserts a <nav> tag. You can have the HTML `id=` and `class=` attributes like `(:nav id=id1 class="class1 class2":)`. A `(:nav:)` markup automatically closes a previously open such tag, and it is not possible to nest such tags.

See also

- BlockMarkup Markup resulting in paragraphs
- ConditionalMarkup The if directive allows portions of a page to be included or excluded from rendering
- CustomMarkup Using the Markup() function for custom wiki syntax; migration to PHP 5.5
- MarkupExpressions String and formatting operations
- Markup Master Index Tabulation of all PmWiki markup

Blocklist

The block list is one of a number of security measures that can be taken to protect your wiki from spam and other unwelcome postings.

Unfortunately, the open-editability of many wiki systems often makes them attractive targets for "link spam" or "wikispam", in which links are added to pages in an effort to increase search engine rankings or drive traffic to other sites. Also, many link spammers have developed automated systems to locate sites that accept visitor input and attempt to flood the site with unwanted links. Also, and harder to deal with, is just plain wiki vandalism where nonsense changes are made, often replacing entire pages.

By far the best countermeasure against wikispam is to restrict editing through the use of passwords (see Passwords and Passwords Admin). Experience has shown that passwords can be effective even if the password is widely known, and even if the password is publicly available on the site itself. However, there are many cases where passwording may be an impediment, so these will generally want to use some form of blocklist.

Blocklist basics

A *blocklist* is a list of IP addresses, phrases, and expressions which are prevented from being added into pages on the website. PmWiki is distributed with a built-in blocklisting capability; blocklists can be enabled by adding the following line to *local/config.php*:

```
    $EnableBlocklist = 1;
```

This tells PmWiki to scan the SiteAdmin.Blocklist page and the "SiteAdmin.Blocklist-Farm" page (and possibly other pages -- see below) looking for phrases and IP addresses to be excluded from posting to the site.

Blocking by word or phrase

The simplest form of block is simply a line containing "`block:`" followed by a word or phrase to be excluded from postings. For example, a line like

```
    block:spam.com
```

in SiteAdmin.Blocklist will block any posts containing the string "spam.com" (case-insensitive) anywhere in the post.

Blocking by IP address

Sometimes we wish to restrict posts coming from particular addresses or address ranges that are known as sources of wikispam. If a blocklist page contains IP addresses of the form "a.b.c.d" or "a.b.c.*", then any posts coming from that address or range will be blocked.

> To find an author's IP address, try hovering the mouse over the author name in the page history for a page.

Blocking by regular expression or pattern

Blocking on simple words can sometimes pose difficulties; for example, a simple "`block:cial`" entry will also block the word "specialist". For these cases it's often helpful to use a regular expression, as in:

```
    block:/\bcial\b/
```

This says to block "cial" only if it doesn't occur in the middle of a larger word. The leading slash (/) after "block:" tells PmWiki to use a regular expression match instead of a simple string match. (Blocklist uses PCRE or "Perl Compatible Regular Expressions"; see http://php.net/manual/en/ref.pcre.php for more information.)

Regular expression to block 'href'

If you want to block '`href`', you can use the following markup:
```
    block:/[^\w\\]href\b/
```
which blocks '`href`', but neither '`\href`' nor '`toughref`'.

The regular expression can be interpreted as follows: Match any character that is **neither** a word character **nor** a '\', followed by `href` which ends in a word boundary.

Letting authors know why they've been blocked

By default, blocklist only tells an author that a particular edit has been blocked, but doesn't give a specific reason for the blocking (e.g., the offending phrase). Setting the following in a local customization file will also provide the reasons for the block:

```
    $EnableWhyBlocked = 1;
```

Managing multiple blocklists

PmWiki allows blocklist entries to come from multiple pages by setting the `$BlocklistPages` variable. By default `$BlocklistPages` is set to "SiteAdmin.Blocklist", as well as any automatically downloaded blocklists as described below. PmWiki will use all entries in all the blocklists for filtering wikispam. Setting a value of `$BlocklistPages` changes the default:

```
$BlocklistPages = array('Main.Blocklist', 'PmWiki.Blocklist');
```

The order of blocklists really doesn't matter -- all of the blocklist pages ultimately get used, and the `unblock:` entries are processed after all of the blocklist pages have been loaded.

Automatically downloaded blocklists

Maintaining blocklists is relatively easy to do, but can become tedious over time. Several groups have formed and maintain "shared blocklists", where a common blocklist is made available to all. PmWiki's blocklist capability has built-in features for automatically downloading and updating such shared blocklists.

If you're just in a hurry to make use of some standard blocklists, make the following setting in *local/config.php*:

```
$EnableBlocklist = 10;
```

This tells PmWiki to not only enable blocklists on the site, but to also configure itself to automatically retrieve and maintain local copies of well-known blocklists such as MoinMaster. These local copies will be saved in SiteAdmin.Blocklist-MoinMaster and refreshed once per day (as determined by the value of `$BlocklistDownloadRefresh`).

To automatically retrieve the SiteAdmin.Blocklist page used at pmwiki.org, add the following setting in *local/config.php*:

```
$BlocklistDownload["$SiteAdminGroup.Blocklist-PmWiki"] = array('format' => 'pmwiki');
```

The blocklist from chongqed.org which we used in the past is no longer available as of 2013.

Ignoring specific entries in a blocklist (unblock)

When using a large master blocklist or blocklists automatically refreshed from external sites, it may be that some entries in the blocklists are inappropriate or overeager and block legitimate content. In this case a wikiadministrator can use "unblock" in a blocklist page to ignore an entry from the blocklist. For example, to allow "spam.com" even if another blocklist has a block entry for it:

```
unblock:spam.com
```

In order for unblocking to work the phrase or pattern following "unblock:" must be *exactly* the same as the original.

Permissions on blocklist pages

In general, an administrator will want to edit-protect the SiteAdmin.Blocklist and any other blocklist pages to prevent arbitrary changes to the blocklist (see Passwords). Since most pages in the SiteAdmin.* group are edit-protected by default anyway, this usually isn't a problem.

Administrators may also wish to read-protect the various blocklist pages so that others do not know the exact phrases and/or IP addresses that are being blocked. (By their nature blocklists tend to contain phrases or terms that may be offensive or inappropriate to some.)

Any pages created via automatic download (see above) are automatically locked against viewing except by administrators.

administrators (intermediate)

Detailed configuration of automatically downloaded blocklists

Automatic downloading of blocklist information is controlled by the `$BlocklistDownload` array. An entry for MoinMaster might look like:

```
$BlocklistDownload[" $SiteAdminGroup.Blocklist-MoinMaster"] = array(
        'url' => ' http://moinmo.in/BadContent?action=raw',
        'format' => 'regex',
        'refresh' => 86400);
```

This says to download the blocklist data from the given url into the SiteAdmin.Blocklist-MoinMaster page, that the entries in the blocklist are regular expressions, and to refresh the information every 86,400 seconds (one day).

If 'refresh' is omitted, then the page will be refreshed at the time interval given by `$BlocklistDownloadRefresh` (default one day). If 'format' is omitted, the page is assumed to have PmWiki-formatted entries as described above. If 'url' is omitted, then the blocklist information is downloaded from a standard location on the pmwiki.org site.

To force a refresh of an automatically downloaded blocklist, simply delete the existing page -- a new version will be installed upon the next blocklist scan. Blocklist pages are checked for download in response to any ?action=edit request.

If you are specifying your Blocklist-Pages in the config.php you have to specify the automatically updated pages too, else they won't be updated or created even if you use `$EnableBlocklist` = 10; .

Farm-wide blocklist

A blocklist can be applied farm-wide (see SharedPages). After these pages are created they can be moved into the farm *shared.d/* directory:

Blocklist Variables

The following variables help control the configuration and operation of blocklists:

`$EnableBlocklist`
: If set to a non-zero value, then blocklists are enabled on the site. If set to a value of ten or higher, then add entries for automatic downloads of standard blocklists.
`$EnableBlocklist` = 1; # enable blocklists
`$EnableBlocklist` = 10; # auto-configure standard blocklists

`$EnableWhyBlocked`
: By default, authors are not told which particular phrases or IP addresses are causing a particular post to be blocked; setting `$EnableWhyBlocked` to 1 provides this information.
`$EnableWhyBlocked` = 1; # give reasons for blocking

`$BlocklistPages`
: An array of pages to be checked for blocklist entries. The elements of the array may contain page variables. Defaults to "SiteAdmin.Blocklist", plus any other automatically downloaded blocklist pages.

`$BlocklistMessageFmt`
: The message to provide the author whenever a post has been blocked.

`$BlockedMessagesFmt`
: If `$EnableWhyBlocked` is set, defines the text to use for each type of block being performed. Currently only 'ip' and 'text' are recognized.
`$BlockedMessagesFmt`['ip'] = "IP address blocked from posting: ";
`$BlockedMessagesFmt`['text'] = "Text blocked from posting: ";

`$BlocklistDownload`
: An array of automatically-downloaded blocklists. The keys of the array are the pages in which the blocklists should be stored, the values contain the url, format, and refresh interval for the downloaded blocklist.

```
# Download the MoinMaster blocklist every twelve hours
$BlocklistDownload["$SiteAdminGroup.Blocklist-MoinMaster"] = array(
  'url' => 'http://moinmo.in/BadContent?action=raw',
  'format' => 'regex',
  'refresh' => 43200);
# Download a shared blocklist from pmwiki.org every day
$BlocklistDownload["$SiteAdminGroup.Blocklist-Shared"] = array(
  'format' => 'pmwiki');
```

`$BlocklistDownloadRefresh`
: The default refresh interval for any `$BlocklistDownload` entries that don't explicitly specify a 'refresh' value.
# perform automatic downloads once per week by default
`$BlocklistDownloadRefresh` = 86400 * 7;

`$BlocklistDownloadFmt`
: The format to use when saving automatically downloaded blocklists.

`$EnableBlocklistImmediate`
: Some cookbook recipes update pages with author input but don't use the built-in data posting routines. If `$EnableBlocklistImmediate` is set (default) and the current action is listed in `$BlocklistActions` (below), then an immediate blocklist scan is performed on the incoming text.

`$BlocklistActions`
: A list of actions for which immediate blocklist checks should be performed (see `$EnableBlocklistImmediate` above).
# perform immediate checks for ?action=comment
`$BlocklistActions`['comment'] = 1;
# perform immediate checks for ?action=postdata
`$BlocklistActions`['postdata'] = 1;

Categories

Purpose of categories

Categories (also known as "tags") are a way to organize and find related pages. Categories are implemented by default in PmWiki, and in most wikis they don't require any special code or markup, they're just a useful convention. The idea is that every page that falls into a particular subject area should have a link to a shared page containing links to other pages on that subject. These pages are created in the *Category* group, and thus these subject areas are called "categories".

Using categories

Getting categories to work requires a single step: adding links to each category. A category named Subject is created by adding a link to Category.Subject on any page. When you add the link to a page, the page can be described as being *in* the category "Subject".

There is a special markup for creating these links which makes categories work more smoothly: [[!Subject]] will create a link to Category.Subject. So [[!Subject]] is a kind of shortcut to the page Subject in the category group.

A Category.GroupFooter file is included in the PmWiki release that contains the line `(:pagelist link={*$FullName} list=normal:)` so that whenever a category page is displayed, it will show a list of links to pages that reference that page in the category group. Like any other page in `wikilib.d` you can modify this page and it will not get overwritten by another release.

It is worth noting that rather than using Category.GroupFooter, the pagelist directive can be added to Category.GroupHeader to similar effect; it just depends on whether you'd prefer to have the list of pages appear before or after any text that you add to the individual category pages (which can be edited just like normal pages).

Because we use the normal PageList `link=` markup, you can use it not only in the category group. If you want to show all pages belonging to the category Subject you can use on any wiki page `(:pagelist link=Category.Subject list=normal:)`.

Similarly, there's no requirement that a "category page" has to be in the*Category* group -- any page can define a "category" of pages that link to it.

An administrator can override the default category group name of "Category" by setting the `$CategoryGroup` variable in *config.php* to another group name. (Normally a change such as this should be done during initial setup on a new wiki; changing this on a wiki with existing content can cause problems with pagelists unless each page with a category is re-saved.)

A page author can also link to a category list without adding the linking page to the category by using [[{Category.Subject$PageUrl} | Subject]]. This will create a link that looks like [[!Subject]] without adding the linking page to the category listing.

Recap

So, by adding the link [[!Subject]] to a page, a link to that page will automatically appear on the page*Category.Subject*, as long as *Category.GroupFooter* has been tweaked appropriately. Thus, you can create a page that automatically creates an alphabetized list of all movies discussed on your wiki by creating links to [[!Movies]] on each film's page; the resulting automatic list would be on the page *Category.Movies* .

authors (advanced)

Category nesting

Categories have the potential for even greater usefulness because`Category.*` pages can themselves be placed into categories! To follow an excellent example from John Rankin, let's suppose we have the following film pages in the categories listed to the right:

```
    Film.ShaunOfTheDead   [[!Horror]] [[!Comedy]] [[!2003]]
    Film.InMyFathersDen   [[!Drama]] [[!2004]]
    Film.TheCorporation   [[!Documentary]] [[!2003]]
```

Now then, we can create `Category.Horror`, `Category.Comedy`, `Category.Drama`, and `Category.Documentary`, and in each one of those pages we put `[[!Genre]]`. In `Category.2003` and `Category.2004`, we put `[[!Year]]`.

So, what happens when we display`Category.Genre` ? We see links to "Comedy", "Drama", "Documentary", and "Horror", because they're in the Genre category. When we click on one of those links, we see all of the films listed in one of those categories. Similarly, if we click on `Category.Year`, we see links to "2003" and "2004", each of which in turn displays the list of films for that year.

Finally, in `Category.Genre` and `Category.Year` we can put `[[!Category]]`, which makes them "top-level" categories reachable from the `Category.Category` page. Voila, we now have an instant "hierarchy":

```
Category.Category
    Category.Genre
        Category.Comedy
            Film.ShaunOfTheDead
        Category.Drama
            Film.InMyFathersDen
        Category.Documentary
            Film.TheCorporation
        Category.Horror
            Film.ShaunOfTheDead
    Category.Year
        Category.2003
            Film.ShaunOfTheDead
            Film.TheCorporation
        Category.2004
            Film.InMyFathersDen
```

Note however that this isn't a "strict" hierarchy--i.e., any page or category can appear simultaneously in multiple categories. For example, `Category.Documentary` could be a member of both the Genre and top-level category listings.

Each category page can have content text before the generated list, e.g., to give a generic description of things in the category. (Or it can be empty, which works fine.) It can also contain associations to related categories ("see also" references). For example, in a tourism wiki, the "bed and breakfast" category might contain a see-also reference to the "self-catering" category.

administrators (intermediate)

The guts of the category markup

As mentioned, all of the necessary markup features for Categories are enabled by default in current releases of PmWiki 2.0, but here's how they work for those who are interested. The use of the Category group as the repository for all categories is determined by setting the `$CategoryGroup` variable, and the special [[!Subject]] markup is activated by a call to the Markup() function:

```
SDV($CategoryGroup,'Category');
Markup('[[!','<links','/\[\[!([^\|\]] ?)\]\]/',
  "<span class='category'>[[$CategoryGroup/$1]]</span>");
```

Coming up with good category schemes

The hard part about using categories is choosing a good vocabulary. Site content managers may wish to follow the Guidelines for the establishment and development of monolingual thesauri (ISO 2788-1986) and the Guidelines for the establishment and development of multilingual thesauri (ISO 5964-1985). Questions to think about include:

- whether a scheme already exists and can be reused
- number of levels in a multilevel scheme (not too shallow, not too deep -- e.g. 3)
- number of categories per page (not too many, not too few -- e.g. 3)
- consistent use of singular (`[[Mercury]] is a [[!planet]]`) or plural (`[[Mercury]] is in the [[!planets]] category`)
- disambiguation and use of phrases (`[[!musical instruments]]` and `[[!medical instruments]]`) or Cookbook:Subgroup Markup (`[[!Instruments*Musical]]` and `[[!Instruments*Medical]]`)

Or you can just let people use whatever category terms they find meaningful. A vocabulary (or "folksonomy") will emerge over time.

Showing a list of categories

To show a list of categories we can use a pagelist for the pages in the category group. For instance the following will list pages in the Category group, put it on page Category.Category for convenience, or on any other page:

```
(:pagelist group=Category list=normal fmt=#title:)
```

But there is a problem: Just adding a category markup to a page will not create a corresponding category page, even though following the link will show the page with a list of pages linking to it!
To have category pages automatically created in group 'Category' add the following to config.php:

```
$AutoCreate['/^Category\./'] = array('ctime' => $Now, 'text' => $page['text']);
```

Change 'Category' to the name of your category group. You can also add more definitions for more category groups, useful if you use a recipe like Cookbook:Tagger which allows multiple category groups.

Linking = Categorizing

Note that categorizing a page (using the `[[!category markup]]`) cannot be distinguished from referring or linking to a category (using the normal `[[link markup]]`), i.e. pages referring to a category become part of that category. This is the subject of a long outstanding feature request that seems to be hard to implement without breaking other functionality. You can link to a category without categorizing the page by using an external link, such as: `[[{Category.MyCategory$PageUrl}|MyCategory]]`. Since the link is external, all pages (not just the category page) will ignore it when listing backlinks.

See also EditVariables#AutoCreate

Last modified by RandyB on August 13, 2016. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Categories

ChangeLog

toc top

See the cookbook recent changes page for additional updates and activity by other developers, or join the PmWiki mailing lists to discuss feature development with us.

Changes made to the subversion pre-release (ZIP) of PmWiki:

- Workaround around Subversion incompatibility with `$Author:...$` string not intended as SVN keyword.

Version 2.2.99 (2017-06-26)

- Fix Preview didn't show changes due to `$ROSPatterns` (PITS:01408).
- Remove markup rules for previewing author signature not needed anymore.
- Fix bug and warning appearing in PHP 4 installations.
- Update Wikipedia intermap entry (secure https).
- Fix bug with `[[<<]]` styles "clear:both".
- Fix incomplete definition of page text variable halts the rendering (PITS:01300).
- Fix `$Version` didn't work as a vardoc link.
- Update documentation.

Version 2.2.98 (2017-05-31)

- Fix WikiStyles where "pct" was incorrectly dropped from some classnames (PITS:01404).
- Hide warning about missing intermap file.
- Add pmwiki-responsive skin, based on modified Skins:2016.
- Responsive skin: Hide icon if PageActions empty. Fix "close" icon didn't appear for the PageActions block. Unrestrict menu height. Switching from portrait view with menu open to landscape: page should not be greyed out (PITS:01406). Landscape view: fix overflow for search form in Epiphany (likely Safari and other AppleWebKit-based browsers). Move the <main> tag up to allow scrolling of the whole #wikibody. Large preformatted blocks will also scroll in the mobile view. Set limit for desktop layout to 50em~800px. Scrollable tables via cosmetic JavaScript.
- Both skins: Set default text color (PITS:01406).
- Fix Deprecated notice for Site.AuthUser password attributes.
- Vardoc links now use MakeLink() to allow a custom LinkPage function, fix bug reported by ChuckG.
- $InclCount now counts per browsed page (for multi-page processing recipes).
- Make $markupid variable available to markup replacement functions.
- Refactor function ReplaceOnSave to allow easier calling from recipes (PITS:01407).
- Enable *.mkv as allowed video extension.
- Fix bug with attachlist markup.
- Fix alternative bold/italics markup in sample-config.php (PITS:01400).
- Fix lost space in markup tables, replace markup tables <code> with <pre> and add style "pre-wrap" (reported by ChuckG).
- Update documentation.

Version 2.2.97 (2017-04-07)

- Fix bug concerning `$ScriptUrl` when `$EnablePathInfo` is set, introduced in 2.2.96, reported by 3 users.
- Update documentation.

Version 2.2.96 (2017-04-05)

- Fix severe PHP code injection vulnerability, reported by Gabriel Margiani.
    - Filter `$pagename` to exclude certain characters.
    - Add $pagename_unfiltered in case a recipe requires it.
- Update documentation.

Version 2.2.95 (2017-02-28)

- Update documentation.

Version 2.2.94 (2017-01-31)

- Strip both .html and .htm extensions (Cookbook:HtmlUrls-Talk).
- Clear $PageExistsCache[`$pagename`] when a page is created or deleted (PITS:01401).
- Update documentation.

Version 2.2.93 (2016-12-31)

- Update documentation.

Version 2.2.92 (2016-11-30)

- Skip checking for `$AllowPassword` if empty or false.
- Enable FmtPageName() to expand PageVariables with asterisks.
- Update documentation.

Version 2.2.91 (2016-09-30)

- Update documentation.

Version 2.2.90 (2016-08-31)

- Add action parameter to upload form URL.
- Add `imgonly` and `imgcaption` CSS classes (PITS:01390).
- Fix plus-links with suffix `[[Page|+]]s` (PITS:01392).
- Update documentation.

Version 2.2.89 (2016-07-30)

- Add identifiers to Site.EditForm elements to enable easier styling.
- Add `$SimpleTableDefaultClassName`, default unset (PITS:00638).
- Add temporary $new['=html'] entry, in SaveAttributes().
- Fix superfluous line breaks in SiteAdmin.AuthList.
- Add optional placeholder attribute in `(:searchbox:)`.
- Add `$SearchBoxInputType`, default 'text'.
- Set `$HTMLStylesFmt` via SDVA() in vardoc.php, urlapprove.php, and xlpage-utf-8.php.
- Fix vardoc.php to recognize and link variables `$pagename`, `$Author`, `$Skin`, and to sort case insensitively.
- Update documentation.

Version 2.2.88 (2016-06-29)

- Fix invalid HTML output of WikiTrail links (PITS:01388).
- Add 4th argument $double_encode to PHSC() for safe replacement of htmlspecialchars().
- Add page variable `{$SiteAdminGroup}` (PITS:00951).
- Update documentation.

Version 2.2.87 (2016-05-31)

- Add `$HTMLTagAttr`, to allow inclusion of lang, manifest and other attributes.
- Add `$EnableRevUserAgent`, `$FmtV`['$DiffUserAgent'].
- Fix relative link in Site.UploadQuickReference.
- Update documentation.

Version 2.2.86 (2016-04-28)

- Fix PageStore() for PHP 7.
- Fix `$DefaultPasswords` for PHP 7.
- Update documentation.

Version 2.2.85 (2016-03-31)

- Add svg(z) and SVG(Z) as embeddable image extensions (PITS:00197, PITS:00435).
- Add *.svgz as allowed upload extension.
- Update documentation.

Version 2.2.84 (2016-02-21)

- Update/fix URL in UPGRADES.txt (PITS:01378).
- Fix indent and outdent CSS for RTL languages (PITS:01379).
- Add `$EnableLinkPlusTitlespaced` (PITS:01140).
- Update documentation.

Version 2.2.83 (2015-12-31)

- Update documentation.

Version 2.2.82 (2015-11-30)

- Enable stripmagic() to process arrays recursively.
- Update documentation.

Version 2.2.81 (2015-10-31)

- Fix single line PageTextVariable definition (reported by HansB).
- Add .ltr and .rtl CSS classes for UTF-8.
- Update documentation.

Version 2.2.80 (2015-09-30)

- Modify `(:searchbox:)` to use type="search" input.
- Update documentation.

Version 2.2.79 (2015-08-27)

- Modify guiedit.js::insMarkup() to accept a custom function name processing the text, and a custom id for the text area.
- Add CSS basic colors 'fuchsia','olive','lime','teal','aqua','orange' and 'grey' as WikiStyles (PITS:01373).
- Add `$EnableROSEscape`, default 0 (PmWiki:TextFormattingRules-Talk).
- Remove 'target' attribute in input forms (breaks PmForm).
- Add HTML5 input types email, url, number, date, search.
- Add attribution in script comments.
- Update documentation.

Version 2.2.78 (2015-07-21)

- Update $RobotPattern with current user agents.
- Accept 'target' attribute in input forms.
- Update documentation.

Version 2.2.77 (2015-06-19)

- Add generic function MakeNames() to process MakePageNames().
- Extend `(:if attachments:)` to specify file and page names (PITS:01087).
- Optimize PageStore::recode() to cache utf8_decode and utf8_encode callbacks.
- Add `{$WikiTitle}` page variable.
- Update documentation.

Version 2.2.76 (2015-05-31)

- Recover posted arrays (indexed or associative, not multidimensional) when a password is required (PITS:00835, PITS:01110).
- Add label argument to checkbox and radio inputs (PITS:01367).
- Enable PHSC() to process arrays recursively.
- Enable processing of arrays as input values (PITS:01032).
- Add CSS classes to standalone image div and caption (PITS:00489, PITS:00497).
- Update documentation.

Version 2.2.75 (2015-04-26)

- Fix uploads to respect $EnableReadOnly.
- Escape HTML special characters when printing failed callback creation.
- Add pmcrypt() for PHP 5.6 compatibility.
- Add markup for HTML5 semantic tags article, section, nav, header, footer, aside, address.
- Update documentation.

Version 2.2.74 (2015-03-28)

- Allow translation of the "OK" string in forms (PITS:01363).
- Update documentation.

Version 2.2.73 (2015-02-28)

- Update documentation.

Version 2.2.72 (2015-01-27)

- Enable markup debug messages even when debug_backtrace() is not available.
- Add `$AbortFunction`.
- Restore ability to set a custom $MarkupWordwrapFunction, add $MarkupWrapTag (related to earlier fix for PITS:01360).
- Update documentation.

Version 2.2.71 (2014-12-29)

- Add `$DraftActionsPattern`.
- Enable "input default source" parameter to contain multiple pages.
- Enable "pagelist request" parameter to contain a list of (dis)allowed parameters.
- Enable Markup() backtrace for ?action=ruleset.
- Fix strict warning for blacklisted uploads (PITS:01359).
- Fix wrong hard wrap in `(:markup:)` code examples (PITS:01360).
- Update documentation.

Version 2.2.70 (2014-11-08)

- Update documentation.

Version 2.2.69 (2014-10-13)

- Fix DRange() for ISO-8601 dates +/- X days.
- Fix wording in Site.UploadQuickReference.
- Update documentation.

Version 2.2.68 (2014-09-01)

- Add Skins: InterMap prefix.
- Add signature to Site.EditQuickReference (PITS:01350).
- Allow $PostConfig entries to be launched after per-page customization, before other stdconfig.php inclusions if values<50.
- Add WikiStyles clear, min and max width and height (PITS:00860), fix %p class=...% with more than one space.
- Update documentation.

Version 2.2.67 (2014-08-02)

- Fix InputDefault/PageTextVariables inconsistency (PITS:01337).
- Update documentation.

Version 2.2.66 (2014-07-02)

- Fix Author in Notifcations when deleting pages (PITS:01112).
- Exclude "_" to be considered as a function name in various $*Patterns.
- Update documentation.

Version 2.2.65 (2014-06-07)

- Fix {$$PseudoVars} containing {*$PageVars} in PageList Templates.
- Fix wording in scripts/.htaccess (PITS:01345).
- Fix fixperms() if directory owner is root (PITS:01346).
- Update documentation.

Version 2.2.64 (2014-05-08)

- Add {(mod)} markup expression.
- Add tel: and geo: URI schemes.
- Add $SysMergePassthru to allow Merge() to use passthru() instead of popen().
- Update documentation.

Version 2.2.63 (2014-04-05)

- Allow form elements to have a dash in the attribute names.
- Strip magic slashes for pagelist/search request vars.
- Allow input attributes readonly, placeholder and autocomplete for HTML5 sites.
- Update documentation.

Version 2.2.62 (2014-02-28)

- Add $CallbackFnTemplates["return"].
- Add 4th argument to Markup_e() - $template.
- Add `$EnableTableAutoValignTop`.
- Update documentation.

Version 2.2.61 (2014-01-31)

- Add `$TableCellAlignFmt`.
- Remove unused snippet in prefs.php (reported by Oliver Betz).
- Remove unused calls to PSS() (reported by John Rankin).
- Update documentation.

Version 2.2.60 (2014-01-12)

- Revert to previous pmwiki.css file.

Version 2.2.59 (2014-01-11)

- Fix checking multiple posted fields in blocklist.php (reported by Randy Brown).
- Allow Markup_e() to accept a callback as well as code.
- Fix "+" shortcut for internal anchor links.
- Disable HTML cache if count($_GET)>1 not >2 (PITS:01278).
- Fix query string if a "?" is encoded to uppercase "%3F".
- Replace CSS font sizes from points (fixed) to percents (relative) for the default skin.
- Fix nested conditionals containing `$pagename` (reported by Benjamin Grassineau).
- Update documentation.

Version 2.2.58 (2013-12-25)

- Allow $LinkUpload to be usable in (:attachlist:).
- Enable customizations of (:input auth_form:).

- Remove unused variable $Block in FormatTableRow(), reported by Klonk.
- Fix `$EnableBlocklistImmediate` to check all posted fields for blocked terms.
- Add $GLOBALS['MarkupToHTML'] to pass parameters such as `$pagename` to markup calls.
- Update documentation.

Version 2.2.57 (2013-11-03)

- Encode international character used for detection of a recode function.
- Enable `$IMapLinkFmt`['Attach:'] to be used in (:attachlist:) links.
- Add `$MakePageNameSplitPattern`.
- Update documentation.

Version 2.2.56 (2013-09-30)

- Work in progress to remove the core dependency of the deprecated "eval" feature of the preg_replace() function (PITS:01319).
- Add functions PCCF(), PPRE(), PPRA(), Markup_e(), migrating all core calls to these functions.
- Fix detection of proper PageStore->recodefn.
- Update documentation.

Version 2.2.55 (2013-09-16)

- Add `$EnableDraftAtomicDiff` (PITS:01007).
- Update documentation.

Version 2.2.54 (2013-08-13)

- Fix broken page history for draft pages, reported by ChuckG.
- Update documentation.

Version 2.2.53 (2013-07-08)

- Show a message when the post has been blocked because of too many unapproved links.
- Update documentation.

Version 2.2.52 (2013-06-08)

- Add docx, pptx, xlsx upload extentions.
- Hide E_DEPRECATED warnings for PHP 5.5.
- Update documentation.

Version 2.2.51 (2013-05-08)

- Update url to MoinMoin's blocklist.
- Comment-out blacklist.chongqed.org as the domain appears to have expired.
- Fix possible XSS vulnerability in prefs.php, discovered today.
- Fix access keys to be a single character.
- Fix $AuthorPage if there is a group named the same as the author (PITS:01259).
- Update documentation.

Version 2.2.50 (2013-04-08)

- Update documentation.

Version 2.2.49 (2013-03-09)

- Add `$UploadBlacklist` array.
- Update documentation.

Version 2.2.48 (2013-02-11)

- Fix bug introduced yesterday with some links, reported by Michael Weiner (PITS:01308).

Version 2.2.47 (2013-02-10)

- Enable tooltip titles for links to anchors on the same page.
- Update documentation.

Version 2.2.46 (2013-01-07)

- Add third parameter to fixperms() explicitly setting the permissions.
- Add `$UploadPermAdd` and `$UploadPermSet` variables.
- Update documentation.

Version 2.2.45 (2012-12-02)

- Cleanup some PHP notices (PITS:01304).

- Update documentation.

Version 2.2.44 (2012-10-21)

- Better display of whitespace in page histories.
- Fix definition for PageTextVariables containing a dash (PITS:00978).
- Update documentation.

Version 2.2.43 (2012-09-20)

- Allow for HTML attribute names to contain dashes, eg. data-transition, data-role etc.
- Remove warning when previewing Site.EditForm.
- Update documentation.

Version 2.2.42 (2012-08-20)

- Convert the line-endings in the docs/ directory to \r\n compatible with Windows.
- Modify PHSC() to call htmlspecialchars() with a single-byte encoding argument.
- Update documentation.

Version 2.2.41 (2012-08-12)

- Change $KeepToken to "\034\034" which is compatible with more encodings.
- Update documentation.

Version 2.2.40 (2012-07-21)

- Add PHSC() helper function as a replacement of htmlspecialchars() for PHP 5.4 (PITS:01292).
- Update documentation.

Version 2.2.39 (2012-06-25)

- Fix URL encoding of attachment links.
- Update documentation.

Version 2.2.38 (2012-05-21)

- Fix "Wrong parameter count for utf8_decode" warning, reported by Simon.
- Update documentation.

Version 2.2.37 (2012-05-01)

- Add page filename encoding functions.
- Better handling of dots in `[[#anchor_1.2]]` sections (PITS:01285).
- Expand PageVariables in PageList templates defaults (PITS:01282).
- Add test for iconv() and mb_convert_encoding(), refactor recode().
- Update documentation.

Version 2.2.36 (2011-12-28)

- Add $EnableOldCharset variable and $page["=oldcharset"] entry.
- Refactor PageStore->recode() to recover Windows-1252 characters.
- Add exit line to xlpage-iso-8859-2.php (PITS:01275).
- Fix difference in defining and removing "invisible" PTVs.
- Update documentation.

Version 2.2.35 (2011-11-11)

- Fix critical PHP injection vulnerability (PITS:01271, reported by Egidio Romano).
- Important change: Disable script loading from XLPage().
- Move the processing of `[[link|+]]` inside LinkPage() and delete markup rule from stdmarkup.php.
- Modify MakeLink() to better handle link titles.
- Add optional $LinkTitleFunction allowing recipes to customize the link titles.
- Fix ReadTrail() to better handle links with titles.
- Add title attributes for the HTML templates in the`$LinkPage*Fmt` variables.
- Add upload extensions svg, xcf, ogg, flac, ogv, mp4, webm, odg, epub.
- Minor optimization for the MarkupExpressions for UTF-8 strings.
- Minor optimization of the rendering of page history.

Version 2.2.34 (2011-10-10)

- Add MarkupExpressions replacements for UTF-8.
- Reset timestamps of Site(Admin).AuthUser to 1000000000, used in upgrades.php.
- Update documentation.

Version 2.2.33 (2011-09-23)

- Fix locked states for Site and SiteAdmin GroupAttributes (reported by Brijesh Kothari).
- Fix intermap.txt entries PITS: and Wikipedia: to point to their current locations.
- Fix refcount.php to produce valid HTML (PITS:01266).

Version 2.2.32 (2011-09-18)

- Add required html xmlns attribute to the print skin template.
- Add PageStore->recode() function.
- Add `$DefaultPageCharset` array.
- Optimize for speed the inline diff for page history when too many lines were added or deleted.
- Update and convert to UTF-8 the documentation.

Note: Due to a manipulation error, a version 2.2.31 was created before it was ready for a release.

Version 2.2.30 (2011-08-13)

- Fix $Charset definition in iso-8859-*.php files.
- Add $EnableRangeMatchUTF8, set it to 1 to enable range matches in UTF-8.
- Update documentation.

Version 2.2.29 (2011-07-24)

- Fix Attach links that were broken with the Path fix in 2.2.28.
- Add $IMapLocalPath array containing InterMap prefixes that should be treated as local.

Version 2.2.28 (2011-07-24)

- Fix potential XSS vulnerability in refcount.php (PITS:01262).
- Fix bug in Path: links (PITS:01260).
- Fix potential XSS vulnerability in custom SitePreferences (PITS:01263).
- Update documentation.

Version 2.2.27 (2011-06-19)

- Add block WikiStyle %justify% (PITS:01253).
- Remove unused <vspace> after a redirection (PITS:01255).
- Add ?nodiff=1 parameter for page history to disable diff rendering and show only restore links.
- Update documentation.

Version 2.2.26 (2011-05-21)

- Fix ReadTrail(), redundant replacing of hashes, already done in MakePageName().
- Update documentation.

Version 2.2.25 (2011-03-22)

- Update documentation.

Version 2.2.24 (2011-02-15)

- Add `{$$PageTrailDepth}` pseudovariable for PageList templates.
- Fix PageVar(), add $authpage array for an authenticated page data, removed `$EnablePageVarAuth`.
- Update documentation.

Version 2.2.23 (2011-01-25)

- Default `$EnablePageVarAuth` to 0 until the resolution of PITS:01242.

Version 2.2.22 (2011-01-16)

- Add `$EnableXLPageScriptLoad` to XLPage() to prevent editors from changing the encoding.
- PageVariables now respect authentications (PITS:01213).
- Add `$EnablePageVarAuth`.
- Update documentation.

Version 2.2.21 (2010-12-14)

- Fix potential XSS vulnerability, reported by DFaure.
- Fix invalid HTML for simple table captions, reported by JL.
- Fix WikiStyles could work not properly if a value was empty like accesskey="".

Version 2.2.20 (2010-12-14)

- Fix Pagelist {$$variable} didn't work in template none (PITS:01212).
- Fix interface access keys in browse mode (PITS:01188).
- Add PmL10n: intermap prefix for the Localization/ group on pmwiki.org (PITS:01180).
- Fix AuthUser excluding members didn't work (PITS:01201).

- Update documentation.

Version 2.2.19 (2010-11-10)

- Update documentation.

Version 2.2.18 (2010-09-04)

- Fix $SaveAttrPatterns to skip nested conditionals (reported by RandyB).
- Fix RecentChanges when an edit summary contains the dollar sign (PITS:01217).
- Fix RDF feed number of elements (PITS:01198).
- Update documentation.

Version 2.2.17 (2010-06-20)

- Add tabindex as a valid form attribute (PITS:01190).
- Collapse adjacent insertions in DiffRenderSource (PITS:01192).
- Fix HandleDownload to flush() output before exit (PITS:01199).
- Fix HandleDownload to respect `$EnableIMSCaching` (PITS:01191).
- Add $PostConfig functions and scripts, loaded after stdconfig.php (PITS:01132).
- Add $AuthUserPat variable for the regexp pattern in AuthUserId() (PITS:01202).
- Pass $authlist as last parameter to $AuthUserFunctions (PITS:01197).
- Fix "exists" conditional to work with old link markup.
- Update documentation.

Version 2.2.16 (2010-05-10)

- Allow "exists" conditional to accept wildcards (PITS:01184)
- Fix GUI button %center% which didn't work correctly.
- Fix incorrectly parsed quote in PQA(), possible script injection (discovered by Hanno Boeck).

Version 2.2.15 (2010-03-27)

- Add `(Auth|Edit)Form` to auto-translated titles.
- Fix `(:if auth LEVEL:)` to respect `$HandleAuth` (PITS:01164).
- Skip loading of the second half of draft.php if $action!="edit".
- Fix bug with `(:template none:)` introduced in 2.2.14, reported by Holger.
- Fix HandleDownload() to use binary file-read.

Version 2.2.14 (2010-02-27)

- Fix inline styles in WikiTrails (PITS:01121).
- Add a negation parameter to pagelist first/last templates (PITS:01127).
- Refactor FPLTemplateFormat(), move repeated code blocks into FPLExpandItemVars().
- Add `$EnableUndefinedTemplateVars` allowing to hide or show undefined template/include {$$variables} (PITS:01152).
- Add "title" attribute to external links (PITS:00657).
- Add FmtPageTitle() to allow automatic i18n titles for RecentChanges and other technical pages (PITS:01157).
- Update documentation.

Version 2.2.13 (2010-02-21)

- Replace deprecated in PHP 5.3 function split() with explode().
- Add $WordDiffFunction default to PHPDiff().
- Use existing border colors as highlighting background.
- Refactor/optimize DiffRenderSource(), merge with DiffRenderInline().
- Change default history to show word-level highlighting.
- Fix bug with `$DiffKeepNum` which kept less revisions than it should.
- Fix RetrieveAuthPage() call from HandleDiff().
- Update documentation.

Version 2.2.12 (2010-02-17)

- Allow a custom $DiffHTMLFunction to skip the line rendering if it returns false.
- Add `$EnableDiffInline`, simple word-level diffs (PITS:00571).
- Update documentation.

Version 2.2.11 (2010-02-14)

- Break PrintDiff() into customizable functions (PITS:01106).
- Add anchors to individual diffs (PITS:00796).
- Remove unused $RecipeInfo definition in markupexpr.php (reported by P.Bowers).
- Add (:head:) and (:headnr:) table directives (PITS:00535).
- Fix `$GroupPattern` and `$NamePattern` in xlpage-utf-8.php.
- Update documentation.

Version 2.2.9, 2.2.10 (2010-01-17)

- Fix i18n string in PasswdVar(), reported by SteP.
- Fix sample-config.php with correct information about `$EnableWSPre` (PITS:01145).
- Fix range searches for wikis in UTF-8 (reported by Maxim).
- Fix global variable $StringFolding in scripts/xlpage-utf-8.php.
- Fix markup for italics in creole.php.
- Fix previews for PTVs, PageList templates and included sections (PITS:01098).
- Add `$DiffKeepNum` - number of revisions kept, even if older than `$DiffKeepDays`.
- Add Yandex to robots.php.
- Change default `$EnableRelativePageVars` to 1 (PITS:01145).
- Add fifth parameter to SetProperty() : keep existing property.
- Add `$EnablePageTitlePriority` (PITS:00266, PITS:00779).
- Update documentation.

Version 2.2.8 (2009-12-07)

- Fix apostrophes in Author field (PITS:01155).
- Fix Condition "exists" for PHP 5.3 (PITS:01156).
- Update documentation.

Version 2.2.7 (2009-11-08)

- Fix GlobToPCRE() to work with !excl and -excl with PHP 5.3 (PITS:01149).
- Fix HandleDownload() correctly quote the filenames (PITS:01150).
- Fix SessionAuth() for PHP 5.3, the $_REQUEST array doesn't contain the $_COOKIE array (PITS:01141).
- Fix default timezone for PHP 5.3 (PITS:01141).
- Update documentation.

Version 2.2.6 (2009-10-04)

- Escape apostrophes for multiline textarea/hidden form fields.
- Fix global unset of $MarkupRules in Markup() and DisableMarkup(), reported by D.Faure.
- Fix call to BuildMarkupRules() in MarkupToHTML(), suggested by Pm.
- Allow disabling of $PageListFilters and $FPLTemplateFunctions if set to -1 and thus allow replacing a core function with a custom one.
- Fix DRange() returned timestamps +1min or +1day when it shouldn't (PITS:01125).
- Add $MarkupWordwrapFunction to allow custom `(:markup:)` line width for multibyte wikis (PITS:00703).
- Add `$MakeUploadNamePatterns` to allow custom filename normalization for uploads.
- Add a fourth argument to PostRecentChanges() to allow this function to be called with a custom `$RecentChangesFmt` array.
- Add `$RecentUploadsFmt`, to allow logging of new uploads to the RecentChanges pages (PITS:00088).
- Fix Notify for some installations in safe_mode (PITS:00976).
- Add `$HTMLHeaderFmt`['guiedit'] variable in guiedit.php to allow customization (PITS:01146).
- Update documentation.

Version 2.2.5 (2009-08-25)

- Add *.7z as accepted upload extension (PITS:00813).
- Fix global variable $HandleAttrFmt in HandleAttr (PITS:01126).
- Allow brackets in input element names (PITS:01131).
- Fix CSS class applied twice (PITS:01071).
- Fix Not-Modified headers could prevent caching (PITS:00802).
- Break FPLTemplate() into configurable sub-parts (PITS:01102).
- Add `(:template none:)` section for PageList templates.
- Fix attr-protected page could be deleted with edit permissions (PITS:00238).
- Update documentation.

Version 2.2.4 (2009-07-16)

- Fix bug with page attributes, which somehow didn't make it in the 2.2.3 release.
- Fix bug with HTML entities in XLPages introduced earlier today in 2.2.3 (reverted, PITS:01114).

Version 2.2.3 (2009-07-16)

- Fix action=logout could incorrectly set a session cookie (PITS:01062).
- Fix page history trim in vardoc.php (PITS:01103).
- Add `$EnableUploadGroupAuth`, use group password for downloads (PITS:01104).
- Fix recursive PTV loops, added `$MaxPageTextVars` (PITS:00915, PITS:01099).
- Fix mkdirp() messages for absolute paths (PITS:00396).
- Fix sample-config.php order for urlapprove.php (PITS:01037).
- Fix broken signature links on preview.
- Fix crypt.php (action=crypt) could malfunction for passwords with quotes or apostrophes.
- Fix `@_site_*` passwords to work in GroupAttributes (PITS:00836, PITS:00998).
- Fix possible XSS vulnerabilities, reported by Michael Engelke.
- Update documentation.

Version 2.2.2 (2009-06-21)

- Fix class in pages not on the breadcrumbs trail, reported by Ed W.
- Fix `tabindex` and `onclick` to guiedit buttons.
- Fix `$GroupPrintHeaderFmt` in print.php (PITS:01073).
- Fix global vars in xlpage-utf-8.php (PITS:00980).
- Fix $txt in LinkPage (reported by Eemeli Aro).
- Add `$EnableNotifySubjectEncode` for international wikis (Cookbook:UTF-8).
- Fix international message in Abort().
- Fix security bug with AuthUser, reported by Eemeli Aro. See Release notes.
- Fix $ActionTitleFmt for login and upload, reported by Eemeli Aro.

Version 2.2.1 (2009-03-28)

- Fix $FPLTemplateMarkupFunction which somehow didn't get in the 2.2.0 archive.
- Fix wikitrails to work cross-group (PITS:00407).
- Add `$EnableRedirectQuiet` variable (PITS:00919).
- Fix {$Title} could display global variables (reported by HansB).
- Fix reloaded form submissions could lose values (reported by DaveG).
- Fix preview while restoring a version from history (PITS:01081).
- Fix relative links with international characters (reported by G. Hermanowicz).
- Add in sample-config.php example call to xlpage-utf-8.php (PITS:01066).
- Update documentation.
- Fix guiedit.php to produce valid HTML.

Version 2.2.0 (2009-01-18)

- Convert beta series to official release series.
- Add $FPLTemplateMarkupFunction (PITS:00984, requested by John Rankin).

Version 2.2.0-beta68 (2008-08-14)

- Fix E_NOTICE errors reported by Dominique Faure.
- Enable `(:redirect:)` directives in pagelists.

Version 2.2.0-beta67 (2008-07-13)

- Add {$LastModifiedTime} page variable.
- Add `$EnableSessionPasswords` variable to control session password usage.
- Add `$SessionEncode` and `$SessionDecode` variables to specify functions for encoding/decoding sensitive session data.
- Updated httpauth.php to use SessionAuth instead of poking in session guts directly.

Version 2.2.0-beta66 (2008-07-04)

- Add content-type/charset to Abort() output (suggested by Petko).
- Close minor XSS vulnerability (PITS:01030).
- Add "nested if" capability.
- Fix bug in $Transition handling that would enable all transitions if any were set (reported by John Rankin).

Version 2.2.0-beta65 (2007-11-17)

- Fix SiteAdmin.AuthList so that it defaults to list=all (reported by Roman).
- Fix pmwiki skin to include xmlns= attribute in <html> tag (PITS:00989, reported by Mateusz Czaplinski and Petko Yotov).

Version 2.2.0-beta64 (2007-11-13)

- Add times to PmWiki date parsing (e.g., 2007-08-09T12:22:04).
- Suppress warning from ini_set in diag.php (suggested by Petko).
- Fix handling of -> links in trails (reported by Eemeli Aro).
- Add .kml and .kmz as valid attachment types.
- Fix handling of &amp; in markup (PITS:00988, reported by Stirling Westrup).
- Fix duplication of language markers in `$XLLangs` (PITS:00987, reported by Stirling Westrup).
- Correct typo in DRange() call in stdmarkup.php (reported by Stirling Westrup).
- Turn on error displays when diagnostics are enabled.
- Default PHP's pcre.backtrack_limit to at least 1000000.

Version 2.2.0-beta63 (2007-07-31)

- Added $SkinDirectivesPattern to allow adjustments to available skin directives (requested by Petko).
- Fix default permissions on Site.AuthUser and Site.AuthList (reported by Scott Connard).
- Add "monospace" to pmwiki.css default (reported by Joshua Timberman, with assistance from H. Fox)
- Fix problem with slashes in wildcards to name= and group= parameters (reported by Ian MacGregor).

Version 2.2.0-beta62 (2007-07-21)

- Fix bug in trails introduced by beta61 (reported by charlequin).

Version 2.2.0-beta61 (2007-07-19)

- Add ability to grab trails by section.
- Add an "ontrail" condition (from suggestions by charlequin).

Version 2.2.0-beta59, 2.2.0-beta60 (2007-07-18)

- Fix problem with upgrade.php on wiki farms (reported by Scott Connard).
- Fix problem with distributed version of Site.AuthUser (reported by Jon Haupt).

Version 2.2.0-beta58 (2007-07-17)

- Significant change: Site.AuthUser, Site.Blocklist, Site.ApprovedUrls, and Site.NotifyList now appear in the SiteAdmin group by default.
    - Note: if you limit groups by setting `$GroupPattern`, you now need to include SiteAdmin (see Cookbook:LimitWikiGroups)
- Abort if ldap: authentication requested and libraries aren't present.
- Added "upgrades.php" script to handle various migration issues.
- Current PmWiki version is now held in SiteAdmin.Status .
- Fix ?action=postupload to follow ?action=upload settings.
- Improvements to SiteAdmin.AuthList page (suggestions and fixes from Ian MacGregor).
- Allow leading underscores in attachment names (requested by Christophe David).

Version 2.2.0-beta57 (2007-06-15)

- Fix AsSpacedUTF8() to work like AsSpaced() (reported by Petko).
- Qualify page links that contain parentheses (reported by Petko).
- Fix bug in `(:input default $:var ... :)` (reported by Crisses).

Version 2.2.0-beta56 (2007-06-13)

- Fix AsSpaced() to not add spaces before leading digit, and treat hyphenated digits as complete numbers.
- Fix infinite recursion in self-referencing page text variables (PITS:00915).
- Fix bug introduced in beta55 not handling end anchors correctly (reported by Roman).

Version 2.2.0-beta55 (2007-06-11)

- Fix attributes to `(:input e_form:)` (PITS:00387, re-reported by Crisses).
- UpdatePage() now calls StopWatch() to record posting.
- Display stopwatch output as part of redirect.
- Fix wiki styles bug when `$EnableLinkPageRelative` is set (reported by Petko).
- Revise TextSection() code to hopefully avoid pcre limits (reported by Kathryn Andersen, Knut Alboldt).
- Add wrap=inline and wrap=none options to page list.

Version 2.2.0-beta53, 2.2.0-beta54 (2007-06-02)

- Improve error message reporting for markup rules (suggestion by Knut Alboldt).
- Clean up more E_NOTICE warnings (reported by Ian MacGregor).
- Add focus= option to (: input:) controls.
- Added CSS `.faqtoc` class, to be able to display only the questions coming from the #includefaq page list template.
- Changed PmWiki.FAQ to use .faqtoc class.
- Fix bug in TextSection (PITS:00935, reported by Jean-Fabrice).
- Fix bug in page list caching of trails.

Version 2.2.0-beta52 (2007-05-26)

- Add per-PageStore attributes (from a suggestion by Tobias Thelen).
- Add `{$PasswdRead}`, `{$PasswdEdit}`, etc. to display page password settings.
- Add Site.AuthList to display all password permissions on a site.
- Reorder $PageListFilters slightly.
- Add "passwd=" option to page list, to return only those pages that have some sort of password attribute on them.
- Add line numbers to StopWatchHTML output.
- Clean up handling of $AuthCascade.

Version 2.2.0-beta51 (2007-05-23)

- Add fmt=count to page list (reminder from Hans).
- Ignore hidden files in skin directories when searching for .tmpl (suggestion by Stephan Becker).
- Clean up queuing of pages to be updated in .pageindex .
- Reset $LinkTargets() at beginning of each UpdatePage() sequence.

Version 2.2.0-beta50 (2007-05-22)

- Fix HTML cache when drafts are enabled, or other recipes using CondAuth().
- Prevent page lists with protected pages from HTML cache.

Version 2.2.0-beta48, 2.2.0-beta49 (2007-05-21)

- Fix spurious value= attribute in <textarea> tag generated by `(:input textarea ... :)`.
- Allow either `(:input default ...:)` or `(:input defaults ...:)`.
- Fix problem with page text variable handling in `(:input defaults:)`.
- Allow either `(:template default:)` or `(:template defaults:)` in page list templates.
- Fix a bug handling dates with suffixes (reported by Crisses).

Version 2.2.0-beta47 (2007-05-20)

- Fix bug with quote handling in `(:include:)` options (reported by Hans).

Version 2.2.0-beta46 (2007-05-19)

- Moved $PageTextVarPatterns definition from scripts/stdmarkup.php to pmwiki.php.
- Ignore Markup() rules that have unresolved $when parameters.
- Fix issue in authuser.php when $auth array isn't set (contributed by Ben Stallings).
- The `(:include:)` directive now performs template argument processing on the included text.
- Optimized `(:pagelist:)` slightly when sorting on page variables.
- Refactored `(:input ... :)` markups.
- Added HandleDispatch(), which allows action handlers to easily redispatch to other actions (and add messages).
- Added FmtTemplateVars(), to perform various template-substitutions.

Version 2.2.0-beta45 (2007-05-02)

- Update pmwiki's date parsing to use a common routine, recognizing dates within strings and restricting range to 1900-2039.
- Add additional parameter to "date" conditional.
- Add if= option to page list (suggested by Crisses).
- Refactor code to use TextSection() and RetrieveAuthSection() functions.
- The value= parameter to `(:input textarea:)` now works properly (including values loaded from $InputValues).
- The `(:input default:)` directive now allows loading input control defaults from another page via the `source=` parameter.
- Remove automatic call to FmtPageName() in `$ROSPatterns`. Add `$ROEPatterns` (from suggestions by JB and others).
- Fix minor variable bugs in scripts/crypt.php.
- Remove E_NOTICE errors (reported by Hans).
- Fix handling of page variables when pagename is empty or not provided.
- Add `$EnableLinkPageRelative` configuration option.
- Clean up handling of arguments to `{(ftime ...)}`.
- Remove mailposts.php call in stdconfig.php (reported by Christophe David).

Version 2.2.0-beta44 (2007-04-16)

- Fix case conversion of U+027D and U+026B (reported by Petko).
- Add `$FTimeFmt` to set default formatting for `{(ftime)}`.
- Add %s conversion to `{(ftime)}` for systems that don't have it by default.
- Report an error if edit form cannot be read (suggested by Hans).
- Don't report ?cannot acquire lockfile when simply browsing pages.
- Add $EnableReadOnly flag to signal when PmWiki is to be run in read-only mode.

Version 2.2.0-beta43 (2007-04-15)

- Update drafts code to add `$EnablePublishAttr` and change button labels when drafts are enabled (PITS:00755).
- Removed no-longer-needed 'compat1x.php' and 'mailposts.php' from distribution.
- Added `$DraftRecentChangesFmt`.
- Added " markup expressions" `{(...)}` into the core.
- Added charset= attribute to saved pages.
- Update pagelist.php and xlpage-utf-8.php to handle case-insensitive searches.
- Added some optimizations to phpdiff.php script to produce more useful history information.

Version 2.2.0-beta42 (2007-03-27)

- Fix a bug with order=title in pagelists (reported by Anno).

Version 2.2.0-beta41 (2007-03-26)

- Added `$EnableWSPre` option, which allows easy adjustment of the "leading space -> preformatted text" (or "whitespace") rule.
- Added a new "pre" wikistyle, to designate blocks that are to be treated as preformatted text.

Version 2.2.0-beta40 (2007-03-24)

- Fix bug with order=title in pagelists when using $Titlespaced (PITS:00906, reported by Feral).
- Report state of allow_url_fopen when downloads fail in blocklist.php.

Version 2.2.0-beta39 (2007-03-23)

- Allow page variable filters to appear as options in `(:template defaults:)` (reported by SteP).

- Updated Site.PageListTemplates to use `(:template:)` directives.
- Remove '#wikileft h1' and '#wikileft h5' from pmwiki default stylesheet.

Version 2.2.0-beta38 (2007-03-22)

- Strip control characters from $ChangeSummary.
- Fix problem with count=m..n where m..n is outside the range of available pages (reported by SteP).
- Allow `(:template default ...:)` to specify a class= option.
- Redirect pagename can now include an anchor (PITS:00558)

Version 2.2.0-beta37 (2007-03-16)

- Allow an optional space after comma separators in wildcard patterns (reported by Han Baas).

Version 2.2.0-beta36 (2007-03-16)

- Allow nested page text variables to work, remove extraneous ENT_NOQUOTES parameter.
- Add new `(:template ...:)` directives for PageList templates.
- Modify count= option to pagelists to allow for alternate ranges.

Version 2.2.0-beta35 (2007-03-05)

- Fix bug in conditional markup parsing (reported by Christophe David).

Version 2.2.0-beta33, 2.2.0-beta34 (2007-03-01)

- Refactor wildcard handling into its own GlobToPCRE function.
- Allow negated wildcards for page variable filters in pagelists (PITS:00878, reported by Jiri)
- Fix wildcards so that spaces no longer separate patterns (use commas).
- Fix handling of '&' prior to `(:input:)` and other directives (reported by Luigi).
- Adjust position of `%define=...%` wiki styles to occur after ampersands.
- Adjust copyright dates on many files.
- Allow spaces around text variable names in page text variable markups.

Version 2.2.0-beta32 (2007-02-28)

- Fix erroneous $EnableCreole item in docs/sample-config.php (reported by Sigurd).
- Added `(:elseif:)` and `(:else:)` markups (PITS:00787).
- Fix global `$Skin` variable handling when using SetSkin from within markup.
- Make sure directives aren't treated like page text variables (reported by Petko).
- Remove call to ResolvePageName() from authuser.php .
- Simplify LDAP authentication for Active Directory sites.
- Cache lowercase/uppercase patterns in AsSpacedUTF8().

Version 2.2.0-beta31 (2007-02-11)

- Fix bug with sorting on pagelist variables (reported by Kathryn Andersen).

Version 2.2.0-beta29, 2.2.0-beta30 (2007-02-09)

- MakePageName now uses the first matching entry of `$PagePathFmt` as the home page of groups without a home page.
- Add AsSpacedUTF8() to handle title spacing in utf-8 (PITS:00875, contributed by Petko, Celok)
- Fix $RequestedPage when running with utf-8.
- Add <meta> content-type tag for utf-8.
- Add an experimental caching system for pagelists.
- Fix $SuffixPattern and link suffixes for utf-8 (PITS:00881, reported by ppip).

Version 2.2.0-beta28 (2007-02-03)

- Update blocklist.php so that all posted fields are checked for block values (PITS:00850).

Version 2.2.0-beta27 (2007-01-25)

- Fix markup processing sequence for `(:input default:)`, `(:input select:)`, etc. (problem noted by Marc).
- Fix default value of `order=` parameter to MakePageList().

Version 2.2.0-beta26 (2007-01-23)

- Fix a bug where pagelist list= option had no effect when reading from trails (from an rss problem noted by Russ Fink).

Version 2.2.0-beta24, 2.2.0-beta25 (2007-01-22)

- Add a scripts/creole.php module for Creole markup (http://www.wikicreole.org/).

- Move WikiWords out of the core defaults -- can be enabled via `$EnableWikiWords`.
- Fix handling of WikiWords following & or #, as in &AElig; and #FFFF00 (reported by Moni Kellermann).
- Adjust FormatTableRow() to support Creole-style tables (using single |'s).
- Update docs/sample-config.php with new configurations and options.
- Added code to allow Abort() to refer to additional information on pmwiki.org.
- Added $EnableSkinDiag, which checks templates for required <!--HTMLHeader--> and <!--HTMLFooter--> directives.
- Removed deprecated $BasicLayoutVars support from skins.php.

Version 2.2.0-beta22, 2.2.0-beta23 (2007-01-17)

- Added $EnableActions, to allow pmwiki.php to be included without generating output (from a suggestion by Wouter Groeneveld).
- Fix bug in "order=" option to `(:pagelist:)` (reported by Mike Bishop).
- Change DisplayStopWatch() function to StopWatchHTML().
- Allow multiple lines for markup:, wiki:, and page: template directives (reported by Marc)

Version 2.2.0-beta21 (2007-01-12)

- Fix <vspace> bug in searchresults output (PITS:00846, reported by M. Czaplinski, marc, and others).
- Fix numerous E_NOTICE warnings and incorrect constants (PITS:00853, contributed by AndrewFyfe).

Version 2.2.0-beta20 (2007-01-11)

- $FeedPageListOpt needs to be declared global in feeds.php.
- Add "404 Not Found" status code to ?invalid page name aborts (PITS:00854, suggested by Athan).
- Remove stale entries from $PageExistsCache when a new PageStore is added (reported by Hans).

Version 2.2.0-beta19 (2006-12-29)

- Have blocklist check $_POST['text'] only when it is set (from a report by Simon).

Version 2.2.0-beta18 (2006-12-28)

- Change `$pagename` parameter in UpdatePage() to be passed by reference (suggestion by J. Meijer).
- Fix $EnableRobotsCloakActions so that it works again with page variables.
- Add "XML Sitemaps" to $RobotPattern.
- Change `$MetaRobots` to return "nofollow,noindex" for non-existent pages.
- Prefer "404 Not Found" to "403 Forbidden" for robots attempting to do invalid actions on non-existent pages.
- Add rel='nofollow' to "create attachment" links.
- Added class='inputbox' to select boxes (suggested by Hans).
- Added .odt, .ods, and .odp file extensions to allowed uploads (suggested by Algis Kabaila, Robin Sheat, and others).
- Clean up some error warnings (PITS:00801, contributed by psvo).
- Set `$ScriptUrl` to 'https:' when accessed via SSL link (suggestions from C. Ridderström, H. Fox, PITS:00410, PITS:00527, PITS:00595).
- Fix bug in link= and trail= options to `(:pagelist:)` (reported by C. Ridderström).

Version 2.2.0-beta17 (2006-12-13)

- Fix spurious hidden field in `(:searchbox:)` output (reported by Hans).
- Fix $CaseConversions array for \xc4\xb1 and \xc5\xbf (reported by Petko Yotov).
- Refactor `(:input:)` markup handling.
- Add `(:input select ...:)` markup (PITS:00567).
- Add `(:input default ...:)` markup -- may change before 2.2.0 release.
- Add ability to set defaults for radio/checkbox/select controls.

Version 2.2.0-beta16 (2006-11-10)

- Fix problem with `(:e_preview:)` directive when viewing an edit form (reported by Dominique Faure).
- Fix out-of-memory problem in scripts/compat1x.php when dealing with large pages to be converted (contributed by Donald Gordon).
- Fix problem of Variable: lines immediately followed by newline (reported by Hans).
- Fix uninitialized variable errors in FormatTableRow() (reported by Bob Sanders).
- Fix second argument of MakeBaseName() (provided by Stirling Westrup).

Version 2.2.0-beta15 (2006-10-16)

- Fix bug with displaying multi-line `(:var:value:)` page text variables (reported by Pico).
- Improve PageStore ls() method slightly, to restrict pagename searches to directories of a given depth (based on an issue reported by Chris Cox).

- Added $IsBlocked status variable to scripts/blocklist.php.
- Added $UnapprovedLink array to report unapproved links.
- Added $TimeISOFmt, $TimeISOZFmt, and $CurrentTimeISO variables.
- Switched scripts/feeds.php to use $TimeISOZFmt instead of $ISOTimeFmt.
- Added `request=` option to `(:pagelist:)`, switched pagelist to default to not use url/form parameters.
- Fixed bug with array `{$$options}` in pagelist.

Version 2.2.0-beta14 (2006-10-06)

- Fix problem with extra parameter to mail when `$NotifyParameters` is empty (reported by Tom Lederer).
- Improve configurability of `$SearchPatterns` (from suggestions by Stirling Westrup).
- Add ability for `$WikiWordCount` to disable wikiword spacing (PITS:00327).

Version 2.2.0-beta13 (2006-10-04)

- Fix handling of angle brackets (and potential XSS) in pagelists combined with page text variables (noted by Pico).

Version 2.2.0-beta12 (2006-10-03)

- Added the UpdatePage() function into the core.

Version 2.2.0-beta11 (2006-10-03)

- Added ability to automatically create targets.
- Added sample code to docs/sample-config.php for automatic generation of Category.* pages.
- Fixed character escapes in pagelist `{$$option}` variables.

Version 2.2.0-beta10 (2006-10-02)

- Added `{$$option}` variables to get option values from `(:pagelist:)` (based on a recipe from Martin Fick).
- Changed `{$PageCount}, {$GroupPageCount}, and {$GroupCount}` to be `{$$PageCount}, {$$GroupPageCount}, and {$$GroupCount}`.
- Added `{$BaseName}` page variable and `$BaseNamePatterns`.

Version 2.2.0-beta9 (2006-10-01)

- Fix bug with `$EnablePageListProtect` (reported by Brent Zupp).
- Added ability to select based on page variables in `(:pagelist:)`.

Version 2.2.0-beta8 (2006-09-30)

- Update scripts/blocklist.php to check only $_POST['text'] instead of entire markup text.
- Fix bug in pagelist.php that wouldn't return correctly formatted array in certain circumstances (noted by Florian Fischer and JDem).

Version 2.2.0-beta7 (2006-09-30)

- Added scripts/blocklist.php to core.
- Updated handling of $PageTextVarPatterns.
- Eliminated need for extra flush() steps in notify.php, pagelist.php.

Version 2.2.0-beta6 (2006-09-27)

- Fix bug with initialization of $FeedPageListOpt in scripts/feeds.php (reported by Roman).
- Fix bug with over-eager `(:textvar:value:)` markup (from a bug reported by Chris Cox).

Version 2.2.0-beta4, 2.2.0-beta5 (2006-09-27)

- Fix bug with name= option in pagelist (reported by Ben Wilson).
- Fix bug with array_merge under PHP 5 (reported by Kathryn Andersen).

Version 2.2.0-beta3 (2006-09-26)

- Remove extra <!----> comment at end of table directives (noted by Ben Stallings).
- Fix directive form of page text variables (reported by Kathryn Andersen).
- Add first version of new modular pagelist code.

Version 2.2.0-beta2 (2006-09-25)

- Add support for `{$:var}` page text variables, and `(:var:...:)` markup.
- Fix default setting of `$EnableRelativePageVars` in docs/sample-config.php .

Version 2.2.0-beta1 (2006-09-25)

- Added `{*$var}` page variables (always the currently browsed page).
- Convert link and page variable handling in (:include:) to be relative to the included page.
- Added `$EnableRelativePageVars` and $EnableRelativePageLinks variables, as well as transition options.

- Added basepage= option to (:include:).
- Updated `$GroupHeaderFmt` and `$GroupFooterFmt` to use basepage= option.
- Adjusted `$MakePageNamePatterns` to automatically strip any #... or ?... from the end of a pagename input string (solution to a problem reported by J. Meijer).

Version 2.1.27 (2006-12-11)

- Backport in bug fix for TableRowFormat (from 2.2.0-beta16).
- Add support for `{*$Variable}` syntax (from 2.2.0 page variables).

Version 2.1.26 (2006-09-11)

- Fix a bug with variable referencing that caused feeds.php to get a confused PCache (reported by Helge Larsen).

Version 2.1.25 (2006-09-08)

- Fixed a bug in authuser.php that would fail if $AuthUser isn't defined (reported by Hans Huijgen).
- Added <!--XMLHeader--> and <!--XMLFooter--> aliases to <!--HTMLHeader--> and <!--HTMLFooter--> directives in skin templates (suggested by John Rankin).
- Added $PageExistsCache (suggested by John Rankin).

Version 2.1.24 (2006-09-06)

- Fixed a bug in authuser.php that had trouble dealing with non-array entries in $AuthUser (reported by Udo).
- Can now specify authorization groups using $AuthUser['@group'] entries.
- Can now specify an Apache .htgroup-formatted file for authorization groups via $AuthUser['htgroup'].

Versions 2.1.21, 2.1.22, 2.1.23 (2006-09-05, 2006-09-06)

- Close a potential security hole with `$FarmD` when register_globals is set "On".
- Correct a syntax error in feeds.php (noted by Ben Wilson).
- Fix a bug that prevented PmWiki from reading page files generated by versions prior to 0.5.6 (discovered by Milan Avramovic).

Version 2.1.20 (2006-09-04)

- Fixed a bug in `(:attachlist:)` when passed a wikiword argument (reported by Kathryn Andersen).
- Changed `$HTMLStylesFmt`['markup'] to honor config.php setting (reported by Hans).

Version 2.1.19 (2006-08-30)

- Corrected a bug in the pageindex code that was causing the .pageindex to not update as quickly as it should.
- Slightly changed the handling of 'width' and 'height' in wikistyles.php, so that they can be be applied as attributes to <object> and <embed> tags.
- Updated the Keep() function to recognize closing block tags as being in the 'B' block pool.
- Fixed a bug with wikistyles and form tags.

Version 2.1.18 (2006-08-28)

- Closed a potential cross-site scripting vulnerability in table markups (reported by JB).
- Added `(:input image:)` markup (requested by JB).
- Fixed problem with ?action=print failing to set `{$Action}` (reported by Bart).

Version 2.1.17 (2006-08-26)

- Added some improvements to IMS caching to better handle logout and authorization actions (PITS:00573, reported by floozy and Henrik Bechmann).

Version 2.1.16 (2006-08-26)

- Added `$SkinLibDirs` variable, to select filesystem and url locations where skins may be found (resolves PITS:00708, as reported by Hagan Fox, with additional suggestions from Ben Wilson).
- Changed `<!--HeaderText-->` to `<!--HTMLHeader-->` in skin templates, and added an optional `<!--HTMLFooter-->` directive (PITS:00767).
- Adjusted the pmwiki and print skins to use the new directives.

Version 2.1.15 (2006-08-25)

- Fixed issue dealing with order of `@_site_*` passwords (reported by Jean-Fabrice and others).
- Added `$LocalDir` variable (requested by John Rankin).
- Removed an unnecessary setting of `$DefaultPage` in *scripts/pgcust.php* (it's now handled by ResolvePageName()).
- Added some variables and changes in wikistyles.php to better support wikipublisher (contributed by John Rankin).
- RetrieveAuthPage (PmWikiAuth) now recognizes a $level of 'ALWAYS' as indicating that access should always be allowed, regardless of current passwords or identities.
- Added filter specifier for AuthUser LDAP authentication (contributed by Balu).

Version 2.1.13, 2.1.14 (2006-08-15, 2006-08-16)

- Updated scripts/authuser.php to allow ldaps://... authentications (contributed by Michael Brenner).
- Fixed problem with numeric passwords introduced in 2.1.beta20 (reported by Christophe David and Dirk Blaas).

Version 2.1.12 (2006-08-07)

- Corrected typo in Site.SideBar file (reported by Judith Zacharie).
- Suppressed warning message for search on sites without a wikilib.d/ directory.
- Added capability for nested divs.
- Use $Transition['nodivnest'] to restore previous non-nesting div/table behavior.
- Including authuser.php now automatically resolves pagename.
- Added `(:noaction:)` directive to turn off actions.
- Fixed bug in wikistyles prior to image blocks.
- Added white-space as allowed wikistyle (suggested by C. Ridderström).
- Allow colons, hyphens, and dots in id= tags.

Version 2.1.11 (2006-06-09)

- Fixed generation of empty paragraphs around `%define=...%` wikistyles (PITS:00753).

Version 2.1.10 (2006-06-04)

- Added a <span> around the RecentChanges link in the pmwiki skin (PITS:00750, suggested by Hagan Fox).
- Changed the $Action variable to $ActionTitle (PITS:00749, reported by Hagan Fox).
- Changed `$FPLTemplatePageFmt` to be an array of pages to be searched for page templates, enabled searching of current page and Site.LocalTemplates page.
- Updated .vspace margin in sidebar for pmwiki skin (PITS:00751, by Hagan Fox).

Version 2.1.9 (2006-06-02)

- Fixed a bug with `[[~Author]]` links (PITS:00530 reported by Klonk, PITS:00611 reported by weijang, PITS:00671 reported by Stirling Westrup, and helpful clues provided by Clayton Curtis).

Version 2.1.8 (2006-06-01)

- Added ability to specify notification entries from *local/config.php* as well as Site.Notify (suggested by Christophe David).
- Fixed $Transition['vspace'] from 2.1.7.

Version 2.1.7 (2006-05-31)

- Adjusted width of edit form for IE browsers (contributed by Roman and H. Fox).
- Suppress authentication failure error from LDAP (PITS:00739).
- Fixed problem with invalid page names resulting in redirect loop (PITS:00723, reported by jojoo).
- Added "Group." and "Group/" page name syntax, resolving PITS:00736 (from a suggestion by Pico).
- Changed handling of "vspace" paragraphs.
- Fixed some XSS vulnerabilities in uploads.php and url links (reported by Moritz Naumann, http://moritz-naumann.com).
- Added notify.php script, allowing finer control of email notifications.

Version 2.1.6 (2006-05-22)

- Optimized performance of urlapprove.php.
- Added `(:if auth xyz PageName:)` syntax.
- Corrected XSS bug in trails.php.
- Slightly improved performance of free links.
- Restore ability to use hyphens in InterMap links (reported by Henrik Bechmann).

Version 2.1.4, 2.1.5 (2006-03-29)

- Fixed problem with pagelist-based feeds (PITS:00709, reported by Jon Haupt).
- Added `{$Action}` page variable. (PITS:00696, reported by Sebastian Pipping).
- Added stripmagic() around variables submitted to authuser.php.
- Fixed problem with multi-term searches containing special characters (PITS:00713, reported by Leo).
- Switched `(:attachlist:)` to use a natural case sort (suggested by H. Fox).

Version 2.1.3 (2006-03-17)

- Re-fixed problem with PHP 5.1.1 and lines= option to `(:include:)` (PITS:00620).
- Fixed empty LDAP password issue (reported by Thomas Lederer).

Version 2.1.2 (2006-03-16)

- Fixed <h1>/<h2> tag mismatches (PITS:00702, reported by Martin Hason).
- Fixed bug with `$AllowPassword` and "nopass" (reported by M. Weiner and bram brambring).
- Improved the speed of RSS and other web feeds when `$EnablePageListProtect` is not set.

Version 2.1.1 (2006-03-13)

- Fixed a bug with multiple authorization groups as a password (PITS:00699, reported by Ari Epstein).
- Updated the authorization code to be a bit more liberal with password/group settings.
- Updated PmWiki.FAQ page to be able to grab FAQ items from other pages in the documentation.

Version 2.1.0 (2006-03-12)

- Many many documentation updates (special thanks to many authors).
- Allow trailing underscores in upload names (requested by Hans).
- Fixed 'ak_print' problem causing accesskey='a' for print (noted by Pico).
- Added code to make sure each anchor is generated only once per page (for XHTML validity).
- Added a $BlockPattern variable to recognize block HTML tags.
- Made an adjustment to Keep() so that it places strings with block HTML into the 'B' pool.
- Adjusted stdmarkup.php to not produce paragraphs for keep blocks in the 'B' pool.
- Corrected a variety of i18n phrases.
- Added class='escaped' to distinguish @@...@@ from [@...@] (from a comment by Hans).
- Slightly changed styling of .faq divs.
- Made the edit textarea a couple of rows smaller to better fit on smaller displays (suggested by H. Fox).

ChangeLog Archive - changes prior to version 2.1.0.

Last modified by Petko on June 26, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/ChangeLog

ConditionalMarkup

toc top

Using the (:if:) Directive

The `(:if:)` directive allows portions of a page to be included or excluded from rendering. The generic forms of the `(:if:)` directive are

```
(:if cond param:) body (:ifend:)
(:if cond param:) body (:else:) body (:ifend:)
(:if cond param:) body (:elseif cond param:) body (:ifend:)
(:if cond param:) body (:elseif cond param:) body (:else:) body (:ifend:)
```

where "cond" names a condition to be tested, and "param" is a parameter or other argument to the condition.

Note that `(:if:)` without parameters and `(:ifend:)` are identical. Also note that `(:if cond:)` automatically closes a previous conditional. For nested multiple levels, see Nested conditionals.

Built-in Conditions

The built-in conditions include:

`(:if name PAGENAME:)`	- current page is named "*PAGENAME*" or "*GROUPNAME.PAGENAME*"
`(:if group GROUPNAME:)`	- current group is named "*GROUPNAME*"
`(:if auth LEVEL PAGENAME:)`	- viewer is authorized - meaning "what they are allowed to do" - matches a "`LEVEL`" where `LEVEL` can be: `read`, `edit`, `upload`, `attr` or `admin`; `PAGENAME` is optional.
`(:if authid:)`	- current viewer is authenticated - meaning they have proven who they are via login - to use this the wiki must include recipe AuthUser or others which set the `$AuthId` variable.
`(:if enabled InvalidLogin:)`	- username and password not authenticated. To use this the wiki must include recipe Cookbook:AuthUser.
`(:if true:)`	- always include text, case sensitive
`(:if false:)`	- always exclude text (same as a comment, but Page Text Variables ARE set), case sensitive
`(:if attachments FILENAMES PAGENAME:)`	- `PAGENAME` has one or more attachments among the specified. A pagename can be omitted, in that case the current page is implied. `FILENAMES` specify an attachment like "pic1.jpg" or attachment patterns separated by commas, like "pic*.jpg,*.png" where asterisk (*) means "anything"; if omitted, any attachment (i.e. "*") is implied. If used in a sidebar, header, or footer, and the`PAGENAME` is not specified, the condition applies to the main page. e.g. `(:if attachments "*.png,*.gif" Groupname.PageName:)`

In the following "if date" examples:

- `DATE` may be year-month. year-month-day is optional.
- `VALUE` can be a recognizable date via strtotime()
- `DATE` (or `DATE1` and `DATE2` below) have a more fixed format which explicitly must exclude spaces. Any spaces in `DATE1` or `DATE2` cause unpredictable results
- "now" or "today" is assumed if`VALUE` is omitted
- dates are in standard format `yyyy-mm-dd` or `yyyymmdd` or `yyyymmddThhmm` (note the "T" between the date and the hour, and also see comment above on format of `VALUE`)
- the ".." cannot have leading (when used with`DATE1`) or trailing spaces (when used with`DATE2`)

`(:if date DATE VALUE:)`	- Evaluates to true if `VALUE` is within `DATE`
`(:if date DATE1.. VALUE:)`	- true if `VALUE` (or current date if omitted) is `DATE1` or later (unlimited)
`(:if date ..DATE2 VALUE:)`	- true if `VALUE` (or current date if omitted) is `DATE2` or earlier (unlimited)
`(:if date DATE1..DATE2 VALUE:)`	- true if `VALUE` (or current date if omitted) is in range `DATE1` to `DATE2` (inclusive)
`(:if enabled VAR:)`	- true if PHP `VAR` is true
`(:if enabled AuthPw:)`	- true if user has entered any password during the current browser session. - This does not mean the user has entered the correct password, just that they entered one.
`(:if equal STRING1 STRING2:)`	- true if `STRING1` equals `STRING2`, use quotes if the string or string variable contains spaces, eg `"MY STRING"`
`(:if match REG_EXPRESSION:)`	- true if current page name matches the regular expression
`(:if exists PAGENAME:)`	- true if the page "*pagename*" or "*groupname.pagename*" exists
`(:if ontrail WikiTrailPage ThisPage:)`	- true if `ThisPage` is in a list used as a trail on "*WikiTrailPage*"

The name and group conditionals will work even for an included page, as the "name" and "group" conditionals always check the currently displayed page, as opposed to the page that the markup appears in.

Note: Although there is no built-in conditional markup to test ?action=, you can use `(:if equal {$Action} ACTION:)` to test what the current action being requested is.

Concatenated conditions

In some cases where built in conditions have a parameter the parameters may be concatenated using a comma, viz:

- `(:if name Name1,Name2,-Name3:)`
- `(:if group -Group1,Group2,Group3:)`

Negated Conditions

Negated forms of conditions also work:

`(:if !attachments:)`	- this page has no attachments
`(:if ! name PAGENAME:)` `(:if name -PAGENAME :)`	- current page is NOT named "`PAGENAME`"
`(:if group -GROUPNAME1,-GROUPNAME2 :)`	- group is not named "`GROUPNAME1`" or "`GROUPNAME2`"

Nesting Conditions

Note that `(:if cond:)` automatically closes a previous conditional. Thus, the following two examples have identical meaning:

- `(:if cond1:) cond1 is true (:if cond2:) cond2 is true (:ifend:)`
- `(:if cond1:) cond1 is true (:ifend:)(:if cond2:) cond2 is true (:ifend:)`

Conditions can be nested from 2.2.beta 66. To have nested conditionals you need to number the if, and the matching else/ifend:

```
(:if cond1:)
  cond1 is true
  (:if2 cond2:)
     cond1 and cond2 are true
  (:elseif2 cond3:)
     cond1 and cond3 are true, cond2 is not
  (:else2:)
     cond1 is true, cond2 and cond3 are not
  (:if2end:)
(:else:)
  cond1 is false, cond2 testing was ignored
(:ifend:)
```

Spaces were added for better readability.

Using wildcard placeholders

The character `*` can be used as a wildcard to represent any character, zero, one, or multiple times.
The character `?` can be used as a wildcard to represent any character exactly once.
Wildcard characters (`*` and `?`) can be used with the *name* and *group* conditional markups, thus:

`(:if name PmCal.2005* :)`	- current page is in group PmCal and begins with 2005
`(:if group PmWiki* :)`	- current page is in group PmWiki or a group beginning with PmWiki
`(:if name Profiles.*,-Profiles.Profiles :)`	- current page is in group `Profiles` but not `Profiles.Profiles`

Using page text variables, page variables and markup expressions

Page text variables (PTVs), page variables (PVs) and markup expressions can be used in conditional markup. They will be assigned/evaluated before the condition(s).

Combining conditions

Conditions (as previously defined) may be combined into more complex conditional expressions using one of these three equivalent forms:

```
(:if expr EXPRESSION :)
(:if [ EXPRESSION ] :)
(:if ( EXPRESSION ) :)
```

Conditions are combined into expressions with boolean operators and brackets. In the next table, A and B are either regular conditions or (round-)bracketed sub-expressions of regular conditions:

Expression	Operator	Result
`A and B`	And	TRUE if both A and B are TRUE.
`A or B`	Or	TRUE if either A or B is TRUE.
`A xor B`	Xor	TRUE if either A or B is TRUE, but not both.
`! A`	Not	TRUE if A is not TRUE.
`A && B`	And	TRUE if both A and B are TRUE.
`A || B`	Or	TRUE if either A or B is TRUE.

Example

```
(:if [ name SomePage and group SomeGroup ]:)    equivalent to (:if name SomeGroup.SomePage:)
```

Important Notes:

- Spaces are *required* around operators and brackets.
- No specific feedback is given for syntax errors or unbalanced brackets.
- Use round brackets (not square) for nested expressions.

Thus, the following is a valid way of building an expression that shows the following contents only when the user is either the administrator, or is logged in and the time is later than the given date:

```
(:if [ auth admin || ( authid && date 2006-06-01.. ) ] :)
```

Nesting with square brackets will silently fail to work as expected:

```
(:if [ auth admin || [ authid && date 2006-06-01 ] ] :)
```
NOTE: Doesn't Work!

A common use of these complex tests are for expressions like:

```
(:if expr auth admin || auth attr || auth edit :)
[[Logout -> {$Name}?action=logout]]
(:ifend:)
```

which provides a *logout* link only when the browser has admin, attr, or edit permissions.

admins (advanced)

Creating new conditions

See Cookbook:ConditionalMarkupSamples.

See also special references for the use of {*$Variables}.

Contact us

toc top

This website "PmWiki" is powered by the open source PmWiki collaborative content management system.

To contact the owners/editors of the "PmWiki" **website**, look around starting at Main or Path:/. Questions or objections about the website content should be directed to them.

To contact the developers and community publishing the PmWiki**software**, please visit http://www.pmwiki.org/.

Last modified by Petko on June 26, 2015.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/ContactUs

toc top

Contributors

toc top

Here's a list of contributors to PmWiki development and improvement. My apologies if I've forgotten anyone -- feel free to add your name if you've been left out, feel free to remove your name if you don't want to be associated with these people. :-)

- GNUZoo - Several recipes, some security and bug fixes
- Scott Duff - pmwe, simple-journal.php, all-around Pm sanity checker
- Ross Kowalski - uploads and printable page research
- John Rankin - WikiTrails, Links, EditQuickReference, notify.php, documentation, debugging
- Joachim Durchholz - hacking documentation, general pest
- Jessica Tishmack - uploads, testing
- Jean-Claude Gorichon - voting
- Janice Heinold - early PmWiki testing and suggestions, documentation
- James Davis - WikiStyles markup, testing
- Isabelle Michaud - floating images markup, Wiki Groups, uploads/attachments
- Glenn Blalock - WikiStyles suggestions, testing, documentation
- Dawn Green - WikiStyles suggestions, uploads, documentation
- Christian Ridderström - pmwiki-mode for Emacs and some other hacks/modifications.
- Carlo Strozzi - Internationalization, PmWiki on Boa, HTML redirection
- Michael Weiner - Modifications to the ToDo, RssFeedDisplay, MyPmWiki, and CommentBox recipes
- Criss Ittermann (aka Crisses/XES) - ye old best seller Blocklist2 that topped the charts for a while and many other recipes
- Rev. Ian MacGregor - I've contributed with monetary donations, skins, bug reports and continued testing. My personal website is powered by PmWiki.
- Petko Yotov - I have been the PmWiki core developer and pmwiki.org webmaster since January 2009 (after having worked with it since 2004). My contributions are at the Change log page, in the PITS issue tracking system and in the mailing lists. My cookbook recipes can be found at my profile page.

Last modified by Ian MacGregor on October 08, 2013.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Contributors

toc top

Creating New Pages

toc top

The first step to create a new page is to edit an existing page and add a link to the page you want to create.

> To link to your new page, you must choose a name for it. The best names describe the page's contents well, so that everyone can remember and type the name easily.
>
> To create a link, surround the page name with**double brackets**. Typing `[[my new page]]` will create a link to my new page. There's a lot you can do with double bracket links.

You can see that the links to my new page all have question marks after them. That's because my new page hasn't been written yet. Clicking the link as second step will take you to an edit form where you could write and finally save the new page.

Another way to create a page: in your browser's address bar (where the page URL is), replace the name of the current page with the name of the page you wish to create, and hit Enter or do whatever you would normally do to go to a new location. PmWiki will then dutifully tell you that the page you entered doesn't exist, but you can click on the "Edit" link in order to create, edit, and save the new page.

The drawback to this method is that there are no links to your new page, so you're the only person who knows it exists. It will be an orphan, unread, unlinked, unloved. That's why adding a link to an existing page or to the SideBar is a better way to create a page.

Learn more:
- You can also organize related pages into groups, and link between pages in different groups.

How do I create a new page?

> Typing [[my new page]] will create a link to the new page. There's a lot you can do with double bracket links.

Why do some new pages have a title with spaces like "Creating New Pages" and others end up with a WikiWord-like title like "CreatingNewPages"?

> The default page title is simply the name of page, which is normally stored as "CreatingNewPages." However, you can override a page's title by using the `(:title Creating New Pages:)` directive. This is especially useful when there are special characters or capitalization that you want in the title that cannot be used in the page name.

Last modified by Ian MacGregor on September 11, 2012.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/CreatingNewPages

toc top

CustomInterMap

Page redirects to InterMap Summary:Redirects to PmWiki.InterMap.

Last modified by Petko on November 08, 2011.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/CustomInterMap

CustomMarkup

Introduction

PmWiki's markup translation engine is handled by a set of rules; each rule searches for a specific pattern in the markup text and replaces it with some replacement text. Internally, this is accomplished by using PHP's " preg_replace" function.

Rules are added to the translation engine via PmWiki's Markup() or Markup_e() functions, which look like

```
Markup($name, $when, $pattern, $replace); # if no evaluation is needed, or if PHP < 5.5
Markup($name, $when, $pattern, $replace_function); # if evaluation is needed

# DEPRECATED, will not work as of PHP 7.2
Markup_e($name, $when, $pattern, $replace); # if evaluation is needed and 5.5<=PHP<=7.1
```

- `$name` is a unique name (a string) given to the rule
- `$when` says when the rule should be applied relative to other rules
- `$pattern` is the pattern to be searched for in the markup text
- `$replace` is what the pattern should be replaced with.
- `$replace_function` is the name of the function which should be called with the match, and should return the replacement.

For example, here's the code that creates the rule for `''emphasized text''` (in *scripts/stdmarkup.php*):

```
Markup("em", "inline", "/''(.*?)''/", "<em>$1</em>");
```

Basically this statement says to create a rule called "em" to be performed with the other "inline" markups, and the rule replaces any text inside two pairs of single quotes with the same text ($1) surrounded by `<em>` and `</em>`.

Sequence in which rules are applied

The first two parameters to Markup() are used to specify the sequence in which rules should be applied. The first parameter provides a name for a rule -- "`em`" in the example above. We could've chosen other names such as "`''`", or even "`twosinglequotes`". In general PmWiki uses the markup itself to name the rule (i.e., PmWiki uses "`''`" instead of "`em`"), but to keep this example easier to read later on we'll use a mnemonic name for now.

The second parameter says that this rule is to be done along with the other "inline" markups. PmWiki divides the translation process into a number of phases:

```
 _begin      start of translation
   {$var}     Page Text Variables happen here.
 fulltext    translations to be performed on the full text
 split       conversion of the full markup text into lines to be processed
 directives  directive processing
 inline      inline markups
 links       conversion of  links, url-links, and WikiWords
 block       block markups
 style       style handling
 _end        end of translation
```

This argument is normally specified as a left-angle bracket ("before") or a right-angle bracket ("after") followed by the name of another rule.

Thus, specifying "inline" for the second parameter says that this rule should be applied when the other "inline" rules are being performed. If we want a rule to be performed with the directives -- i.e., before inline rules are processed, we would specify "directives" or "<inline" for the second parameter.

{$var} and (:if ...:) conditionals

A significant rule in terms of ordering is "{$var}" which substitutes variables -- if you say "<{$var}" then your markup will be processed before variables are substituted whereas if you say ">{$var}" then your markup will be processed after variables are substituted. This happens before conditional (:if...:) expressions, which is why page text variables are processed even if they are defined inside (:if false:).

Markup regular expression definition

The third parameter is a Perl-compatible regular expression. Basically, it is a slash, a regular expression, another slash, and a set of optional modifiers.

The example uses the pattern string `"/''(.*?)''/"`, which uses `''(.*?)''` as the regular expression and no options. (The regular expression says "find two single quotes in succession, then as few arbitrary characters as are needed to make the match find something, then two additional single quotes in succession"; the parentheses "capture" a part of the wikitext for later use.)

Replacement text

The fourth parameter is the replacement text that should be inserted instead of the marked-up wikitext. You can use`$1`, `$2`, etc. to insert the text from the first, second etc. parenthesised part of the regular expression.

In the example, we have `"<em>$1</em>"`, which is an `<em>`, the text matched by the first parentheses (i.e. by the `.*?` section of the pattern), and `</em>`.

Here's a rule for `@@monospaced@@` text:

```
Markup("@@", "inline", "/@@(.*?)@@/", "<code>$1</code>");
```

and for a `[:comment ...:]` directive that is simply removed from the output:

```
Markup("comment", "directives", "/\\[:comment .*?:\\]/", '');
```

Okay, now how about the rule for `'''strong emphasis'''`? We have to be a bit careful here, because although this translation should be performed along with other inline markup, we also have to make sure that the rule for `'''` is handled *before* the rule for `''`, because `'''` also contains `''`. The second parameter to Markup() can be used to specify the new rule's relationship to any other rule:

```
Markup("strong", "<em", "/'''(.*?)'''/", "<strong>$1</strong>");
```

This creates a rule called "strong", and the second parameter "<em" says to be sure that this rule is processed before the "em" rule we defined above. If we wanted to do something after the "em" rule, we would use ">em" instead. Thus, it's possible to add rules at any point in PmWiki's markup translation process in an extensible manner. (In fact, the "inline", "block", "directives", etc., phases above are just placeholder rules used to provide an overall sequence for other rules. Thus one can use "<inline" to specify rules that should be handled before any other inline rules.)

If you want to disable available markup just call e.g.:

```
DisableMarkup("strong");
```

PmWiki's default markup rules are defined in the*scripts/stdmarkup.php* file. To see the entire translation table as the program is running, the scripts/diag.php module adds "`?action=ruleset`", which displays the set of defined markup rules in the sequence in which they will be processed. You can see it at CustomMarkup?action=ruleset. You must first enable the action by setting `$EnableDiag` = 1 in your configuration file.

Other common examples

Define a custom markup to produce a specific HTML or Javascript sequence

Suppose an admin wants to have a simple "`(:example:)`" markup that will always produce a fixed HTML string in the output, such as for a webring, Google AdSense display, or Javascript. The Markup() call to do this would be:

```
Markup('example', 'directives',
  '/\\(:example:\\)/',
  Keep("<div class='example'><p>Here is a
    <a target='_blank' href='http://www.example.com'>link</a> to
    <em>example.com</em></p></div>") );
```

- The first argument is a unique name for the markup ("example").
- The second argument says to perform this markup along with other directives.
- The third argument is the pattern to look for "(:example:)".
- The fourth argument is the HTML that "(:example:)" is to be replaced with. We use the Keep() function here to prevent the output from being further processed by PmWiki's markup rule -- in the above example, we don't want the http://www.example.com url to be again converted to a link.

Define a markup to call a custom function that returns content

The /e modifier has been deprecated and should not be used in ongoing development. See below for more details.

For older PHP versions (< 7.2) an 'e' option on the `$pattern` parameter causes the `$replace` parameter to be treated as a PHP expression to be evaluated instead of replacement text. To avoid using the deprecated e/ parameter, a markup to produce a random number between 1 and 100 might now look like:

```
Markup('random', 'directives',
  '/\\(:random:\\)/',
  "MyRandomFunction");
function MyRandomFunction() {
  return rand(1, 100);
}
```

This calls the PHP built-in rand() function and substitutes the directive with the result. Any function can be called, including functions defined in a local customization file or in cookbook recipes.

Arguments can also be passed by using regular expression capturing parentheses, thus

```
Markup('randomargs', 'directives',
  '/\\(:random (\\d+) (\\d+):\\)/',
  "MyRandomFunction");
function MyRandomFunction($m) {
  return rand($m[1], $m[2]);
}
```

will cause the markup `(:random 50 100:)` to generate a random number between 50 and 100.

> Note: the /e modifier in regular expressions is deprecated since PHP version 5.5, and removed since PHP version 7. The reason for this is, that malicious authors could pass strings that could cause arbitrary and undesirable PHP functions to be executed.

For a PmWiki function to help with parsing arbitrary sequences of arguments and key=value pairs, seeCookbook:ParseArgs.

Migration to PHP 5.5 and `Markup_e()`

Since PHP version 5.5, the `/e` evaluation modifier is deprecated and some hosting providers don't allow its use.

Recent versions of the PmWiki core (2.2.58 and newer) allow new ways to define markup rules without being dependent on the `/e` eval modifier. The historical ways to define markup rules are not removed and work, but may be incompatible with PHP 5.5 installations.

Note: whether your replacement pattern needs evaluation or not, you must use `Markup()` and not `Markup_e()`. The latter is deprecated and should no longer be used for new recipes and customizations, and old recipes using `Markup_e` should be upgraded to the new format.

The examples below all require PmWiki 2.2.58 (2013-12-25) or newer but the latest version is recommended.

THE SHORT ANSWER: If your markup regular expression (the 3rd argument) contains an "e" after the closing slash (i.e., `/regex/e` or `/regex/se` or etc) AND your 4th argument is entirely surrounded with double-quotes then you may be able to get away with simply following these simple steps:

1. Delete the "e" from after the closing slash in the 3rd argument
2. Create a new replacement function with $m as argument.
3. In your function, the previous occurrences of '$1', '$2', etc. are now found as $m[1], $m[2], etc.
4. In your function, call `extract($GLOBALS['MarkupToHTML']);` in order to get the current `$pagename` and $markupid.
5. Set the name of the replacement function as 4th argument of the Markup() call.

In some cases this will not suffice - it depends on how quoting was done - but in many cases following these simple steps will result in PHP 5.5+ compatibility.

If you try those steps and are still having problems then continue to read below for a deeper understanding.

The following is acceptable for PHP 5.5+ (compatible with PmWiki 2.2.58+, will also work in PHP 5.4 and older)

- `Markup($name, $when, $pattern, $replace);`
  - `$pattern` can no longer have an "`/e`" modifier
  - `$replace` can be a function name (callback) which will be called with the array of matches as argument
  - instead of a string, the fourth parameter can be a definition of an anonymous function (note you can use anon functions this way since PHP 5.3.0+).

- `Markup_e($name, $when, $pattern, $replace);` DEPRECATED, should no longer be used

Examples:

- For PHP 5.4 and older, this was acceptable:
```
Markup('randomargs', 'directives',
  '/\\(:random (\\d+) (\\d+):\\)/e',
  "rand('$1', '$2')"
  );
```

- For PHP 5.5 and newer, $replace is callback, we call Markup():
  ```
  Markup('randomargs', 'directives',
    '/\\(:random (\\d+) (\\d+):\\)/',
    "MyRandom"
    );
  function MyRandom($matches) {
    return rand($matches[1], $matches[2]);
  }
  ```
  This will also work in PHP 5.4 and older

Other example:

- PHP 5.4 or older:
  ```
  Markup('Maxi:','<links',
    "/\\b([Mm]axi:)([^\\s\"\\|\\[\\]]+)(\"([^\"]*)\")?/e",
    "Keep(LinkMaxi(\$pagename,'$1','$2','$4','$1$2'),'L')"
    );
  ```

- PHP 5.5 or newer, PmWiki 2.2.58+, $replace is a function name:
  ```
  Markup('Maxi:','<links',
    "/\\b([Mm]axi:)([^\\s\"\\|\\[\\]]+)(\"([^\"]*)\")?/",
    "LinkMaxi"
    );
  function LinkMaxi($m) {
    extract($GLOBALS['MarkupToHTML']); # to get $pagename
    # do stuff with $m[1], $m[2], etc.
    return Keep($out, 'L');
  }
  ```
  This will also work in PHP 5.4 and older

- $replace can also be a callback function, we call Markup():
  ```
  Markup('Maxi:','<links',
    "/\\b([Mm]axi:)([^\\s\"\\|\\[\\]]+)(\"([^\"]*)\")?/",
    "CallbackMaxi"
  );
  function CallbackMaxi($m) {
    extract($GLOBALS["MarkupToHTML"]); # to get $pagename
    return Keep(LinkMaxi($pagename,$m[1],$m[2],$m[4],$m[1].$m[2]),'L');
  }
  ```
  This will also work in PHP 5.4 and older

The above may seem complicated, but it is actually simpler to use your own callback function:

```
Markup('mykey', 'directives',
  '/\\(:mydirective (.*?) (.*?):\\)/i',
  'MyFunction'
);
function MyFunction($matches) {
  extract($GLOBALS["MarkupToHTML"]);

  # ... do stuff with $matches ...

  return $out; # or return Keep($html);
}
```

If you have any questions about the new way to define custom markup, you can ask us at the talk page or on the mailing lists.

FAQ

How can I embed JavaScript into a page's output?

> There are several ways to do this. The Cookbook:JavaScript recipe describes a simple means for embedding static JavaScript into web pages using custom markup. For editing JavaScript directly in wiki pages (which can pose various security risks), see the JavaScript-Editable recipe. For JavaScript that is to appear in headers or footers of pages, the skin template can be modified directly, or <script> statements can be inserted using the `$HTMLHeaderFmt` array.

How would I create a markup (*(:nodiscussion:)*) that will set a page variable (*{$HideDiscussion}*) which can be used by *(:if enabled HideDiscussion:)* in .PageActions?

> Add the following section of code to your config.php
> ```
> SDV($HideDiscussion, 0);  #define var name
> Markup('hideDiscussion', '<{$var}',
>  '/\\(:nodiscussion:\\)/', 'setHideDiscussion');
> function setHideDiscussion() {
> ```

```
      global $HideDiscussion;
      $HideDiscussion = true;
    }
```

This will enable the `(:if enabled HideDiscussion:)` markup to be used. If you want to print the current value of {$HideDiscussion} (for testing purposes) on the page, you'll also need to add the line:
`$FmtPV['$HideDiscussion'] = '$GLOBALS["HideDiscussion"]';`

It appears that (.*?) does not match newlines in these functions, making the above example inoperable if the text to be wrappen in <em> contains new lines.

If you include the "s" modifier on the regular expression then the dot (.) will match newlines. Thus your regular expression will be "/STUFF(.*?)/s". That s at the very end is what you are looking for. If you start getting into multi-line regexes you may be forced to look at the m option as well - let's anchors (^ and $) match not begin/end of strings but also begin/end of lines (i.e., right before/after a newline). Also make sure your markup is executed during the fulltext phase.

How can the text returned by my markup function be re-processed by the markup engine?

If the result of your markup contains more markup that should be processed, you have two options. First is to select a "when" argument that is processed earlier than the markup in your result. For example, if your markup may return [[links]], your "when" argument could be `"<links"` and your markup will be processed before the links markup. The second option is to call the PRR() function in your markup definition or inside your markup function. In this case, after your markup is processed, PmWiki will restart all markups from the beginning.

How do I get started writing recipes and creating my own custom markup?

(alternate) Introduction to custom markup for Beginners

How do I make a rule that runs once at the end of all other rule processing?

Use this statement instead of the usual `Markup()` call:

```
    $MarkupFrameBase['posteval']['myfooter'] = "\$out = onetimerule(\$out);";
```

CustomWikiStyles

This page describes the predefined Wiki Styles and how a Wiki Administrator can define additional Wiki Styles as a local customization for all pages (in local/config.php) or specific groups (in local/$Group.php).

All predefined Wiki Styles are setup in the global array `$WikiStyle`. To define your own Wiki Styles, add the setting of the correspondent WikiStyle within the array.

Predefined Wiki Styles

The following array-values are set by `scripts/wikistyles.php` using the SDV()-function (so you can overwrite them by setting them prior in config.php or farmconfig.php):

markup:	**definition:**
text colors:	(equivalent to `%define=xxxx color=xxxx%`)
`black`	`$WikiStyle['black']['color'] = 'black';`
`white`	`$WikiStyle['white']['color'] = 'white';`
`red`	`$WikiStyle['red']['color'] = 'red';`
`yellow`	`$WikiStyle['yellow']['color'] = 'yellow';`
`blue`	`$WikiStyle['blue']['color'] = 'blue';`
`gray`	`$WikiStyle['gray']['color'] = 'gray';`
`silver`	`$WikiStyle['silver']['color'] = 'silver';`
`maroon`	`$WikiStyle['maroon']['color'] = 'maroon';`
`green`	`$WikiStyle['green']['color'] = 'green';`
`navy`	`$WikiStyle['navy']['color'] = 'navy';`
`purple`	`$WikiStyle['purple']['color'] = 'purple';`
list-styles:	
`decimal`	`$WikiStyle['decimal']['apply'] = 'list';` `$WikiStyle['decimal']['list-style'] = 'decimal';`
`roman`	`$WikiStyle['roman']['apply'] = 'list';` `$WikiStyle['roman']['list-style'] = 'lower-roman';`
`ROMAN`	`$WikiStyle['ROMAN']['apply'] = 'list';`

	`$WikiStyle['ROMAN']['list-style'] = 'upper-roman';`
`alpha`	`$WikiStyle['alpha']['apply'] = 'list';` `$WikiStyle['alpha']['list-style'] = 'lower-alpha';`
`ALPHA`	`$WikiStyle['ALPHA']['apply'] = 'list';` `$WikiStyle['ALPHA']['list-style'] = 'upper-alpha';`

special:

open links in a new browser-window:

`newwin`	`$WikiStyle['newwin']['target'] = '_blank';`

Turns markup into a comment via display:none CSS

`comment`	`$WikiStyle['comment']['display'] = 'none';`

wiki styles

`frame`	border:1px solid #cccccc; padding:4px; background-color:#f9f9f9;
`lfloat`	float:left; margin-right:0.5em;
`rfloat`	float:right; margin-left:0.5em;
`thumb`	
`lframe`	frame lfloat
`rframe`	frame rfloat
`cframe`	
`pre`	block white-space:pre
`sidehead`	block class:sidehead

Author-Defined Wiki Styles

1. The first index of the array defines the style name (e.g. mynewstyle, projectentry etc)
2. the second index defines the attribute name (e.g. color, background-color, etc.)
3. the value set defines the attribute value (e.g. red, bold, #00ffcc, etc.)

Sample: If you want to define a (site-wide) style the same as the page style

```
    %define=projectentry color:red%
```

use

```
    $WikiStyle['projectentry']['color'] = 'red';
```

The `$WikiStyle['projectentry']['apply']` variable may be defined if the wikistyle concerns a particular tag. It may be `'item' (for li|dt)`, `'list' (for ul|ol|dl)`, `'div'`, `'pre'`, `'img'`, `'p'`or the combining `'block' (for p|div|ul|ol|dl|li|dt|pre|h[1-6])`. Example:

```
 $WikiStyle['top']['apply'] = 'item';
 $WikiStyle['top']['class'] = 'top';
```

then a markup

```
 * %top% An important list-item
```

will output

```
 <li class="top">An important list-item</li>
```

Printer-Friendly Styles

If your custom-styles (in local/config.php) are getting very colorful it might be useful to disable them in print-view. This can be done easily by putting them into a condition.

```
if($action!="print") {
  // your custom-styles
}
```

Notes

To be done:

Questions:

I tried this but background didn't work, thou border and float worked?/Vincent 2008-04-08

```
$WikiStyle['vMenu']['background']='#ffffcc' ;
$WikiStyle['vMenu']['float']='left' ;
$WikiStyle['vMenu']['border']='1px dotted red' ;
```

Try using `$WikiStyle['vMenu']['background-color']='#ffffcc';` -- unlike `background`, `background-color` is defined in the $WikiStyleCSS array, which is checked for valid properties.

Q: How would I set an image to the left of a paragraph in a WikiStyle? I'd like to provide an icon for paragraphs that are notes, important, warnings, etc.

See WikiStylesPlus and Callout.

FAQ

How can I remove underlining from a link, but make it underlined blue when the mouse hovers?

Put in pub/css/local.css:

```
.noul a {text-decoration: none;}
.noul a:hover {text-decoration: underline; color: blue;}
```

Then use this markup:

```
%noul% [[Link]] %%
```

DebugVariables

`$AbortFunction`
: A custom function name replacing the built-in Abort() function.

`$EnableDiag`
: The following actions are available only if you set `$EnableDiag = 1;` in your configuration file. They can be used for debugging and should not be set in a production environment.

?action=**ruleset**
displays a list of all markups in 4 columns:
- column 1 = markup-name (1. parameter of markup())
- column 2 = when will rule apply (2. parameter of markup())
- column 3 = PmWiki's internal sort key (derived from #2)
- column 4 = Debug backtrace information for potentially incompatible rules (filename, line number, pattern)

(see Custom Markup Using the Markup() function for custom wiki syntax; migration to PHP 5.5).
To see more than what `?action=ruleset` gives you, apply the Cookbook:MarkupRulesetDebugging recipe: it can also show the pattern and the replacement strings.
- doesn't make use of PmWiki's authorization mechanisms.

?action=**phpinfo**
displays the output of phpinfo() and exits. No page will be processed
- doesn't make use of PmWiki's authorization mechanisms.

?action=**diag**
displays a dump of all global vars and exits. No page will be processed
- doesn't make use of PmWiki's authorization mechanisms.

`$EnableIMSCaching`
: A variable which, when set equal to 1, recognizes the "If-Modified-Since" header coming from browsers and allows browsers to use locally cached pages. Disabled by default to help the administrator customize its page without needing permanent reloading.

`$EnableStopWatch`
: This activates an internal stopwatch that shows how long it takes to render a page. (If you have a wiki that composes a HTML page from multiple pages, such as a normal layout with a sidebar, you'll get separate timings for each subpage and for the total page.)

The timings can be displayed by adding `<!--function:StopWatchHTML 1-->` in the wiki template.

Valid values are:

```
$EnableStopWatch = 0; # No timings (the default). No HTML will be generated.
$EnableStopWatch = 1; # Wall-clock timings only.
$EnableStopWatch = 2; # Wall-clock and CPU usage timings. Won't work on Windows.
```

See Stopwatch for more details.

See also:
- scripts/refcount.php is useful for debugging

Is it possible for someone with admin priviledges to always have access to debugging tools, without letting everyone else access them?

> You can easily & automatically allow debugging for anyone with admin priviledges (meanwhile leaving it off for everyone else) by including this line in config.php - *just be sure that 1) `$EnableDiag` is either null or set to 0, and 2) to include it near the end of config.php, AFTER declaring your passwords, and after any AuthUser or other priviledge settings*:

```
        if (CondAuth($pagename, 'admin'))  $EnableDiag = 1; # allows admin to always call phpinfo, etc
```

DeletingPages

To delete a wiki page, edit the page, select (highlight) all text in the edit textarea, and replace it with the single word

```
delete
```

It is a good idea to add a comment to the `summary` field explaining why you deleted the page. (The field `summary` is usually found just below the edit textarea).

On saving the change the page is deleted. As an added safety feature, the deleted page still exists on the server (with a timestamp) and can be restored to the former page by the wiki administrator.

If you suspect that a page has been deleted but aren't sure, have a look at the wikigroup's RecentChanges. Erasing a page counts as editing the page, and the activity is recorded there and on Site.AllRecentChanges.

The default word used for page deletion ("delete") can be changed in `config.php` by setting the variable `$DeleteKeyPattern` (see Edit Variables). If there is a danger of malicious page deletion it may be a good idea to change the delete word to something more obscure. There is also a recipe for creating a separate delete action at Cookbook:DeleteAction.

Removing deleted pages

The deleted pages are retained in the same `wiki.d` directory in which they were created. They are renamed with an extension of `,del-123456789` where `123456789` is a unique number (timestamp). A wiki administrator may log into the server via FTP or SSH and periodically remove these files.

One way to remove the files is to delete them. If you have shell access, you could use different commands, for example, go into the wiki.d directory and type one of these lines:

```
rm -f *,del-*
find . -name '*,del-*' -delete
```

Alternatively, the Cookbook:CleanUp recipe can purge those unused files. See also BackupAndRestore.

How is a Wiki Group deleted?

> An admin can remove the group pages from `wiki.d/`. Note that a wiki page may also have related uploads.
>
> Fully deleting a group via the wiki isn't possible, since a delete counts as an "update" which causes the Recent Changes page to be re-created. It is possible to modify the site's configuration to allow deletion of the group's RecentChanges page -- see Cookbook:RecentChangesDeletion.

How is a Category deleted?

> To delete a category, delete all the `[[!Category]]` references from all pages where they occur, then delete the category page as explained above.

maintenance

DesignNotes

Here are some of the features and notes about PmWiki's design decisions. Many of these derive directly from the PmWiki Philosophy and lots of discussion on the mailing lists.

- PmWiki:Flat File Advantages - why PmWiki uses flat files to store pages instead of an SQL database
- PmWiki:Hierarchical Groups - why PmWiki doesn't support nested groups

- PmWiki:Page Locking - how PmWiki works without locking pages (see also simultaneous edits)
- PmWiki:Page File Format - the format of PmWiki's page files
- PmWiki:Search Improvements - why PmWiki has a native search engine
- PmWiki:File Permissions - some information about PmWiki's file permission settings
- PmWiki:Wiki Group Motivation - why WikiGroups
- PmWiki:WYSIWYG - why not WYSIWYG.

Why doesn't PmWiki use hierarchical / nested groups?

It essentially comes down to figuring out how to handle page links between nested groups; if someone can figure out an obvious, intuitive way for authors to do that, then nested groups become plausible. See Design Notes and PmWiki:Hierarchical Groups.

Why don't PmWiki's scripts have a closing ?> tag?

All of PmWiki's scripts now omit the closing ?> tag. The tag is not required, and it avoids problems with unnoticed spaces or blank lines at the end of the file. Also, some file transfer protocols may change the newline character(s) in the file, which can also cause problems. See also the Instruction separation page in the PHP manual.

Does PmWiki support WYSIWYG editing (or something like the FCKEditor)?

Short answer: PmWiki provides GUI buttons in a toolbar for common markups, but otherwise does not have WYSIWYG editing. For the reasons why, see PmWiki:WYSIWYG.

Categories: PmWiki Developer

Last modified by Caroline Guénette on March 05, 2012.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/DesignNotes

Documentation Index

The pages below describe various aspects of using, administering and troubleshooting a PmWiki installation, as well as aspects of the PmWiki community.

As you know, documentation is *always* incomplete. Feel free to help yourself and others by contributing to it. Just edit the pages on pmwiki.org. You might want to follow or contribute to the documentation guidelines.

Table of Contents

Beginner Topics for Creating and Editing Pages

- Basic editing - PmWiki's basic edit syntax
- Creating new pages - How to create a new page
- Links - Multiple mechanisms for creating links
- Images - Placing images in pages
- Text formatting rules - A list of some of the markup sequences available

Intermediate Editing Topics

- Markup master index - Tabulation of all PmWiki markup

- Uploads - Allow authors to upload files, also known as page attachments
- Tables - Simple tables with double pipe markup, one row per line
- Table directives - Directives for table processing
- Wiki styles - Modifying the style of page contents
  - Wiki style examples - Styling text for colour and other attributes

- Access keys - Access keys are keyboard shortcuts for tasks that would otherwise require a mouse
- Page directives - Directives to specify page titles, descriptions, keywords, and display
- Include other pages - Include contents from other PmWiki pages
- InterMap links - Interwiki links definition and use
- Conditional markup - The if directive allows portions of a page to be included or excluded from rendering
- Page variables - variables that are associated with pages
- Page text variables - Page variables automatically made available through natural or explicit page markup
- Markup expressions - String and formatting operations
- Forms - How you can embed input forms into wiki pages

- Simultaneous edits - Handling multiple attempts to edit a page nearly simultaneously

Organizing and Protecting Pages

- Wiki structure - PmWiki structural support for page organization
- Wiki groups - Organising pages into related groups
- Group headers - Group Header and Group Footer page usage
- Wiki trails - Trails from lists items from a single page
- Page history - History of previous edits to a page

- Passwords - General use of passwords and login
- Categories - Categories are a way to organize and find related pages
- Page lists - Listing pages by multiple criteria with templated output
- Attach lists - Get a list of files uploaded and attached to a group using (:attachlist:)(Directives to specify page titles, descriptions, keywords, and display)
- Deleting pages - Removing wiki pages
- Wiki elements -
- Special pages -

PmWiki Site Administration

Installation and maintenance

- Installation - Obtaining and installing PmWiki
- Initial setup tasks - First steps following a fresh installation
- Upgrades - How to upgrade an existing PmWiki installation
- Backup and Restore - background information and some basic backup and restore procedures
- Uploads administration - Administration of PmWiki uploads
- Security - Resources for securing your PmWiki installation

Customisation

- Custom markup - Using the Markup() function for custom wiki syntax; migration to PHP 5.5
- Custom wiki styles - Predefined PmWiki styles & adding custom wiki styles
- Internationalizations - Language internationalisation of web pages
- Local customizations - Customize your PmWiki installation through `config.php` and `local.css`
- Group customizations - How to customize a subset of your wiki
- Skins - Change the look and feel of part or all of PmWiki
- Skin templates - Skin templates (.tmpl files)
- Site Preferences - Customisable browser setting preferences: Access keys, edit form
- Web feeds - Web feed notification of changes
- Wiki Farms - Running multiple wikis from a single installation

Troubleshooting

- Frequently answered questions
- Answers to some other questions
- FAQ Candidate - more answered questions
- Questions
- How to get assistance
- Troubleshooting - Advice for troubleshooting an installation
- Available actions - documentation for developers

Security

- AuthUser - Authorization system that uses usernames and passwords
- Blocklist - Blocking IP addresses, phrases, and expressions to counteract spam and vandalism.
- Notify - How to receive email messages whenever pages are changed on the whole wiki site, individual groups or selected watchlists of pages
- Passwords administration - More password options for the administrator
- Ref count - Link references counts on pages
- Url approvals - Require approval of Url links

Development

- Cookbook:Module Guidelines - Guidelines for creating, distributing, and maintaining a recipe for the Cookbook.
- Variables - Variables available for local customisation
- Functions - How some of the functions in pmwiki.php work
- Page file format - Create wiki formatted pages in bulk and for upload to your pmwiki site

About PmWiki

- Audiences - Patrick Michaud's comments regarding the "audiences" for which PmWiki was designed
- Contributors - A list of contributors to PmWiki development and improvement
- Mailing lists - The email discussion lists available and their archives
- PmWiki philosophy - This page describes some of the ideas that guide the design and implementation of PmWiki
- Design notes - Some of the features and notes about PmWiki's design decisions
- Release notes - Notes about new versions, important for upgrades
- Change log - Log of changes made to PmWiki by Release

- References - References to PmWiki media coverage
- Glossary - Terms related to PmWiki

Last modified by Petko on June 09, 2017.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/DocumentationIndex

Drafts

PmWiki has the capability to stage *draft* versions of a page prior to them becoming "official". All of the draft pages end in "-Draft" by default (this can be changed by setting `$DraftSuffix`). Multiple interim edits to a page can be temporarily saved in a "-Draft" copy of a page until the draft is ready to be published to the original.

When the site administrator sets `$EnableDrafts` in a local customization file, the "Save" button on the edit page is split into separate "Publish" and "Save draft" buttons.

The "Save draft" button causes any edits to be saved to a "-Draft" copy of the original page, leaving the original page intact. Subsequent requests to edit the page (either the original or -Draft) bring up the draft copy for further editing.

The "Publish" button saves back to the original non-Draft copy of the page, removing any -Draft page that may have been created.

By default, saving drafts and publishing are available to anyone with 'edit' permissions (see Passwords). However, the site administrator can also set the `$EnablePublishAttr` configuration variable, which provides a separate 'publish' permission that is required to publish to the original page.

When "publishing", how the page's history is handled depends on whether the administrator has set the `$EnableDraftAtomicDiff` variable.

- When set to 1, "publishing" a draft version will clear the "draft" history, leaving a single "diff" between the latest and the previous "published" versions. Note that this will delete the author names, dates and contributions of the intermediate, unpublished versions, so the change "Summary" you enter should summarize the changes.
- Otherwise, all of the "draft" history entries are kept. The final "publish" history entry will show changes since the most recent "draft" version.

Drafts and pagelists (and RSS)

The drafts module also sets pagelists (and thus RSS feeds) to ignore "-Draft" pages by default; one has to do list=all or similar in order to have draft pages included in a pagelist or RSS feed.

How do I moderate all postings?

> Start by enabling drafts to change the "Save" button into separate "Publish" and "Save draft" buttons. Then set `$EnablePublishAttr`. This adds a "publish" authorization level to distinguish editing of page drafts from publishing.

Last modified by ChuckG on May 24, 2017.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Drafts

EditVariables

To set many of the variables below specify them in `config.php`.

`$AutoCreate`
: Used in conjunction with the AutoCreateTargets edit function, this array records any sets of pages which should be created automatically if they don't exist. The syntax is

```
$AutoCreate[REGEXP] = PAGE_PARAMETERS;
```

where `REGEXP` is a regular expression which will identify the pages to be autocreated, and `PAGE_PARAMETERS` is an array of attributes for the page to be created. For example

```
$AutoCreate['/^Category\\./'] = array('ctime' => $Now);
```

will create a blank page with a current creation time for any missing Category page.

`$DefaultPageTextFmt`
: The text that should be displayed when browsing non-existent pages. As default PmWiki uses the contents of Site.PageNotFound:

```
$DefaultPageTextFmt = '(:include $[{$SiteGroup}.PageNotFound]:)';
```

`$DeleteKeyPattern`
: The pattern used to determine if a page should be deleted. The default is to remove pages that contain only the single word "delete" (and optional spaces).

1. Change delete word to "remove":

```
$DeleteKeyPattern = "^\\s*remove\\s*$";
```

2. Delete any page with no visible text, i.e., empty:

```
$DeleteKeyPattern = "^\\s*$";
```

`$DiffKeepDays`

The `$DiffKeepDays` variable sets the minimum length of time that a page's revision history is kept. By default it is set to 3650 days, or a little less than ten years. You can change this value in a customization file to be something smaller, e.g.:

```
$DiffKeepDays = 30; # keep revisions at least 30 days
```

Note that a specific page revision isn't removed from the page until the first edit after the time specified by `$DiffKeepDays` has elapsed. Thus, it's still possible for some pages to have revisions older than `$DiffKeepDays` -- such revisions will be removed the next time those pages are edited.

`$DiffKeepNum`

This variable contains the minimum number of changes to be kept in the page history, even if some of them are older than the limit defined by `$DiffKeepDays`. It prevents lost history of pages that are older, but have few changes.

```
$DiffKeepNum = 50; # keep at least 50 revisions (default is 20)
```

`$DraftActionsPattern`

The actions which allow full loading of the draft.php functionnality for custom actions. Default is 'edit'. You can enable drafts for other actions like:

```
$DraftActionsPattern = 'edit|pmform|translate';
    # Enable drafts for actions 'edit', 'pmform' and 'translate'.
```

`$DraftSuffix`

The suffix to use for draft versions of pages (default "-Draft").

`$EditFunctions`

This array contains the sequence of functions that are called when a page is edited. It can be customized to provide additional functions to be called as part of the editing process. The standard setting is:

```
$EditFunctions = array('EditTemplate', 'RestorePage', 'ReplaceOnSave',
  'SaveAttributes', 'PostPage', 'PostRecentChanges', 'AutoCreateTargets', 'PreviewPage');
```

Many recipes manipulate this array, so it is recommended, instead of redefining the complete array to add your custom functions, to use functions like array_unshift(), array_push() and array_splice().

`$EditRedirectFmt`

The page to which an author is sent after pressing "Save" or "Cancel" from an edit form. Defaults to "$FullName", which sends the author to the page just edited, but can be changed to specify another page.

1. Redirect to Main.HomePage:
   ```
   $EditRedirectFmt = 'Main.HomePage';
   ```
2. Redirect to HomePage of current group:
   ```
   $EditRedirectFmt = '{$Group}.HomePage';
   ```

`$EditTemplatesFmt`

Name of the page (or an array of names) to be used as the default text for any newly created pages.

1. Use 'Main.NewPageTemplate' as default text of all new pages:
   ```
   $EditTemplatesFmt = 'Main.NewPageTemplate';
   ```
2. Use 'Template' in the current group for new pages:
   ```
   $EditTemplatesFmt = '$Group.Template';
   ```
3. Use 'Template' in the current group if it exists, otherwise use 'Main.NewPageTemplate':
   ```
   $EditTemplatesFmt = array('$Group.Template', 'Main.NewPageTemplate');
   ```

See Cookbook:EditTemplates for more information.

`$EnableDrafts`

When set to '1', enables the "Save draft" button and built-in handling of "draft" versions of pages, where:

1. Initial "Save draft" of an existing page ("PageName") saves changes to a new name ("PageName-Draft").
2. Subsequent attempts to edit PageName causes PageName-Draft to be edited.
3. Subsequent selections of "Save draft" cause PageName-Draft to be saved.
4. Pressing "Publish" causes PageName-Draft to be posted to PageName, and deleted.

Turn on draft edits:

```
$EnableDrafts = 1;
```

A related variable, `$EnablePublishAttr`, adds a new "publish" authorization level to distinguish editing of drafts from publishing them.

`$EnableDraftAtomicDiff`

When set to 1, "publishing" a draft version will clear the "draft" history, leaving a single "diff" between the latest and the previous "published" versions. Note that this will delete the author names, dates and contributions of the intermediate, unpublished versions. (Drafts need to be enabled, see `$EnableDrafts`.)

`$EnableGUIButtons`

When set to '1', turns on the graphical buttons in the "Edit Page" form.

1. Turn on graphical edit buttons:
   ```
   $EnableGUIButtons = 1;
   ```

`$EnablePostAuthorRequired`
: When set to '1', posting of pages requires the author to provide an author name. Otherwise, authors can post without a name.
    1. Require authors to provide a name:
    ```
    $EnablePostAuthorRequired = 1;
    ```

`$EnableRevUserAgent`
: When set to '1', the page history will store the "User agent" string from the browser of the writer (by default this feature is disabled). This can be useful for tracking bugs in custom applications, by examining the disk files in wiki.d.
    1. Store browser user agent with page diffs:
    ```
    $EnableRevUserAgent = 1;
    ```

`$GUIButtons`
: Allows the configuration of the buttons which appear at the top of the text area when editing a page. See scripts/guiedit.php for the default definition. Note that the 5th element can be HTML code rather than just the url of a gif - this allows more flexibility in defining the related javascript.

`$HandleEditFmt`
: Like $HandleBrowseFmt, this specifies the entire output format for `?action=edit` for a page.

`$IsPagePosted`
: Set to a true value if the page is actually saved (e.g., this is used to tell the RecentChanges handlers if they need to update).

`$PageEditFmt`
: By default, this is the HTML to be displayed for an edit form.

`$PageEditForm`
: Specifies the edit form for `?action=edit`. Defaults to ' `$SiteGroup`.EditForm'.

`$ROEPatterns`
: With this array you can add a pattern as *key* and set a text *value* which replace it on every edit request, using preg_replace function. Specifically it is replaced when the page is loaded into the editform, whenever a preview is done, and when the page is saved (from PmWiki 2.2.0beta45). See Cookbook:ROEPatterns for examples.

`$ROSPatterns`
: With this array you can add patterns as *key* and set a text *value* which will replace it when the edited page is posted (as signaled by $EnablePost). It is not replaced when the page is loaded into the editform nor when a preview is done, but it is replaced only when the page is saved. See Cookbook:ROSPatterns for examples.

`$EnableROSEscape`
: If set to 1, the `$ROEPatterns` and `$ROSPatterns` replacements will skip escaped text (surrounded by `[=...=]` or `[@...@]`). If not set, or if set to 0, the replacements will happen even inside escaped text.

Categories: PmWiki Developer

FAQ

This page will attempt to summarize some of the more commonly asked questions. The answers are on the corresponding pages (see link). If you have a question which isn't answered here, you can leave your question on the Questions page or search for documentation using the search facility. More documentation can be found on the documentation index page.

Introduction

What is PmWiki?

PmWiki is a wiki-based system for collaborative creation and maintenance of websites. See PmWiki.

What can I do with it?

PmWiki pages look and act like normal web pages, except they have an "Edit" link that makes it easy to modify existing pages and add new pages into the website, using basic editing rules. You do not need to know or use any HTML or CSS. Page editing can be left open to the public or restricted to small groups of authors. Feel free to experiment with the Text Formatting Rules in the "Wiki sandbox". The website you're currently viewing is built and maintained with PmWiki.

What are the requirements?

See the PmWiki requirements page.

Where can I find documentation?

See the documentation index page.

How can I download PmWiki?

See the download page.

How do I install PmWiki?

Instructions for installation are on the installation page.

How do I get help with PmWiki?

See Mailing lists and How to get assistance.

How do you pronounce "Michaud"?

"Michaud" is french pronounced "mee show", the trailing D is silent.

Basic PmWiki editing rules

I'm new to PmWiki, where can I find some basic help for getting started?

The Basic Editing page is a good start. From there, you can just follow the navigational links at the top or the bottom of the page (they are called Wiki Trails) to the next pages, or to the Documentation Index page, which provides an outline style index of essential documentation pages, organized from basic to advanced.

How do I include special characters such as Copyright (©) and Trademark (® or ™) on my wiki pages?

See special characters on how to insert special characters that don't appear on your keyboard.

How can I preserve line-breaks from the source text?

PmWiki normally treats consecutive lines of text as being a paragraph, and merges and wraps lines together on output. This is consistent with most other wiki packages. An author can use the `(:linebreaks:)` directive to cause the following lines of markup text in the page to be kept as separate lines in the output. Or a wiki administrator can set in *config.php* `$HTMLPNewline = '<br/>';` to force literal new lines for the whole site.

Can I just enter HTML directly?

By default (and by design), PmWiki does not support the use of HTML elements in the editable markup for wiki pages. There are a number of reasons for this described in the PmWiki Philosophy and Audiences. Enabling HTML markup within wiki pages in a collaborative environment may exclude some potential authors from being able to edit pages, and pose a number of display and security issues. However, a site administrator can use the Cookbook:Enable HTML recipe to enable the use of HTML markup directly in pages.

Where can I find more documentation?

See the documentation index and the markup master index pages.

Creating New Pages

How do I create a new page?

Typing [[my new page]] will create a link to the new page. There's a lot you can do with double bracket links.

Why do some new pages have a title with spaces like "Creating New Pages" and others end up with a WikiWord-like title like "CreatingNewPages"?

The default page title is simply the name of page, which is normally stored as "CreatingNewPages." However, you can override a page's title by using the `(:title Creating New Pages:)` directive. This is especially useful when there are special characters or capitalization that you want in the title that cannot be used in the page name.

Links

How do I create a link that will open as a new window?

Use the `%newwin%` wikistyle, as in:

`%newwin% http://example.com/ %%`	http://example.com/

How do I create a link that will open a new window, and configure that new window?

This requires javascript. See Cookbook:PopupWindow.

How do I place a mailing address in a page?

Use the `mailto:` markup, as in one of the following:

`* mailto:myaddress@example.com`
`* [[mailto:myaddress@example.com]]`
`* [[mailto:myaddress@example.com | email me]]`
`* [[mailto:myaddress@example.com?subject=Some subject | email me]]`
	• myaddress@example.com
	• mailto:myaddress@example.com
	• email me
	• email me

The markup `[[mailto:me@example.com?cc=someoneelse@example.com&bcc=else@example.com&subject=Pre-set Subject&body=Pre-set body | display text]] =]` lets you specify more parameters like the message body and more recipients (may not work in all browsers and e-mail clients).

See also Cookbook:DeObMail for information on protecting email addresses from spammers.

How can I enable links to other protocols, such as nntp:, ssh:, xmpp:, etc?

See Cookbook:Add Url schemes

How do I make a WikiWord link to an external page instead of a WikiPage?

Use link markup. There are two formats:

```
[[http://example.com/ | WikiWord]]
[[WikiWord -> http://example.com/]]
```

How do I find all of the pages that link to another page (i.e., backlinks)?

In the wiki search form, use `link=Group.Page` to find all pages linking to Group.Page.

Use the `link=` option of the `(:pagelist:)` directive, as in

```
(:pagelist link=SomePage list=all:)   -- show all links to SomePage
(:pagelist link={$FullName} list=all:)  -- show all links to the current page
```

Note that (with a few exceptions) includes, conditionals, pagelists, searchresults, wikitrails, and redirects are not evaluated for Wikilinks, and so any links they put on the page will not be found as backlinks. All other directives and markup, for example links brought to the page by (:pmform:), will be found.

What link schemes does PmWiki support?

See PmWiki:Link schemes

How do I open external links in a new window or mark them with an icon?

See Cookbook:External links

How can I use an image as a link?

Use [[Page| Attach:image.jpg]] or [[http://site | http://site/image.jpg]] See Images#links

Why my browser does not follow local file:// links?

For security reasons, most browsers will only enable file:// links if the page containing the link is itself on the local drive. In other words, most browsers do not allow links to file:// from pages that were fetched using http:// such as in a PmWiki site. See also Cookbook:DirList for a workaround.

Images

Is it possible to link an image on PmWiki without using a fully qualified URL?

Yes. For images that are attachments, the general format is `Attach:Groupname./image.gif`. To link to an image that is on the same server, use `Path:/path/to/image.gif`.

Can I attach a client image file on PmWiki?

Yes, see Uploads .

How can I include a page from another group that contains an attached image?

Include the page in the normal way, ie `(:include GroupName.Pagename:)`. In the page to be included (that contains the image) change `Attach:filename.ext` to `Attach:{$Group}./filename.ext`.

Why, if I put an image with rframe or rfloat and immediatly after that I open a new page section with ! the section title row is below the image instead of on the left side?

Because the CSS for **headings** such as ! contains an element **clear:both** which forces this behaviour. Redefine the CSS locally if you want to stop this happening, but I think the bottom border (that underlines the heading) would need further re-definition. I just use bolding for the title, and 4 dashes below ---- to separate a new section, and it saves the effort of fiddling with the core definitions.

Unlike the **lframe** and **rframe** directives, **cframe** does not fully honour the width setting. While the frame itself resizes to match the request, the enclosed image does not, and retains its original width. Effect is the same in IE and Fx. I've added an example beneath the standard example above.

Is it possible to disallow all images? I already disabled uploads but I also want to disallow external images from being shown on my wiki pages.

Yes, add to config.php:

```
DisableMarkup('img');
$ImgExtPattern = "$^";
```

How can I make it so that when I place an image in a page, the block of text it is in is a <p> (paragraph) rather than a <div> (division)?

If you just want it to happen for a single image (instead of all), then try putting `[==]` at the beginning of the line, as in:

```
[==] http://www.pmwiki.org/pub/pmwiki/pmwiki-32.gif
```

Having `[==]` at the beginning of a line forces whatever follows to be part of a paragraph.

Is there any way to use relative paths for images?

See Cookbook:RelativeLinks and `$EnableLinkPageRelative`.

Is there a way to attach a BMP and have it display rather than link?

Add to config.php the following line:
```
$ImgExtPattern = "\\.(?:gif|jpg|jpeg|png|bmp|GIF|JPG|JPEG|PNG|BMP)";
```
Note that BMP images are uncompressed and quite heavy. You may wish to convert them to PNG (lossless) or JPG (lossy) format, and thus reduce 5-20 times their filesizes.

Is there a way to have a table to the left or right of an image?

Yes, see TableAndImage.

Uploads

When I upload a file, how do I make the link look like "file.doc" instead of " Attach:file.doc Δ"?

Use parentheses, as in `[[(Attach:)file.doc]]`. There is also a configuration change that can eliminate the `Attach:` -- see Cookbook:AttachLinks.

Why can't I upload files of size more than 50kB to my newly installed PmWiki?

Out of the box PmWiki limits the size of files to be uploaded to 50kB. Add
```
$UploadMaxSize = 1000000; # limit upload file size to 1 megabyte
```
to your *config.php* to increase limit to 1MB (for example). See UploadsAdmin for how to further customize limits. Note that both PHP and webservers also place their own limits on the size of uploaded files.

Why does my upload exit unexpectedly with "Incomplete file received"?

You may be running out of space in a 'scratch' area, used either by PmWiki or by PHP. On *nix, check that you have sufficient free space in /tmp and /var/tmp.

How do I make it so that the upload link still allows one to make another upload (if someone wants to replace the old version of a file with a newer version, for example). Currently you only get the upload link when there is no file in the upload directory.

Use the Attach page action, and click on the delta symbol (Δ) shown against each of files listed. If you can't see the attach action either uploads are not enabled, you are not authorized to upload, or the attach action has been commented out or is missing. See also available actions.

How do I hide the "Attach:" for all attachments

See Cookbook:AttachLinks, note that this does not currently work for `[[Attach:my file.ext]]`.

How can I link a file that have a 4-letter file extension such like 'abc.pptx'?

See Cookbook:Upload Types

How can I prevent others from using the url's of my images on their site

See Cookbook:Prevent Hotlinking

How can I display a file that lacks a correct extension? (e.g. you are using Cookbook:LinkIcons)

A file can be displayed by addition of a "false" extension to the URL. For example, if the url is `http://example.com/dox/mydoc`, add a fake query string on the end with the desired extension (e.g., `http://example.com/dox/mydoc?format=.docx`). If query strings are unsuitable, a fragment identifier should work, e.g. `http://example.com/dox/mydoc#.docx`.

Tables

How do I create a basic table?

Tables are created via use of the double pipe character: `||`. Lines beginning with this markup denote rows in a table; within such lines the double-pipe is used to delimit cells. In the examples below a border is added for illustration (the default is no border).

Basic table

```
|| border=1 rules=rows frame=hsides
|| cell 1 || cell 2 || cell 3 ||
|| cell 1 || cell 2 || cell 3 ||
```

cell 1	cell 2	cell 3
cell 1	cell 2	cell 3

How do I create cell headers?

Header cells can be created by placing ! as the first character of a cell. Note that these are *table headers*, not *headings*, so it doesn't extend to !!, !!!, etc.

Table headers

```
|| border=1 rules=cols frame=vsides
||! cell 1 ||! cell 2 ||! cell 3 ||
|| cell 1  ||  cell 2 ||  cell 3 ||
```

cell 1	**cell 2**	**cell 3**
cell 1	cell 2	cell 3

How do I obtain a table with thin lines and more distance to the content?

"Thin lines" is tricky and browser dependent, but the following works for Firefox and IE (Nov. 2009):

Thin lines and cell padding

```
||border="1" style="border-
collapse:collapse" cellpadding="5"
width=66%
||!Header ||! Header || '''Header'''||
||cells   ||  with   ||      padding||
||        ||         ||             ||
```

Header	Header	Header
cells	with	padding

How do I create an advanced table?

See table directives

My tables are by default centered. When I try to use '||align=left' they don't align left as expected.

Use ||style="margin-left:0px;" instead.

How can I specify the width of columns?

You can define the widths via custom styles, see Cookbook:FormattingTables and `$TableCellAttrFmt`. Add in config.php : `$TableCellAttrFmt = 'class=col$TableCellCount';`
And add in pub/css/local.css :

```
table.column td.col1 { width: 120px; }
table.column td.col3 { width: 40px; }
```

How can I display a double pipe "||" in cell text using basic table markup?

Escape it with `[=||=]` to display || unchanged.

How do I apply styles to the elements of the table, like an ID to the table row, or a class/style to the TD?

See $WikiStyleApply.

Table directives

Can I define table headers using the table directive markup?

Yes, use `(:head:)` or `(:headnr:)` with PmWiki version 2.2.11 or newer. See also Cookbook:AdvancedTableDirectives.

Is it possible to do nested tables?

Yes, if you nest simple tables inside advanced tables. See also Cookbook:AdvancedTableDirectives.

Is it possible to add background images to tables and table cells?

Yes, see Cookbook:BackgroundImages.

Is it possible to apply styles to the elements of the table, like an ID to the table row, or a class/style to the TD?

Yes, see $WikiStyleApply.

Is it possible to automatically generate columns or rows in tables, i.e. without having to do a lot of counting?

Yes, this is possible with the Cookbook:CreateColumns recipe - it allows you to specify a certain number of columns, and/or to specify a certain number of items per column. Plus, someone has provided some similar markup on the TableDirectives-Talk page.

Page Directives

Can I get `(:redirect:)` to return a "moved permanently" (HTTP 301) status code?

Use `(:redirect PageName status=301:)`.

Is there any way to prevent the "redirected from" message from showing at the top of the target page when I use `(:redirect:)`?

From version 2.2.1 on, set in config.php `$EnableRedirectQuiet=1;` and in the page `(:redirect OtherPage quiet=1:)` for a quiet redirect.

Is there any method for redirecting to the equivalent page in a different group, i.e. from BadGroup/thispage => GoodGroup/thispage using similar markup to

Page redirects to Goodgroup.{Name} ?
(:redirect Goodgroup.{$Name}:) works if you want to put it in one page.

If you want it to work for the entire group, put (:redirect Goodgroup.{*$Name}:) into Badgroup.GroupHeader - however, that only works with pages that really exist in Goodgroup; if you visit a page in Badgroup without a corresponding page of the same name in Goodgroup, instead of being redirected to a nonexistant page, you get the redirect Directive at the top of the page.

With (:if exists Goodgroup.{*$Name}:)(:redirect Goodgroup.{*$Name}:)(:ifend:) in Badgroup.GroupHeader you get redirected to Goodgroup.Name if it exists, otherwise you get Badgroup.Name without the bit of code displayed.

How can a wiki enable linebreaks by default, i.e. without having the directive `(:linebreaks:)` in a page or in a GroupHeader?

Add to config.php such a line:
`$HTMLPNewline = '<br/>';`

Include Other Pages

What's the maximum number of includes that can exist in a page?

My site seems to stop including after 48 includes. (`$MaxIncludes`)

By default, PmWiki places a limit of 50 include directives for any given page, to prevent runaway infinite loops and other situations that might eat up server resources. (Two of these are GroupHeader and GroupFooter.) The limit can be modified by the wiki administrator via the `$MaxIncludes` variable.

Is there any way to include from a group of pages without specifying by exact name, e.g. between Anchor X and Y from all pages named IFClass-* ?

This can be achieved using page lists.

There appears to be a viewing issue when the included page contains the (:title:) directive.

In a default installation, the *last* title in the page overrides previous ones so you can place your (:title :) directive at the bottom of the page, after any includes. See also `$EnablePageTitlePriority`.

How to test to see if the page is part of another page?

```
(:if ! name {PmWiki.IncludeOtherPages$FullName}:)
%comment% name of this page is not the same as the page this text was sourced from
->[[{PmWiki.IncludeOtherPages$FullName}#anchor | more ...]]
(:ifend:)
```

more ...

Inter Map

Are InterMap names case sensitive?

Yes, thus `eAdmin:` is a different InterMap link than `EAdmin:`.

How can I achieve a *localmap.txt* mapping with the effect of `Pics: Path:/somepathto/pics/`?

Use the following:
`Pics: /somepathto/pics/`

How can I define an InterMap in PHP?

Use the following:

```
        $LinkFunctions['PmWikiHome:'] = 'LinkIMap';
        $IMap['PmWikiHome:'] = 'http://pmwiki.org/wiki/$1';
```

Page specific variables

Is there a variable like $LastModified, but which shows me the creation time?

No, but you can create one in config.php. For instance:

```
    # add page variable {$PageCreationDate} in format yyyy-mm-dd
    $FmtPV['$PageCreationDate'] = 'strftime("%Y-%m-%d", $page["ctime"])';
```

If you like the same format that you define in config.php with `$TimeFmt` use

```
 $FmtPV['$Created'] = "strftime(\$GLOBALS['TimeFmt'], \$page['ctime'])";
```

How can I test if a variable is set and/or not empty?

Use `(:if ! equal "{$Variable}" "":) $Variable is not empty. (:ifend:)`. Note that undefined/inexistent variables appear as empty ones.

Categories: PmWiki Developer

Wiki Group

How can I get rid of the 'Main' group in urls for pages pointing to Main?

See Cookbook:Get Rid Of Main.

How can I limit the creation of new groups?

See Cookbook:Limit Wiki Groups.

Why doesn't [[St. Giles and St. James]] work as a link? (It doesn't display anything.)

Because it contains periods, and destroys PmWiki's file structure, which saves pages as Group.PageName. Adding those periods disrupts this format. Links may only contain words. If you need a link precisely as shown, the page must be named eg StGilesAndStJames then you can use the (:title:) directive to have the page's title appear with periods (:title St. Giles and St. James:). (Although in US grammar the period is often omitted and in UK grammar the period must be omitted for contractions like St).

How can I delete a wiki group?

Normally you can't, as this requires an admin with server-side access to delete the file that makes up the group's RecentChanges page. But there is an option method of making it possible to delete RecentChanges pages from within the wiki if the admin enables the code found on Cookbook:RecentChanges Deletion.

How can I delete a wiki group's Group.RecentChanges page?

Normally you can't, as this requires an admin with server-side access to delete a file. But there is an optional method of making it possible to delete RecentChanges pages from within the wiki if the admin enables the code found on Cookbook:RecentChanges Deletion.

Can I delete a wiki group inside wiki.d folder on the server to eliminate the group?

Yes, if you delete all files named YourGroup.*, the pages from that group will be removed from the wiki. Note that the documentation (group PmWiki) and the site configuration (groups Site and SiteAdmin) that exist in the default installation, are located in wikilib.d and not in wiki.d, and some recipes provide files located in a wikilib.d subdirectory in the cookbook directory. (You shouldn't delete the groups Site and SiteAdmin, required for normal function.)

How can I list all pages in a WikiGroup?

In a wiki page use `(:pagelist group=GroupName list=all:)` or in a search box type`GroupName/ list=all`.

GroupHeaders and GroupFooters

How do I set the same header or footer for all pages/groups?

The header and footer for each page are controlled by the variables `$GroupHeaderFmt` and `$GroupFooterFmt`. If your site-wide header and footer pages are Site.SiteHeader and Site.SiteFooter, you can add this in config.php:

```
### If you use Site.SiteHeader and Group.GroupHeader
$GroupHeaderFmt = '(:include {$SiteGroup}.SiteHeader'
  . ' basepage={*$FullName}:)(:nl:)' . $GroupHeaderFmt;

### If you use Site.SiteHeader instead of Group.GroupHeader
$GroupHeaderFmt = '(:include {$SiteGroup}.SiteHeader'
  . ' basepage={*$FullName}:)(:nl:)';

### If you use Site.SiteFooter and Group.GroupFooter
$GroupFooterFmt .= '(:nl:)(:include {$SiteGroup}.SiteFooter'
  . ' basepage={*$FullName}:)';

### If you use Site.SiteFooter instead of Group.GroupFooter
$GroupFooterFmt = '(:nl:)(:include {$SiteGroup}.SiteFooter'
  . ' basepage={*$FullName}:)';
```

Note that single quotes must be used in the lines above.

See also the Cookbook:AllGroupHeader recipe.

Instead of using an additional page, you could set any wiki text in `$GroupHeaderFmt`, for example:

```
$GroupHeaderFmt .= "Global message here.";
```

Wiki Trails

What's the difference between a PageList and a WikiTrail?

The pagelist directive dynamically generates a list of pages. There are many ways to generate the list, including using a WikiTrail as the source. The pagelist directive then displays the pages that match the criteria using an optional template - for example displaying each page name on a separate line as a link or including the entire content. The pagelist directive currently does not have built-in navigation markup that you can put on the pages in the list. By contrast, WikiTrails are simply specified via links on an "index" page and you *can* put previous-next navigation markup on each page. The two serve very different purposes. WikiTrails are useful for specifying the pages in web feeds, for creating a "tour" through a predefined set of pages, and many other things.

Page History

Is there a way to remove page history from page files?

1. Administrators can clean page histories using the Cookbook:ExpireDiff recipe.

2. Administrators with FTP file access can download individual pages from the wiki.d directory, open them in a text editor, manually remove history, and re-upload the files to wiki.d/ directory. Care must be exercised, when manually editing a page file, to preserve the minimum required elements of the page and avoid corrupting its contents. See PageFileFormat#creating.

3. Edit the page. Select *all* the contents of the edit text area and cut them to the clipboard. Enter`delete` into the text area and click on the *save and edit* button. Select *all* the contents of the edit text area and paste the contents of the clipboard

over them. Click on the *save* button. This will remove all of the page's history up to the final save in which the pasted material is re-added.

How can I restrict viewing the page history `(?action=diff)` to people with edit permission?

In the *local/config.php* file, set

```
$HandleAuth['diff'] = 'edit';
```

In case of this restriction is set up on a farm, and you want to allow it on a particular wiki, set in your local/config.php :

```
$HandleAuth['diff'] = 'read';
```

Passwords

How can I password protect all the pages and groups on my site? Do I really have to set passwords page by page, or group by group?

Administrators can set passwords for the entire site by editing the config.php file; they don't have to set passwords for each page or group. For example, to set the entire site to be editable only by those who know an "edit" password, an administrator can add a line like the following to local/config.php:

```
$DefaultPasswords['edit'] = pmcrypt('edit_password');
```

For more information about the password options that are available only to administrators, see PasswordsAdmin.

I get http error 500 "Internal Server Error" when I try to log in. What's wrong?

This can happen if the encrypted passwords are not created on the web server that hosts the PmWiki.
The PHP crypt() function changed during the PHP development, e.g. a password encrypted with PHP 5.2 can not be decrypted in PHP 5.1, but PHP 5.2 can decrypt passwords created by PHP 5.1.
This situation normally happens if you prepare everything on your local machine with the latest PHP version and you upload the passwords to a webserver which is running an older version.
The same error occurs when you add encrypted passwords to local/config.php.

Solution: Create the passwords on the system with the oldest PHP version and use them on all other systems.

How can I create private groups for users, so that each user can edit pages in their group, but no one else (other than the admin) can?

Modify the edit attribute for each group to id:username, e.g. set the edit attribute in JaneDoe.GroupAttributes to id:JaneDoe.

There is a more automatic solution, but it's probably not a good idea for most wikis. Administrators can use the AuthUser recipe and add the following few lines to their local/config.php file to set this up:

```
    $group = FmtPageName('$Group', $pagename);
    $DefaultPasswords['edit'] = 'id:'.$group;
    include_once("$FarmD/scripts/authuser.php");
```

This automatically gives edit rights to a group to every user who has the same user name as the group name. Unfortunately it also gives edit rights to such a user who is visiting a same-named group not just for pages in that group, but for any page on the wiki that relies on the site's default edit password. This can create security holes.

How come when I switch to another wiki within a farm, I keep my same authorization?

PmWiki uses PHP sessions to keep track of authentication/authorization information, and by default PHP sets things up such that all interactions with the same server are considered part of the same session.

An easy way to fix this is to make sure each wiki is using a different cookie name for its session identifier. Near the top of one of the wiki's local/config.php files, before calling authuser or any other recipes, add a line like:
session_name('XYZSESSID');
You can pick any alphanumeric name for XYZSESSID; for example, for the cs559-1 wiki you might choose session_name('CS559SESSID');
This will keep the two wikis' sessions independent of each other.

Is it possible to test the password level for display and/or if condition? Example: * (:if WriterPassword:) (display Edit link) (:ifend:)

You can use `(:if auth edit:)`. See ConditionalMarkup.

Deleting Pages

How is a Wiki Group deleted?

An admin can remove the group pages from `wiki.d/`. Note that a wiki page may also have related uploads.

Fully deleting a group via the wiki isn't possible, since a delete counts as an "update" which causes the Recent Changes page to be re-created. It is possible to modify the site's configuration to allow deletion of the group's RecentChanges page -- see Cookbook:RecentChangesDeletion.

How is a Category deleted?

To delete a category, delete all the `[[!Category]]` references from all pages where they occur, then delete the category page as explained above.

maintenance

PmWiki Installation

Should I rename pmwiki.php to index.php?

Renaming pmwiki.php is not recommended. Instead, create an *index.php* file that contains this single line

```
<?php include_once('pmwiki.php');
```

How do I make pmwiki.php the default page for a website?

Create an *index.php* file that runs PmWiki from a subdirectory (*pmwiki/* for example) and place it in the site's web document root (the main directory for the website).

```
<?php chdir('pmwiki'); include_once('pmwiki.php');
```

Note: You will also need to explicitly set the `$PubDirUrl` variable (e.g. to `"http://example.com/pmwiki/pub"`) in *local/config.php* .

How do I enable "Clean URLs" that are shorter and look like paths to my wiki pages? Why does pmwiki.org appear to have a directory structure rather than "?n=pagename" in URLs?

See Cookbook:CleanUrls.

How can I run PmWiki on a standalone (offline, portable) machine ?

See Cookbook:Standalone or Cookbook:WikiOnAStick.

Upgrades

FAQ

How can I determine what version of PmWiki I'm running now?

See version - Determining and displaying the current version of PmWiki (pmwiki-2.2.99).

How can I test a new version of PmWiki on my wiki without changing the prior version used by visitors?

The easy way to do this is to install the new version in a separate directory, and for the new version set (in local/config.php):

```
    $WikiLibDirs = array(&$WikiDir,
      new PageStore('/path/to/existing/wiki.d/{$FullName}'),
      new PageStore('wikilib.d/{$FullName}'));
```

This lets you test the new version using existing page content without impacting the existing site or risking modification of the pages. (Of course, any recipes or local customizations have to be installed in the new version as well.)

Then, once you're comfortable that the new version seems to work as well as the old, it's safe to upgrade the old version (and one knows of any configuration or page changes that need to be made).

Uploads Administration

How do I disable uploading of a certain type of file?

Here's an example of what to add to your *local/config.php* file to disable uploading of .zip files, or of files with no extension:

```
     $UploadExtSize['zip'] = 0;  # Disallow uploading .zip files
     $UploadExtSize[''] = 0;     # Disallow files with no extension
```

How do I attach uploads to individual pages or the entire site, instead of organizing them by wiki group?

Use the `$UploadPrefixFmt` variable (see also the Cookbook:UploadGroups recipe).

```
$UploadPrefixFmt = '/$FullName'; # per-page, in Group.Name directories
$UploadPrefixFmt = '/$Group/$Name'; # per-page, in Group directories with Name subdirectories
$UploadPrefixFmt = ''; # site-wide
```

For `$UploadDirQuota` - can you provide some units and numbers? Is the specification in bytes or bits? What is the number for 100K? 1 Meg? 1 Gig? 1 Terabyte?

Units are in bytes.

```
$UploadDirQuota = 100*1024;          # limit uploads to 100KiB
$UploadDirQuota = 1000*1024;         # limit uploads to 1000KiB
$UploadDirQuota = 1024*1024;         # limit uploads to 1MiB
$UploadDirQuota = 25*1024*1024;      # limit uploads to 25MiB
$UploadDirQuota = 2*1024*1024*1024; # limit uploads to 2GiB
```

Is there a way to allow file names with Unicode or additional characters?

Yes, see `$UploadNameChars`

Where is the list of attachments stored?

It is generated on the fly by the
markup.

Security

How do I report a possible security vulnerability of PmWiki?

Pm wrote about this in a post to pmwiki-users from September 2006. In a nutshell he differentiates two cases:

1. The possible vulnerability isn't already known publicly: In this case please contact us by private mail.
2. The possible vulnerability is already known publicly: In this case feel free to discuss the vulnerability in public (e.g. on pmwiki-users or in the PITS).

See his post mentioned above for details and rationals.

What about the botnet security advisory at http://isc.sans.org/diary.php?storyid=1672?

Sites that are running with PHP's *register_globals* setting set to "On" and versions of PmWiki prior to 2.1.21 may be vulnerable to a botnet exploit that is taking advantage of a bug in PHP. The vulnerability can be closed by turning *register_globals* off, upgrading to PmWiki 2.1.21 or later, or upgrading to PHP versions 4.4.3 or 5.1.4.
In addition, there is a test at PmWiki:SiteAnalyzer that can be used to determine if your site is vulnerable.

Wiki Vandalism and Spam

Assumptions

you are using a Blocklist and Url approvals.
You don't want to resort to password protecting the entire wiki, that's not the point after all.
Ideally these protections will be invoked in `config.php`

How do I stop pages being deleted, eg password protect a page from deletion?

Use Cookbook:DeleteAction and password protect the page deletion action by adding `$DefaultPasswords['delete'] = '*';` to `config.php` or password protect the action with `$HandleAuth['delete'] = 'edit';`
or `$HandleAuth['delete'] = 'admin';` to require the edit or admin password respectively.

How do I stop pages being replaced with an empty (all spaces) page?

Add `block: /^\s*$/` to your blocklist.

how do I stop pages being completely replaced by an inane comment such as *excellent site*, *great information*, where the content cannot be blocked?

Try using the newer automatic blocklists that pull information and IP addresses about known wiki defacers.

(OR) Try using Cookbook:Captchas or Cookbook:Captcha (note these are different).

(OR) Set an edit password, but make it publicly available on the Site.AuthForm template.

How do I password protect the creation of new groups?

See Cookbook:Limit Wiki Groups

How do I password protect the creation of new pages?

See Cookbook:Limit new pages in Wiki Groups

How do I take a whitelist approach where users from known or trusted IP addresses can edit, and others require a password?

Put these lines to local/config.php:

```
## Allow passwordless editing from own turf, pass for others.
if ($action=='edit'
 && !preg_match("/^90\\.68\\./", $_SERVER['REMOTE_ADDR']) )
 { $DefaultPasswords['edit'] = pmcrypt('foobar'); }
```

Replace 90.68. with the preferred network prefix and foobar with the default password for others.

For a single IP, you may use

```
if($_SERVER['REMOTE_ADDR'] == '127.0.0.1') { # your IP address here
 $_POST['authpw'] = 'xxx';                   # the admin password
}
```

Please note the security issues : this means that you have your admin passwords in clear in config.php and someone with access to the filesystem can read them (for example a technician of your hosting provider) ; your IP address may change from time to time (unless you have a fixed IP contract with your ISP). When that happens, someone with your old IP address will be logged in automatically as admin on your wiki. It is extremely unlikely to become a problem, but you should know it is possible ; if you are behind a router, all other devices which pass through that router will have the same IP address for PmWiki - your wifi phone, your wife's netbook, a neighbour using your wifi connection, etc. All these people become admins of your wiki. Again, you should evaluate if this is a security risk ; In some cases, your ISP will route your traffic through the same proxy as other people. In such a case, thousands of people may have the same IP address.

See also Cookbook:AuthDNS & Cookbook:PersistentLogin

How do I password protect page actions?

See Passwords for setting in config.php
```
$HandleAuth['pageactionname'] = 'pageactionname'; # along with :
$DefaultPasswords['pageactionname'] = pmcrypt('secret phrase');
```
or
```
$HandleAuth['pageactionname'] = 'anotherpageactionname';
```

How do I moderate all postings?

Enable PmWiki.Drafts

- Set `$EnableDrafts`, this relabels the "Save" button to "Publish" and a "Save draft" button appears.
- Set `$EnablePublishAttr`, this adds a new "publish" authorization level to distinguish editing from publishing.

How do I make a read only wiki?

In config.php set an "edit" password.

How do I restrict access to uploaded attachments?

See

- instructions for denying public access to the uploads directory
- see Cookbook:Secure attachments

How do I hide the IP addresses in the "diff" pages?

If the user fills an author name, the IP address is not displayed. To require an author name, set in config.php such a line:

```
  $EnablePostAuthorRequired = 1;
```

The IP address can also be seen in a tooltip title when the mouse cursor is over the author name. To disable the tooltip, set in config.php:
```
$DiffStartFmt =
  "<div class='diffbox'><div class='difftime'><a name='diff\$DiffGMT'
href='#diff\$DiffGMT'>\$DiffTime</a>
   \$[by] <span class='diffauthor'>\$DiffAuthor</span> - \$DiffChangeSum</div>";
```

How do I stop some Apache installations executing a file which has ".php", ".pl" or ".cgi" anywhere in the filename

Use `$UploadBlacklist`

How do I stop random people from viewing the ?action=source (wiki markup) of my pages? I have `(:if auth edit:)` text that I don't want the world to see.

`$HandleAuth['source'] = 'edit';` or `$HandleAuth['source'] = 'admin';`

Custom Markup

How can I embed JavaScript into a page's output?

There are several ways to do this. The Cookbook:JavaScript recipe describes a simple means for embedding static JavaScript into web pages using custom markup. For editing JavaScript directly in wiki pages (which can pose various security risks), see the JavaScript-Editable recipe. For JavaScript that is to appear in headers or footers of pages, the skin template can be modified directly, or <script> statements can be inserted using the `$HTMLHeaderFmt` array.

How would I create a markup (*(:nodiscussion:)*) that will set a page variable (*{$HideDiscussion}*) which can be used by *(:if enabled HideDiscussion:)* in .PageActions?

Add the following section of code to your config.php
```
     SDV($HideDiscussion, 0);  #define var name
     Markup('hideDiscussion', '<{$var}',
      '/\\(:nodiscussion:\\)/', 'setHideDiscussion');
     function setHideDiscussion() {
       global $HideDiscussion;
       $HideDiscussion = true;
     }
```

This will enable the `(:if enabled HideDiscussion:)` markup to be used. If you want to print the current value of {$HideDiscussion} (for testing purposes) on the page, you'll also need to add the line:
```
$FmtPV['$HideDiscussion'] = '$GLOBALS["HideDiscussion"]';
```

It appears that (.*?) does not match newlines in these functions, making the above example inoperable if the text to be wrappen in <em> contains new lines.

If you include the "s" modifier on the regular expression then the dot (.) will match newlines. Thus your regular expression will be "/STUFF(.*?)/s". That s at the very end is what you are looking for. If you start getting into multi-line regexes you may be forced to look at the m option as well - let's anchors (^ and $) match not begin/end of strings but also begin/end of lines (i.e., right before/after a newline). Also make sure your markup is executed during the fulltext phase.

How can the text returned by my markup function be re-processed by the markup engine?

If the result of your markup contains more markup that should be processed, you have two options. First is to select a "when" argument that is processed earlier than the markup in your result. For example, if your markup may return [[links]], your "when" argument could be `"<links"` and your markup will be processed before the links markup. The second option is to call the PRR() function in your markup definition or inside your markup function. In this case, after your markup is processed, PmWiki will restart all markups from the beginning.

How do I get started writing recipes and creating my own custom markup?

(alternate) Introduction to custom markup for Beginners

How do I make a rule that runs once at the end of all other rule processing?

Use this statement instead of the usual `Markup()` call:

```
$MarkupFrameBase['posteval']['myfooter'] = "\$out = onetimerule(\$out);";
```

Internationalizations

If my wiki is internationalized by *config.php*, how do I revert a specific group to English?

Use `$XLLangs = array('en');` in the group's group customization file.

If my wiki is in English and I want just one page, or group, in Spanish do I say `XLPage('es','PmWikiEs.XLPage');` in the group or page configuration file?

Yes, that is usually the best method. If you were doing this with many scattered pages, or with several languages, you might find it easier to maintain if you load the translations all in config.php like this:

```
XLPage('es','PmWikiEs.XLPage');
XLPage('fr','PmWikiFr.XLPage');
XLPage('ru','PmWikiRu.XLPage');
 $XLLangs = array('en');
```

Then in each group or page configuration file, you'd just use `$XLLangs` = array('es'); to set the language to use (in this case, Spanish). Note that though this method is easier to maintain, its somewhat slower because it loads all the dictionaries for each page view, even if they won't be used.

What does the first parameter of this function stand for? How can it be used?

The XLPage mechanism allows multiple sets of translations to be loaded, and the first parameter is used to distinguish them.

For example, suppose I want to have translations for both normal French and "Canadian" French. Rather than maintain two entirely separate sets of pages, I could do:

```
XLPage('fr-ca', 'PmWikiFrCa.XLPage');
XLPage('fr', 'PmWikiFr.XLPage');
```

PmWikiFr.XLPage would contain all of the standard French translations, while PmWikiFrCA.XLPage would only need to contain "Canada-specific" translations -- i.e., those that are different from the ones in the French page.

The first parameter distinguishes the two sets of translations. In addition, a*config.php* script can use the `$XLLangs` variable to adjust the order of translation, so if there was a group or page where I only wanted the standard French translation, I can set

```
$XLLangs = array('fr', 'en');
```

and PmWiki will use only the 'fr' and 'en' translations (in that order), no matter how many translations have been loaded with XLPage().

How can I add a translation for an individual string in a PHP file?

Use the XLSDV() function to provide a translation for a specific (English) string. For instance, with this in config.php

```
XLSDV('nl', array('my English expression'=>'mijn Nederlandse uitdrukking'));
```

any instance of the variable expression `$[my English expression]` in wiki mark-up will be displayed as *my English expression* in default (English) context, but as *mijn Nederlandse uitdrukking* in Dutch (nl) context, i.e. when `XLPage('nl',...)` has been called for that page in config.php or a cookbook recipe.

If you need to get a translation in a PHP file, use the `XL()` function:

```
  $local_string = XL("my English expression");
```

But beware: XLPage() uses XLSDV() internally for its translation pairs, too, and only the first definition is accepted! Thus, if the Dutch XLPage already contains a translation and you want to override that, you need to use your XLSDV('nl',...) *before* calling the correspondent XLPage('nl',...). Otherwise, by using XLSDV() *after* XLPage() - e.g. within a recipe that is included later in config.php - your translation will only work as long nobody defines 'my English expression' in that XLPage.

Local Customizations

There's no "config.php"; it's not even clear what a "local customisation file" is!

The "sample-config.php" file in the "docs" folder, is given as an example. Copy it to the "local" folder and rename it to "config.php". You can then remove the "#" symbols or add other commands shown in the documentation. See also Group Customizations.

Can I change the default page something other than Main.HomePage (`$DefaultPage`)?

Yes, just set the `$DefaultPage` variable to the name of the page you want to be the default. You might also look at the `$DefaultGroup` and `$DefaultName` configuration variables.

```
$DefaultPage = 'ABC.StartPage';
```

Note the recommendations in `$DefaultName` and the need to set `$PagePathFmt` as well if you are changing the default startup page for groups.

How do I get the group / page name in a local configuration file (e.g. *local/config.php*)?

Use the following markup in pmwiki-2.1.beta21 or newer:

```
    ## Get the group and page name
    $pagename = ResolvePageName($pagename);
    $page = PageVar($pagename, '$FullName');
    $group = PageVar($pagename, '$Group');
    $name = PageVar($pagename, '$Name');
```

Note the importance of the order of customizations in config.php above to avoid caching problems.

If you need the verbatim group and page name (from the request) early in config.php, `$pagename` is guaranteed to be set to

1. Any value of ?n= if it's set, or
2. Any value of ?pagename= if it's set, or
3. The "path info" information from REQUEST_URI (whatever follows SCRIPT_NAME), or
4. Blank otherwise

according to this posting

Can I remove items from the wikilib.d folder on my site?

The files named Site.* and SiteAdmin.* contain parts of the interface and the configuration and they should not be removed. The other files named PmWiki* contain the documentation and could be removed.

How do I customize my own 404 error page for non-existent pages?

To change the text of the message, try editing the Site.PageNotFound page.

Is the order of customizations in config.php important? Are there certain things that should come before or after others in that file?

Yes, see Order of the commands in config.php.

Group Customizations

How can I apply CSS styles to a particular group or page?

Simply create a *pub/css/Group.css* or *pub/css/Group.Page.css* file containing the custom CSS styles for that group or page. See also Cookbook:LocalCSS.

Why shouldn't passwords be set in group (or page) customization files? Why shouldn't group or page passwords be set in config.php?

The reason for this advice is that per-group customization files are only loaded for the current page. So, if `$DefaultPasswords['read']` is set in *local/GroupA.php*, then someone could use a page in another group to view the contents of pages in GroupA. For example, Main.WikiSandbox could contain:

    (:include GroupA.SomePage:)

and because the *GroupA.php* file wasn't loaded (we're looking at Main.WikiSandbox -->*local/Main.php*), there's no read password set.

The same is true for page customization files.

Isn't that processing order strange? Why not load per page configuration last (that is after global configuration an per group configuration)?

Many times what we want to do is to enable a certain capability for a group of pages, but disable it on a specific page, as if it was never enabled. If the per-group config file is processed first, then it becomes very difficult/tedious for the per-page one to "undo" the effects of the per-group page. So, we load the per-page file before the per-group.

If a per-page customization wants the per-group customizations to be performed first, it can use the techniques given above (using *include_once()* or setting `$EnablePGCust = 0;`).

Skins

How do I change the Wiki's default name in the upper left corner of the Main Page?

Put the following config.php

```
$WikiTitle = 'My Wiki Site';
```

The *docs/sample-config.php* file has an example of changing the title.

How can I embed PmWiki pages inside a web page?

Source them through a PHP page, or place them in a frame.

How do I change the font or background color of the hints block on the Edit Page?

Add a CSS style to pub/css/local.css: `.quickref {background:...; color:... }`. The hints are provided by the Site.EditQuickReference page, which is in the PmWiki or Site wikigroup. Edit that page, and change the "bgcolor" or specify the font "color" to get the contrast you need.

Skin Templates

How do I customize the CSS styling of my PmWiki layout?

See Skins for how to change the default PmWiki skin. See also Skins, where you will find pre-made templates you can use to customize the appearance of your site. You can also create a file called *local.css* in the *pub/css/* directory and add CSS selectors there (this file gets automatically loaded if it exists). Or, styles can be added directly into a local customization file by using something like:

```
$HTMLStylesFmt[] = '.foo { color:blue; }';
```

Where can the mentioned "translation table" be found for adding translated phrases?

See Internationalizations.

Is it possible to have the edit form in full page width, with no sidebar?

If the sidebar is marked with `<!--PageLeftFmt-->`, adding `(:noleft:)` to Site.EditForm will hide it when a page is edited.

Can I easily hide the Home Page title from the homepage?

Yes, you can use in the wiki page either `(:title Some other title:)` to change it or `(:notitle:)` to hide it.

Is it possible to hide the Search-Bar in the default PmWiki Skin?

Yes, please see Cookbook:HideSearchBar.

Web Feeds

How do I include text from the page (whole page, or first X characters) in the feed body? (note: markup NOT digested)

```
    function MarkupExcerpt( $pagename) {
      $page = RetrieveAuthPage( $pagename, 'read', false);
      return substr(@$page['text'], 0, 200);
    }

     $FmtPV['$MarkupExcerpt'] = 'MarkupExcerpt($pn)';
    $FeedFmt['rss']['item']['description'] = '$MarkupExcerpt';
```

Does this mean if I want to include the time in the rss title and "summary" to rss body I call `$FeedFmt` twice like so:

```
 $FeedFmt['rss']['item']['description'] = '$LastSummary';
 $FeedFmt['rss']['item']['title'] = '{$Group} / {$Title} @ $ItemISOTime';
```

From mailing list Feb 13,2007, a response by Pm: Yes

How can I use the RSS <enclosure> tag for podcasting?

For podcasting of mp3 files, simply attach an mp3 file to the page with the same name as the page (i.e., for a page named

Podcast.Episode4, one would attach to that page a file named "Episode4.mp3"). The file is automatically picked up by ?action=rss and used as an enclosure.

The set of potential enclosures is given by the $RSSEnclosureFmt array, thus

```
$RSSEnclosureFmt = array('{$Name}.mp3', '{$Name}.wma', '{$Name}.ogg');
```

allows podcasting in mp3, wma, and ogg formats.

How to add "summary" to the title in a rss feed (ie. with `?action=rss`)?

Add this line in you `local/config.php`

```
$FeedFmt['rss']['item']['title'] = '{$Group} / {$Title} : $LastModifiedSummary';
```

How to add "description" to the title in an rss feed, and summary to the body?

Add these lines to your `local/config.php`

```
$FeedFmt['rss']['item']['title'] = '{$Group} / {$Title} : {$Description}';
$FeedFmt['rss']['item']['description'] = '$LastModifiedSummary';
```

NOTES:

- you need to replicate these lines for each type (atom, rdf, dc) of feed you provide.
- the RSS **`description`**-tag is not equivalent to the pmWiki *`$Description`* variable, despite the confusing similarity.

Some of my password-protected pages aren't appearing in the feed... how do I work around this?

From a similar question on the newsgroup, Pm's reply:

The last time I checked, RSS and other syndication protocols didn't really have a well-established interface or mechanism for performing access control (i.e., authentication). As far as I know this is still the case.

PmWiki's WebFeeds capability is built on top of pagelists, so it could simply be that the `$EnablePageListProtect` option is preventing the updated pages from appearing in the feed. You might try setting `$EnablePageListProtect`=0; and see if the password-protected pages start appearing in the RSS feed.

The "downside" to setting `$EnablePageListProtect` to zero is that anyone doing a search on your site will see the existence of the pages in the locked section. They won't be able to read any of them, but they'll know they are there!

You could also set `$EnablePageListProtect` to zero only if ?action=rss:

```
    if ($action == 'rss')  $EnablePageListProtect = 0;
```

This limits the ability to see the protected pages to RSS feeds; normal pagelists and searches wouldn't see them.

Lastly, it's also possible to configure the webfeeds to obtain the authentication information from the url directly, as in:

```
    .../Site/AllRecentChanges?action=rss&authpw=secret
```

The big downside to this is that the cleartext password will end up traveling across the net with every RSS request, and may end up being recorded in Apache's access logs.

How to add feed image?

Add the following to *local/config.php* (this example is for `?action=rss`):

```
     $FeedFmt['rss']['feed']['image'] =
     " <title>Logo title</title>
      <link>http://example.com/</link>
      <url>http://example.com/images/logo.gif</url>
      <width>120</width>
      <height>60</height>";
```

Do not forget NOT to start with a '<' as there would be no <image> tag around this... See here.

How do I insert RSS news feeds into PmWiki pages?

See Cookbook:RssFeedDisplay.

How can I specify default feed options in a configuration file instead of always placing them in the url?

For example, if you want `?action=rss` to default to `?action=rss&group=News&order=-time&count=10`, try the following in a local customization file:

```
        if ($action == 'rss')
```

```
            SDVA($_REQUEST, array(
              'group' => 'News',
              'order' => '-time',
              'count' => 10));
```

Are there ways to let people easily subscribe to a feed?

On some browsers (Mozilla Firefox), the visitor can see an orange RSS icon in the address bar, and subscribe to the feed by clicking on it. To enable the RSS icon, add this to config.php :

```
$HTMLHeaderFmt['feedlinks'] = '<link rel="alternate" type="application/rss+xml"
  title="$WikiTitle" href="$ScriptUrl?n=Site.AllRecentChanges&amp;action=rss" />
<link rel="alternate" type="application/atom+xml" title="$WikiTitle"
  href="$ScriptUrl?n=Site.AllRecentChanges&amp;action=atom" />';
```

You can also add such a link, for example in your SideBar,`[[Site.AllRecentChanges?action=atom | Subscribe to feed]]`.

Can I create an RSS feed for individual page histories?

See Cookbook:PageFeed.

How do I create a custom FeedPage similar to RecentChanges or AllRecentChanges, but with only certain groups or pages recorded?

See Cookbook:CustomRecentChanges. In a nutshell, you'll declare a `$RecentChangesFmt` variable with your dedicated FeedPage, and then wrap it in a condition of your choice. For example:

```
   if (PageVar($pagename, '$Group')!='ForbiddenGroup') {
     $RecentChangesFmt['Site.MyFeedPage'] =
       '* [[{$FullName}]]  . . . $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
   }
```

How can I update my RSS feed to show every edit for pages on that feed, not just new pages added to the feed?

Add unique guid links for each edit to your to config.php file (see PITS entry):

```
   $FeedFmt['rss']['item']['guid'] = '{$PageUrl}?guid=$ItemISOTime';
```

Alternatively, you can create the option for edit monitoring by adding a qualifier for RSS links. This allows the user to choose between default *new pages* RSS feeds and *new edits* RSS feeds (pmwiki.org has this option enabled).

```
   ## For new pages updates: http://example.com/wiki/HomePage?action=rss
   ## For edits updates: http://example.com/wiki/HomePage?action=rss&edits=1
   if(@$_REQUEST['edits'] && $action == 'rss')
     $FeedFmt['rss']['item']['guid'] = '{$PageUrl}?guid=$ItemISOTime';
```

Basic PmWiki editing rules

Troubleshooting

My wiki displays warnings "Deprecated: preg_replace(): The /e modifier is deprecated, use preg_replace_callback instead".

This is caused by a change in PHP version 5.5 for the preg_replace() function. PmWiki no longer relies on the deprecated feature since version 2.2.56 (it is recommended to upgrade to the latest version) but many recipes do. Note that even if the warning points to a line in pmwiki.php, the problem comes from a local configuration or recipe.

Recipes and Skins are currently being updated for PHP 5.5. Check if there are more recent versions published by their maintainers on the Cookbook. If you update your PmWiki and recipes, and still see the warnings, here is how to find out which recipes cause them:

For PmWiki version 2.2.71 or newer, in config.php, enable diagnostic tools:

```
 $EnableDiag = 1;
```

Then visit your wiki with the action 'ruleset', for example http://www.pmwiki.org/wiki/PmWiki/PmWiki?action=ruleset or follow a link like `[[HomePage?action=ruleset]]`. This page will list all markup rules; those potentially incompatible with PHP 5.5 will be flagged with filenames, line numbers and search patterns triggering the warning.

If the ?action=ruleset page shows no flagged rules, it is possible that either your recipes call the preg_replace() function directly, or they define various search-replace patterns in incompatible ways. In these cases, your warning should display the file name and line number causing problems, if not, here is how to track it. In config.php disable all recipes: included files from the cookbook directory, or a custom skin, or any line containing "Patterns". You can insert # at the beginning of a line to disable it. Then test the wiki: if you have disabled everything, the warning message should disappear.

Next, re-enable your customizations one after another, every time testing the wiki. If at some point the warnings re-appear, you'll know that the customization you just enabled is not compatible with PHP 5.5.

You can contact the authors of the broken recipes and (kindly) ask them to update their recipes for PHP 5.5 - recent PmWiki versions add new helper functions which make it easy, see CustomMarkup. If you cannot have the recipes fixed by their authors, tell us and we'll try to fix them.

Note that many hosting providers allow you to run different versions of PHP. See the documentation of your hosting plan to learn how to enable a PHP version earlier than 5.5.

Finally, it is possible to suppress these warnings in PHP 5.5, by setting this line at the beginning of config.php:
`error_reporting(E_ALL & ~E_NOTICE & ~E_DEPRECATED);`
This should be a temporary solution, left only until your recipes are fixed.

After a PHP upgrade, some of the pages on my wiki are completely blank, empty, some have blank or missing sections, but the sidebar and the action links are visible.

This can be caused by a change in PHP 5.4 which affects the function `htmlspecialchars()`.

The easiest temporary fix would be in your `php.ini`, or in `.user.ini` to change the `default_charset` directive to an 8-bit charset, for example cp1252:

```
default_charset = "Windows-1252"
```

Or, this may sometimes work in `pmwiki/local/config.php`:

```
ini_set("default_charset", "Windows-1252");
```

A more permanent fix would be to upgrade your installation to a more recent PmWiki version, your recipes, and in your own recipes or modules replace all calls to `htmlspecialchars()` with `PHSC()`, a PmWiki helper function for such cases.

The "blank" pages come from the fact that in PHP 5.4 the default encoding switched from an 8-bit encoding to variable-bit validated UTF-8, and that an incorrect UTF-8 string will be rejected. If your wiki uses an 8-bit encoding, it is virtually certain that it is not valid UTF-8. Worse, even if you do use UTF-8 some browsers may submit invalid bits. So the PHSC() function always pretends that it converts an 8-bit encoding where all bits are allowed.

Why am I seeing strange errors after upgrading?

Make sure all of the files were updated, in particular *pmwiki.php*.

This question sometimes arises when an administrator hasn't followed the advice, which used to be less prominent, on the installation and initial setup tasks pages and has renamed *pmwiki.php* instead of creating an *index.php* wrapper script. If you have renamed *pmwiki.php* to *index.php*, then the upgrade procedure won't have updated your *index.php* file. Delete the old version and create a wrapper script so it won't happen again.

Sometimes an FTP or other copy program will fail to transfer all of the files properly. One way to check for this is by comparing file sizes.

Be sure all of the files in the *wikilib.d/* directory were also upgraded. Sometimes it's a good idea to simply delete the *wikilib.d/* directory before upgrading. (Local copies of pages are stored in *wiki.d/* and not *wikilib.d/*.)

Make sure that the file permissions are correct. The official files have a restricted set of permissions that might not match your site's needs.

If you use a custom pattern for `$GroupPattern` make sure that it includes Site (`$SiteGroup`) and since PMWiki 2.2 also SiteAdmin (`$SiteAdminGroup`). Otherwise migration may fail (e.g. missing SiteAdmin for PMWiki 2.2 and later) and/or login does not work.
Additionally Main (`$DefaultGroup`) should be included too.

I'm suddenly getting messages like "`Warning: fopen(wiki.d/.flock): failed to open stream: Permission denied...`" and "`Cannot acquire lockfile`"... what's wrong?

Something (or someone) has changed the permissions on the *wiki.d/.flock* file or the *wiki.d/* directory such that the webserver is no longer able to write the lockfile. The normal solution is to simply delete the *.flock* file from the *wiki.d/* directory -- PmWiki will then create a new one. Also be sure to check the permissions on the *wiki.d/* directory itself. (One can easily check and modify permissions of the *wiki.d/* directory in FileZilla (open-source FTP app) by right-clicking on the file > File attributes)

My links in the sidebar seem to be pointing to non-existent pages, even though I know I created the pages. Where are the pages?

Links in the sidebar normally need to be qualified by a WikiGroup in order to work properly (use [[Group.Page]] instead of [[Page]]).
Also: Make sure you type Side**B**ar with a capital B.

Why am I seeing "`PHP Warning: Cannot modify header information - headers already sent by ...`" messages at the top of my page.

If this is the first or only error message you're seeing, it's usually an indication that there are blank lines, spaces, or other characters before the `<?php` or after the `?>` in a local customization files such as `config.php`. Double-check the file and make sure there is nothing before the initial `<?php`. It's often easiest and safest to eliminate any closing `?>` altogether. On Windows, it may be, but shouldn't be, necessary to use a hex editor to convert LFCR line endings to LF line endings in the `local\config.php` file.

When you save the file, the encoding/charset should be either cp1252/Windows1252 or UTF-8 without Byte Order Mark. NotePad++ is an editor that can do this.

When you transfer the files, tell your FTP manager to use text mode transfer, or, if that doesn't help, binary mode transfer.

If the warning is appearing after some other warning or error message, then resolve the other error and this warning may go away.

How do I make a PHP Warning about `function.session-write-close` go away?

If you are seeing an error similar to this

```
Warning: session_write_close() [function.session-write-close]:
open(/some/filesystem/path/to/a/directory/sess_[...]) failed: No such file
or directory (2) in /your/filesystem/path/to/pmwiki.php on line NNN
```

PmWiki sometimes does session-tracking using PHP's session-handling functions. For session-tracking to work, some information needs to be written in a directory on the server. That directory needs to exist and be writable by the webserver software. For this example, the webserver software is configured to write sessions in this directory

```
/some/filesystem/path/to/a/directory/
```

but the directory doesn't exist. The solution is to do at least one of these:

- **Create the directory** and make sure it's writable by the webserver software
- Provide a session_save_path value that points to a directory that is writable by the server, e.g. in config.php:

```
session_save_path('/home/someuser/tmp/sessions'); # unix-type OS
session_save_path('C:/server/tmp/sessions'); # Windows
```

Why is PmWiki prompting me multiple times for a password I've already entered?

This could happen like out of nowhere if your hosting provider upgrades to PHP version 5.3, and you run an older PmWiki release. Recent PmWiki releases fix this problem.

Alternatively, this may be an indication that the browser isn't accepting cookies, or that PHP's session handling functions on the server aren't properly configured. If the browser is accepting cookies, then try setting `$EnableDiag`=1; in *local/config.php*, run PmWiki using `?action=phpinfo`, and verify that sessions are enabled and that the session.save_path has a reasonable value. Note that several versions of PHP under Windows require that a session_save_path be explicitly set (this can be done in the *local/config.php* file). You might also try setting session.auto_start to 1 in your php.ini.

See also the question I have to log in twicebelow.

I edited *config.php*, but when I look at my wiki pages, all I see is "`Parse error: parse error, unexpected T_VARIABLE in` *`somefile`* `on line` *`number`*."

You've made a mistake in writing the PHP that goes into the *config.php* file. The most common mistake that causes the T_VARIABLE error is forgetting the semi-colon (;) at the end of a line that you added. The line number and file named are where you should look for the mistake.

Searches and pagelists stopped working after I upgraded -- no errors are reported, but links to other pages do not appear (or do not appear as they should) -- what gives?

Be sure all of the files in the *wikilib.d/* directory were also upgraded. In particular, it sounds as if the Site.PageListTemplates page is either missing (if no links are displayed) or is an old version (if the links do not appear as they should). Also make sure that read-permissions (attr) are set for the pages Site.PageListTemplates and Site.Search.

Some of my posts are coming back with "403 Forbidden" or "406 Not Acceptable" errors, or "Internal Server Error". This happens with some posts but not others.

Your webserver probably has mod_security enabled. The mod_security "feature" scans all incoming posts for forbidden words or phrases that might indicate someone is trying to hack the system, and if any of them are present then Apache returns the 403 Forbidden or 406 Not Acceptable error. Common phrases that tend to trigger mod_security include "curl ", "wget", "file(", and "system(", although there are many others (depending on the configuration, percent signs, html tags, international characters).

Since mod_security intercepts the requests and sends the "forbidden" message before PmWiki ever gets a chance to run, it's not a bug in PmWiki, and there's little that PmWiki can do about it. Instead, one has to alter the webserver configuration to disable mod_security or reconfigure it to allow whatever word it is forbidding. Some sites may be able to disable mod_security by placing `SecFilterEngine off` in a *.htaccess* file.

I get the following message when attempting to upload an image, what do I do?

```
Warning: move_uploaded_file(): SAFE MODE Restriction in effect. The script whose uid is 1929 is not
allowed to access /home/onscolre/public_html/pmwikiuploads/Photos owned by uid 33 in
/home/onscolre/public_html/pmwiki/scripts/upload.php on line 198

PmWiki can't process your request

?cannot move uploaded file to
/home/onscolre/public_html/pmwikiuploads/Photos/FoundationPupilsIn1958.jpeg

We are sorry for any inconvenience.
```

Your server is configured with PHP Safe Mode enabled. Configure your wiki to use a site-wide uploads prefix, then create the *uploads/* directory manually and set 777 permissions on it (rather than letting PmWiki create the directory).

I'm starting to see "Division by zero error in pmwiki.php..." on my site. What's wrong?

It's a bug in PmWiki that occurs only with the tables markup and only for versions of PHP >= 4.4.6 or >= 5.2.0. Often it seems to occur "out of nowhere" because the server administrator has upgraded PHP. Try upgrading to a later version of PmWiki to remove the error, or try setting the following in *local/config.php*:

```
    $TableRowIndexMax = 1;
```

I have to log in twice (two times) (2 times). -or- My password is not being required even though it should. -or- I changed the password but the old password is still active. -or- My config.php password is not over-riding my farmconfig.php password.

It could happen if (farm)config.php, or an included recipe, directly calls the functions CondAuth(), or RetrieveAuthPage(), PageTextVar(), PageVar() and possibly others, before defining all passwords and before including AuthUser (if required).

The order of config.php is very significant.

When editing an existing page, The "Save" causes a no-response of your server (not a blank page, no response at all, an endless connexion try). To get back the hand, it is necessary to request for another page (by clicking on its link in the menu for instance). And horror!, the ...?action=edit is then inhibited, it becomes impossible to edit any page.

When the editing of a page is initiated a file names `.flock` is created in the `wiki.d` repository. As long as this file exists it is impossible to edit any page. This file denotes an edition in progress and is automatically destroyed when leaving successfully an edit action by "Save". In case of a crash of the editing, this file is not destroyed. The remedy is, with an FTP client parameterized to show hidden files, to remove the `.flock` file. And all get back OK. This behavior is typically caused by a bug which provokes (directly or indirectly), an endless loop in a recipe concerned by the edited page.

I get the error "Data Mismatch - Locking FAILED!"

This is probably not a PmWiki error. PmWiki cannot create a lock file due to an underlying file system problem. For example the disk quota has been exceeded (e.g. by an error log file or file uploads), or there are problems with file system permissions.

Auth User

Can I specify authorization group memberships from with *local/config.php*?

Yes -- put the group definition into the $AuthUser array (in config.php):

```
        $AuthUser['@editors'] = array('alice', 'carol', 'bob');
```

Can I have multiple admin groups?

Yes, define the groups with `array('@admins', '@moderators');` like this:

```
  $DefaultPasswords['admin'] = array( pmcrypt('masterpass'), # global password
    '@admins', '@moderators', # +users in these groups
    'id:Fred', 'id:Barney');  # +users Fred and Barney
```

I'm running multiple wikis under the same domain name, and logins from one wiki are appearing on other wikis. Shouldn't they be independent?

This is caused by the way that PHP treats sessions. See PmWiki.AuthUser#sessions for more details.

Is there any way to record the time of the last login for each user when using AuthUser? I need a way to look for stale accounts.

See Cookbook:UserLastAction.

Though every setting seems correct, authentication against LDAP is not working. There is nothing in ldap log, what's wrong?

Be sure ldap php module is installed (on debian apt-get install php(4|5)-ldap ; apache(2)ctl graceful)

The login form asks for username and password, but only password matters.

Username can be left blank and it still signs in under the account. Is this intentional and if so, can I change it so that the username and password must both be entered? - X 1/18/07 Never mind I think this has something to do with using the admin password. I created a test account and it's working ok.

Make sure you are not entering the admin password when testing the account because, if the password is equal to the admin password, it will authenticate directly through the config.php file and skip any other system.

Do note that even with AuthUser activated you can still log in with a blank username and only entering the password. In that case any password you enter will be "accepted" but only passwords which authenticate in the given context will actually give you any authorization rights. Using this capability AuthUser comfortably coexists with the default password-based system.

If you want to require both username and password, then you need to set an admin id**before** including authuser.php:

```
## Define usernames and passwords.
$AuthUser['carol'] = '$1$CknC8zAs$dC8z2vu3UvnIXMfOcGDON0';

## Enable authentication based on username.
include_once('scripts/authuser.php');

# $DefaultPasswords['admin'] = pmcrypt('secret');
$DefaultPasswords['admin'] = 'id:carol';
```

A username and password will then be required before login is successful.

Is there any way to hide IP addresses once someone has logged in so that registered users can keep their IP addresses invisible to everyone except administrators? - X 1/18/07

Yes, see solution provided at PITS:00400.

Is there a way that people could self-register through AuthUser?

You can see HtpasswdForm or UserAdmin for recipes providing this feature.

I would like it that after I have AuthUser turned and a user is authenticated to get on my site, that if I have a password put on a particular page or group that they don't get the AuthUser form to show up (username and password), but only the typical field for password?

See this thread of the mailing list.

Passwords Admin

There seems to be a default password. What is it?

There isn't any valid password until you set one. Passwords admin describes how to set one.

PmWiki comes "out of the box" with `$DefaultPasswords`['admin'] set to '*'. This doesn't mean the password is an asterisk, it means that default admin password has to be something that encrypts to an asterisk. Since it's impossible for the pmcrypt() function to ever return a 1-character encrypted value, the admin password is effectively locked until the admin sets one in config.php.

How do I use passwd-formatted files (like .htpasswd) for authentication?

See AuthUser, Cookbook:HtpasswdForm or Cookbook:UserAuth2.

Is there anything I can enter in a GroupAttributes field to say 'same as the admin password'? If not, is there anything I can put into the config.php file to have the same effect?

Enter '@lock' in GroupAttributes?action=attr to require an admin password for that group.

How do I edit protect, say, all RecentChanges pages?

see Security#wikivandalism.

How can I read password protect all pages in a group except the HomePage using configuration files?

As described in PmWiki.GroupCustomizations per-group or per-page configuration files should not be used for defining passwords. The reason is that per-group (or per-page) customization files are only loaded for the current page. So, if

`$DefaultPasswords['read']` is set in *local/GroupA.php*, then someone could use a page in another group to view the contents of pages in GroupA. For example, Main.WikiSandbox could contain:

(:include GroupA.SomePage:)

and because the *GroupA.php* file wasn't loaded (we're looking at Main.WikiSandbox -->*local/Main.php*), there's no read password set.

How can I password protect the creation of new pages?

See Cookbook:LimitWikiGroups, Cookbook:NewGroupWarning, Cookbook:LimitNewPagesInWikiGroups.

How do I change the password prompt screen?

If your question is about how to make changes to that page... edit Site.AuthForm. If your question is about how to change which page you are sent to when prompted for a password, you might check out the Cookbook:CustomAuthForm for help.

How do I change the prompt on the attributes (`?action=attr`) screen?

Simply create a new page at Site.AttrForm, and add the following line of code to`config.php`:

```
$PageAttrFmt = 'page:Site.AttrForm';
```

Note that this only changes the text above the password inputs on the attributes page, but doesn't change the inputs themselves - the inputs have to be dealt with separately. See Cookbook:CustomAttrForm for more info.

I get http error 500 "Internal Server Error" when I try to log in. What's wrong?

This can happen if the encrypted passwords are not created on the web server that hosts the PmWiki.
The crypt function changed during the PHP development, e.g. a password encrypted with PHP 5.2 can not be decrypted in PHP 5.1, but PHP 5.2 can decrypt passwords created by PHP 5.1.
This situation normally happens if you prepare everything on your local machine with the latest PHP version and you upload the passwords to a webserver which is running an older version.
The same error occurs when you add encrypted passwords to local/config.php.

Solution: Create the passwords on the system with the oldest PHP version and use them on all other systems.

I only want users to have to create an 'edit' password, which is automatically used for their 'upload' & 'attr' passwords (without them having to set those independently). How do I do this?

By setting `$HandleAuth` like so:

```
      $HandleAuth['upload'] = 'edit';
      // And to prevent a WikiSandbox from having it's 'attr' permissions changed
      // except by the admin (but allowing editors to change it on their own pages/group)
      if(($group=="Site") || ($group=="Main") || ($group=="Category") ||
             ($group=="SiteAdmin") || ($group=="PmWiki") ) {
 $HandleAuth['attr'] = 'admin';  // for all main admin pages, set 'attr' to 'admin' password
      } else {
 $HandleAuth['attr'] = 'edit';  // if you can edit, then you can set attr
      }
```

Design Notes

Why doesn't PmWiki use hierarchical / nested groups?

It essentially comes down to figuring out how to handle page links between nested groups; if someone can figure out an obvious, intuitive way for authors to do that, then nested groups become plausible. See Design Notes and PmWiki:Hierarchical Groups.

Why don't PmWiki's scripts have a closing ?> tag?

All of PmWiki's scripts now omit the closing ?> tag. The tag is not required, and it avoids problems with unnoticed spaces or blank lines at the end of the file. Also, some file transfer protocols may change the newline character(s) in the file, which can also cause problems. See also the Instruction separation page in the PHP manual.

Does PmWiki support WYSIWYG editing (or something like the FCKEditor)?

Short answer: PmWiki provides GUI buttons in a toolbar for common markups, but otherwise does not have WYSIWYG editing. For the reasons why, see PmWiki:WYSIWYG.

Categories: PmWiki Developer

FilePermissions

toc top

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

Last modified by simon on September 10, 2011.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/FilePermissions

FmtPageName

This page describes an internal function in PmWiki's engine called FmtPageName(). The contents are not intended for those with a weak heart ;-)

Also see: PmWiki.Functions

`FmtPageName`($fmt, `$pagename`)

Returns `$fmt`, with $variable and internationalisation substitutions performed, under the assumption that the current page is `pagename`. As a rule is used to pre-process all variables which by convention have a "Fmt" suffix (like`$GroupFooterFmt`), but also other strings that require interpolation, notably the page template (.tmpl) file. See PmWiki.Variables for an (incomplete) list of available variables, PmWiki.Internationalizations for internationalisation.

The function `FmtPageName()` applies internationalization-substitutions and $Variable-substitions to the string $fmt under the assumption that the current page is `$pagename`.

The substitutions go as follows:

1. Replace any sequences of the form `$XyzFmt` with value of any corresponding global variable.
2. Process the string for any `$[...]` phrases (internationalized phrase), using the currently loaded translation tables.
3. Replace any instances of `{$ScriptUrl}` with `$ScriptUrl` (to defer processing to the URI processing phase)
4. Replace any instances of standard Page Variables (beginning with an upper case letter, followed by at least one word character) with their values. Note that PVs of the form `{Group.Page$Var}` are **not** replaced. If there are no more $-sequences, then return the formatted string and exit the function
5. Perform any pattern replacements from the array `$FmtP`. Typically this is used to handle things like $Name and $Group etc that are specific to the name of the current page. *?? Appears to be used in robots.php to hide actions from robots.*
6. Replace any remaining instances of Page Variables with their values. Note that these variables are in the form`$Var` rather than the usual PV form of `{$Var}`.
7. If `$EnablePathInfo` isn't set, convert `URIs` to use the syntax `$ScriptUrl`?n=<Group>.<Name> instead of `$ScriptUrl`/<Group>/<Name>. In any case, replace `$ScriptUrl` with its value. If there are no more $-sequences, then return the formatted string and exit the function
8. Replace any $-sequences with global variables (caching as needed) of the same name (in reverse alphabetical order, and filtering out any unsafe globals) *
9. Replace any $-sequences with values out of the array `$FmtV`.

Note that `FmtPageName`() is automatically aware of any global variables. However, since modifying global variables may be expensive, the array `$FmtV` exists as a way to avoid rebuilding the variable cache for values that change frequently.

Security

According to PM, as a general rule it's unwise to be calling FmtPageName() on strings that are coming from page markup, as this exposes the ability for people to view the values of variables that perhaps they shouldn't see. This is also why page variables (which come from markup) use PageVar() and PageTextVar() and don't go through FmtPageName().

Availability of Variables in FmtPageName

To be very specific, here's what Pm wrote regarding different ways of defining a variable that can be used by FmtPageName (when it is formatting a string):

- Set a global variable. FmtPageName() automatically performs substitution on all global variables that aren't arrays. If the variable is going to change value over repeated calls to FmtPageName, it's probably better to use `$FmtV` as in the next item.

- Set a value in the `$FmtV` array. `$FmtV`['$MyVariable']='something' means to replace instances of '$MyVariable' with 'something'. Use this for variables that change value frequently over multiple calls to FmtPageName.

- Set a pattern/replacement in the `$FmtP` array. This is normally done for substitutions that have to be dynamic somehow based on the pagename being referenced, such as '$Title', '$Group', '$Name', '$PageUrl', etc.

Also see: Cookbook:Functions#FmtPageName

Finally, here's something else Pm wrote that is related and explains why we have this function:

> In order to produce its output, PmWiki has to do a variety of string substitutions:
>
> 1. Generating the full name, group, title, or url of a page (other than the currently displayed page)
> 2. Substituting the values of global variables
> 3. Performing internationalization substitutions
> 4. Converting `$ScriptUrl`/$Group/$Name to `$ScriptUrl`?n=$Group.$Name for sites that cannot handle PATH_INFO urls
> 5. Other substitutions needed by specific functions
>
> PmWiki centralizes all of that substitute-a-dynamic-value-in-a-string into the FmtPageName() subroutine. Because some things are extremely dynamic, such as the url or group for an arbitrary page that is not the current one, those things cannot be simple global PHP variables. Or, if they do become global variables, they're variables that cannot be trusted to hold a value for very long because some other routine (that may happen to be formatting a string for a different page) will come along and change that global variable for whatever it happens to be doing.
>
> A limited set of $-substitutions -- basically anything that corresponds to a page attribute -- are not PHP variables and are only available through the FmtPageName() subroutine. The complete set of these special substitutions is $Group, $Name, $FullName, $PageUrl, $Title, $Titlespaced, $Namespaced, $Groupspaced, $LastModifiedBy, $LastModifiedHost, and $LastModified. These items cannot just be standard PHP variables because often PmWiki needs to obtain the url, name, group, title, etc. of a page other than the one currently being viewed by a browser.
>
> At the moment, $Title, $LastModified, $LastModifiedBy, and $LastModifiedHost can only work if the page's attributes have been loaded and cached using the PCache function. So, to get at these values one must typically do:
>
> ```
> $page = ReadPage($pagename);
> PCache($pagename, $page);
> $ptitle = FmtPageName('$Title', $pagename);
> $pauthor = FmtPageName('$LastModifiedBy', $pagename);
> ```

Forms

This page explains how you can embed input forms into wiki pages.

Input forms don't actually handle processing of the form data -- the feature simply allows creation of forms inside wiki pages. Forms processing can be found in the Cookbook (see below).

Markup

Two directives are used to begin and end forms:

```
(:input form "url" method:)
...
(:input end:)
```

The `(:input form:)` directive starts a form that will post to*url* (optional *action*=url) using the supplied *method* (optional *method*=method). The *url* must be in quotes if not specified via*action*=. If the url is omitted, then the current page is assumed. If *method* is omitted then "POST" is assumed. An optional `name="FormName"` argument can be used to name the form. You can explicitly state `action=url` or `method=get` or you can simply use them as positional parameters.

If your site uses ?n=Group.Page to specify the pagename then having a field`(:input hidden name=n value={$FullName}:)` will allow your form to post to the current page as an alternative to fully specifying the action=url.

The `(:input end:)` directive ends the current form.

Note that this feature doesn't ensure that the form output is correct HTML -- it assumes the author knows a little bit of what he or she is doing. Notably, (:input form:) and (:input end:) shouldn't appear inside tables, and all form fields and controls should be inside an (:input form:)...(:input end:) block.

Standard input controls

The standard input controls are:

```
    (:input text name value size=n:)
    (:input hidden name value:)
    (:input password name value:)
    (:input search name value:)
    (:input number name value min=x max=y step=z:)
    (:input email name value:)
    (:input url name value:)
    (:input date name value:)
    (:input radio name value label checked=checked:)
    (:input checkbox name value label checked=checked:)
    (:input select name value label:)
    (:input default default-name default-value:)
    (:input submit name value:)
    (:input textarea name [=value=] rows=n cols=n:)
    (:input reset name label:)
    (:input file name label:)
    (:input image name "src" alt:)
```

Where *name* and *value* are in the HTML syntax: name="addr" value="808 W Franklin".

For most controls the markup has the form:

```
    (:input type name value parameter=value:)
```

where *type* is the type of input element (described below),*name* is the name of the control,*value* is its initial value, and parameters are used to specify additional attributes to the control. If *value* contains spaces, enclose it in quotes; if it contains newlines (for textarea and hidden elements), enclose it in `[=...=]`.

For example, the following creates a text input control with a size of 30 characters:

```
(:input text authorid "Jane Doe" size=30:)
```

Jane Doe

For convenience, an author can also specify name and value arguments directly using`name=` and `value=` attributes (same as HTML):

```
(:input text name=authorid value="Jane Doe" size=30:)
```

Jane Doe

For the `textarea` control a value can be set from PmWiki 2.2.0beta45 onwards. Enclose the value in`[=...=]` if it contains spaces or new lines.

The `submit` control will more often be written as:

```
    (:input submit value=label:)
```

Here's a more complete example, e.g., for a login prompt:

```
(:input form "http://www.example.com":)
(:input hidden action login:)
||     Name:||(:input text username:)         ||
|| Password:||(:input password password:)     ||
||          ||(:input checkbox terms yes "Accept Terms":) ||
||          ||(:input submit value="Log In":) ||
(:input end:)
```

Name:	
Password:	
	☐ Accept Terms
	Log In

General form field attributes

- `(:input ... focus=1:)` Setting `focus=1` causes that field to receive the initial focus when the form is first opened.
- The following advanced HTML attributes are supported: `name, value, id, class, rows, cols, size, maxlength, action, method, accesskey, tabindex, multiple, checked, disabled, readonly, enctype, src, alt`. For a more detailed description, see their counterparts in the w3c reference: HTML forms (not all of them can be used for all types of form fields).

`(:input select ... :)`

The basic form of a select box is a sequence of options:

```
(:input form:)
(:input select name=abc value=1 label=alpha :)
(:input select name=abc value=2 label=beta  :)
(:input select name=abc value=3 label=gamma :)
(:input submit:)
(:input end:)
```

alpha ▾ Submit

The values can be specified positionally:

```
(:input select abc 1 alpha :)
```

We can specify the size of the selection box:

```
(:input select abc 1 alpha size=3 :)
```

You can specify a multiple select box (only the first item needs to have "size=3 multiple" attributes):

```
(:input select abc 1 alpha size=3 multiple:)
```

To have an element selected, use `selected=selected`:

```
(:input select abc 2 beta selected=selected:)
```

Note that to have two select boxes inline, not only should you give them different`name=` parameters, but also place a separator, like a character, `&nbsp;` or even the null sequence `[==]` between them:

```
(:input form:)
(:input select name=FIRST value=1:)(:input select name=FIRST value=2:)[==]
(:input select name=SECOND value=3:)(:input select name=SECOND value=4:)
(:input end:)
```

1 ▾ 3 ▾

See Also

- Cookbook:Input Default
- Cookbook:Form Validation
- Cookbook:Form Extensions
- Cookbook:Input Forms and JavaScript

Compatible recipes:

- Cookbook:PmForm
- Cookbook:Fox
- Cookbook:Wiki Forms
- Cookbook:ProcessForm

Functions

This page describes some of the internal workings of PmWiki by explaining how some of the functions in pmwiki.php work. For a more brief list/overview on functions useful to for instance cookbook writers, see Cookbook:Functions.

To use this functions you have to make sure that all relevant internal variables have been initialized correctly. See Custom Markup and Custom Actions for more information on how these functions are typically called via `Markup()` or `$HandleActions[]`.

`pmcrypt($password, $salt = null)`

The pmcrypt() function is intended to be a safe replacement for the PHP 5.6+ crypt() function without providing a $salt, which would raise a notice. If a salt is provided, crypt() is called to check an existing password. If a salt is not provided, password_hash() will be called to create a cryptographically strong password hash.

`PCCF($php_code, $callback_template='default', $callback_arguments = '$m')` Deprecated since PHP 7.2

The PCCF() function (*PmWiki Create Callback Function*) can be used to create callback functions used with preg_replace_callback. It is required for PHP 5.5, but will also work with earlier PHP versions.

The first argument is the PHP code to be evaluated.

The second argument (optional) is the callback template, a key from the global $CallbackFnTemplates array. There are two templates that can be used by recipe authors:

- 'default' will pass $php_code as a function code
- 'return' will wrap $php_code like "return $php_code;" (since PmWiki 2.2.62)

The third argument (optional) is the argument of the callback function. Note that PmWiki uses the '$m' argument to pass the matches of a regular expression search, but your function can use other argument(s).

PCCF() will create an anonymous (lambda) callback function containing the supplied code, and will cache it. On subsequent calls with the same $php_code, PCCF() will return the cached function name.

See http://php.net/create_function.

`PPRA($array_search_replace, $string)`

The PPRA() function (*PmWiki preg_replace array*) can be used to perform a regular expression replacement with or without evaluation, for PHP 5.5 compatibility.

Since PmWiki 2.2.56, PmWiki uses this function to process the following arrays: `$MakePageNamePatterns`, `$FmtP`, `$QualifyPatterns`, `$ROEPatterns`, `$ROSPatterns`, $SaveAttrPatterns, `$MakeUploadNamePatterns`. Any custom settings should continue to work for PHP 5.4 and earlier, but wikis running on PHP 5.5 may need to make a few changes.

The first argument contains the 'search'=>'replace' pairs, the second is the "haystack" string to be manipulated.

The 'replace' parts of the array can be strings or function names. If the 'replace' part is a callable function name, it will be called with the array of matches as a first argument via preg_replace_callback(). If not a callable function, a simple preg_replace() will be performed.

Previously, PmWiki used such constructs:

```
  $fmt = preg_replace(array_keys( $FmtP), array_values( $FmtP), $fmt);
```

It is now possible to use simply this:

```
  $fmt = PPRA( $FmtP, $fmt);
```

Note that since PHP 5.5, the search patterns cannot have an /e evaluation flag. When creating the $array_search_replace array, before PHP 5.5 we could use something like (eg. for `$MakePageNamePatterns`):

```
  '/(?<=^| )([a-z])/e' => "strtoupper('$1')",
```

Since PHP 5.5, we should use this (will also work in PHP 5.4 and earlier):

```
  '/(?<=^| )([a-z])/' => PCCF("return strtoupper(\$m[1]);"),
```

Note that the /e flag should be now omitted, instead of '$0', '$1', '$2', we should use $m[0], $m[1], $m[2], etc. in the replacement code, and there is no need to call PSS() in the replacement code, as backslashes are not automatically added.

`PPRE($search_pattern, $replacement_code, $string)` Deprecated since PHP 7.2

The PPRE() function (*PmWiki preg_replace evaluate*) can be used to perform a regular expression replacement with evaluation.

Since PHP 5.5, the preg_replace() function has deprecated the /e evaluation flag, and displays warnings when the flag is used. The PPRE() function automatically creates a callback function with the replacement code and calls it.

Before PHP 5.5, it was possible to use such calls:

```
  $fmt = preg_replace('/\\$([A-Z]\\w*Fmt)\\b/e','$GLOBALS["$1"]',$fmt);
```

Since PHP 5.5, it is possible to replace the previous snippet with the following (also works before PHP 5.5):

```
  $fmt = PPRE('/\\$([A-Z]\\w*Fmt)\\b/','$GLOBALS[$m[1]]',$fmt);
```

Note that the /e flag should be now omitted, instead of '$0', '$1', '$2', we should use $m[0], $m[1], $m[2], etc. in the replacement code, and there is no need to call PSS() in the replacement code, as backslashes are not automatically added.

`Qualify($pagename, $text)`

Qualify() applies `$QualifyPatterns` to convert relative links and references into absolute equivalents. This function is called by usual wiki markups that include text from other pages. It will rewrite links like `[[Page]]` into `[[Group/Page]]`, and page (text) variables like `{$Title}` into `{Group.Page$Title}` so that they work the same way in the source page and in the including page. See also `$QualifyPatterns` and `RetrieveAuthSection()`.

`PHSC($string_or_array, $flags=ENT_COMPAT, $encoding=null, $double_encode=true)`

The PHSC() function (*PmWiki HTML special characters*) is a replacement for the PHP function htmlspecialchars.

The htmlspecialchars() function was modified since PHP 5.4 in two ways: it now requires a valid string for the supplied encoding, and it changes the default encoding to UTF-8. This can cause sections of the page to become blank/empty on many sites using the ISO-8859-1 encoding without having set the third argument ($encoding) when calling htmlspecialchars().

The PHSC() function calls htmlspecialchars() with an 8-bit encoding as third argument, whatever the encoding of the wiki (unless you supply an encoding). This way the string never contains invalid characters.

It should be safe for recipe developers to replace all calls to htmlspecialchars() with calls to PHSC(). Only the first argument is required when calling PHSC().

Unlike htmlspecialchars(), the PHSC() function can process arrays recursively (only the values are converted, not the keys of the array).

`PSS($string)`

The PSS() function (*PmWiki Strip Slashes*) removes the backslashes that are automatically inserted in front of quotation marks by the /e option of PHP's preg_replace function. PSS() is most commonly used in replacement arguments to Markup(), when the pattern specifies /e and one or more of the parenthesized subpatterns could contain a quote or backslash. ("PSS" stands for "PmWiki Strip Slashes".)

> From PM: PmWiki expects PSS() to always occur inside of double-quoted strings and to contain single quoted strings internally. The reason for this is that we don't want the $1 or $2 to accidentally contain characters that would then be interpreted inside of the double-quoted string when the PSS is evaluated.
>
> ```
>     Markup('foo', 'inline', '/(something)/e', 'Foo(PSS("$1"))'); # wrong
>     Markup('foo', 'inline', '/(something)/e', "Foo(PSS('$1'))"); # right
> ```

Note, the markup definitions with `Markup_e()` do NOT need to use PSS() in the replacement strings.

Example

This is a fictitious example where PSS() should be used. Let us assume that you wish to define a directive`(:example:)` such that `(:example "A horse":)` results in the HTML

```
     <div>"A horse"</div>.
```

Here is how the markup rule can be created:

```
     Markup('example', 'directives',
            '/\\(:example\\s(.*?):\\)/e',
            "Keep('<div>'.PSS('$1').'</div>')");
```

We need to use PSS() around the '$1' because the matched text could contain quotation marks, and the /e will add backslashes in front of them.

`stripmagic($string)`

This function should be used when processing the contents of`$_POST` or `_GET` variables when they could contain quotes or backslashes. It verifies `get_magic_quotes()`, if true, strips the automatically inserted escapes from the string.

The function can process arrays recursively (only the values are processed).

`FmtPageName`($fmt, `$pagename`)

Returns `$fmt`, with $variable and $[internationalisation] substitutions performed, under the assumption that the current page is `pagename`. See PmWiki.Variables for an (incomplete) list of available variables, PmWiki.Internationalizations for internationalisation. Security: not to be run on user-supplied data.

This is one of the major functions in PmWiki, see PmWiki.FmtPageName for lots of details.

`Markup`($name, $when, $pattern, $replace)

Adds a new markup to the conversion table. Described in greater detail at PmWiki.CustomMarkup.

This function is used to insert translation rules into the PmWiki's translation engine. The arguments to`Markup()` are all strings, where:

`$name`
: The string names the rule that is inserted. If a rule of the same name already exists, then this rule is ignored.

`$when`
: This string is used to control *when* a rule is to be applied relative to other rules. A specification of "`<xyz`" says to apply this rule prior to the rule named "xyz", while "`>xyz`" says to apply this rule after the rule "xyz". See CustomMarkup for more details on the order of rules.

`$pattern`
: This string is a regular expression that is used by the translation engine to look for occurences of this rule in the markup source.

`$replace`
: This string will replace the matched text when a match occurs, or the function name that will return the replacement text.

Also see: PmWiki.CustomMarkup and Cookbook:Functions#Markup

`MarkupToHTML`(`$pagename`, $str)

Converts the string `$str` containing PmWiki markup into the corresponding HTML code, assuming the current page is `$pagename`.

Also see: Cookbook:Functions#MarkupToHTML

`mkdirp`($dir)

The function `mkdirp`($dir) creates a directory, `$dir`, if it doesn't already exist, including any parent directories that might be needed. For each directory created, it checks that the permissions on the directory are sufficient to allow PmWiki scripts to read and write files in that directory. This includes checking for restrictions imposed by PHP's safe_mode setting. If `mkdirp`() is unable to successfully create a read/write directory, `mkdirp`() aborts with an error message telling the administrator the steps to take to either create $dir manually or give PmWiki sufficient permissions to be able to do it.

`MakeLink`(`$pagename`, $target, $txt, $suffix, $fmt)

The function `MakeLink`(`$pagename`, $target, $txt, $suffix, $fmt) returns an html-formatted anchor link. Its arguments are as follows:

```
 $pagename is the source page
$target is where the link should go
$txt is the value to use for '$LinkText' in the output
$suffix is any suffix string to be added to $txt
$fmt is a format string to use
```

If $txt is NULL or not specified, then it is automatically computed from $target.

If $fmt is NULL or not specified, then MakeLink uses the default format as specified by the type of link. For page links this means the `$LinkPageExistsFmt` and `$LinkPageCreateFmt` variables, for intermap-style links it comes from either the `$IMapLinkFmt` array or from `$UrlLinkFmt`. Inside of the formatting strings, $LinkUrl is replaced by the resolved url for the link, $LinkText is replaced with the appropriate text, and $LinkAlt is replaced by any "title" (alternate text) information associated with the link.

Also see: PmWiki:MakeLink and Cookbook:Functions#MakeLink

`MakeUploadName`(`$pagename`, $x)

`MakeUploadName()` simply takes a string `$x` (representing an attachment's name) and converts it to a valid name by removing any unwanted characters. It also requires the name to begin and end with an alphanumeric character, and as of 2.0.beta28 it forces any file extensions to lowercase. This function is defined in `scripts/upload.php` and only used when uploads are enabled.

`SessionAuth`(`$pagename`, $auth=NULL)

`SessionAuth()` manages keeping authentication via cookie-sessions. Session contains ever password or vaidated id and associated groups from previous calls.It adds elements passed by `$auth` to session. It also writes every element saved in session to `$AuthPw`(passwords) and `$AuthList`(ids and groups).

`IsAuthorized`($chal, $source, &$from)

`IsAuthorized` takes a pageattributesstring (e. g. "id:user1 $1$Ff3w34HASH...") in`$chal`. `$source` is simply returned and used for building the authcascade (pageattributes - groupattributes - `$DefaultPassword`). `$from` will be returned if `$chal` is empty, because it is not checked before calling `IsAuthorized()`, this is needed for the authcascade.`IsAuthorized()` returns an array with three values: `$auth` `1` - authenticated, `0` - not authenticated, `-1` - refused; `$passwd`; `$source` from the parameter list.

`CondAuth` (`$pagename`, 'auth level')

`CondAuth` implements the ConditionalMarkup for `(:if auth level:)`. For instance `CondAuth($pagename,'edit')` is true if authorization level is 'edit'. Use inside local configuration files to build conditionals with a check of authorization level, similar to using `(:if auth level:)` on a wiki page.

Note that CondAuth() should be called after all authorization levels and passwords have been defined. For example, if you use it with Drafts, you should include the draft.php script before calling CondAuth():

```
$EnableDrafts = 1;
$DefaultPasswords['publish'] = pmcrypt('secret');
include_once("$FarmD/scripts/draft.php");
if (! CondAuth($pagename, 'edit')) { /* whatever */ }
```

Best is to use CondAuth() near the bottom of your config.php script.

`RetrieveAuthPage($pagename, $level, $authprompt=true, $since=0)`

where:

```
 $pagename   - name of page to be read
$level      - authorization level required (read/edit/auth/upload)
$authprompt - true if user should be prompted for a password if needed
$since      - how much of the page history to read
               0 == read entire page including all of history
               READPAGE_CURRENT == read page without loading history
               timestamp == read history only back through timestamp
```

The $since parameter allows PmWiki to stop reading from a page file as soon as it has whatever information is needed -- i.e., if an operation such as browsing isn't going to need the page's history, then specifying READPAGE_CURRENT can result in a much faster loading time. (This can be especially important for things such as searching and page listings.) However, if combined with UpdatePage, the updated page will have no history.

Use e.g. `$page = @RetrieveAuthPage('Main.MyPage', 'read')` to obtain a page object that contains all the information of the correspondent file in separate keys, e.g. `$page['text']` will contain a string with the current wiki markup of Main.MyPage. Use this generally in preference to the alternative function `ReadPage($pagename, $since=0)` since it respects the authorisation of the user, i.e. it checks the authorisation level before loading the page, or it can be set to do so. `ReadPage()` reads a page regardless of permission.

Passing 'ALWAYS' as the authorization level (instead of 'read', 'edit', etc.) will cause RetrieveAuthPage to always read and return the page, even if it happens to be protected by a read password.

`RetrieveAuthSection($pagename, $pagesection, $list=NULL, $auth='read')`

RetrieveAuthSection extracts a section of text from a page. If $pagesection starts with anything other than '#', it identifies the page to extract text from. Otherwise RetrieveAuthSection looks in the pages given by $list, or in `$pagename` if $list is not specified.

- The selected page is placed in the global $RASPageName variable.
- The caller is responsible for calling Qualify() as needed, i.e. if you need to control how unqualified page and variable names shall be resolved.
  - To have them act as in the original text, let`Qualify()` resolve them relative to the source page.
  - If the imported text was not meant as wikitext but as some other kind of markup that might happen to contain double pairs of square brackets, or dollar signs inside curly brackets, you probably don't want to `Qualify()` them. If you output them into wikitext, you'll probably need to `Keep()` them to prevent later stages of processing from interpreting them in context of the target page.
  - If your code produces wikitext for an auxiliary page that is meant to be included by another page higher up in the inclusion chain, and want links and variables to work as if they were in the auxiliary page, use the auxiliary page's "GroupName.PageName" as the `$pagename` argument for `Qualify()`.

Provides a way to limit the array that is returned by ReadPage, so that it only pulls the content up to a specific section marker. For example, pulling from start of page to '##blogend':

```
function FeedText($pagename, &$page, $tag) {
  $text = RetrieveAuthSection($pagename, '##blogend');
  $content = MarkupToHTML($pagename, $text);
  return "<$tag><![CDATA[$content]]></$tag>";
}
```

The '##blogend' argument says to read from the beginning of the page to just before the line containing themarker. See IncludeOtherPages for more information about the section specifications.

This version won't read text from pages that are read-protected; if you want to get text even from read-protected pages, then

```
  $text = RetrieveAuthSection($pagename, '##blogend', NULL, 'ALWAYS');
```

```
UpdatePage( $pagename, $old (page object), $new (page object));
```

More Technical Notes

`UpdatePage()` allows cookbook recipes to mimic the behavior of editing wiki pages via the browser. Internally, PmWiki does several house keeping tasks which are accessible via this function (preserving history/diff information, updating page revision numbers, updating RecentChanges pages, sending email notifications, etc._

- "Page object" refers to an array pulled from`RetrieveAuthPage($pagename, $level, $authprompt=true, $since=0);` (preferred), or `ReadPage($pagename);` (disregards page security). Note that $new['text'] should contain all page data for the new version of the page.
- If a page doesn't exist, UpdatePage() will attempt to create it.
- Ignoring $old (e.g. `UpdatePage($pagename, '', $new);`) will erase all historical page data---a*tabula rasa*.
    - If you retrieved $old using RetrieveAuthPage(`$pagename`,$auth,$prompt,READPAGE_CURRENT) and set $new=$old, then UpdatePage will also erase all historical data

`UpdatePage()` cannot be called directly from config.php because there are necessary initializations which occur later in pmwiki.php. It is not enough to just load stdconfig.php. If you want to use `UpdatePage()` you will need to do it within a custom markup, a custom markup expression, or a custom action.

Categories: PmWiki Developer

Last modified by Sven on June 24, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Functions

Glossary

toc top

This page describes various terms related to PmWiki.

Author
: Any user with privileges to write to the wiki.

Admin
: The person (or people) who controls access to the wiki, configures the wiki, and generally is the person who installed the wiki.

Configuration file
: A specially-named PHP script file where local customizations can take place for a farm, a wiki, a group, or a page.

Default configuration
: The way Pm has chosen to set all settings, or an individual setting, by default. For example, `$EnablePathInfo` is disabled by default. A wiki with no *local/config.php* file is using the default configuration. Likewise, a farm that only defines `$FarmPubDirUrl` in *farmconfig.php* is using the default configuration.

Farm
: A group of wikis that share code. Content and formats may or may not be shared. For more farm-related terms, including several which have been deprecated, see WikiFarmTerminology

Farm-wide configuration file
: A WikiFarm's *local/farmconfig.php* file, where any settings (besides `$FarmPubDirUrl`) customize the default configuration for all of the wikis in a farm.

Full page name
: The *full page name* consists of a group and a name, e.g.`Main.WikiSandbox`. The variable for the full page name is `{$FullName}`, which for this page is `PmWiki.Glossary`. Similarly, the variable for the group is`{$Group}` which here is `PmWiki`.

Group
: A collection of associated wiki pages; by default this appears in the page name as "Group.PageName". Attributes can be set on all pages in the group simultaneously. The variable for the group is `{$Group}`, which here is `PmWiki`.

Local configuration file
: A specially-named PHP script where local customizations can take place for an individual wiki. For an entire wiki it's named *local/config.php*. Individual groups and pages can also have their own local configuration files.

Local customization
: Any deviation from the default configuration. A related phrase is "farm-wide customization".

Page file name
: The *page file name* is the name of the file that normally stores the data of a page in the directory`wiki.d/`. This file name is normally built directly from the page name.

Page link
: A *page link* is something that is used to generate a link to a page. For example, the markup`[[wiki sandbox]]`,

`[[(wiki) sandbox]]`, `WikiSandbox`, `Main/WikiSandbox`, `[[Main/wiki sandbox]]`, `[[Main.WikiSandbox | click here]]`, etc all specify a link to the page 'Main.WikiSandbox'. In each case PmWiki uses the context of the link to generate a page name from the page link -- normally by capitalizing each word found in the link and stripping any characters that aren't considered valid in page names.

Page name
: The *page name* is a string that PmWiki uses to refer to a page - i.e. it *names the page*. This could also be considered a *handle* for the page. The variable for the page name is simply called `{$Name}`, which for this page is `Glossary`.
Note that there is no whitespace in page names, and by default PmWiki capitalizes each word in a page's name. There is however a variable `{$Namespaced}` where spaces have been inserted, e.g. for the page WikiSandbox this variable would be `Wiki Sandbox`.
Note that PmWiki also uses the page name to locate per-group and per-page customization files in the `local/` subdirectory. For example, browsing Main.WikiSandbox would cause `local/Main.WikiSandbox.php` and `local/Main.php` to be loaded if these files existed.

Page title
: A *page title* is the title element of a page, i.e. what is usually shown above the page and in the browser window's name. This title is normally set via the directive `(:title:)`, but if no such directive is given the title will be automatically generated from the page name. The title of a page is accessed via either the variable `{$Title}` or the variable `{$Titlespaced}`. The latter differs in that it uses the spaced version of the name.

Page URI
: Page names are used in URIs to tell PmWiki which page is to be loaded or acted upon. The normal form of a page URI is usually one of these two

```
http://www.example.com/pmwiki/pmwiki.php?n=Main.WikiSandbox
http://www.example.com/pmwiki/pmwiki.php/Main.WikiSandbox
```

Note that various aliasing and rewriting tricks can be used to modify this, but PmWiki expects to obtain a page name from the parameter 'n' or from the `PATH_INFO` component following the URI of the script (`pmwiki.php`).
Note that the parameter 'n' takes precedence over `PATH_INFO` if both are available.

Wikifarm
: Synonymous for " farm".

Last modified by simon on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Glossary

GroupCustomizations

toc top

One of the purposes of Wiki Groups is to allow a Wiki Administrator to customize the features of PmWiki on a per-group basis. Here is where *per group customizations* come into play.

- The *local/* subdirectory is used to hold local configuration files.
- The *pub/css/* subdirectory is used to hold local css files.

To perform local customizations for a particular WikiGroup,
- place the customizations in a file called "*<groupname>.php*" (where *<groupname>* is the actual name of the page group in question) in the *local/* subdirectory
- place the css customizations in a file called "*<groupname>.css*" (where *<groupname>* is the actual name of the page group in question) in the *pub/css/* subdirectory.

These files will be automatically processed after processing any local customizations in the *config.php* and *local.css* files.

For example, to change the image displayed in the upper-left corner of pages in the "GroupName" WikiGroup, one could create *local/GroupName.php* containing

```
<?php
  $PageLogoUrl = "/myimages/chess.gif";
```

The example's effect would cause all pages in the GroupName Wiki Group to use "/myimages/chess.gif" as the logo image instead of the default.

To add markup to the beginning or end of each page in a wiki group, see Group headers.

Per-page customizations

PmWiki also allows per-page customizations, simply use the full name of the page to be customized instead of the group. For example, one can use the file *local/Chess.HomePage.php* to set local customizations for Chess.HomePage.

Almost any customization that would be placed in *config.php* can be used as a per-group or per-page customization.

An important exception to this is setting **per-group or per-page customizations for recipe scripts** included in config.php. Most recipe scripts would need any customization variables defined before the script is included. Instead of using a per-group or per-page customization php file, use a per-group or per-page conditional statement in config.php, before including the recipe script. Example:

```
$page = PageVar($pagename, '$FullName');
```

```
     $group = PageVar($pagename, '$Group');
     //per-group customizations:
     if($group=='GroupName') {
        $RecipeVariable = 'valueA';
        etc. ...
     }
     //per-page customizations:
     if($page=='GroupName.PageName) {
        $RecipeVariable = 'valueB';
        etc. ...
     }
     //include recipe after variables are set:
     include_once('cookbook/recipescript.php');
```

Note that this method cannot be used to set `$DefaultPasswords`, you should use Group or Page attributes. See Passwords and PasswordsAdmin for more information.

Processing order

For all local customizations, PmWiki first processes the *local/config.php* file, and then looks for a per-page customization file in the *local/* subdirectory to process, followed by any per-group customization file. If no per-page or per-group customizations are loaded, then PmWiki loads *local/default.php*. If a per-page customization wants to have the per-group customizations loaded first, it can do so directly by using PHP's `include_once()` function. For more information see wiki cascades.

Custom CSS styles per group or per-page

To apply CSS styles to pages of a specific group named Group Name, create a file named *GroupName.css* in the *pub/css/* directory and add the CSS style rules there. To apply styles to a specific page, create a file *GroupName.PageName.css* in this directory with your style rules. Any CSS rules to be applied for all wiki pages can be put into *pub/css/local.css*.

```
    /pub/css/GroupName.css:

      body { background: #F4C4B4; }
```

Preventing group-Level configurations

Any customization file can set `$EnablePGCust`=0; to prevent later page/group/default customizations from being automatically loaded. If a per-page customization needs to have the per-group customizations loaded first, it can do so directly by using PHP's `include_once()` function.

Authentication

Any passwords required for a group should be set in the group's Group Attributes page (see Passwords Administration) and not in a group customization file.

Consider Wiki Farms

Wiki Groups are an easy way to host multiple sites in a single PmWiki installation by giving each site its own group. Another approach is to use Wiki Farms, which allows each site to have its own set of Wiki Group and local customization files. Read about

If you are looking for nested group levels, you may want to consider Pm's design considerations on hierarchical groups.

How can I apply CSS styles to a particular group or page?

> Simply create a *pub/css/Group.css* or *pub/css/Group.Page.css* file containing the custom CSS styles for that group or page. See also Cookbook:LocalCSS.

Why shouldn't passwords be set in group (or page) customization files? Why shouldn't group or page passwords be set in config.php?

> The reason for this advice is that per-group customization files are only loaded for the current page. So, if `$DefaultPasswords['read']` is set in *local/GroupA.php*, then someone could use a page in another group to view the contents of pages in GroupA. For example, Main.WikiSandbox could contain:
>
> > (:include GroupA.SomePage:)
>
> and because the *GroupA.php* file wasn't loaded (we're looking at Main.WikiSandbox -->*local/Main.php*), there's no read password set.
>
> The same is true for page customization files.

Isn't that processing order strange? Why not load per page configuration last (that is after global configuration an per group configuration)?

> Many times what we want to do is to enable a certain capability for a group of pages, but disable it on a specific page, as if it was never enabled. If the per-group config file is processed first, then it becomes very difficult/tedious for the per-page one to "undo" the effects of the per-group page. So, we load the per-page file before the per-group.
>
> If a per-page customization wants the per-group customizations to be performed first, it can use the techniques given above (using *include_once()* or setting `$EnablePGCust = 0;`).

GroupHeaders and GroupFooters

Every WikiGroup can have GroupHeader and GroupFooter pages that contain markup that should be included at the beginning or end of each page within the group. This feature is useful for:

- adding a disclaimer or heading to all of the pages of a group
- defining custom WikiStyles that may be used for all pages in a group
- replacing the default headers and/or footers for pages in a group (e.g., using`(:noheader:)` and or `(:nofooter:)` -- see PageDirectives).

To create a group header, just create a new page called`YourGroup.GroupHeader`. Group headers allow authors to create groups with custom headers and footers without having to coordinate with a wiki administrator.

The default GroupHeader or GroupFooter can be suppressed on an individual page (such as a group's HomePage) by using the `(:nogroupheader:)` and `(:nogroupfooter:)` markups *on that page*.

If a generic GroupHeader is used in one wikigroup (say, the Site wikigroup), then the code can be easily duplicated in the GroupHeader of any other group by using `(:include Site.GroupHeader:)`. See IncludeOtherPages.

If you want a header or footer to appear when you print a page (action**print**), simply create a page called `YourGroup.GroupPrintHeader` or `YourGroup.GroupPrintFooter` and fill it with your markup.

You can also set the variable `$GroupPrintHeaderFmt` and `$GroupPrintFooterFmt` in the same way as`$GroupHeaderFmt` and `GroupFooterFmt` in order to change the header used when`action=print`.

See also
- Cookbook:All group header
- Cookbook:Wiki footer

How do I set the same header or footer for all pages/groups?

> The header and footer for each page are controlled by the variables `$GroupHeaderFmt` and `$GroupFooterFmt`. If your site-wide header and footer pages are Site.SiteHeader and Site.SiteFooter, you can add this in config.php:

```
    ### If you use Site.SiteHeader and Group.GroupHeader
    $GroupHeaderFmt = '(:include {$SiteGroup}.SiteHeader'
      . ' basepage={*$FullName}:)(:nl:)' . $GroupHeaderFmt;

    ### If you use Site.SiteHeader instead of Group.GroupHeader
    $GroupHeaderFmt = '(:include {$SiteGroup}.SiteHeader'
      . ' basepage={*$FullName}:)(:nl:)';

    ### If you use Site.SiteFooter and Group.GroupFooter
    $GroupFooterFmt .= '(:nl:)(:include {$SiteGroup}.SiteFooter'
      . ' basepage={*$FullName}:)';

    ### If you use Site.SiteFooter instead of Group.GroupFooter
    $GroupFooterFmt = '(:nl:)(:include {$SiteGroup}.SiteFooter'
      . ' basepage={*$FullName}:)';
```

> Note that single quotes must be used in the lines above.
>
> See also the Cookbook:AllGroupHeader recipe.
>
> Instead of using an additional page, you could set any wiki text in `$GroupHeaderFmt`, for example:

```
$GroupHeaderFmt .= "Global message here.";
```

I18nVariables

toc top

This page describes the variables used by PmWiki for Internationalizations (i18n).

`$DefaultPageCharset`
: Fix and correctly handle some pages with missing or wrong attributes when UTF-8 is enabled.

`$EnableXLPageScriptLoad`
: This variable, if set to 0, will disable the 'xlpage-i18n' parameter in XLPage definitions and thus it will prevent editors from (accidentally) loading scripts and changing the website encoding. Note that if you use this variable, you should include the required scripts, eg. xlpage-utf-8.php, from config.php.

`$VarPagesFmt`
: An array which contains the PageNames where you can find lists (trails) of pages containing variable definitions. To be modified when documentation is not in English. See *scripts/vardoc.php*.

`$XL`
: An array (hash) which contains pairs of language identifiers and translation hashes. Each translation hash maps a given lookup key (or phrase) into a corresponding text string for the given language. Thus, it is essentially a multi-lingual dictionary used for phrase translation. It is also used for handling user preference mappings. Thus, the 'e_row' value that one finds on the Site.Preferences page is loaded into $XL during preference processing.

`$XLLangs`
: An array that contains the names of the currently active language definitions. Only dictionaries in $XL that are named in `$XLLangs` are used by the $[...] markup when performing a translation.

See also:
- `$TimeFmt`

Last modified by HaganFox on September 14, 2016.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/I18nVariables

toc top

Images

toc top

To place an image into a page, enter the address (url) of the image into the markup text. Any *alternate text* (used for tooltips and for browsers that do not display images) is placed in double quotes immediately following the image url. A caption can follow separated by a vertical bar (|), and may include simple formatting

```
http://pmichaud.com/img/misc/pc.jpg"Paper
clips" | [- %newwin% [[
Wikipedia:Paper_clips | Paper clips ]] are
''fun'' to work with. -]
```

Paper clips are *fun* to work with.

Images can also be specified as uploaded files (i.e., `Attach:image.jpeg`) and using InterMap links. By default PmWiki supports the following image types for embedding into the page:

```
gif, jpg, jpeg, png, svg, svgz
```

(See also Uploads and Notes below for image files that lack extensions.)

To create a link to an image (like http://pmichaud.com/img/misc/pc.jpg as opposed to displaying the image itself), use double brackets to mark the link, as in `[[http://pmichaud.com/img/misc/pc.jpg]]` or `[[Attach:image.jpeg]]`.

To have an image link to another location, use the image as the link text as in

```
[[http://pmwiki.org/ |
http://pmichaud.com/img/misc/pc.jpg"PmWiki"]]
```

or `[[http://example.com|Attach:Groupname./image.jpeg]]`.

Tool tips or alternate text

A tool tip, or alternate text, can be displayed when the cursor hovers over the image by placing the description in double quotes directly following the image's URL. The description cannot contain any formatting.

```
http://pmichaud.com/img/misc/pc.jpg"Coloured
paper clips"
```

Captions

A caption can be added to an image using a vertical bar and the caption text.

```
http://pmichaud.com/img/misc/pc.jpg |
'''Figure 1'''
```

Figure 1

Image alignment

Normally, images are displayed "in line" with the surrounding text.

Use `%center%` to center an image on its own paragraph,`%right%` to align it to the right.

```
Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo
consequat.

%center%http://pmichaud.com/img/misc/pc.jpg"Paper
clips"%%
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Floating images

To left or right-align an image with text wrapping around it, use the`%lfloat%` or `%rfloat%` wiki styles.

```
%lfloat text-align=center margin-top=5px
margin-right=25px margin-bottom=5px margin-
left=25px%
http://pmichaud.com/img/misc/gem.jpg |
'''Rock on!'''
'''The image is left-aligned, with margins
set; the caption is centered; the text
wraps on the right side of the image.'''

Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat. Lorem
ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna
aliqua.
```

Rock on!

The image is left-aligned, with margins set; the caption is centered; the text wraps on the right side of the image.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

The `[[<<]]` markup breaks floating text, and the text continues at the bottom of the image.

```
%lfloat%
http://pmichaud.com/img/misc/gem.jpg
'''The image is left-aligned, and the text
wraps on the right side of the image. The
text after the ''[@[[<<]]@]'' markup
continues below the image.'''
[[<<]]
```

The image is left-aligned, and the text wraps on the right side of the image. The text after the *[[<<]]* markup continues below the image.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim

```
Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat.
```

ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Use the `%lframe%` or `%rframe%` styles to float an image and place a frame around the image and its caption. The frame can be formatted via wikistyles:

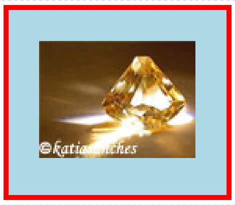
```
%rframe%
http://pmichaud.com/img/misc/gem.jpg |
'''Rock on!'''
'''The image is right-aligned, and the text
wraps on the left side of the image.'''

Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat. Lorem
ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna
aliqua. Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et
dolore magna aliqua.
```

Rock on!

The image is right-aligned, and the text wraps on the left side of the image.

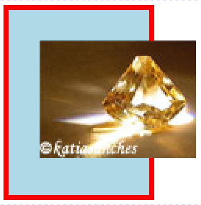
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```
%cframe width=100px bgcolor=lightblue
border='3px solid red' padding=20px%
http://pmichaud.com/img/misc/gem.jpg
```



Example to show failure to fully apply width setting:-

```
%cframe width=50px bgcolor=lightblue
border='3px solid red' padding=20px%
http://pmichaud.com/img/misc/gem.jpg
```

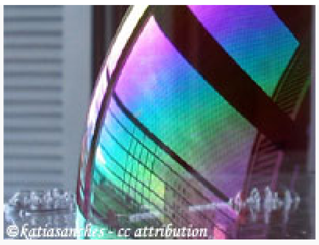


Use `%block rframe%` to set off multiple images and caption text to be stacked vertically in a right-floating frame.

```
%block rframe width=300px%http://pmichaud.com/img/misc/gem.jpg\\
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua.\\\
http://pmichaud.com/img/misc/bubble.jpg\\
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua.%%

Text subsequent to the defined block wraps to the left of the frame. Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua.
```

Text subsequent to the defined block wraps to the left of the frame. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

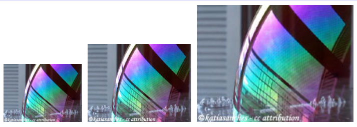
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Resizing images

To resize an image via wikistyles, use `%width=50px%` or `%height=50px%` in front of an image. The `%thumb%` wikistyle is a helpful shortcut for `%width=100px%`.

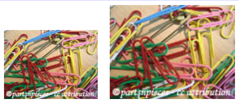
```
%width=50px% http://pmichaud.com/img/misc/bubble.jpg \
%height=50px% http://pmichaud.com/img/misc/bubble.jpg \
%thumb% http://pmichaud.com/img/misc/bubble.jpg
```



Note: Resizing an image via wikistyles only affects how it is displayed in a browser; it does not reduce the transfer size of the image itself - hence resizing via wikistyles is not an acceptable method of generating a page-full of images, or 'gallery'.

If you want a resized image within a link, you have to specify the size before the link as well as close it off with a %%.

```
%width=60%[[http://pmwiki.org/ | http://pmichaud.com/img/misc/pc.jpg"PmWiki"]]%% \
%height=60%[[http://pmwiki.org/ | http://pmichaud.com/img/misc/pc.jpg"PmWiki"]]%% \
```

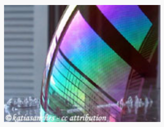


To open the link in new window, you place %newwin% before the size specification.

```
%newwin%[[http://pmwiki.org/ | http://pmichaud.com/img/misc/pc.jpg"PmWiki"]]%%
```

Resized images using `%thumb%` can also be floated with frames, as well as made into links.

```
%lframe thumb% [[http://pmichaud.com/img/misc/bubble.jpg | http://pmichaud.com/img/misc/bubble.jpg"Burst
the bubble"]] | [-Bubble-]
%lframe thumb% http://pmichaud.com/img/misc/pc.jpg"Clip the ticket" | [-Paper Clips-]
%lframe thumb% [[DocumentationIndex | http://pmichaud.com/img/misc/gem.jpg"Visit the Documentation
Index"]] | [[DocumentationIndex | [-Rock On-]]]
```


Bubble

Paper Clips

Rock On

Images as links

To use an image as a link specify an image instead of text in the link markup.

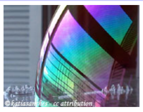
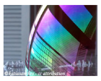
```
[[PmWiki/Links | http://pmichaud.com/img/2003/atc-1.gif"Information about wiki links"]]
```

Notes

- **An image file that lacks a correct extension** can be displayed by addition of a "false" extension to the URL. For example, if the url is `http://example.com/script/tux`, add a fake query string on the end with the desired extension (e.g., `http://example.com/script/tux?format=.png`). If query strings are unsuitable, a fragment identifier should work, e.g. `http://example.com/script/tux#file.png`.

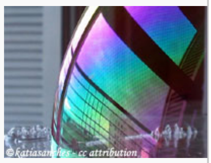
- Relative width is possible by using percentages.

```
%width=10pct% http://pmichaud.com/img/misc/bubble.jpg \
%height=30pct% http://pmichaud.com/img/misc/bubble.jpg
```

- **Flowing text is possible, like captions, within a frame**

```
>>lframe width=130px<<
%thumb width=130% [[http://pmichaud.com/img/misc/bubble.jpg |
http://pmichaud.com/img/misc/bubble.jpg"Burst the bubble"]] | [--Long caption for an image like
[[http://pmichaud.com/img/misc/bubble.jpg | the bubble]]. This is just to show some text flowing within
the frame.--]
>><<
```


Long caption for an image like the bubble. This is just to show some text flowing within the frame.

See also

- Cookbook:Images - adding image galleries, automatic thumbnails, background images and more.

Credits

The images on this page were obtained from http://flickr.com and are redistributed under a Creative Commons License.

Is it possible to link an image on PmWiki without using a fully qualified URL?

Yes. For images that are attachments, the general format is `Attach:Groupname./image.gif`. To link to an image that is on the same server, use `Path:/path/to/image.gif`.

Can I attach a client image file on PmWiki?

Yes, see Uploads .

How can I include a page from another group that contains an attached image?

Include the page in the normal way, ie `(:include GroupName.Pagename:)`. In the page to be included (that contains the image) change `Attach:filename.ext` to `Attach:{$Group}./filename.ext`.

Why, if I put an image with rframe or rfloat and immediatly after that I open a new page section with ! the section title row is below the image instead of on the left side?

Because the CSS for **headings** such as ! contains an element **clear:both** which forces this behaviour. Redefine the CSS locally if you want to stop this happening, but I think the bottom border (that underlines the heading) would need further re-definition. I just use bolding for the title, and 4 dashes below ---- to separate a new section, and it saves the effort of fiddling with the core definitions.

Unlike the **lframe** and **rframe** directives, **cframe** does not fully honour the width setting. While the frame itself resizes to match the request, the enclosed image does not, and retains its original width. Effect is the same in IE and Fx. I've added an example beneath the standard example above.

Is it possible to disallow all images? I already disabled uploads but I also want to disallow external images from being shown on my wiki pages.

Yes, add to config.php:

```
    DisableMarkup('img');
    $ImgExtPattern = "$^";
```

How can I make it so that when I place an image in a page, the block of text it is in is a <p> (paragraph) rather than a <div> (division)?

If you just want it to happen for a single image (instead of all), then try putting `[==]` at the beginning of the line, as in:

```
[==] http://www.pmwiki.org/pub/pmwiki/pmwiki-32.gif
```

Having `[==]` at the beginning of a line forces whatever follows to be part of a paragraph.

Is there any way to use relative paths for images?

See Cookbook:RelativeLinks and `$EnableLinkPageRelative`.

Is there a way to attach a BMP and have it display rather than link?

Add to config.php the following line:
`$ImgExtPattern = "\\.(?:gif|jpg|jpeg|png|bmp|GIF|JPG|JPEG|PNG|BMP)";`
Note that BMP images are uncompressed and quite heavy. You may wish to convert them to PNG (lossless) or JPG (lossy) format, and thus reduce 5-20 times their filesizes.

Is there a way to have a table to the left or right of an image?

Yes, see TableAndImage.

Last modified by Petko on June 24, 2016. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Images

IncludeOtherPages

toc top

The `(:include:)` directive makes it possible to insert (or "transclude") the contents of other pages into the current wiki page. All of the include directives below perform a straight text inclusion. In particular, any page links in the included text are assumed to link to pages in the current group if not otherwise qualified.

Syntax

The basic syntax is

- (:include PageName:)
  with pagename includes the full page from the same group.
- {Group/PageName$: PTVar}:
  includes a named variable from a page, Group and PageName are options

The full syntax is

- (:include FullName# fromanchor# toanchor lines=*12..34* self=*0* basepage=*abc* variable=*value* :)
  includes a page according to the parameters supplied. Parameters are optional.

Parameters

The directive can have multiple Name parameters with or without anchors, and multiple template variable parameters.

Named pages

```
(:include PageName:)
(:include Group.PageName:)
(:include Page1 Page2 Group1.Page3 Group2.Page4:)
```

Includes the entire text of another page into the current page. Multiple pages may be specified, but only the first available is included.

You can use the above feature to *display an error message if an include fails*. Create a page, eg. Site.IncludeFailed containing the error message. You can use any page name. Then, in your include markup, append this page at the end of the page list:

```
 (:include Page1 Page2 Page3 Site.IncludeFailed:)
```

A slightly more complex approach is outlined at the talk page.

#From#To anchors

`(:include PageName#from#to:)`	include lines from *PageName* between the `[[#from]]` and `[[#to]]` anchors
`(:include PageName#from#:)`	include all lines after `[[#from]]` to the end of the page
`(:include PageName##to:)`	include all lines from the start of the page to `[[#to]]`
`(:include PageName#from:)`	include everything between `[[#from]]` and the next anchor
`(:include PageName#:)`	include everything from the top of the page to the first anchor

Note: do not put whitespace between "#from" "#to"

Note: text on the same line as a closing anchor but preceding the closing anchor will **NOT** be included in the text. Example Below:

```
[[#start]]some text on the first line
some text on the last line [[#end]]
```

The above, when included via `(:include PageName#start:)` will have the text on the first line but not the text on the last line.

`(:include Page1 Page2 #from#to:)`
: Include from the first available of Page1, Page2 between the `[[#from]]` and `[[#to]]`

Note: put whitespace between "Page2" and "#from#to". The same anchors "#from#to" should be in both pages. If proper anchors are missing in the first available of Page1, Page2 the whole contents of the page is included.
This does not seem to work in 2.2 betas. See Cookbook:IncludeSection for a fix.

`(:include Page1#from1#to1 Page2#from2#to2:)`
: Include the first from the first available of Page1 (between the `[[#from1]]` and `[[#to1]]`) or Page2 (between the `[[#from2]]` and `[[#to2]]`)

Note: Previous versions of PmWiki allowed whitespace between `#from` and `#to` anchors even though it was not designed to. Newer versions do not allow whitespace anymore. To re-enable this "exploited misbehavior" put this into your config.php or farmconfig.php

```
    Markup('includeanchors', '<include', '/(\\(:include.*?#\\w+)\\s+(#\\w+)/', '$1$2');
```

Lines=from..to

`(:include PageName lines=10:)` `(:include PageName lines=5..10:)` `(:include PageName lines=5..:)`
: Include the first 10 lines, lines 5-10, or lines 5 and up from *PageName*. A "line" in this context refers to a line of source. *Thus a line may be a paragraph that wraps over several lines on the screen, or a completely blank line.*

`(:include Page1 Page2 Page3 lines=1..5:)`
: Include the first five lines from the first available of Page1, Page2, or Page3. (To include lines from a list of pages, use a separate include for each.)

Self=

`(:include PageName self=0:)`
: The parameter `self` can be `0` or `1`. It tells the include directive if it is allowed to include the current page. This is useful if PageName is a variable like `{$Name}` and you want to prevent the directive from including the current page.

Page text variables

`{Group/PageName$:Var}`
: Includes definition list values from an (optional) page as page text variables. These are defined using a definition list (`:item:description`), simple colon delimiter (`item:description`), or special markup (`(:item:description:)`).

Basepage=

`(:include PageName basepage=BasePageName:)`
: Include PageName, but treat all relative links and page variables on *PageName*` as relative to *BasePageName*.

If `basepage=` is provided all relative links and page variables are interpreted relative to basepage. So, if one creates `TemplateName` as

```
    Name: {$:Name}
    Address: {$:Address}
```

then the directive

```
    (:include TemplateName basepage=PageName:)
```

will retrieve the contents of `TemplateName`, treating any page variables and links as being relative to `PageName`. In particular, the values for `{$:Name}` and `{$:Address}` will be taken from `PageName`, but things like `{$Title}` and `{$LastModifiedBy}` would also work here.

Basepage usage

The primary purpose of basepage is to allow the inclusion of pages in a way that mimics the 2.1.x behavior where page variables and links are interpreted relative to the currently displayed page. This is done with:

```
     (:include SomeOtherPage basepage='' :)
  -or-
     (:include SomeOtherPage basepage={*$FullName} :)
```

It also allows GroupHeader and GroupFooter to have their page variables and links be relative to the currently displayed page (instead of GroupHeader and GroupFooter):

```
  ## PmWiki default  $GroupHeaderFmt setting
   $GroupHeaderFmt =
     '(:include {$Group}.GroupHeader self=0 basepage={*$FullName}:)(:nl:)';
```

Otherwise, using IncludeOtherPages inside of a GroupHeader would display 'GroupHeader' and not the name of the currently displayed page.

The basepage= parameter is general enough that it can also be used as a templating engine, so that we can grab a template page containing variables that are then filled in with values from another page:

```
     (:include TemplatePage basepage=DataPage :)
```

And, of course, a single TemplatePage can actually contain multiple templates delimited by anchors, so that we end up with a syntax eerily similar [1] to pagelist-templates:

```
    (:include TemplatePage#abc basepage=DataPage :)
```

So then TemplatePage can use a syntax like:

```
    [[#abc]]
    ...template stuff here...
    [[#abcend]]
```

and it's possible to display TemplatePage as a template without it being interpreted... same as we do for Site.PageListTemplates.

[1]Okay, maybe it's not so eerie, given that the pagelist template code actually uses the same function as (:include:) to grab its templates. But it's still a useful parallel.

Specifying variables as parameters: Use sections as templates

You can also specify variable values inline with the include statement, and refer to the variables in the template using the `{$$variable1}` format:

```
     (:include TemplatePage variable1="value" variable2="value2":)
```

This assumes that a site has `$EnableRelativePageVars` enabled, which is recommended in PmWiki 2.2.0 -- but was disabled by default in version 2.2.8 and earlier.

For example, on my included page ("template") I might have this:

`[[#ivars]]`
`Hi, {$$Name}, how are you today?`
`[[#ivarsend]]`
	Hi, {$$Name}, how are you today?

Then, including that section above (that section is available via the section `{$FullName}#ivars`) you get this type of behavior:

`(:include {$FullName}#ivars Name=Sam:)`	Hi, Sam, how are you today?

If a value contains spaces, quote it:

`(:include {$FullName}#ivars Name="my friend":)`	Hi, my friend, how are you today?

See also `$EnableUndefinedTemplateVars`.

Specific markup

`(:nl:)` acts like a new line in the*markup*, only if there isn't one already.

The purpose of `(:nl:)` is to be able to write things like "`(:include Page1:)(:nl:)(:include Page2:)`" which guarantees that the first line of Page2 is treated as a separate line from the last line of Page1, but without inadvertently generating a blank line between them.

See this thread and this thread for more info.

`(:nl:)` is not intended to put a newline character in the output!

See Also

- Page text variables Page variables automatically made available through natural or explicit page markup
- Cookbook:IncludeUrl

Styling Note

By default, Included pages or lines cannot be distinguished from other text on the page. To provide a visual indication that this text is special, you can apply Wiki Styles. For example:

```
%define=leftborder border-left="2px solid #88f" margin-left="2px" padding="1px 0 3px 10px"%
What is PmWiki?
>>leftborder<< (:include PmWiki.PmWiki lines=1..4:)
>><<
''Have a very nice day!''
```

What is PmWiki?

> PmWiki is a wiki-based system for collaborative creation and maintenance of websites.
>
> PmWiki pages look and act like normal web pages, except they have an "Edit" link that makes it easy to modify existing pages and add new pages into the website, using basic editing rules. You do not need to know or use any HTML or CSS. Page editing can be left open to the public or restricted to small groups of authors.

Have a very nice day!

Parameter References

Any parameters supplied to an include statement (whether they are keywords or not) are accessible inside the included page as a special `{$$...}` variable of the same name. This feature can be used to provide extra information to use when displaying the included page.

Notes

- You can also say `(:include My/Page#myanchor lines=4:)` which starts from, and includes, the line with the anchor `[[#myanchor]]` for four lines.

Notes about use with conditional markup

The `(:include ...:)` markup is processed after conditional markup is evaluated.
Therefor you can include a page or page section as part of a condition, like

```
    (:if some condition:)(:include SomePage#section:)(:if:)
```

But `(:include SomePage#section:)` doesn't look to see if `[[#section]]` is part of a conditional, like

```
    (:if some condition:)[[#section]]...[[#sectionend]](:ifend:)
```

`(:include SomePage#section:)` will ignore such a condition.

When testing variables in included pages the context of the page (source or target) can be useful. Seespecial references for details.

What's the maximum number of includes that can exist in a page?

> My site seems to stop including after 48 includes. (`$MaxIncludes`)
>
> By default, PmWiki places a limit of 50 include directives for any given page, to prevent runaway infinite loops and other situations that might eat up server resources. (Two of these are GroupHeader and GroupFooter.) The limit can be modified by the wiki administrator via the `$MaxIncludes` variable.

Is there any way to include from a group of pages without specifying by exact name, e.g. between Anchor X and Y from all pages named IFClass-* ?

> This can be achieved using page lists.

There appears to be a viewing issue when the included page contains the (:title:) directive.

> In a default installation, the *last* title in the page overrides previous ones so you can place your (:title :) directive at the bottom of the page, after any includes. See also `$EnablePageTitlePriority`.

How to test to see if the page is part of another page?

```
(:if ! name {$FullName}:)
%comment% name of this page is not the same as the page this text was sourced from
->[[{$FullName}#anchor | more ...]]
(:ifend:)
```

Last modified by Sven on June 25, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/IncludeOtherPages

InitialSetupTasks

toc top

Once you have PmWiki running on your site (see Installation), you can customize it for your particular needs.

Most PmWiki configuration is performed in files called *local/config.php* and *pub/css/local.css*. Some configuration is done on special pages in the Site and SiteAdmin groups, such as the Site.SideBar menu.

The local configuration file (*local/config.php*)

When you first install PmWiki, the *local/config.php* file does not exist. Copy the *sample-config.php* file (in the *docs/* directory) to *local/config.php* and use it as a starting point. You could create *local/config.php* from scratch, but *sample-config.php* is already populated with many of the options you might want to adjust.

Here is a simple *config.php* file:

```
<?php if (!defined('PmWiki')) exit();
$WikiTitle = "My New Wiki";
$PageLogoUrl = "http://example.com/mylogo.gif";

# Uncomment these if needed
#$ScriptUrl = 'http://example.com/pmwiki/pmwiki.php';
#$PubDirUrl = 'http://example.com/pmwiki/pub';

$DefaultPasswords['admin'] = pmcrypt('onesecret');

$EnableUpload = 1;
$DefaultPasswords['upload'] = pmcrypt('secrettwo');

# Uncomment and change these if needed
# putenv("TZ=EST5EDT"); # if you run PHP 5.0 or older
# date_default_timezone_set('America/New_York'); # if you run PHP 5.1 or newer

$TimeFmt = '%B %d, %Y, at %I:%M %p EST';
```

Note that *config.php* begins with `<?php` . The PHP end tag `?>` is optional, and can be left off to improve compatibility with some operating systems. Be sure there aren't any blank lines or spaces before the `<?php` or after any closing `?>`, or else you may get strange PHP error messages at the beginning of your wiki pages.

The *config.php* file above sets the value of PHP variables used by PmWiki:

- The `$WikiTitle` variable gives the name of your site as it will appear in a user's browser title bar.
- The `$ScriptUrl` and `$PubDirUrl` variables tell your wiki where it is located. Often pmwiki can guess, but if you have difficulty with links not working or skins not being found then try uncommenting these lines.
- The `$PageLogoUrl` variable specifies the URL of the icon image that will appear in the upper-left corner of each wiki page.
- The `$DefaultPasswords['admin']` sets an administrative password.
- Setting `$EnableUpload` to "1" enables Uploads ("Attached files"). `$DefaultPasswords['upload']` sets an upload password.
- The TZ environment variable defines a particular time zone (see Cookbook:ChangeTimeFormat). If your site runs on PHP 5.1 or newer, it is recommended to use the function date_default_timezone_set, see below.
- The date_default_timezone_set tells PHP what the default time zone is. For other ways to set the time zone, and a list of identifiers, see the online PHP manual.
- The `$TimeFmt` variable defines the appearance of time strings and (along with TZ) localizes the wiki to a specific time zone (see Cookbook:ChangeTimeFormat).

By setting these (and other) variables in *local/config.php*, you can change the look and feel of PmWiki from its default, sometimes substantially so. See PmWiki.Variables for a list of variables that PmWiki uses, and see PmWiki:PmWikiUsers for examples of sites that use PmWiki in customized ways.

Other common setup tasks

The following variables are often requested when preparing a new wiki

- Author required when editing a page `$EnablePostAuthorRequired = 1;`
- Set the `$DefaultGroup`

These common Cookbook recipes are also often installed immediately

- Clean Urls - Remove the `?n=Group.Page` arguments from the end of URLs

If you prepare an international wiki, potentially with characters in different alphabets (Cyrillic, Greek, Chinese) or many diacritical symbols (Czech + French), please look at PmWiki.UTF-8 and Cookbook:UTF-8.

Security

Review and set up any security required.

Setting an administrative password

The pages in the Site group (except the Site.SideBar) are locked by default. In order to edit pages in this group you need to create a site-wide admin password in *local/config.php*. To set the site-wide admin password to "`mysecret`", change the line to the following:

```
$DefaultPasswords['admin'] = pmcrypt('mysecret');
```

You must use the `pmcrypt()` function, but set the password to a value with meaning for you. SeePasswordsAdmin for details about making the password more secure.

Don't modify or rename *pmwiki.php*

PmWiki has been designed so that all customizations can be made without changing the distribution files -- one of its design goals is to provide seamless upgrades. PmWiki never writes to files in the *local/* or *cookbook/* directories, so placing your customizations here makes it easier to track the changes and upgrade PmWiki without losing the changes.

When changing the configuration of your site, always change the*local/config.php* file or add files to the*cookbook/* or *pub* directories. Do not change *pmwiki.php* or the files in the*scripts/* directory because the files are supposed to be overwritten upon upgrading.

You shouldn't rename*pmwiki.php* either. If you rename the file it will not be overwritten during an upgrade of the software and there will be a version mismatch. Many administrators add an *index.php* "wrapper script" in the pmwiki directory that contains the following single line:

```
<?php include('pmwiki.php');
```

Just make an text-file. Paste <?php include('pmwiki.php'); into it. Save the file as index.php Send it via FTP to the same directory as pmwiki.php is located.

Other organisation

Upload directories

By default Pmwiki uses an upload directory for each group (see Uploads administration. Deciding on accepting the default, or choosing an alternative (eg one directory for the entire wiki, or one directory per page) is best done when setting up your wiki.

Page store directories

By default Pmwiki uses a single page store directory (wiki.d). Deciding on accepting the default (recommended), or choosing the alternative (one directory per group) is best done when setting up your wiki. [1]

Other customization

After setting up *local/config.php* file, you may wish to make other local customizations. See the PmWiki Cookbook for a large number of customizations that have been contributed. And don't fear Cookbook recipes - they're well prepared, so that most of them require only to download a single file, add a one-line include command to `config.php`, and *voilà!* - they're working!

If you (or others sharing your server) want to maintain more than one wiki on the same server, seeWiki Farms.

Now what?

Don't forget to join a PmWiki mailing list, where you can email other wiki administrators for help on customizing PmWiki and participate in discussions about PmWiki improvements. Once you have your site operational, be sure to add it to PmWiki:PmWikiUsers so others will know about it!

Last modified by Petko on April 26, 2015.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/InitialSetupTasks

PmWiki Installation

This page explains how to download and install PmWiki 2.1 and 2.2. Here's a list of related pages:

- Requirements - Pre-requisites for running the PmWiki wiki engine
- Upgrades - How to upgrade an existing PmWiki installation
- Wiki Farms - Running multiple wikis from a single installation
- Change Log - Log of changes made to PmWiki by Release

Improvements to these instructions are always appreciated. Just report any problems you encounter to the pmwiki-users mailing list or use the PmWiki Issue Tracking System.

Installing PmWiki

If you upgrade, please read the page Upgrades and Release notes.

1. Download

Download the latest *stable* version of PmWiki as a

- zip archive (pmwiki-latest.zip), or a
- gzipped tarball (pmwiki-latest.tgz) from http://pmwiki.org/pub/pmwiki/, or

Download the latest *beta* version from the PmWiki:Subversion page.

2. Unpack

Unpack the archive (`tar zxvf pmwiki-latest.tgz` or `unzip pmwiki-latest.zip`). This will create a *pmwiki-x.y.z* directory containing the PmWiki software. For example, the current "latest" should unpack to a directory named *pmwiki-2.2.99.* The files in this directory include:

```
README.txt        An introductory document
pmwiki.php        The main PmWiki script
local/            Configuration scripts (local configuration files)
cookbook/         Recipes (add-ons) from the Cookbook
docs/             Brief documentation, sample configuration scripts
pub/              Publicly accessible files
pub/css/          Extra CSS stylesheet files [1]
pub/guiedit/
pub/skins/        Layout templates for custom look and feel
scripts/          Scripts that are part of PmWiki
wikilib.d/        Bundled default PmWiki pages
```

The *pmwiki-x.y.z* directory needs to be placed into a location accessible by your webserver (e.g., in a *public_html* directory of some sort). You can place files and directories using a number of methods -- FTP, or a Unix `mv` or `cp` command generally does the job.

> Note: It is recommended to change the "*pmwiki-x.y.z*" directory name to be simply "*pmwiki*" or just "*wiki*".

3. Create directories

In most cases PmWiki will do this for you. Open a web browser to the *pmwiki.php* script on the server (i.e., not the one on your local computer or accessed using a file://... URL). PmWiki will then analyze your system configuration and provide instructions (if needed) for creating the *wiki.d/* directory which will be used to hold the pages created for your site.

Otherwise, there are two ways to achieve this. (Use Filezilla or WinSCP to change FTP file/folder permissions.)

> 3a. You can create the *wiki.d/* directory manually, and then give it full write permissions (i.e., "`chmod 777 wiki.d`"). Use this method when "safe mode" is activated in the server's PHP installation.
>
> 3b. On some systems you can let PmWiki create *wiki.d/* by temporarily changing the permissions on the directory containing the *pmwiki.php* file to 2777. In Unix this is commonly done by changing to the directory containing *pmwiki.php* and executing the command
>
> ```
> chmod 2777 .
> ```
>
> (note the dot at the end). The `chmod` command also works in many FTP programs. Creating *wiki.d/* in this manner will (1) make the directory writable so the web server can create the data directory it needs for the wiki files, (2) preserve group ownership of the directory so the installer account can manipulate the files created in this directory, and (3) make it more difficult for other accounts on the same server to access the files in *wiki.d/*.

After establishing directory permissions, try opening a browser to the *pmwiki.php* script again. If all is well, the *wiki.d* directory will have been created and you'll see the default home page.

Important: If you used method 3b, you should reset permissions by executing "`chmod 755 .`" in the directory containing pmwiki.php.

See also FilePermissions.

4. Initialize

Check out Initial Setup Tasks for other tasks you may want to perform to begin customizing your PmWiki installation. You might also want to peruse the Release Notes for further information.

5. Set language

If you want to use PmWiki in a different language download the international language pack as zip archive (i18n-all.zip) from http://pmwiki.org/pub/pmwiki/i18n/. Then extract it and copy the files into the *wikilib.d/* directory as described above. Besides the *-all* file you can also download your country localization file only.

Languages available are:

There are two directories in the decompressed i18n archive, *scripts* and *wikilib.d*. Copy the files respectively contained in these directories to the *scripts* and *wikilib.d* of your PmWiki directory. For example, for French localization, PmWikiFr.* and PmWiki.* must be contained in the **same** directory.

Then, enable localization by adding an instruction to local/config.php to load the language translation page of your choice. For instance, `XLPage('fr','PmWikiFr.XLPage');` loads the French language page PmWikiFr.XLPage.

Read more about this on Internationalizations.

Notes

- The PmWiki distribution deliberately doesn't include an *index.php* file. You can easily add your own "wrapper script" in the same directory as *pmwiki.php*. Create a new file called *index.php* with the following single line of text (missing a closing "`?>`" tag deliberately):

```
        <?php include_once('pmwiki.php');
```

Resist the temptation to rename *pmwiki.php* to *index.php* because if you rename the file it will not be overwritten during an upgrade.

- If using the Unix *tar* command to unpack the archive in step 2 above, be sure that the files are created with sufficient permissions for the webserver to be able to access them. Usually you can ensure this by typing `umask 002` on the command line before unpacking the tar file.

- When installing on Windows you should take a look at Cookbook:SimultaneousEdits to enable simultaneous edits on that platform.

- Additional tips can be found at Troubleshooting.

See also:
- Cookbook:PHP
- Cookbook:InstallOnIIS

Should I rename pmwiki.php to index.php?

Renaming pmwiki.php is not recommended. Instead, create an *index.php* file that contains this single line

```
<?php include_once('pmwiki.php');
```

How do I make pmwiki.php the default page for a website?

Create an *index.php* file that runs PmWiki from a subdirectory (*pmwiki/* for example) and place it in the site's web document root (the main directory for the website).

```
<?php chdir('pmwiki'); include_once('pmwiki.php');
```

Note: You will also need to explicitly set the `$PubDirUrl` variable (e.g. to `"http://example.com/pmwiki/pub"`) in *local/config.php* .

How do I enable "Clean URLs" that are shorter and look like paths to my wiki pages? Why does pmwiki.org appear to have a directory structure rather than "?n=pagename" in URLs?

See Cookbook:CleanUrls.

How can I run PmWiki on a standalone (offline, portable) machine ?

See Cookbook:Standalone or Cookbook:WikiOnAStick.

InterMap

The InterMap (also called InterWiki in some other wikis) is a system for defining links between WikiWikiWeb sites that was first developed by UseMod and Meatball (see UseMod:InterWiki and Meatball:InterWiki). The method is to use a word shortcut that stands for a defined path. InterMap links have the form `MapPrefix:PagePath`, where the host prefix is converted to a partial URL based on entries in the site's *intermap.txt* and *localmap.txt* files.

The default intermap.txt

The default *intermap.txt* distributed with PmWiki (in the *scripts/* directory) includes the following InterMap entries:

```
PmWiki:      http://www.pmwiki.org/wiki/PmWiki/
Cookbook:    http://www.pmwiki.org/wiki/Cookbook/
Wiki:        http://www.c2.com/cgi/wiki?
UseMod:      http://www.usemod.com/cgi-bin/wiki.pl?
Meatball:    http://www.usemod.com/cgi-bin/mb.pl?
Wikipedia:   http://en.wikipedia.org/wiki/
PITS:        http://www.pmwiki.org/wiki/PITS/
PmL10n:      http://www.pmwiki.org/wiki/Localization/
Path:
```

The page Site.InterMap

Site.InterMap includes the following entries:

Thus, " PmWiki:Variables" becomes " http://www.pmwiki.org/wiki/PmWiki/" + "Variables", a link to the PmWiki.Variables page on the official PmWiki web site, Wiki:FrontPage is a link to the front page of the first WikiWikiWeb, and Wikipedia:Stonehenge takes you to the Wikipedia article about the famous megaliths in England.

Usage in a wiki page

Like other links, you can use the double-bracket syntax to get different link text:

`* [[Meatball:StartingPoints | starting points]] over at Meatball`
`* [[starting points -> Meatball:StartingPoints]] over at Meatball`
	• starting points over at Meatball
	• starting points over at Meatball

If you want to link just to what the intermap says (e.g. http://www.wikipedia.com/wiki/ for Wikipedia), then do `[[Wikipedia:. | Wikipedia's main page]]`, which produces Wikipedia's main page. Note the . (period) after the Map: reference.

The special `Path:` InterMap entry can be used to create "relative urls" in links.

Custom InterMap prefixes

The actual set of InterMap links at any site is defined by the site administrator via the Site.InterMap page and the *local/localmap.txt* file.

An intermap entry takes the following format:

```
MapPrefix:      http://example.com/partial/url/
```

The InterMap entry can be for any of the link schemes supported by PmWiki.
You can create your own InterMap links by doing one or more of the following:

- Modify the page called *Site.InterMap* and place entries like the ones above in it.
- Create a file called *local/localmap.txt* and place entries like the ones above in it.
- In a WikiFarm installation you can create a file called *local/farmmap.txt* and there place entries like the ones above in it. These prefixes will be common to all the wikis in the farm.
- Ensure that there is a space after the colon.

Do not edit the file *scripts/intermap.txt* directly! If you do, you'll lose your changes when you upgrade PmWiki.

Variables and InterMap links

It's possible to use variables within your InterMap entries. The following entries create `ThisWiki:` and `ThisPage:` shortcuts:

```
    ThisWiki:       $ScriptUrl
    ThisPage:       {$PageUrl}
```

You can also define InterMap entries where the text of the entry is substituted into the middle of the URL. Just include '$1' in the URL where you want the substitution to take place. For example:

```
    Jargon: http://catb.org/~esr/jargon/html/$1.html
```

would cause `Jargon:F/feature-creep` to be converted to `http://catb.org/~esr/jargon/html/F/feature-creep.html`.

Tips and tricks

It is possible to document your intermap prefixes directly in the page Site.InterMap. The extra text will not cause a performance penalty, nor will it break the definition of prefixes. However, be aware that anything matching a line starting with a word and a colon (**:**) will be considered to define a prefix.

The order in which various sources are checked for definitions of prefixes is controlled by the variable `$InterMapFiles`. Currently the precedence (highest to lowest is as follows):

- *local/localmap.txt*
- *`$SiteGroup`.InterMap*
- *`$FarmD`/local/farmmap.txt*
- *`$FarmD`/scripts/intermap.txt*

Are InterMap names case sensitive?

Yes, thus `eAdmin:` is a different InterMap link than `EAdmin:`.

How can I achieve a *localmap.txt* mapping with the effect of `Pics: Path:/somepathto/pics/`?

Use the following:
`Pics: /somepathto/pics/`

How can I define an InterMap in PHP?

Use the following:

```
            $LinkFunctions['PmWikiHome:'] = 'LinkIMap';
            $IMap['PmWikiHome:'] = 'http://pmwiki.org/wiki/$1';
```

Last modified by mfwolff on May 06, 2016.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/InterMap

toc top

Internationalizations

toc top

PmWiki supports internationalization (internationalisation) of web pages, allowing accented characters to appear in page names and almost complete customization of PmWiki's prompts. Most customization is provided via the XLPage() function in PmWiki, which loads a set of translation variables from a wiki page (typically named XLPage, but it can be named anything you wish).

The rest of this page is devoted to the installation, configuration and usage of other language(s) support. If you are looking for tools and help to *localize* PmWiki in your language, or how you can improve the existing translations, start on pmwiki.org with the page Localization - The Translation Portal.

Loading Translation Pages

Pages for many other languages have already been created and maintained at the pmichaud.com site. You can download an archive of these translations from http://www.pmwiki.org/pub/pmwiki/i18n/ . Simply download the appropriate language archive(s), and unpack the archive(s) into the directory containing your *pmwiki.php* installation. Each archive contains a number of page files that are placed in your *wikilib.d/* directory, and some special scripts for translations that use a character set other than iso-8859-1 (PmWiki's default). You can also use UTF-8 charset.

Once the translation pages are installed, you enable a language by adding a call to XLPage() in your *config.php* file. For example, to select French language prompts, one would specify

```
    include_once("scripts/xlpage-utf-8.php"); # optional
    XLPage('fr','PmWikiFr.XLPage');
```

which says to load the translations for French ('fr') from the page PmWikiFr.XLPage. The include_once line is recommended if you start a new wiki, and it should be placed *before* the XLPage line (for languages with alphabets other than the Latin, the include_once line is required). These lines should be placed near the beginning of config.php, *but after any `$WikiDir` and `$WikiLibDirs` setting (if you have such setting)*.

It's possible to load multiple pages; so if you want to create your own local translations without changing the ones you got from an i18n archive, just create another page (see below) and load it first. Be sure that you load first the page with your local changes:

```
XLPage('fr','PmWikiFr.XLPageLocal');  # my local translations
XLPage('fr','PmWikiFr.XLPage');       # from i18n.tgz
```

If your intention is to offer multiple languages on your site, and use Wiki Groups as language selectors, you may want to place this code in local customizations files (see Group Customizations). For example, if your site is published in French and English, and the French pages are in a group called Fr, you could create a file named Fr.php in the *local/* directory which contains:

```
<?php if (!defined('PmWiki')) exit();
##change to French language
XLPage('fr','PmWikiFr.XLPage');
```

You may wish to create a page called*PmwikiFr.php* with the same content to access the French documentation in the PmwikiFr group. *En.php* is not necessary in this case since English is the default language.

An alternative to the above would be to add to*config.php* the following, which tests if there is an XLPage in a group, and if it finds one it gets loaded (any language):

```
$xlpage = FmtPageName('$Group.XLPage', $pagename);
if (PageExists($xlpage)) XLPage($xlpage, $xlpage);
```

With this method you would need to copy any relevant XLPage into any group which needs the different language support.

Another way (advanced) would be insert into config.php this script, it asks to the web server the headers received from the user's browser and select a language; example with spanish and english:

```
$lang = substr($_SERVER['HTTP_ACCEPT_LANGUAGE'], 0, 2);
switch ($lang){
   case "es":
      XLPage('es','PmWikiEs.XLPage');
      break;
   case "en":
      XLPage('en','PmWikiEn.XLPage');
      break;
   default:
      XLPage('en','PmWikiEn.XLPage');
      break;
}
```

See also
- Cookbook:MultiLanguage
- Cookbook:MultiLanguageViews

Creating New Translations

If language pages don't exist for your desired language, it's easy to create one! An XLPage translation file simply contains lines of the form

```
'phrase' => 'translated phrase',
```

where "phrase" is an internationalized phrase (denoted by `$[phrase]`) in PmWiki's $...Fmt variables, and "translated phrase" is what should be printed in your particular language. For example, the line (in `PmWikiFr.XLPage`)

```
'Search' => 'Rechercher',
```

converts "`$[Search]`" to "Rechercher" on output. The file Localization:XLPageTemplate is a good starting point for creating a new XLPage and has most of PmWiki's key phrases already listed in it.

If you create new versions of PmWiki pages in other languages, please consider adding them to themain PmWiki site so that they can be made available to others in the i18n archives! (Be sure to check out The Localization Portal for further information on effectively internationalizing PmWiki.)

> The term "i18n" is commonly used as an abbreviation for the English word "internationalization". The abbreviation is derived from the fact that there are 18 letters between the "i" and the final "n" and few people want to type them all out.

Enabling "Special" Characters in WikiLinks

To enable "special" characters like for example German umlauts in WikiLinks, it is necessary to configure the server locale to ensure that PmWiki uses the proper character set definition.

If this is not possible due to limited access to the server configuration, PmWiki can be configured to use a specific locale by using the XLPage options (see XLPageTemplate).

For German umlauts, you'd need for example:

- `'Locale' => 'deu',` <- for Windows servers, see MSDN List of locale identifiers
- `'Locale' => 'de_DE',` <- for Linux servers; for the UTF-8 encoding, on some installations you may need to set `'de_DE.utf8'` or `'de_DE.UTF-8'`.

Note that the locale identifier depends on the operation system and perhaps on the specific installation.

Notes

If my wiki is internationalized by *config.php*, how do I revert a specific group to English?

> Use `$XLLangs = array('en');` in the group's group customization file.

If my wiki is in English and I want just one page, or group, in Spanish do I say `XLPage('es','PmWikiEs.XLPage');` in the group or page configuration file?

> Yes, that is usually the best method. If you were doing this with many scattered pages, or with several languages, you might find it easier to maintain if you load the translations all in config.php like this:

```
XLPage('es','PmWikiEs.XLPage');
XLPage('fr','PmWikiFr.XLPage');
XLPage('ru','PmWikiRu.XLPage');
 $XLLangs = array('en');
```

> Then in each group or page configuration file, you'd just use `$XLLangs` = array('es'); to set the language to use (in this case, Spanish). Note that though this method is easier to maintain, its somewhat slower because it loads all the dictionaries for each page view, even if they won't be used.

What does the first parameter of this function stand for? How can it be used?

> The XLPage mechanism allows multiple sets of translations to be loaded, and the first parameter is used to distinguish them.
>
> For example, suppose I want to have translations for both normal French and "Canadian" French. Rather than maintain two entirely separate sets of pages, I could do:

```
XLPage('fr-ca', 'PmWikiFrCa.XLPage');
XLPage('fr', 'PmWikiFr.XLPage');
```

> PmWikiFr.XLPage would contain all of the standard French translations, while PmWikiFrCA.XLPage would only need to contain "Canada-specific" translations -- i.e., those that are different from the ones in the French page.
>
> The first parameter distinguishes the two sets of translations. In addition, a*config.php* script can use the `$XLLangs` variable to adjust the order of translation, so if there was a group or page where I only wanted the standard French translation, I can set

```
 $XLLangs = array('fr', 'en');
```

> and PmWiki will use only the 'fr' and 'en' translations (in that order), no matter how many translations have been loaded with XLPage().

How can I add a translation for an individual string in a PHP file?

> Use the XLSDV() function to provide a translation for a specific (English) string. For instance, with this in config.php

```
XLSDV('nl', array('my English expression'=>'mijn Nederlandse uitdrukking'));
```

> any instance of the variable expression `$[my English expression]` in wiki mark-up will be displayed as *my English expression* in default (English) context, but as *mijn Nederlandse uitdrukking* in Dutch (nl) context, i.e. when `XLPage('nl',...)` has been called for that page in config.php or a cookbook recipe.
>
> If you need to get a translation in a PHP file, use the `XL()` function:

```
  $local_string = XL("my English expression");
```

But beware: XLPage() uses XLSDV() internally for its translation pairs, too, and only the first definition is accepted! Thus, if the Dutch XLPage already contains a translation and you want to override that, you need to use your XLSDV('nl',...) *before* calling the correspondent XLPage('nl',...). Otherwise, by using XLSDV() *after* XLPage() - e.g. within a recipe that is included later in config.php - your translation will only work as long nobody defines 'my English expression' in that XLPage.

Last modified by Jimmy_Olano on June 13, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Internationalizations

Introduction

toc top

What is PmWiki?

PmWiki is a wiki-based system for collaborative creation and maintenance of websites. See PmWiki.

What can I do with it?

PmWiki pages look and act like normal web pages, except they have an "Edit" link that makes it easy to modify existing pages and add new pages into the website, using basic editing rules. You do not need to know or use any HTML or CSS. Page editing can be left open to the public or restricted to small groups of authors. Feel free to experiment with the Text Formatting Rules in the "Wiki sandbox". The website you're currently viewing is built and maintained with PmWiki.

What are the requirements?

See the PmWiki requirements page.

Where can I find documentation?

See the documentation index page.

How can I download PmWiki?

See the download page.

How do I install PmWiki?

Instructions for installation are on the installation page.

How do I get help with PmWiki?

See Mailing lists and How to get assistance.

How do you pronounce "Michaud"?

"Michaud" is french pronounced "mee show", the trailing D is silent.

Last modified by Petko on December 10, 2012. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Introduction

LayoutVariables

toc top

Variable substitutions in the skin template are all managed by the FmtPageName() function from pmwiki.php. Pmwiki variable substitutions available on pages are managed by the substitutions from stdmarkup.php or superseded in local/config files.

`$ActionSkin`
: This array is used to override the current skin when performing a given action. The most common use is to set `$ActionSkin['print']='foo'` to use the 'foo' skin when printing, regardless of what the `$Skin` variable is set to.

`$WikiTitle`
: A variable which contains the Wiki title as displayed in the browser tab and at the top of the browser window.

`$EnablePageTitlePriority`
: A variable defining how to treat multiple `(:title ...:)` page directives (added in PmWiki 2.2.9).
`$EnablePageTitlePriority = 0;` # PmWiki default, last encountered title wins (the title may be changed from included pages or GroupFooter).
`$EnablePageTitlePriority = 1;` # First title wins; if a title is defined in the page, directives from included pages cannot change it.

`$EnableDiffInline`
: If set to 0, this variable switches off the word-level highlighting on the markup in thepage history.
`$EnableDiffInline = 0;` # Disable colors, show plain text differences

`$HTMLTagAttr`
: A string containing attributes of the `<html...>` tag in the skin template, default empty. For example, to add a "lang" attribute, set in config.php:

```
$HTMLTagAttr = 'lang="en" xml:lang="en"';
```

For this variable to work in a custom skin, add it in the template file, for example:

```
<html xmlns="http://www.w3.org/1999/xhtml" $HTMLTagAttr>
```

`$HTMLStylesFmt`
: An array of CSS statements to be included in the page's output along with other HTML headers. This array provides an easy place for scripts to add custom CSS statements.

`$HTMLHeaderFmt`
: An array of HTML text to be included in the page's <head> section, at the point where the skin template specifies a `<!--HTMLHeader-->` directive. This array provides an easy place for scripts to add custom HTML headers.

For example, if you want to specify a logo for all the pages of your wiki (a png image for Firefox (and others...), an ico for Internet Explorer):

```
$HTMLHeaderFmt['logo'] =
  '<link href="http://path/to/logo.png" type="image/png" rel="icon" />
  <link href="http://path/to/logo.ico" type="image/x-icon" rel="shortcut icon" />';
```

Another example, if you want to get the rss notification on some browsers (the rss icon in firefox for instance):

```
$HTMLHeaderFmt['rss'] =
  '<link rel="alternate" type="application/rss+xml" title="Rss All recent Changes"
     href="$ScriptUrl/Site/AllRecentChanges?action=rss" />';
```

`$HTMLFooterFmt`
: Like `$HTMLHeaderFmt` above, this contains an array of HTML text to be included near the end of an HTML document, at the point where the skin template specifies a `<!--HTMLFooter-->` directive (usually just before a closing </body> tag). Primarily used by scripts to add custom HTML output after the body of the page output.

`$MetaRobots`
: Sets the value of the `<meta name='robots' ... />` tag generated by PmWiki to control search engine robots accessing the site. PmWiki's default setting tells robots to not index anything but the normal page view, and to not index pages in the PmWiki wiki group. Explicitly setting `$MetaRobots` overrides this default.

```
# never index this site
 $MetaRobots = 'noindex,nofollow';
# disable the robots tag entirely
 $MetaRobots = '';
```

`$MessagesFmt`
: An array of HTML text to be displayed at the point of any `(:messages:)` markup. Commonly used for displaying messages with respect to editing pages.

`$RecentChangesFmt`
: An array specifying the format of the RecentChanges listing.

The key of the array specifies the page where changes will be logged, as in

```
$RecentChangesFmt['$SiteGroup.AllRecentChanges']
```

The value of the array specifies the format in which the changes will be logged, as in

```
'* [[{$Group}.{$Name}]]  . . . $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]'
```

Note the two consecutive spaces before the three dots (. . .). The two spaces separate two parts of the format: the first part doesn't change (e.g. a link to the changed page) and the second part does change (e.g. the date and author of the change). Upon saving a page, PmWiki removes a line that matches the first part and adds a line with the current format before the first line with 2 spaces. This way, any line without two consecutive spaces stays at the top of the recent changes page.

You can use and adapt the following to change the format (put it in config.php):

```
$RecentChangesFmt['$SiteGroup.AllRecentChanges'] =
  '* [[{$Group}.{$Name}]]  . . . $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
$RecentChangesFmt['$Group.RecentChanges'] =
  '* [[{$Group}/{$Name}]]  . . . $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
```

Note that changes made to the format will only affect new edits. In other words, you will need to edit a page for your new format to be visible. Note also that you need to have two spaces between the page name and the other information about the edit.

Also note that this variable has other uses, such as not reporting at all to RecentChanges and AllRecentChanges as found here PmWiki Questions.

`$RecentUploadsFmt`
: An array specifying the format for uploaded files at the RecentChanges listing. It is similar to `$RecentChangesFmt`. If enabled, newly uploaded files will be logged to the RecentChanges pages. Default is disabled. See Cookbook:RecentUploadsLog for more information.

`$DraftRecentChangesFmt`

An array specifying the format of the RecentChanges listing when saving Draft pages.

`$RecentChangesFmt` is set to `$DraftRecentChangesFmt` when a Draft page is saved. For example, you could save drafts in a separate Recent Draft Changes page and not list in the normal group's Recent Changes page:

```
$DraftRecentChangesFmt['$Group.RecentDraftChanges'] =
  '* [[{$Group}/{$Name}]]  . . . $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
$DraftRecentChangesFmt['$Group.RecentChanges'] = '';
```

`$RCLinesMax`

The maximum number of lines to be stored in RecentChanges pages. The default is zero, meaning "no limit".

```
$RCLinesMax = 1000;       # maintain at most 1000 recent changes
```

`$PageRedirectFmt`

The text to be used when a page is redirected via the `(:redirect:)` markup.

```
$PageRedirectFmt = '<p><i>redirected from $FullName</p>';
$PageRedirectFmt = '';
```

For display options, see also the FAQ on PageDirectives.

`$WikiStyle`

An array which contains the predefined WikiStyles which can be used on a textpage.
See: PmWiki.CustomWikiStyles

`$WikiStyleApply`

An array which defines the scope of wiki styling per HTML element. Default settings are:

```
'item' => 'li|dt',
'list' => 'ul|ol|dl',
'div' => 'div',
'pre' => 'pre',
'img' => 'img',
'block' => 'p(?!\\sclass=)|div|ul|ol|dl|li|dt|pre|h[1-6]',
'p' => 'p(?!\\sclass=)'
```

This defines that we can apply wiki styling on:

- LI elements using the *item* keyword
- UL, OL, DL elements using the *list* keyword
- etc.

An example of applying scope to an LI element is below. For more information refer to WikiStyle scope.

`* %apply=item red%Here is a red styled list item`
`* This item would not be styled.`
	• Here is a red styled list item
	• This item would not be styled.

You can add additional HTML elements to $WikiStyleApply to apply wiki styles to other HTML elements. For example to allow styling on table rows, or anchor tags.

`$MaxIncludes`

Controls the number of times that pages can be included via the `(:include:)` and other directives, used to control recursion and otherwise pose a sanity check on page contents. `$MaxIncludes` defaults to 50, but can be set to any value by the wiki administrator.

```
$MaxIncludes = 50;           # default
$MaxIncludes = 1000;         # allow lots of includes
$MaxIncludes = 0;            # turn off includes
```

`$Skin`

Lists the name(s) of skins to load, unless overridden by `$ActionSkin`. Normally `$Skin` contains a single string which is a the name of a skin directory, but it may also be an array of names, in which case the first skin found from the list is used.

`$SkinDirUrl`

Set by *scripts/skins.php* to be the base url of the current skin's directory (i.e., within a 'pub/skins/' directory). This variable is typically used inside of a skin .tmpl file to provide access to .css files and graphic images associated with the skin.

`$SkinLibDirs`

An array which, given the filesystem path (array key) to a skin (or a directory containing several skins), provides the corresponding URL (array value).

The array key is the directory containing the skin.tmpl and skin.php files, as seen by the PmWiki program. It does not have to be publicly accessible.

The value is the URL (web address) of the directory containing the .css, .gif, and other files which appear in the HTML code sent by PMWiki to the browser. This directory must be publicly accessible.

By default `$SkinLibDirs` is set to:

```
$SkinLibDirs = array(
  "./pub/skins/\$Skin" => "$PubDirUrl/skins/\$Skin",
  "$FarmD/pub/skins/\$Skin" => "$FarmPubDirUrl/skins/\$Skin");
```

Extra details: When PMWiki is searching for a skin it looks for a directory named for the skin in the array index/keys, and if it finds it then it will use the files in that directory and also the files in the matching array value url. The two sides normally point to the same publicly accessible directory, but they do not have to.

`$PageLogoUrl`
: is the url that refers to a logo image which most skins display somewhere in the page's header (top left usually).

`$EnablePathInfo`
: Changes the handling of the page URL. When set to`1` page URL will be `...wiki.php/Main/Main`, when set to `0` (default) it will be `...wiki.php?n=Main.Main`.

`$EnableFixedUrlRedirect`
: When PmWiki is given a partial page name (e.g., just the name of a WikiGroup), it uses `$PagePathFmt` in order to make a complete page name from the partial one, then issues a "redirect" to the browser to tell it to reload the page with the correct full page name. Setting `$EnableFixedUrlRedirect`=0; blocks the redirect, so that PmWiki continues processing with the adjusted page name rather than issuing the redirect.

`$GroupHeaderFmt`
: Defines the markup placed at the top of every page. Default value is:

```
$GroupHeaderFmt = '(:include {$Group}.GroupHeader self=0 basepage={*$FullName}:)(:nl:)';
```

`$GroupPrintHeaderFmt`
: Defines the markup placed at the top of every page when`action=print`. Default value is:

```
SDV($GroupPrintHeaderFmt,'(:include $Group.GroupPrintHeader basepage={*$FullName}:)(:nl:)');
```

`$GroupFooterFmt`
: Defines the markup placed at the bottom of every page. Default value is:

```
$GroupFooterFmt = '(:nl:)(:include {$Group}.GroupFooter self=0 basepage={*$FullName}:)';
```

`$GroupPrintFooterFmt`
: Defines the markup placed at the bottom of every page when`action=print`. Default value is:

```
SDV($GroupPrintFooterFmt,'(:nl:)(:include $Group.GroupPrintFooter basepage={*$FullName}:)');
```

`$PageNotFoundHeaderFmt`
: Specifies the HTTP header to send when attempting to browse a page that doesn't exist. Some webserver packages (notably Microsoft's "Personal Web Server") require that this variable be changed in order to work.

```
# default
$PageNotFoundHeaderFmt = 'HTTP/1.1 404 Not Found';
# return all pages as found
$PageNotFoundHeaderFmt = 'HTTP/1.1 200 Ok';
```

Beware when expecting to return the content of a Group(header|footer) for an non existent page! By default PmWiki returns 404 (because the page does not exist), despite there is some content to show. Firefox shows the content, while Internet Explorer displays its default 404 page. `$PageNotFoundHeaderFmt` MUST be set to return 200 as described above in order to get the expected behaviour with all browsers.

`$HTMLVSpace`
: Setting `$HTMLVSpace = '';` in a local customizationfile (e.g.,`local/config.php`) prevents insertion of spacer paragraphs (`<p class='vspace'></p>`) in generated HTML code. To limit this change to a single skin, place the`$HTMLVSpace = '';` statement in a skin.php file, preceded by the statement `global $HTMLVSpace;`.

`$HTMLPNewline`
: This variable allows to enable linebreaks by default, i.e. without having the directive (:linebreaks:) in a page or in a GroupHeader. To enable line breaks, add to config.php such a line:
`$HTMLPNewline = '<br/>';`

`$SimpleTableDefaultClassName`
: This variable can contain a CSS classname to be used for simple tables, if a "class=" attribute is not defined in the wiki page (default unset):
`$SimpleTableDefaultClassName = "wikisimpletable";`
See for sample code PITS:00638.

`$TableCellAttrFmt`
: For Tables, defines the HTML attributes given to each`<td>` or `<th>` cell in the output. Can contain references to $TableCellCount which holds the horizontal column number of the current cell.

`$TableCellAlignFmt`

For Tables, defines the HTML attributes for alignment of each `<td>` or `<th>` cell. Default is `" align='%s'"` where %s will be replaced with 'center', 'left' or 'right'. For a valid HTML5 output you may want to change this in config.php:

```
$TableCellAlignFmt = " class='%s'";
```

then define the CSS classes td.center, td.right and td.left (also th).

`$TableRowAttrFmt`
: For Tables, defines the HTML attributes given to each `<tr>` element in the output. Can contain references to $TableRowCount to give the absolute row number within the table, or $TableRowIndex to provide a repeating row index from 1 to `$TableRowIndexMax`.

```
# Give each row a unique CSS class based on row number (tr1, tr2, tr3, ... )
$TableRowAttrFmt = "class='tr\$TableRowCount'";
# Give each row alternating CSS classes (ti1, ti2, ti1, ti2, ti1, ... )
$TableRowIndexMax = 2;
$TableRowAttrFmt = "class='ti\$TableRowIndex'";
```

`$TableRowIndexMax`
: The maximum value for $TableRowIndex in Tables.

```
# Set rows indexes as 1, 2, 3, 1, 2, 3, 1, 2, ...
$TableRowIndexMax = 3;
```

`$EnableTableAutoValignTop`
: Advanced tables are intended for layout, and automatically insert the`valign='top'` attribute if there is no `valign` attribute defined in the markup source. Setting this variable to 0 in config.php will prevent the automatic addition.

```
$EnableTableAutoValignTop = 0; # disable automatic valign='top' attr
```

`$FmtV`['$TableCellCount']
: PMWiki internal variable - Horizontal column number of the current cell. For use in`$TableCellAttrFmt` and `$TableRowAttrFmt`. Administrators can use in `$TableCellAttrFmt` and/or `$TableRowAttrFmt`.

```
Example: $TableCellAttrFmt = 'class=col\$TableCellCount';
```

`$FmtV`['$TableRowCount']
: PMWiki internal variable - Current row number. Administrators can use in `$TableCellAttrFmt` and/or `$TableRowAttrFmt`.

```
Example: TableRowAttrFmt = "class='row\$TableRowCount'";
```

`$FmtV`['$TableRowIndex']
: PMWiki internal variable - Row index number derived from `$TableRowIndexMax`. (1,2,3,1,2,3,...). Administrators can use in `$TableCellAttrFmt` and/or `$TableRowAttrFmt`.

```
Example: $TableRowAttrFmt = "class='ind\$TableRowIndex'";
```

See also: Edit Variables

LinkVariables

`$EnableLinkPageRelative`
: When enabled, causes PmWiki to use relative urls for page links instead of absolute urls.

```
$EnableLinkPageRelative = 1;
```

`$EnableLinkPlusTitlespaced`
: When enabled, a link written like `[[Name|+]]` will display the "Spaced Title". Default is to display the "Title" of the page. See the page PageVariables for `{$Title}` and `{$Titlespaced}`.

`$PagePathFmt`
: This array lists the order in which PmWiki looks for the page that you *most likely* are attempting to link to. The default is listed below. Look at Cookbook:PagePaths for some ideas.

```
array('{$Group}.$1', '$1.$1', '$1.{$DefaultName}')
```

`$LinkPageExistsFmt`
: The (HTML) string to output for links to already existing wiki pages. Defaults to

```
<a class='wikilink' href='\$LinkUrl'>\$LinkText</a>
```

`$LinkPageCreateFmt`
: The (HTML) string to output for links to non-existent wiki pages. The default is to add a '?' after the link text with a link to the page edit/create form. Defaults to

```
<a class='createlinktext' href='\$PageUrl?action=edit'>\$LinkText</a>
<a class='createlink' href='\$PageUrl?action=edit'>?</a>
```

`$LinkPageCreateSpaceFmt`
: Same as `$LinkPageCreateFmt`, but used when the link text has a space in it.

`$LinkPageSelfFmt`
: The (HTML) string to output for self-referencing links (i.e. links to the page itself). Defaults to

```
<a class='selflink' href='\$LinkUrl'>\$LinkText</a>
```

`$UrlLinkFmt`
: The (HTML) string to output for URL-links that begin with 'http:', 'ftp:', etc. Defaults to

```
<a class='urllink' href='\$LinkUrl' title='\$LinkAlt' rel='nofollow'>\$LinkText</a>
```

`$IMapLinkFmt`
: an array of link formats for various link "schemes". Not set as default.
Examples of custom formats to allow different styling via classes:
Links to http: standard url links:

```
$IMapLinkFmt['http:'] = "<a class='httplink urllink' href='\$LinkUrl'>\$LinkText</a>";
```

Links to https: secure pages:

```
$IMapLinkFmt['https:'] = "<a class='httpslink urllink' href='\$LinkUrl'>\$LinkText</a>";
```

Links to PmWiki: InterMap shortcut:

```
$IMapLinkFmt['PmWiki:'] = "<a class='pmwikilink urllink' href='\$LinkUrl'>\$LinkText</a>";
```

`$InterMapFiles`
: An array consisting a list of files and pages containing InterMap entries to be loaded.

`$MakePageNameFunction`
: Name of a custom function to replace MakePageName(), which converts strings into valid page names.

`$MakePageNamePatterns`
: `$MakePageNamePatterns` is an array of regular expression replacements that is used to map the page link in a *free link* such as `[[free link]]` into a page name. Currently the default sequence is:

```
"/'/" => '',      # strip single-quotes
"/[^$PageNameChars]+/" => ' ',         # convert everything else to space
'/((^|[^-\\w])\\w)/' => PCCF("return strtoupper(\$m[1]);"),
'/ /' => ''
```

Note that if you change `$MakePageNamePatterns`, the documentation links may break. This can be fixed by re-setting `$MakePageNamePatterns` to the default in local/PmWiki.php.

`$MakePageNameSplitPattern`
: See Cookbook:DotsInLinks.

`$WikiWordCountMax`
: The maximum number of times to convert each WikiWord encountered on a page. Defaults to 1,000,000. Common settings for this variable are zero (disable WikiWord links) and one (convert only the first occurrence of each WikiWord).

```
$WikiWordCountMax = 0;     # disable WikiWord links
$WikiWordCountMax = 1;     # convert only first WikiWord
```

`$WikiWordCount`
: An array that allows the number of WikiWord conversions to be set on a per-WikiWord basis. The default is to use `$WikiWordCountMax` unless a value is set in this array. By default PmWiki sets `$WikiWordCount['PmWiki']=1` to limit the number of conversions of "PmWiki".

```
$WikiWordCount['PhD']=0;     # Don't convert "PhD"
$WikiWordCount['WikiWord']=5;  # Convert WikiWord 5 times
# the following lines keep a page from linking to itself
$title = FmtPageName('$Title_',$pagename);
$WikiWordCount[$title]=0;
```

`$EnableRedirectQuiet`
: Enable the `quiet=1` parameter for the redirect directive. On publicly edited wikis it is advisable not to enable quiet redirects.

```
$EnableRedirectQuiet = 0;     # disable quiet redirects (default)
$EnableRedirectQuiet = 1;     # enable quiet redirects
```

`$QualifyPatterns`
: An array of regular expression replacements applied when text from one page is included in another, used by the function Qualify(). The two default patterns rewrite links like `[[Page]]` into `[[Group/Page]]`, and page (text) variables like `{$Title}` into `{Group.Page$Title}` so that they work the same way in the source page and in the including page.

Categories: PmWiki Developer

Links

toc top

A key feature of wiki-based systems is the ease of creating hyperlinks (or short **links**) in the text of a document. PmWiki provides multiple mechanisms for creating such links.

Links to other pages in the wiki

To create an internal link to another page, simply enclose the name of the page inside double square brackets, as in `[[wiki sandbox]]` or `[[installation]]`. This results in links to wiki sandbox and installation, respectively.

PmWiki creates a link by using the text inside the double brackets. It does this by removing spaces between the words, and automatically capitalizing the first letter of each word following spaces or other punctuation (like ~). Thus `[[Wiki Sandbox]]`, `[[wiki sandbox]]`, and `[[WikiSandbox]]` all display differently but create the same link to the page titled `WikiSandbox`. Or in other words, PmWiki will automatically create the "link path name" using the page name in *CamelCase*, but the "link text" will display in the format you have entered it.

Some PmWiki sites (default not) will recognize words written in CamelCase, called a WikiWord, automatically as a link to a page of the same name.

Links with different link text

There are three ways to get a different link text:

1. **Hide link text**. Link text within (parentheses) will not be not displayed, so that `[[(wiki) sandbox]]` links to *WikiSandbox* but displays as sandbox. For addresses actually containing parentheses, use %28 and %29 http://www.example.com/linkwith%28parenthese%29.

2. **Change link text**. You can specify another link text after a vertical brace, as in `[[WikiSandbox | a play area]]`, or you can use an arrow (`->`) to reverse the order of the link text and the target, as in `[[a play area -> WikiSandbox]]`. Both links displays as a play area.

3. **Show page title instead of page name**. The use of special characters in the page name is not a problem for PmWiki, but on some servers it may be better to use only plain A-Z letters for the page "name" (which is also a filename), and set the page "title" to the extended or international characters with the (:title PageTitle:) directive within the page. The page title can be shown instead of the page name with the `[[PageName|+]]` link markup, e.g. page `BasicEditing` contains the directive (:title Basic PmWiki editing rules:) with the result that a link written as `[[BasicEditing|+]]` will display as Basic PmWiki editing rules. See also `$EnableLinkPlusTitlespaced`.
   Since PmWiki version 2.2.14 this works also for those technical pages that have an entry in the XLPage, without the need to add the (:title PageTitleName:) directive within that page (for more details see Localization.Localization).

On top of above ways, a suffix can be added to the end of a link, which becomes part of the link text but not of the target page name.
Note: This feature works with the `[[PageName|+]]` markup only since Version 2.2.90.

What to type	What it looks like

`* [[(wiki) sandbox]]`
`* [[(wiki) sandbox]]es`
`* [[WikiSandbox
	wiki sandbox]],`
	`* [[WikiSandbox
		wiki sandbox]]es`
		`* [[BasicEditing
			+]]`	• sandbox
				• sandboxes
				• wiki sandbox,
				• wiki sandboxes
				• Basic PmWiki editing rules

Links with tool tip

From version 2.2.14 PmWiki can show tooltip titles with the following format:
external link
`[[http://pmwiki.org"external tool tip title" | external link ]]`, eg external link or http://pmwiki.org
internal link
`[[Links"internal tool tip title" | internal link ]]`, eg internal link or Links
Anchor links
`[[#name"anchor tool tip title"|anchor link text]]` (since Version 2.2.48), eg anchor link text or #name
InterMap link
`[[Wikipedia:Wiki"tool tip title"| InterMap link ]]`, eg InterMap link or Wikipedia:Wiki

Links to nonexistent pages

Links to nonexistent pages are displayed specially, to invite others to create the page. See Creating new pages to learn more.

Links to pages in other wiki groups

Links as written above are links between pages of the same group. To create a link to a page in another group, add the name of that other group together with a dot or slash as prefix to the page name. For example, links to `Main/WikiSandbox` could be written as:

What to type	What it looks like
`* [[Main.WikiSandbox]]`	• Main.WikiSandbox

`* [[Main/WikiSandbox]]`
`* [[(Main.Wiki)Sandbox]]`
`* [[Main.WikiSandbox | link text]]`
`* [[Main.WikiSandbox | +]]`
	• WikiSandbox
	• Sandbox
	• link text
	• Wiki Sandbox

To link to the "default home page" of a group, the name of the page can be omitted:

`* [[Main.]]`
`* [[Main/]]`
	• Main.
	• Main

See Wiki Group to learn more about PmWiki groups.

Category links

Categories are a way to organize and find related pages. The idea is that every page that falls into a particular subject area should have a link to a shared page containing links to other pages on that subject. These *shared pages* are created in the special group `Category`, and thus these subject areas are called "categories".

Adding a page to the category `Subject` is simple by adding the `[[!Subject]]` markup somewhere on that page. This will create a link to the page `Category.Subject`. So `[[!Subject]]` is a kind of link shortcut to the page `Category.Subject`. See Categories to learn more.

User page links

Similar is `[[~Author]]` a link shortcut to the page `Author` in the special group `Profiles`. PmWiki automatically creates this type of link for the *current author*, when it encounters three tilde characters (~) in a row (~~~) in the page text. The current author is the name found in the "Author" field, when you create or modify a page. The current date and time is appended when four tilde characters in a row are encountered (~~~~).

So, when the Author field contains "Author":
~~~ markup will be replaced by: Author
~~~~ markup will be replaced by: Author October 10, 2010, at 04:50 PM

Link shortcuts

`[[PageName|#]]` creates a reference link as shown below[1].

Links to specific locations within a page -- "anchors"

To define a location, or bookmark, within a page to which you may jump directly, use the markup `[[#name]]`. This creates an " anchor" that uniquely identifies that location in the page. Then to have a link jump directly to that anchor, use one of

- `[[#name|link text]]` within the same page, or
- `[[PageName#name]]` or `[[PageName#name|link text]]` for a location on another page
- The form `[[PageName(#name)]]` may be useful for hiding the anchor text in a link.

For example, here's a link to the Intermaps section, below.

Notes:
- The anchor itself must begin with a letter, **not a number**.
- Valid characters for anchor names are letters, digits, dash (-), underscore (_), and the period (.).
- A link to an anchor must have the **same capitalization as the anchor** itself.
- Spaces are not allowed in an anchor: "`[[#my anchor]]`" won't work, "`[[#myanchor]]`" will.
- All anchor names in a page should be unique.

Sections

While in HTML the purpose of anchors is mostly for jumping to a position in the text, in PmWiki they serve an internal purpose, too: Each anchor also creates a section, because sections are defined as the part of the page between their start anchor and the next anchor. For more details, see Page Sections.

Links to actions

To link to a specific action for the current page use `[[{$FullName}?action=actionname|linkname]]`.

Examples:
- `[[{$FullName}?action=edit|Edit]]` for editing
- `[[{$FullName}?action=diff|differences]]` for differences.

Links outside the wiki

Links to external sites (URLs)

Links to external sites simply begin with a prefix such as 'http:', 'ftp:', etc. Thus `http://google.com/` and `[[http://google.com/]]` both link to Google. As with the above, an author can specify the link text by using the vertical brace or arrow syntax, as in `[[http://google.com/ | Google]]` and `[[Google -> http://google.com]]`.

If the external link includes (parentheses), escape these using %28 for "(" and %29 for ")" :

```
[[http://en.wikipedia.org/wiki/Wiki_%28disambiguation%29 | link to "Wiki (disambiguation)" ]]
```

link to "Wiki (disambiguation)"

The recipe Cookbook:FixURL makes it easy to encode parentheses and other special characters in link addresses.

Links to intranet (local) files

Not all browsers will follow such links (some Internet Explorer versions reportedly follow them). You can link to a file system by including the prefix `'file:///'`. So `file:///S:\ProjPlan.mpp` and `[[Shared S drive->file:///S:\]]` are both valid links. On a Windows file system you may want to use network locations (eg \\server1\rootdirectory\subdirectory) rather than drive letters which may not be consistent across all users. Not all browsers will follow such links.

See also Cookbook:DirList.

Link characteristics

Links as References

Links may also be specified as **References**, so the target appears as an anonymous *numeric* reference rather than a *textual* reference. The following markup is provided to produce sequential reference numbering within a PmWiki page:

Formatting the link as: `[[http://google.com |#]]` produces: [2] as the link.

Subsequent occurrence of the reference link format on the same page will be incremented automatically as per the following example: Entering `[[http://pmwiki.com |#]]` produces [3], `[[#intermaps |#]]` produces [4], and so on for further reference links.

Intermaps

Inter Map links are also supported (see Inter Map). In particular, the `Path:` InterMap entry can be used to create links using relative or absolute paths on the current site (e.g., `Path:../../somedir/foo.html` or `Path:/dir/something.gif`).

Links that open a new browser window

To have a link open in another window, use `%newwin%...%%`:

- `%newwin% http://pmichaud.com %%` produces http://pmichaud.com
- `%newwin% [[http://google.com/ | Google]] %%` produces Google
- `%newwin% [[Main.WikiSandbox]] %%` produces Main.WikiSandbox

You can also specify that links should open in a new window via the `%target=_blank%...%%` attribute:

```
The following link %target=_blank%
http://pmichaud.com %%
will open in a new window.
```

The following link http://pmichaud.com will open in a new window.

Links that are not followed by robots

Prefix a link with %rel=nofollow% to advise robots and link checkers not to follow it.

Links and CSS Classes

PmWiki automatically gives classes to several types of links. Among other things, this enables you to format each type differently.

Note: This may be an incomplete list.

.selflink
    A link to the current page. Useful in sidebars to show "you are here".
.wikilink
    A link to another page within the wiki.
.urllink
    A link to a page outside the wiki.

Notes

Note: The default behavior of "+" above can be overridden to display the spaced title, rather than simply the title by adding the following to config.php:

```
## [[target |+]] title links
Markup('[[|+', '<[[|',
  "/(?>\\[\\[([^|\\]]+))\\|\\s*\\+\\s*]]/e",
  "Keep(MakeLink(\$pagename, PSS('$1'),
                PageVar(MakePageName(\$pagename,PSS('$1')), '\$Titlespaced')
               ),'L')");
```

How do I create a link that will open as a new window?

Use the `%newwin%` wikistyle, as in:

`%newwin% http://example.com/ %%`	http://example.com/

How do I create a link that will open a new window, and configure that new window?

This requires javascript. See Cookbook:PopupWindow.

How do I place a mailing address in a page?

Use the `mailto:` markup, as in one of the following:

`* mailto:myaddress@example.com`
`* [[mailto:myaddress@example.com]]`
`* [[mailto:myaddress@example.com | email me]]`
`* [[mailto:myaddress@example.com?subject=Some subject | email me]]`
	• myaddress@example.com
	• mailto:myaddress@example.com
	• email me
	• email me

The markup `[[mailto:me@example.com?cc=someoneelse@example.com&bcc=else@example.com&subject=Pre-set Subject&body=Pre-set body | display text]] =]` lets you specify more parameters like the message body and more recipients (may not work in all browsers and e-mail clients).

See also Cookbook:DeObMail for information on protecting email addresses from spammers.

How can I enable links to other protocols, such as nntp:, ssh:, xmpp:, etc?

See Cookbook:Add Url schemes

How do I make a WikiWord link to an external page instead of a WikiPage?

Use link markup. There are two formats:

```
[[http://example.com/ | WikiWord]]
[[WikiWord -> http://example.com/]]
```

How do I find all of the pages that link to another page (i.e., backlinks)?

In the wiki search form, use `link=Group.Page` to find all pages linking to Group.Page.

Use the `link=` option of the `(:pagelist:)` directive, as in

```
(:pagelist link=SomePage list=all:)   -- show all links to SomePage
(:pagelist link={$FullName} list=all:)  -- show all links to the current page
```

Note that (with a few exceptions) includes, conditionals, pagelists, searchresults, wikitrails, and redirects are not evaluated for Wikilinks, and so any links they put on the page will not be found as backlinks. All other directives and markup, for example links brought to the page by (:pmform:), will be found.

What link schemes does PmWiki support?

See PmWiki:Link schemes

How do I open external links in a new window or mark them with an icon?

See Cookbook:External links

How can I use an image as a link?

Use [[Page| Attach:image.jpg]] or [[http://site | http://site/image.jpg]] See Images#links

Why my browser does not follow local file:// links?

For security reasons, most browsers will only enable file:// links if the page containing the link is itself on the local drive. In other words, most browsers do not allow links to file:// from pages that were fetched using http:// such as in a PmWiki site. See also Cookbook:DirList for a workaround.

LocalCustomizations

A Wiki Administrator can make a lot of customizations simply by setting variables in the */local/config.php* and defining cascading style sheets in */pub/css/local.css* files. Any group or page can also have its own configuration file and configuration css file.

This page describes how customizations work in general, see PmWiki.Documentation Index for specific customizations that are commonly performed at many PmWiki installations, including:

- Skins - Change the look and feel of part or all of PmWiki
- Internationalizations - Language internationalisation of web pages
- Custom Markup - Using the Markup() function for custom wiki syntax; migration to PHP 5.5
- InterMaps - Interwiki links definition and use

local/config.php

From its inception, PmWiki has been designed so that Wiki Administrators can greatly customize the way PmWiki displays pages and the markup sequences used to generate pages. (This is even mentioned explicitly in PmWiki Philosophy #4 Collaborative Maintenance.) As a result, the core *pmwiki.php* script makes extensive use of PmWiki.Variables to determine how markup sequences will be processed and what each individual page will output.

The simplest type of customization is merely setting a variable to 1 (or TRUE). Here's an example that enables ?action=diag and ?action=phpinfo actions:

```
$EnableDiag = 1;
```

You can begin a line with a "#" (an octothorpe, a.k.a. a hash symbol or pound sign) to add a comment. Additionally, some built-in PmWiki variables take values other than 1 or 0 (true or false). Here's another example that customizes the wiki's behavior with respect to search engine web robots (see Cookbook:ControllingWebRobots):

```
# Remove the default "rel='nofollow'" attribute for external links.
$UrlLinkFmt = "<a class='urllink' href='\$LinkUrl' title='\$LinkAlt'>\$LinkText</a>";
```

The *scripts/* subdirectory (below the directory holding the *pmwiki.php* script) has many customizations. The PmWiki Cookbook contains many example customizations (recipes) that you can download into the *cookbook/* subdirectory, The first few lines of each of these scripts generally contain instructions about how to enable (and use) the feature provided by the script.

These customizations are included in your *config.php* site configuration. For most scripts this is done by simply adding lines like:

```
include_once("cookbook/recipefile.php");
```

and

```
include_once("scripts/scriptfile.php");
```

at the end of the *config.php* file to enable them.

Some of the scripts are automatically enabled for you via the *scripts/stdconfig.php* script unless you disable it by setting `$EnableStdConfig=0;` in *local/config.php*.

Order of the commands in config.php (link)

The following order is recommended:

- define `$ScriptUrl` and `$PubDirUrl`, if needed,
- define any custom PageStore class, like SQLite, CompressedPageStore or PerGroupSubDirectories,
- next include_once scripts/xlpage-utf-8.php,
- next call XLPage() which needs the definitive (rw) `$WikiDir` already set in order to find the wiki page containing the translations,
- next include authuser.php (if needed), because PmWiki caches some group and page authorization levels when a page is accessed,
- next include any other scripts and recipes,
- any direct function call in config.php, like ResolvePageName(), CondAuth(), PageTextVar(), PageVar(), RetrieveAuthPage(), or others, if possible, should be done near the end of config.php.

*Note, each part is **not** required, but if your wiki needs it, this is the recommended order in config.php.*

Character encoding of config.php

The encoding used when you save `config.php` has an effect. Your text editor should allow you to save config.php in the encoding of your wiki. (The default encoding of PmWiki is ISO-8859-1, for new wikis it is recommended to enable UTF-8.)

Newer operating systems like GNU/Linux, FreeBSD and Apple generally default to saving text files in Unicode/UTF-8; in Windows the default encoding is ANSI/Windows-1252 which is almost the same as PmWiki's ISO-8859-1.

The following *free/libre software* text editors can edit and save a file in different encodings:

- Cross-platform: Kate (for KDE), Geany, Arachnophilia, SciTE, Bluefish, vim and others.
- Windows: Notepad++, ConTEXT, Notepad 2.
- OS X: Aquamacs.

Note that if you use the UTF-8 encoding, you should save your files*"without Byte Order Mark (BOM)"*.

Over time PmWiki will be updated to default to Unicode/UTF-8 encoding, which allows all possible alphabets and languages. See UTF-8 for more information.

pub/css/local.css

You can create this file and set there some custom CSS styles which will override any styles set by skins. For example:

```
h1, h2, h3, h4, h5 { color: #880000; } /*dark red titles*/
a { text-decoration: none; } /* don't underline links */
```

Don't modify pmwiki.php or other core files

You should strongly resist the temptation to directly modify the*pmwiki.php* script or the files in the*scripts/* subdirectory. Any modifications you make to these files will probably be overwritten whenever you upgrade. Instead, look at some of the sample scripts for examples of customizations that can be performed from *config.php*. You can even create your own script to do a customization and use `include_once(...)` to include it from *config.php*. If you do make your own customization script, you can safely put it in the *cookbook/* subdirectory--it won't get overwritten by an upgrade there. You might also want to submit your customization to the pmwiki-users mailing list or the Cookbook so that others can benefit from your effort and so that it can perhaps be included in future releases of PmWiki.

FAQ

There's no "config.php"; it's not even clear what a "local customisation file" is!

> The "sample-config.php" file in the "docs" folder, is given as an example. Copy it to the "local" folder and rename it to "config.php". You can then remove the "#" symbols or add other commands shown in the documentation. See also Group Customizations.

Can I change the default page something other than Main.HomePage (`$DefaultPage`)?

> Yes, just set the `$DefaultPage` variable to the name of the page you want to be the default. You might also look at the `$DefaultGroup` and `$DefaultName` configuration variables.
>
> ```
> $DefaultPage = 'ABC.StartPage';
> ```
>
> Note the recommendations in `$DefaultName` and the need to set `$PagePathFmt` as well if you are changing the default startup page for groups.

How do I get the group / page name in a local configuration file (e.g. *local/config.php*)?

> Use the following markup in pmwiki-2.1.beta21 or newer:
>
> ```
> ## Get the group and page name
> $pagename = ResolvePageName($pagename);
> $page = PageVar($pagename, '$FullName');
> $group = PageVar($pagename, '$Group');
> $name = PageVar($pagename, '$Name');
> ```
>
> Note the importance of the order of customizations in config.php aboveto avoid caching problems.
>
> If you need the verbatim group and page name (from the request) early in config.php, `$pagename` is guaranteed to be set to

1. Any value of ?n= if it's set, or
2. Any value of ?pagename= if it's set, or
3. The "path info" information from REQUEST_URI (whatever follows SCRIPT_NAME), or
4. Blank otherwise
   according to this posting

Can I remove items from the wikilib.d folder on my site?

The files named Site.* and SiteAdmin.* contain parts of the interface and the configuration and they should not be removed. The other files named PmWiki* contain the documentation and could be removed.

How do I customize my own 404 error page for non-existent pages?

To change the text of the message, try editing the Site.PageNotFound page.

Is the order of customizations in config.php important? Are there certain things that should come before or after others in that file?

Yes, see Order of the commands in config.php

MailingLists

toc top

There are several mailing lists available for PmWiki.

[pmwiki-users]
This is a great resource where a very helpful group of people will answer questions and discuss PmWiki development. As of 2016, traffic is around 20-40 messages a month.
If you ask a question on the list and it doesn't get answered, don't feel let down. Just ***ask it again***. It probably slipped by unnoticed.

Archives are available from:
http://www.pmichaud.com/pipermail/pmwiki-users/
http://news.gmane.org/gmane.comp.web.wiki.pmwiki.user (searchable)
http://groups.google.com/group/pmwiki-users (searchable, but disconnected from main list in 2012.04)
http://www.mail-archive.com/pmwiki-users@pmichaud.com/info.html (searchable)

Search pmwiki-users archives at gmane

[pmwiki-users-de]
A mailing list for german-speaking users of PmWiki. Archived at
http://www.pmichaud.com/pipermail/pmwiki-users-de

[pmwiki-users-es]
Lista de usuarios PmWiki en Español.

[pmwiki-users-fr]
A mailing list for french-speaking users of PmWiki.

[pmwiki-devel]
This list was created to lower the traffic on pmwiki-users, it focuses on discussions surrounding code development for PmWiki (both core and recipe development).

Archives are available from:
http://www.pmichaud.com/pipermail/pmwiki-devel/
http://news.gmane.org/gmane.comp.web.wiki.pmwiki.devel (searchable)
http://groups.google.com/group/pmwiki-devel (searchable)
http://www.mail-archive.com/pmwiki-devel@pmichaud.com/info.html (searchable)

[pmwiki-announce]
Announcements of new version releases and urgent information, about 1-2 messages per month. If you use PmWiki in a production environment, this low-volume list is highly recommended. The archive is at:
http://www.pmichaud.com/pipermail/pmwiki-announce

Suggestions:

- If you reply to a digest message, please remove the messages irrelevant to your reply before sending it back to the list.
  - It's also helpful to change "Re: pmwiki-users Digest, Vol [...]" to "Re: [the original subject]" because some mail programs determine threads based on the subject.
- If you address a reply to a single list member, please take the [pmwiki-users] off the subject line, or it's possible for your message to get lost in the mailing list traffic. Many people filter list traffic to a separate mailbox.
- If you ask a question, you should disable "digest" mode, this way you'll receive the replies as soon as people post them, and you could follow-up. In digest mode you might receive the replies a week or two later.

Changing mail list settings

Here are some tips regarding changing the mailing list settings:

- Logging in...
  - First go to http://www.pmichaud.com/mailman/listinfo/pmwiki-users and enter your e-mail address in the field at the bottom of the page, to the left of the button *Unsubscribe or edit options*.

  - Next you need to enter your password. As you've probably forgotten this, use the button*Remind* at the bottom of the page to get a new password.
  - Finally enter the password you should get momentarily via e-mail.

- You can directly go to the options web page through a URI such as the following:
  `http://host.pmichaud.com/mailman/options/pmwiki-users/<user>%40<domain>`
  where `<user>` is everything before the `@` in an e-mail address, and `<domain>` is everything after (For those who wonder, the `%40` in the URI just stands for `@`'.

- You can also obtain various help by sending an email to pmwiki-users-request@pmichaud.com with the text `help` in either the subject or the body.

Newsgroups (NNTP)

You may be interested, that the lists are also accessible as newsgroups.

The NNTP server is:

- news.gmane.org [1]

The groups are:

- gmane.comp.web.wiki.pmwiki.user
- gmane.comp.web.wiki.pmwiki.announce
- gmane.comp.web.wiki.pmwiki.user.de

MarkupExpressions

The `{(...)}` "expression markup" allows for a variety of string and formatting operations to be performed from within markup. Operations defined by this recipe include *substr*, *ftime*, *strlen*, *rand*, *mod*, *toupper / tolower*, *ucfirst*, *ucwords*, *pagename* and *asspaced*. Markup expressions can be nested, using the markup `{(...(...)...)}`.

substr

The "substr" expression extracts portions of a string. The arguments are

1. the string to be processed. Always quote the string to be processed.
2. the initial position of the substring. Note that the initial position argument is zero-based (i.e., the first character is referenced via a "0").
3. the number of characters to extract

`{(substr "PmWiki" 2 3)}`	`Wik`
`{(substr "PmWiki" 2)}`	`Wiki`
`{(substr "PmWiki" 0 1)}`	`P`
`{(substr "PmWiki" 0 -3)}`	`PmW`
`{(substr "PmWiki" -3)}`	`iki`

To obtain the last n characters of a string use `{(substr "string" -n)}`
To truncate the last n characters of a string use `(substr "string" 0 -n)}`

ftime

"Ftime" expressions are used for date and time formatting. The generic form is

```
{(ftime "fmt" "when")}
{(ftime fmt="fmt" when="when")}
```

where *fmt* is a formatting string and *when* is the time to be formatted. The arguments can be in either order and may use the optional "fmt=" and "when=" labels.

Examples:

`{(ftime)}`	`July 11, 2017`
`{(ftime fmt="%F %H:%M")}`	`2017-07-11 20:46`
`{(ftime %Y)}`	`2017`
`{(ftime fmt=%T)}`	`20:46:56`
`{(ftime when=tomorrow)}`	`July 12, 2017`
`{(ftime fmt="%Y-%m-%d" yesterday)}`	`2017-07-10`
`{(ftime "+1 week" %F)}`	`2017-07-18`
`{(ftime fmt=%D "+1 month")}`	`08/11/17`
`{(ftime fmt="%a%e %b" when="next week")}`	`Mon17 Jul`

The *fmt* parameter is whatever is given by "fmt=", the first parameter containing a '%', or else the site's default. The formatting codes are described at http://php.net/strftime. In addition to those, '%F' produces ISO-8601 dates, and '%s' produces Unix timestamps. Some common formatting strings:

```
    %F                 # ISO-8601 dates      "2017-07-11"
    %s                 # Unix timestamp      "1499798816"
    %H:%M:%S           # time as hh:mm:ss    "20:46:56"
    %m/%d/%Y           # date as mm/dd/yyyy  "07/11/2017"
    "%A, %B %d, %Y"    # in words            "Tuesday, July 11, 2017"
```

The *when* parameter understands many different date formats. The when parameter is whatever is given by "when=", or whatever parameter remains after determining the format parameter. Some examples:

```
    2007-04-11            # ISO-8601 dates
    20070411              # dates without hyphens, slashes, or dots
    2007-03               # months
    @1176304315           # Unix timestamps (seconds since 1-Jan-1970 00:00 UTC)
    now                   # the current time
    today                 # today @ 00:00:00
    yesterday             # yesterday @ 00:00:00
    "next Monday"         # relative dates
    "last Thursday"       # relative dates
    "-3 days"             # three days ago
    "+2 weeks"            # two weeks from now
```

Note: If you want to convert a Unix timestamp you**must** prefix with the @. Thus, "`{(ftime "%A, %B %d, %Y" @1231116927)}`".

The *when* parameter uses PHP's strtotime function to convert date strings according to the GNU date input formats; as of this writing it only understands English phrases in date specifications.

The variable `$FTimeFmt` can be used to override the default date format used by the "ftime" function. The default `$FTimeFmt` is `$TimeFmt`.

strlen

The "strlen" expression returns the length of a string. The first argument is the string to be measured.

`{(strlen "{$:Summary}")}`	32

rand

The "rand" expression returns a random integer. The first argument is the minimum number to be returned and the second argument is the maximum number to be returned. If called without the optional min, max arguments rand() returns a pseudo-random integer between 0 and RAND_MAX. If you want a random number between 5 and 15 (inclusive), for example, use (rand 5 15).

`{(rand)}`
`{(rand 1 99)}`
	1273340782
	96

mod

The advanced "mod" expression returns the modulo (remainder) of the division of two numbers. It may be used in advanced PageList templates together with `{$$PageCount}` to insert markup every (modulo) entries, for example to create alternate styled "zebra" table rows, or to insert a line/row break. (See also PageLists, WikiStyles and ConditionalMarkup.)

```
>>comment<<
%define=bg1 item bgcolor=#f88%
%define=bg2 item bgcolor=#ff8%
%define=bg0 item bgcolor=#8f8%
[[#altrows]]
* %bg{(mod {$$PageCount} 3)}% {=$Name}
({$$PageCount})
[[#altrowsend]]
>><<
(:pagelist fmt=#altrows group=PmWiki
count=10:)
```

- AccessKeys (1)
- Audiences (2)
- AuthUser (3)
- AvailableActions (4)
- BackupAndRestore (5)
- BasicEditing (6)
- BasicVariables (7)
- Blocklist (8)
- BlockMarkup (9)
- Categories (10)

toupper / tolower

The "toupper" and "tolower" expressions convert a string into uppercase or lowercase. The first argument is the string to be processed.

`{(toupper "{$:Summary}")}`
`{(tolower "{$:Summary}")}`
	`STRING AND FORMATTING OPERATIONS`
	`string and formatting operations`

ucfirst / ucwords

The "ucfirst" expression converts to uppercase the first character of the string, and "ucwords", the first character of each word. The first argument is the string to be processed.

`{(ucfirst "{$:Summary}")}`
`{(ucwords "{$:Summary}")}`
	`String and formatting operations`
	`String And Formatting Operations`

pagename

The "pagename" expression builds a pagename from a string. The first argument is the string to be processed.

`{(pagename "{$:Summary}")}`	`PmWiki.StringAndFormattingOperations`

asspaced

The "asspaced" expression formats wikiwords. The first argument is the string to be processed.

`{(asspaced "{$FullName}")}`	`Pm Wiki.Markup Expressions`

Nesting expressions

Markup expressions can be nested. Omit the curly braces for the inner expressions:

`{(tolower (substr "Hello World" 2))}`	`llo world`

Notes

- For PmWikis version 2.2.33 or older, the string-processing expressions may not work properly on multibyte UTF-8 characters. Newer versions should work fine.

See also

- Page variables, Page text variables
- Conditional markup
- Cookbook:MarkupExpressionSamples — custom markup expression samples
- Cookbook:MarkupExprPlus

Last modified by simon on September 04, 2014.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/MarkupExpressions

toc top

Markup Master Index

toc top

This page contains the most frequently used wiki markup, briefly. Follow the links in each section to learn more.

Table of contents

Markup concepts introduction

PmWiki markup can be applied to 'blocks' of text, and to text 'lines'. PmWiki markup is also used to read and save page, group, and wiki metadata through the use of variables. PmWiki markup can be used to process metadata variables though expressions and pagelists. PmWiki provides a wiki style markup that can be applied to text, lists, paragraphs, and blocks.

Text markup, also known as wikitext is variable, see below, and in general broadly follows wiki conventions. Text markup only applies to single lines of text, delimited by a newline.

Block markup is applied to multiple lines of text as paragraph blocks and division blocks.

In PmWiki the most important markup is the directive. The directive is signified by parenthesis and a colon, viz: `(:....:)`. The directive provides most of PmWiki's functionality.

Markup expressions `{(...)}`, variable markup `{$...}`, and wiki styles `%...%` also provide PmWiki functionality.

Links

See Links

External links

```
http://example.com
[[http://example.com]]
[[http://example.com"tool tip"]]
[[http://example.com | link text]]
[[link text -> http://example.com]]
```

Page links

`[[PageName]]`
`[[page name]]`
`[[page (name)]]`
`[[PageName | link text]]`
`[[PageName | + ]]` *(titled link)*
`[[PageName | # ]]` *(anonymous numerical reference link)*
`[[PageName"tool tip"]]`
`[[link text -> PageName]]`
`[[#anchor]]` *(to create an anchor)*
`[[#anchor | link text]]` *(to refer to an anchor)*
`[[anchor | # ]]` *(anonymous numerical reference link)*
`[[PageName#anchor | link text]]` *(to refer to an anchor in another page)*

See also WikiWord on how to enable `WikiWord` links.

WikiGroup links

See Links and Categories

```
[[GroupName/]] or [[Group name/]]
[[GroupName"tool tip"]]
[[GroupName.]]
[[GroupName/PageName]] or [[GroupName/page name]]
[[(GroupName.)page name]]

[[~Author Name]]
[[~Author Name | +]]
[[~Author Name | #]]
[[~Author Name | link text]]
[[~Author Name"tool tip"]]
[[!Category Name]]
```

InterMap links

See InterMap

```
[[Path:/path/local_document.html]]
[[Wikipedia:WikiWikiWeb]]
```

Email links

```
mailto:someone@example.com
[[(mailto:)someone@example.com]]
[[mailto:someone@example.com | display text]]
```

```
[[display text -> mailto:someone@example.com]]
```

Upload links

See Uploads and Images

```
Attach:file.odt
[[(Attach:)file.odt]]
[[Attach:file.odt | alternative text ]]
[[Attach:file with spaces.pdf]]
[[Attach:Groupname./file with spaces.pdf]]
```

Link Schemes

In addition to `http:`, `https:`, `mailto:` PmWiki also supports:

```
ftp:
news:
gopher:
nap:
file:
tel:
geo:
```

Images

See Images and Uploads

Images as Images

`http://example.com/image.gif`
`http://example.com/image.gif"alt text"`
`Attach:image.gif"My image"`
`Attach:Groupname./image.gif"image in another group"`
`Attach:Groupname.Pagename/image.gif"image on another page"`
`%lfloat% Attach:image.gif | Caption %%` *(could be %rfloat%, %center%, %rframe%, %lframe%, %frame%)*
`%width=200px% Attach:image.gif %%`
`%thumb% Attach:image.gif %%`

Images as links

```
[[Attach:image.gif]]
[[(Attach:)image.gif]]
[[PageName | Attach:image.gif"alt text"]]
[[http://example.com/ | Attach:image.gif"alt text"]]
[[http://example.com/ | http://example.com/image.png"alt text"]] | Caption
%rframe thumb% [[Attach:image.gif | Attach:image.gif"alt text"]] | Caption
```

Start-of-line markup

See Text formatting rules

Lists

See List styles, Wiki styles and Cookbook:Outline lists

`* unordered list`
`** deeper list`
`# ordered list`
`# %item value=#%` arbitrary start number
`# %decimal%, %roman%, %ROMAN%, %alpha%, %ALPHA%`
`:term:definition`
Also
`Q:` start a question paragraph
`A:` start an answer paragraph

Headings

```
!! Heading
!!! Deeper heading
```

Paragraph blocks

```
-> indented text
-< hanging indent
```

`<space> preformatted text`
`[@...@]` preformatted block
`----` (horizontal rule)
`blank line` is vertical space
`\` at end of line joins next line
`\\` at end of line produces a line break
`\\\` at the end of a line produces a blank line, even within a list item
`[[<<]]` produces a line break that clears floating content

Division blocks

See Block markup, Wiki styles and Page directives

```
>>wikistyle<<
>><<
(:div class="" style="":)
(:divend:)
```

Text markup

See Text formatting rules

Character format

```
''emphasized''
'''strong'''
'''''strong emphasis'''''
@@monospaced@@
[-small-], [--smaller--]
[+big+], [++bigger++]
'-small-', '+big+'
'^superscript^', '_subscript_'
{+inserted+} (underscore)
{-deleted-} (strikethrough)
[@escaped code@]
[=escaped text=]
```

Posting markup

`~~~` (author's signature)
`~~~~` (author's signature and date)
`(:encrypt` *`phrase`*`:)` -- replaced with encrypted form of*phrase*

Tables

Plain rows and columns of text

See Tables

```
||table attributes
||!table caption!||
||left aligned || centered || right aligned||
||!column heading||
||spanned columns ||||||
```

Structured tables

See Table directives

```
(:table attr:)
(:cellnr attr:)
(:cell attr:)
(:tableend:)
```

Directives

Page directives

See Page directives

```
(:redirect PageName:)

(:(no)spacewikiwords:)
(:(no)linkwikiwords:)
(:(no)linebreaks:)
```

Display

See Page directives Group headers

```
(:noheader:), (:nofooter:)
(:notitle:)
(:noleft:), (:noright:)
(:nogroupheader:), (:nogroupfooter:)
(:noaction:)
```

Metadata

See Page directives, Comment markup, Page variables

```
(:title text:)
(:keywords word, ...:)
(:description text:)
(:comment text:)
```
`{Group/PageName$:variable}` includes from `(:variable:text:)`

Include

See Include other pages, Page text variables

```
(:include PageName:)
(:include PageName#start#end lines=n paras=n:)
(:include Page1 Page2 Page3:)
```
`{Group/PageName$:Var}` includes from `(:name:text:)`
`(:nl:)` separate included text by conditional line break

Conditional markup

See Conditional markup

`(:if (!)` *cond param*`:)...(:ifend:)`
`(:if (!)` *cond param*`:)...(:else:)...(:ifend:)`
`(:if (!)` *cond param*`:)...(:elseif (!)` *cond param*`:)...(:ifend:)`

Pagelists

See Page lists

```
(:searchbox label=label order=-time:)
(:searchresults incl -excl group=abc fmt=def:)
(:pagelist incl -excl group=abc fmt=def:)
```

Other directives

See Page directives
`(:` attachlist`:)`
`(:` PageDirectives#markup`:) [=...=]`
```
(:markup:)...(:markupend:)
(:markup class=horiz:)...(:markupend:)
(:markup caption='...':)...(:markupend:)
(:messages:)
```

Forms

See Forms

```
(:input form method=get action=url enctype=multipart/form-data:)
     (:input default name=xyz value="abc":)
     (:input text name=first value="Bob" size=20:)
     (:input textarea name=xyz [=value=] rows=2 cols=80:)
     (:input submit name=post value="Go" accesskey=g:)
     (:input reset:)
     (:input hidden name=action value=edit:)
     (:input radio name=xyz value="abc" checked=1:)
     (:input checkbox name=xyz value="abc" checked=1:)
     (:input password name=authpw:)
     (:input file name=upload:)
     (:input image name=xyz src="http:..." alt="Alt Text":)
     (:input select name=xyz value="val1" label="Value 1":)
     (:input select name=xyz value="val2" label="Value 2":)
(:input end:)
```

See also PmWiki Edit forms.

Wiki trails

See Wiki trails

```
<<|[[TrailPage]]|>>
<|[[TrailPage]]|>
^|[[TrailPage]]|^
```

Page variables

See Page variables, Page text variables, Page lists
`{$variable}`, `{pagename$variable}`, `{groupname.pagename$variable}`
`{$:variable}`, `{pagename$:variable}`, `{groupname.pagename$:variable}`
Set a page text variable
`(:name:description:)`
`:name:description`
`name:description`
See special references
`{*$variable}`
`{*$:variable}`
Page list templates special variables
`{=$variable}`, `{<$variable}`, `{>$variable}`,
`{=$:variable}`, `{<$:variable}`, `{>$:variable}`,

Expressions

See Markup expressions
`{(function args)}`

Notify

The *notify.php* script allows a site administrator to configure PmWiki to send email messages whenever pages are changed on the wiki site. Notifications can be configured so that multiple page changes over a short period of time are combined into a single email message (to avoid flooding mailboxes).

This feature is useful for sites and pages that have infrequent updates, as it eliminates the need to frequently check RecentChanges pages just to see if anything has changed.

In order for notifications to work, the notify.php script must be enabled in the site'slocal customization. Usually this is as simple as placing the following in *local/config.php*:

```
$EnableNotify = 1;
```

Notification configuration

Once enabled, the notification system gets its configuration from the SiteAdmin.NotifyList wiki page. The SiteAdmin.NotifyList page contains entries of the form:

> notify=alice@example.com

This says that information about page changes should be periodically emailed to*alice@example.com*. The SiteAdmin.NotifyList page can contain multiple "notify=" lines to cause notifications to be sent to multiple addresses; the "notify=" lines can be concealed by placing them inside of an `(:if false:)` conditional section on the page.

NOTE: Do not put any spaces around the equal sign! Notifications will fail silently if you have... This is a really easy mistake to make because all of the other assignments have spaces around the equal sign.

> notify=fred@example.com rather than notify = fred@example.com

Notification options

The basic syntax is

```
notify=email@address name=abc group=def trail=ghi squelch=123 delay=123
```

A number of options exist for limiting the pages that result in a notification. The `group=` and `name=` parameters can be used to restrict notifications to specific pages or groups:

> # send notifications about the Main group to alice@example.com
> notify=alice@example.com **group**=Main
>
> # notify bob@example.com of any changes to the home page
> notify=bob@example.com **name**=Main.HomePage
>
> # notify charles@example.com of changes to pages except in Main
> notify=charles@example.com **group**=-Main

(Note: The options are similar to the PageList syntax.)

For maintaining arbitrary lists of pages, i.e., "watchlists", it's generally easier to build a trail of pages to be watched. The following entry in SiteAdmin.NotifyList will send alice@example.com an email containing changes to any of the pages listed in the Profiles.Alice trail:

```
# notify Alice of changes to pages listed in Profiles.Alice
notify=alice@example.com trail=Profiles.Alice
```

Note that once this entry has been added to SiteAdmin.NotifyList, Alice can easily change her watchlist by editing the Profiles.Alice page, and doesn't need to edit the SiteAdmin.NotifyList page. In particular, this means that an administrator can restrict editing of SiteAdmin.NotifyList, yet allow individuals to maintain custom watchlists in other pages.

Limitations of this feature:

- only manually-added links on a trail will be acknowledged by the Notify List (no "group=" or other pagelist syntax, nor any "Group.RecentChanges" links, will generate notifications)
- using an `(:include:)` directive on the page SiteAdmin.NotifyList is not an operational work-around.
- PageTextVariables are not resolved - you can't get the notification mail address from the profile page.

This is probably a good place to point out that edit access to SiteAdmin.NotifyList should be controlled, otherwise malicious persons can use the notification capability to flood others' electronic mailboxes. By default, SiteAdmin.NotifyList is blocked against reading or edits except by the admin (as is the case for most pages in the SiteAdmin group).

Adding notification entries via local customizations

Notification entries can also be added via the `$NotifyList` array in *local/config.php*. Simply add a line like the following:

```
$EnableNotify = 1;
$NotifyList[] = 'notify=alice@example.com group=Main';
$NotifyList[] = 'notify=bob@example.com name=Main.HomePage';
```

Controlling notification frequency

To prevent flooding of recipients' mailboxes, the notify script uses a "**squelch**" value as the minimum amount of time that must elapse between messages sent to any given email address. The *default squelch setting is 10800 seconds (three hours)*, which means that once a recipient address is sent a notification message, it will not receive another for at least three hours. Any edits that occur during the squelch interval are queued for the next notification message.

The site administrator can change the default squelch interval via the `$NotifySquelch` parameter

```
# enable notifications
$EnableNotify = 1;
$NotifySquelch = 86400; # wait at least one day (in seconds) between notifications
```

In addition, individual addresses can specify a custom squelch parameter in the SiteAdmin.NotifyList page:

```
# Alice receives at most one email per day
notify=alice@example.com squelch=86400

# Bob can get notifications hourly
notify=bob@example.com trail=Profiles.Bob squelch=3600

# Charles uses the site default squelch
notify=charles@example.com
```

Controlling notification delay

Because a page will often receive several edits in rapid succession (e.g., a long post followed by several minor edits), a site administrator can also set a `$NotifyDelay` value that specifies how long to wait after an initial post before sending notifications:

```
# enable notifications
$EnableNotify = 1;
$NotifySquelch = 86400; # wait at least one day between notifications
$NotifyDelay = 300; # wait five minutes after initial post
```

Note that the squelch and delay values are minimums; notifications are sent on the first execution of PmWiki after the delay period has expired. For inactive sites, this could be much longer than the specified delay periods. This isn't really considered an issue since timely notifications are less important on relatively inactive sites. However, changes within the squelch time after the last notification will remain unnoticed if the wiki is not even visited for a long period after. If this matters it might be necessary to make the server call pmwiki.php regularly (e.g. cron job).

Custom delay parameters cannot be specified for individual addresses in the SiteAdmin.NotifyList page:

    # the delay= parameter will be ignored
    notify=edgar@example.com trail=Profiles.Edgar **delay**=600

Note for Windows installations

Sites running PHP under Windows may not have PHP's mail function configured correctly. Such sites may need to add a line like

```
ini_set('SMTP','smtp.server.com');
```

to *config.php*, where *smtp.server.com* is the name of your host's preferred outgoing mail server. You may also need to set the sendmail_from value if that is not configured:

```
ini_set('sendmail_from','noreply@foo.com');
```

Notify Variables

`$EnableNotify`
: Tells *stdconfig.php* to enable the notify script.
`$EnableNotify` = 1; # enable notify
`$EnableNotify` = 0; # disable notify

`$NotifyFrom`
: Return email address to be used in the sent email.
`$NotifyFrom` = 'wiki@example.com';
`$NotifyFrom` = 'Wiki server <wiki@example.com>';

`$NotifyDelay`
: The length of time (seconds) to wait before sending mail after the first post. Defaults to zero - posts are sent as soon as any squelch period has expired.
`$NotifyDelay` = 300; # send mail 5+ min after first post

`$NotifySquelch`
: The default minimum time (seconds) that must elapse between sending mail messages. Useful when `$NotifyDelay` is set to a small value to keep the number of mail notification messages down. Defaults to 10800 (three hours). Individual recipients can override this value in the SiteAdmin.NotifyList page.
`$NotifySquelch` = 43200; # wait 12+ hours between mailings

`$NotifyItemFmt`
: The text to be sent for each changed item in the post. The string "$PostTime" is substituted with the time of the post (controlled by `$NotifyTimeFmt` below).
# default
`$NotifyItemFmt` = ' * $FullName . . . $PostTime by`$Author`';

# include the page's URL in the message
`$NotifyItemFmt` =
" * \$FullName . . . \$PostTime by \`$Author`\n \$PageUrl";

# include the change summary and link to the page's history in the message
`$NotifyItemFmt` =
" * {\$FullName} . . . \$PostTime by {\`$Author`}
\n Summary: {\$LastModifiedSummary}\n {\$PageUrl}?action=diff";

`$NotifyTimeFmt`
: The format for dates/times in $PostTime above. Defaults to the value of `$TimeFmt`.
`$NotifyTimeFmt` = '%Y-%m-%d %H:%M'; # 2004-03-20 17:44

`$NotifyBodyFmt`
: The body of the message to be sent. The string "$NotifyItems" is replaced with the list of posts (as formatted by `$NotifyItemFmt` above). Use single quotation marks ' to prevent substring "$NotifyItems" from being untimely evaluated as variable in config.php.
`$NotifyBodyFmt` = "Changed items:\n\n" . '$NotifyItems' . "\n\n Best regards...";

`$NotifySubjectFmt`
: The subject line of the mail to be sent.

`$NotifyHeaders`
: String of extra mail headers to be passed to the mail() function.

`$NotifyParameters`
: String of additional parameters to be passed to PHP's mail() function [1].

`$NotifyFile`

The scratch file where Notify keeps track of recent posting information. Defaults to `"$WorkDir/.notifylist"`. Note that this file must generally be writable by the webserver process.

`$NotifyListPageFmt`
: The name of the page containing `notify=` lines for use by *notify.php*. Defaults to `$SiteAdminGroup.NotifyList`.

`$NotifyList`
: An array of `notify=` specifications that can be specified from a local customization file (used in addition to entries in SiteAdmin.NotifyList).
# send notifications to alice@example.com
`$NotifyList`[] = 'notify=alice@example.com';

`$EnableNotifySubjectEncode`
: Apply a standard (base64) encoding for the e-mail subject. Notify e-mails from international wikis may otherwise have unreadable subjects (added for version 2.2.2).
`$EnableNotifySubjectEncode` = 1; # encode subject
`$EnableNotifySubjectEncode` = 0; # use subject as is (default)
To fix encodings with the message body, add to config.php the following line (after XLPage and/or UTF-8):
`$NotifyHeaders = "Content-type: text/plain; charset=$Charset";`

Notification only for major edits

It is possible to send notifications only in case of major edits. In your config.php, replace "`$EnableNotify`=1;" with the following:

if (@$_POST['diffclass'] != 'minor') `$EnableNotify`=1;

This way, only 'major' edits send notify messages (when the author doesn't select the checkbox for minor edit). If you want minor edits and not major edits to send the message then you would use:

if (@$_POST['diffclass'] == 'minor') `$EnableNotify`=1;

instead.

Disabling notifications for downloads

If you use "$EnableDirectDownloads=0;" (eg. for privacy on a password-protected wiki) then attached images may generate duplicate notification messages. To prevent that disable notifications for downloads via

if ($action != 'download') `$EnableNotify`=1;

That way, only page views (and not images within the page) will generate notifications. See PITS:01159 for more information.

Last modified by Petko on December 04, 2013. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Notify

OtherVariables

toc top

`$FmtV`
: This variable is an array that is used for string substitutions at the end of a call to `FmtPageName()`. For each element in the array, the "key" (interpreted as a string) will be replaced by the corresponding "value". The variable is intended to be a place to store substitution variables that have frequently changing values (thus avoiding a rebuild of the variable cache making `FmtPageName()` faster). Also see `$FmtP`. *Values of `$FmtV` are set by the internal functions FormatTableRow, LinkIMap, HandleBrowse, PreviewPage, HandleEdit, PmWikiAuth, and PasswdVar, apparently to set values for system generated string substitutions like PageText.*

`$FmtP`
: This variable is an array that is used for pattern substitutions near the beginning of a call to `FmtPageName`. For each element in the array, the "key" (interpreted as a pattern) will be replaced by the corresponding value evaluated for the name of the current page. This is for instance used to handle $-substitutions that depend on the pagename passed to `FmtPageName()`. Also see `$FmtV`. From robots.php: *If $EnableRobotCloakActions is set, then a pattern is added to `$FmtP` to hide any "?action=" url parameters in page urls generated by PmWiki for actions that robots aren't allowed to access. This can greatly reduce the load on the server by not providing the robot with links to pages that it will be forbidden to index anyway.*

`$FmtPV`
: This variable is an array that is used for defining Page Variables. New variables can be defined with `$FmtPV['$VarName'] = 'variable definition';` which can be used in markup with `{$VarName}`. Please note that the contents of `$FmtPV['$VarName']` are `eval()`ed to produce the final text for `$VarName`, so the contents must be a PHP expression which is valid at the time of substitution. In particular, this does not work:

```
#This doesn't work
$FmtPV['$MyText'] = "This is my text."; # WARNING: Doesn't work!
```

The problem is that the text `This is my text.` is not a valid PHP expression. To work it would need to be placed in quotes, so that what actually gets stored in `$FmtPV['$MyText']` is `"This is my text."` which **is** a valid PHP expression for a text string. Thus the correct way to do this would be with an extra set of quotes:

```
#This will work
$FmtPV['$MyText'] = '"This is my text."';
```

This also has implications for how internal PHP or PmWiki variables are accessed. To have the page variable `$MyVar` produce the contents of the internal variable `$myvar`, many folks try the following which does not work:

```
#This doesn't work either!
$myvar = SomeComplexFunction();
$FmtPV['$MyVar'] = $myvar; # WARNING: Doesn't work!
```

There are several correct ways to do this, depending on whether you need the value of the `$myvar` variable as it was at the time the `$FmtPV` entry was created, or at the time that a particular instance of `$MyVar` is being rendered on a page. For most simple page variables that don't change during the processing of a page its more efficient to set the value when the entry is created:

```
$myvar = SomeComplexFunction();
$FmtPV['$MyVar'] = "'" . $myvar . "'"; #capture contents of $myvar
```

NOTE: If `$myvar` should contain single quotes, the above won't work as is, and you'll need to process the variable to escape any internal quotes.

For more complex cases where an internal variable may have different values at different places in the page (possibly due to the effects of other markup), then you need to make the `$FmtPV` entry make an explicit reference to the global value of the variable (and the variable had better be global) like this:

```
global $myvar;
$FmtPV['$MyVar'] = '$GLOBALS["myvar"]';
```

Finally, there's nothing to stop you from simply having the evaluation of the `$FmtPV` entry execute a function to determine the replacement text:

```
# add page variable {$Today}, formats today's date as yyyy-mm-dd
$FmtPV['$Today'] = 'strftime("%Y-%m-%d", time() )';
```

Once again, please note that the values of the elements of `$FmtPV` are `eval()`ed so always sanitize any user input. The following is very insecure:

```
$FmtPV['$Var'] = $_REQUEST['Var']; # critically insecure, allows PHP code injection
$FmtPV['$Var'] = '"'. addslashes($_REQUEST['Var']).'"'; # critically insecure, allows PHP code injection
```

See the recipe Cookbook:HttpVariables for a better way to use these variables.

See Cookbook:MoreCustomPageVariables for more examples of how to use `$FmtPV`.

`$MaxPageTextVars`
: This variable prevents endless loops in accidental recursive PageTextVariables which could lock down a server. Default is 500 which means that each PageTextVariable from one page can be displayed up to 500 times in one wiki page. If you need to display it more than 500 times, set in config.php something like
`$MaxPageTextVars = 10000;` # ten thousand times

`$PageCacheDir`
: Enables the cache of most of the HTML for pages with no conditionals. The variable contains the name of a writable directory where PmWiki can cache the HTML output to speed up subsequent displays of the same page. Default is empty, which disables the cache. See also `$PageListCacheDir`.

```
# Enable HTML caching in work.d/
$PageCacheDir = 'work.d/';
```

Last modified by Petko on December 19, 2015. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/OtherVariables

PageDirectives

toc top

PmWiki uses a number of directives to specify page titles, descriptions, page keywords, and control the display of various components. Directive keywords are not case sensitive, e.g. Description, description, and DESCRIPTION are equivalent.

`(:attachlist:)`
: Shows a list of attachments of the current group or page, depending on whether attachments are organised per group or

per page. The attachlist is displayed at the foot of the uploads page form.
The parameter to (:attachlist:) always resolves to a pagename. The directive then displays all of the attachments currently available for that page.

Options
- `(:attachlist NAME:)` shows a list of attachments of the group or page NAME.
- `(:attachlist ext=xxx:)` specifies an extension for filtering by type of file.
- `(:attachlist *:)` shows the uploads directory and permits browsing of all uploaded files by directory (*will not work if `$EnableDirectDownload` is set to 0*).

`(:description text:)`
: Descriptive text associated with the page. (Generates a `<meta name='description' content='...' />` element in the page output.)

`(:keywords word1, word2, ...:)`
: Identifies keywords associated with the page. These are not displayed anywhere, but are useful to help search engines locate the page. (Essentially, this generates a `<meta name='keywords' content='...' />` element in the output.)

`(:linebreaks:)`, `(:nolinebreaks:)`
: Honors any newlines in the markup; i.e., text entered on separate lines in the markup will appear as separate lines in the output. Use `(:nolinebreaks:)` to cause text lines to automatically join again.

`(:linkwikiwords:), (:nolinkwikiwords:)`
: Enables/disables WikiWord links in text. Note, this setting requires WikiWords to be enabled, see `$EnableWikiWords`. See also `$LinkWikiWords`.

`(:markup:) ... (:markupend:)` or `(:markup:)[=...=]`
: Can be used for markup examples, showing first the markup and then the result of the markup.

Options
- `(:markup class=horiz:)` will show the markup side by side instead of one upon the other.
- `(:markup caption='...':)` adds a caption to the markup example.
- `(:markupend:)` is not required when using `(:markup:) [=...=]`.

Note that the placement of newlines is very important for this markup. If you are using the `[=...=]` option then the opening `[=` MUST occur on the same line as the `(:markup:)`. If you are using the full `(:markup:) ... (:markupend:)` form then your markup code must appear AFTER a newline after the initial `(:markup:)`.

`(:messages:)`
: Displays messages from PmWiki or recipes, for instance from editing pages.

`(:noaction:)`
: Turns off the section of the skin marked by <!--PageActionFmt--> thru <!--/PageActionFmt-->. In the pmwiki skin, this turns off the display of the actions at the top-right of the page (other skins may locate the actions in other locations). The actions at the bottom of the page are still available.

`(:nogroupheader:)`
`(:nogroupfooter:)`
: Turns off any groupheader or groupfooter for the page. (See GroupHeaders.)

`(:noheader:), (:nofooter:)`
`(:noleft:), (:noright:), (:notitle:)`
: If supported by the skin, each of these turns off the corresponding portion of the page.

`(:redirect PageName:)`
: Redirects to another wiki page.

`(:redirect PageName#anchor:)`
: Redirects to an anchor within a page

`(:redirect PageName status=301 from=name quiet=1:)`
: Redirects the browser to another page, along with a redirect message. For security reasons this only redirects to other pages within the wiki and does not redirect to external urls. The `status=` option can be used to return a different HTTP status code as part of the redirect. The `from=` option limits redirects to occuring only on pages matching the wildcarded *name* (helpful when `(:redirect:)` is in another page). The `quiet=1` option allows the target page not to display a link

back to the original page (`$EnableRedirectQuiet` variable should be set to 1).

`(:spacewikiwords:)`, `(:nospacewikiwords:)`
: Enables/disables automatic spacing of WikiWords in text.

`(:title text:)`
: Sets a page's title to be something other than the page's name. The title text can contain apostrophes and other special characters. If there are multiple titles in a page, the last one encountered wins (see also `$EnablePageTitlePriority` about how to change it).

Can I get `(:redirect:)` to return a "moved permanently" (HTTP 301) status code?

> Use `(:redirect PageName status=301:)`.

Is there any way to prevent the "redirected from" message from showing at the top of the target page when I use `(:redirect:)` ?

> From version 2.2.1 on, set in config.php `$EnableRedirectQuiet=1;` and in the page `(:redirect OtherPage quiet=1:)` for a quiet redirect.

Is there any method for redirecting to the equivalent page in a different group, i.e. from BadGroup/thispage => GoodGroup/thispage using similar markup to

> Page redirects to Goodgroup.{Name} ?
> (:redirect Goodgroup.{$Name}:) works if you want to put it in one page.
>
> If you want it to work for the entire group, put (:redirect Goodgroup.{*$Name}:) into Badgroup.GroupHeader - however, that only works with pages that really exist in Goodgroup; if you visit a page in Badgroup without a corresponding page of the same name in Goodgroup, instead of being redirected to a nonexistant page, you get the redirect Directive at the top of the page.
>
> With (:if exists Goodgroup.{*$Name}:)(:redirect Goodgroup.{*$Name}:)(:ifend:) in Badgroup.GroupHeader you get redirected to Goodgroup.Name if it exists, otherwise you get Badgroup.Name without the bit of code displayed.

How can a wiki enable linebreaks by default, i.e. without having the directive `(:linebreaks:)` in a page or in a GroupHeader?

> Add to config.php such a line:
> `$HTMLPNewline = '<br/>';`

PageFileFormat

toc top

You may have many documents that you would like to use a local program to format in a format PmWiki can display.

You could open each document and copy/paste the content to new pmwiki pages or you could format the document in advance and upload it using an FTP client.

Only two lines are necessary in a PmWiki page file:

```
version=pmwiki-2.1.0 urlencoded=1
text=Markup text
```

"version=" tells PmWiki that the values are urlencoded. The actual value doesn't matter, as long as "urlencoded=1" appears somewhere in the line.

"text=" *needs* to have the markup text with newlines converted to "%0a" and percent signs converted to "%25".

> In addition, *PmWiki writes* pages with '<' encoded as "%3c" (to help with security), but it doesn't require that <'s be encoded that way in order to be able to read the page. More conversions are possible to be added in the future.

In order to let the `(:pagelist :)` markup work, make sure the filename begins with an uppercase letter.

In order to have the `(:pagelist link= ... :)` markup on other pages list this page, a third attribute is required:

```
targets=GroupName1.Pagename1,GroupName2.Pagename2,...
```

"targets=" is a comma delimited list of all links from the current page (no space following the comma).

Keys you could see in a raw PmWiki file:

version
: Version of PmWiki used to create the file

agent
: Author's browser when saving the page

author
: Last author to save page

charset
: The character encoding of the page text

csum
: Change summary

ctime
: Page creation time

description
: Page description. Used to fill `<meta name='description' />` if set via `(:description page'sdecription text:)`

host
: Host created this page

name
: Name of the page (e.g., `Main.WikiSandbox`)

passwdattr
: encrypted version of the password required to change attributes

passwdedit
: encrypted version of the password required to edit

passwdread
: encrypted version of the password required to read

passwdupload
: encrypted version of the password required to upload

rev
: Number of times the page has been edited

targets
: Targets for links in the page

text
: The page's wiki markup

time
: Time the page was last saved (seconds since 1 Jan 1970 00:00 UTC)

title
: Page title set via `(:title The Page Title:)`.

newline
: Character used for newlines (deprecated)

updatedto
: The version to which PmWiki has been updated to by `upgrades.php` (only on SiteAdmin.Status)

Below these you will see information used to keep track of the page's revision history.

Creating a Page for Distribution

A simple way to create a wikipage file to use for distribution (for example with a recipe or a skin) is to create the page with PmWiki and then use a text editor to delete all lines but *version*, *text*, and *ctime*. Example:

```
version=pmwiki-2.1.0 ordered=1 urlencoded=1
text=This is a line.%0aThis is another.
ctime=1142030000
```

Keeping track of page history

Inside of a page file, PmWiki stores the latest version of the markup text, and uses this to render the page. The page history is kept as a sequence of differences between the latest version of the page and each previous version.

PmWiki normally puts the page history at the end of each page file in reverse chronological sequence, and sets the "ordered=1" items in the header. If an operation needs only the most recent version of a page, then PmWiki will stop reading and processing a page file at the point where the history begins, potentially saving a lot of time and memory. If the "ordered=1" flag isn't present, PmWiki makes no assumptions about the ordering of items in the pagefile and processes the entire file.

Load pages from text files

See Cookbook: Import text.

Unix utility to extract wiki text

This one-line sed command extracts and prints the text of a PmWiki 2.x file (could be aliased, eg*pmcat*):

```
sed -n 's/^text=//; s/%0a/\n/gp; s/%3c/</gp; s/%25/%/gp' GroupName.PageName
```

The following unix script (tested on MacOSX) will extract and decode the current text from a wiki file:

```
    #!/bin/tcsh
    # wtext - extract wiki text
    #
    # wtext wikifile > output

    set fn = "$1"
    if ("$fn" == "") then
      echo "need input file parameter"
      exit 999
    endif
    if (! -f $fn) then
      echo "$fn does not exist"
      exit 999
    endif
    rm sedin.$$ >& /dev/null
    set ch = `grep ^newline= $fn | cut -d= -f2`
    if ("$ch" == "") set ch = "%0a"
    cat <<eof > sedin.$$
    s/^text=//
    s/$ch/\
    /g
    s/%3c/</g
    s/%25/%/g
    eof
    grep "^text=" "$1" | sed -f sedin.$$
    rm sedin.$$ >& /dev/null
```

See also

- Cookbook:AdminByShell A collection of ways to assist sysadmin of pmwiki using shell tools
- Cookbook:PageTopStore A PageStore alternative which doesn't mangle page contents when viewed outside PmWiki

Categories: PmWiki Developer

PageHistory

When PmWiki is called with '?action=diff', it displays a summary of past edits on a page. Each past edit is shown in a box which shows lines added, changed or deleted during that edit in a before & after format.

Below each box is a "Restore" link. Clicking the link will open an edit box with the page as it was *before* that edit. You can make changes or simply click Save to restore the text.

There are two additional options specific to PageHistory:

- **Hide minor edits** - hides any edit that the author marked as 'minor'. - This is done by adding "&minor=n" to "?action=diff". The default value for this is to show minor edits with "&minor=y"

- **Show changes to output** - It shows changes to the rendered output (as opposed to the normal display which shows changes to the wiki-markup). This is done by adding "&source=n" to "?action=diff". You can show changes to markup (the default behavior from 2.2.13) with "&source=y".

- You can set both by using "?action=diff&source=y&minor=y".

In the default mode "Show changes to markup", you can disable word-level highlighting of differences by adding to config.php such a line:

```
  $EnableDiffInline = 0;
```

A page's history is kept for the number of days given by the `$DiffKeepDays` and `$DiffKeepNum` variables (set by the site's wiki administrator). When a page is edited, any page history information older than both these values is automatically discarded.

Note that a specific page revision isn't removed from the page until the first edit *after* the time specified by `$DiffKeepDays` has elapsed. Thus, it's still possible for some pages to have revisions older than `$DiffKeepDays` -- such revisions will be removed the next time those pages are edited.

See also

- recent changes special pages

- Cookbook:ExpireDiff
- Cookbook:LimitDiffsPerPage
- Cookbook:ViewDiff
- Cookbook:TrackChanges

Is there a way to remove page history from page files?

1. Administrators can clean page histories using the Cookbook:ExpireDiff recipe.

2. Administrators with FTP file access can download individual pages from the wiki.d directory, open them in a text editor, manually remove history, and re-upload the files to wiki.d/ directory. Care must be exercised, when manually editing a page file, to preserve the minimum required elements of the page and avoid corrupting its contents. See PageFileFormat#creating.

3. Edit the page. Select *all* the contents of the edit text area and cut them to the clipboard. Enter `delete` into the text area and click on the *save and edit* button. Select *all* the contents of the edit text area and paste the contents of the clipboard over them. Click on the *save* button. This will remove all of the page's history up to the final save in which the pasted material is re-added.

How can I restrict viewing the page history `(?action=diff)` to people with edit permission?

In the *local/config.php* file, set

```
$HandleAuth['diff'] = 'edit';
```

In case of this restriction is set up on a farm, and you want to allow it on a particular wiki, set in your local/config.php :

```
$HandleAuth['diff'] = 'read';
```

PageListTemplates

Default page list templates

PmWiki's default templates for page lists are in Site.PageListTemplates, which is replaced during upgrades. These default templates can be supplemented or overridden with custom templates stored in other locations.

If the page name is not specified as part of the template name, PmWiki's default configuration looks for templates in the following locations in the following order

1. the current page
2. Site.LocalTemplates,
3. Site.PageListTemplates

Administrators can change those locations by using the `$FPLTemplatePageFmt` variable.

If the template is on the current page, the current page must be saved for changes involving the template to show up *(preview alone will not work)*.

Custom page list templates

Custom templates are used in the same way as default templates: by referencing the desired format with the `fmt=#`*`anchor`* option. There are several ways to indicate which template to use:

- `fmt=#custom` uses the #custom section from the current page, Site.LocalTemplates, or Site.PageListTemplates, (sections are denoted by `[[#custom]]` anchors.
- `fmt=MyTemplatePage#custom` uses a custom format from page `MyTemplatePage` from its `#custom` section.
- `fmt=MyTemplatePage` uses a custom format from the entire page `MyTemplatePage`.
- `fmt=custom` uses custom format which is defined in a cookbook script as *custom*.

See Cookbook:PagelistTemplateSamples for examples of custom pagelist formats.

Creating page list templates

A pagelist template contains standard pmwiki markup. When creating pagelist output, pmwiki iterates over each page returned from the pagelist and will include the pagelist template markup once for every page in the list.

Special references

During the page list iteration pmwiki sets 3 special page references: **=**, **<** and **>**. These special page references are updated on each pagelist iteration and can be used with the page variables syntax, such as *{=$variable}*, to define a pagelist template which will format the pagelist output. The meaning of the special references are:

= current page	so *{=$Title}*	displays the title of the current page in the iteration
< previous page	so *{<$Group}*	displays the group of the previous page in the iteration
> next page	so *{>$Name}*	displays the name of the next page in the iteration

The > and < references are most useful to help structure pagelist output before and after the actual pagelist. Some common tests used to structure pagelist output are:

`(:template first:)`	`(:if equal {<$Group}:)`	Iteration is at the beginning of list
`(:template last:)`	`(:if equal {>$Group}:)`	Iteration is at the end of list
`(:template first {=$Group}:)`	`(:if ! equal {=$Group} {<$Group}:)`	Iteration is at the first item in a group
`(:template last {=$Group}:)`	`(:if ! equal {=$Group} {>$Group}:)`	Iteration is at the last item in a group
`(:template defaults:)`		Default options to be used in the pagelist command
`(:template each:)`		Signifies the repeated part

Note: the markup in column 2 is deprecated.
See also page variable special references.

Page list template special markup

Pagelist templates may have special sections

- `(:template first ...:)` and `(:template ! first ...:)`
- `(:template last ...:)` and `(:template ! last ...:)`

to specify output for the first or last page in the list or a group (use`! first` and `! last` for output *except* for the first/last page).

There's also a

- `(:template defaults ...:)` to allow a template to specify default options,
- `(:template each ...:)` to signify the repeated part, and
- `(:template none:)` whose content will appear if no page was found (from version 2.2.5).

These allow Pagelist templates to be easily separated into "sections" that are included or not included in the output based on a variety of conditions. These are intended to be improved versions of the (:if ...:) conditions that have traditionally been used to control pagelist output (however, the (:if:) conditions still work as before).

First, Each, Last, None

The simplest versions of the directives are:

```
(:template first:)      # markup to display only for first page in list
(:template ! first:)    # markup to display for every page in list but the first
(:template each:)       # markup to display for each page in list
(:template last:)       # markup to display only on last page in list
(:template ! last:)     # markup to display for every page in list but the last
(:template none:)       # markup to display only if no pages were found
```

So, a pagelist template can specify:

```
(:template first:)
Pages in the list:
(:template each:)
* [[{=$FullName}]] [-{=$:Summary}-]
(:template last:)
Displayed {$$PageCount} pages.
```

In addition, the "first" and "last" options can have control break arguments that identify markup to be displayed on the first or last page within a particular control section. For example, to specify markup to be displayed upon reaching the first or last page of a group, one can use

```
(:template first {=$Group}:)
(:template last {=$Group}:)
```

Thus, instead of writing control breaks using directives, as in

```
(:if ! equal {<$Group} {=$Group}:)
Group: {=$Group}
(:ifend:)
* [[{=$FullName}]]
```

one can now write

```
(:template first {=$Group}:)
Group: {=$Group}
(:template each:)
```

```
* [[{=$FullName}]]
```

Page text variables and page variables can also be used, for example

```
(:template default $:Maintainer=- order=$:Maintainer,name:)
(:template first {=$:Maintainer}:)
```

Default options

In addition, a template may specify default options to be used in the pagelist command. For example, a pagelist template to display a list of pages by their titles (and sorted by title) might use:

```
[[#bytitle]]
(:template defaults order=title:)
* [[{=$FullName}|+]]
[[#bytitleend]]
```

Then an author could write (:pagelist fmt=#bytitle:) and the pages would automatically be sorted by title without having to specify an additional "order=title" option to the (:pagelist:) directive.

To specify multiple parameters to an option enclose the parameters in double quotes, eg to sort by apage text variable and then the page name

```
(:template defaults order="$:Database,name" :)
```

Examples

(:template defaults ... :)
: default options for pagelists using this template

(:template each:)
: markup for each page in the pagelist

(:template first:)
: markup output only for the first page in the pagelist

(:template last:)
: markup output only for the last page in the pagelist

(:template first {=$Group}:)
: markup output only for a page where the value of {=$Group} has just changed

(:template last {=$Group}:)
: markup output only for a page where the value of {=$Group} will change with the next page

So, we have:

```
[[#template]]
(:template defaults order=name:)
(:template first:)
Pages ordered by group
(:template first {=$Group}:)

Pages in group [[{=$Group}/]]
(:template each:)
* [[{=$FullName}]]
(:template last {=$Group}:)
   {=$Group} contains {$$GroupPageCount} pages.
(:template last:)
   {$$PageCount} pages total.
[[#templateend]]
```

Page list template additional page variables

Additional Page Variables that are only available during pagelist are:

```
{$$PageCount}       The current page count of this iteration
{$$GroupCount}      The current group count of this iteration
{$$GroupPageCount}  The current page count within the current group of this iteration
{$$option} The argument option values from (:pagelist:)
```

Use of {$$*option*}: For example {$$trail} returns the page name entered in the trail= option of the pagelist directive. You can make up custom "options" with no other purpose than being displayed in the pagelist.

Redirect

To enable searches that return only one page to automatically redirect to that page add the following to a pagelist template where the "jump to a page" functionality is desired:

```
(:template last:)
(:if equal {$$PageCount} 1:)(:redirect {=$FullName}:)(:ifend:)
```

Closure of markup

Any open tables, divs, or other structures inside of (:pagelist:) are, by default, automatically closed at the end of the pagelist command. In other words, (:pagelist:) acts like its own complete page, as opposed to generating markup that is then inserted into the enclosing page.

For example a table generated by the (:cell:) directive in the first (:pagelist:) command is automatically closed at the end of the pagelist. The (:cell:) in the second (:pagelist:) command then starts a new table.

Note that the (:table:) directive doesn't actually start a new table, it's the (:cell:) or (:cellnr:) directive that does it. All that the (:table:) directive does is set attributes for any tables that follow.

Usage

It is advisable to not modify the page Site.PageListTemplates directly so that you will still benefit from upgrades. Instead, modify your Site.LocalTemplates page (which is not part of the PmWiki distribution). Cookbook:PagelistTemplateSamples has many examples of custom pagelist formats.

Other recipes

In addition, the Cookbook has other recipes for special `fmt=` options, including `fmt=dictindex` (alphabetical index) and `fmt=forum` (forum postings).

Last modified by simon on April 20, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PageListTemplates

PageLists

toc top

PmWiki comes with two directives for generating lists of pages -- `(:pagelist:)` and `(:searchresults:)`. Both directives are basically the same and each accepts the parameters documented below. The primary difference between the two is that searchresults generates the "Results of search for ..." and "### pages found out of ### searched" messages around the results.

The `(:searchbox:)` directive generates a search form (input text box) to submit search queries. The markup generally accepts the same parameters as `(:pagelist:)`, which makes it possible to restrict, order and format searchresults in the same ways that are described below for a `(:pagelist:)`. For more information about the `(:searchbox:)` directive, and the ways in which it differs from a `(:pagelist:)`, skip to the section below.

Basic syntax

- `(:pagelist:)`
  without any arguments shows a bulleted list of all pages, as links, ordered alphabetically and in groups.
- `(:pagelist group=`*ab*` name=`*cd*` fmt= `*template*` list=`*ef*` order=`*gh*` count=`*123*` link=`*ij*` trail=`*kl*` wrap=`*mn*` passwd=`*op*` if=`*qr*` $:`*ptv*`=`*st*` $`*pv*`=`*uv*` cache=0 `*argument1* - *argument2* *etc*` variable=`*value*` class=`*class* *request=1* *req=1*` :)`
  shows a pagelist according to the parameters supplied. Parameters are optional.
- `(: searchbox value=`*abc*` size=`*99*` target=`*def*` label="label":)`
- `(: searchresults:)`

Parameters

Any argument supplied within `(:pagelist:)` that isn't in the form `'key=value'` is treated as text that either must (or must not) exist in the page text.

The minus sign (-) can be used to indicate things that should be excluded. Thus

```
(:pagelist trail=PmWiki.DocumentationIndex list=normal apple -pie:)
```

lists all "normal" pages listed in the Documentation Index trail that contain the word "apple" but not "pie".

With page text variables

You can also use page text variables as a *key* to list pages according to the existence of a page text variable. Eg :

```
(:pagelist $:pagetextvar=avalue:)
```

lists pages having *$:pagetextvar* set to *avalue*.
Minus sign (-), wildcards (?*) and a comma separated list of values also works when specifying a selection based on pagetextvariables. Eg :

```
(:pagelist $:apagetextvar=t*,-test:)
```

lists pages having $:apagetextvar like 't*' but not 'test'.
Examples:

PTV is set (is not empty):	`(:pagelist $:MyPageTextVariable=- :)`
PTV is empty or not set: (ie, is not set to one char followed by 0 or more	`(:pagelist $:MyPageTextVariable=-?* :)`

chars)	
PTV is not VALUE:	`(:pagelist $:MyPageTextVariable=-VALUE :)`
PTV is set and not YES:	`(:pagelist $:MyPageTextVariable=?*,-YES :)`

Be aware that if using `(:pagelist $:MyPTV=$:YourPTV :)` PTVs include PmWiki formatting, so you may not get the matches you expect. Currently the only way around this is to use wild cards, so if the formatting is embedded you may be out of luck.

NOTE: Pagelist does not evaluate MarkupExpressions when working with PTVs. So if your page text variables is defined using a markup expression to set the value, pagelist will see the literal values of the text of your markup expression rather than the result of your expression. (e.g., the PTV definition `(:foo:{(substr abcdef 2 4)}:)` will be seen by pagelist as an open-curly-brace followed by an open-paren followed by s-u-b-s-t-r, etc. rather than being seen as b-c-d-e) Any processing of the markup expression in the output of your pagelist occurs in subsequent rules (after pagelist) within the context of the current page and thus these values cannot be used for sorting or selecting pages. (source)

With page variables (PV)

Page variables can be used within pagelists in the same way as page text variables. See Page Text Variables above for more details. Simply use $var instead of $:var.

group= and name=

The "`group=`" and "`name=`" parameters limit results to pages in a specific group or with a specific name:

All pages in the Pmwiki group:	`(:pagelist group=PmWiki :)`
All pages except those in the PmWiki or Site groups:	`(:pagelist group=-PmWiki,-Site :)`
All RecentChanges pages	`(:pagelist name=RecentChanges :)`
All pages except RecentChanges	`(:pagelist name=-RecentChanges :)`

Wildcards

Name and group parameters can contain *wildcard* characters that display only pages matching a given pattern:

- An asterisk (*) represents zero or more characters
- A question mark (?) represents exactly one character

Examples:

All pages in any group beginning with "PmWiki"	`(:pagelist group=PmWiki* :)`
All pages in any group beginning with "PmWiki", except for Chinese	`(:pagelist group=PmWiki*,-PmWikiZh :)`
All pages in the PmCal group with names starting with "2005":	`(:pagelist name=PmCal.2005* :)`

All Cookbooks with names beginning with a A and a B letter
note the different separators used for the same result
	`(:pagelist group=Cookbook name=A*,B*   :)`
	`(:pagelist group=Cookbook name="A* B*" :)`
	`(:pagelist group=Cookbook name=[AB]*   :)`
	`(:pagelist group=Cookbook, name=[AB]*   :)`

If you want to use multiples conditions in name you need to use quotes or commas to delimit the string. For example

```
key="one value,another value"
```

trail=

The "`trail=`" option obtains the list of pages to be displayed from a WikiTrail:

- Display pages in the documentation by modification time
  `(:pagelist trail=PmWiki.DocumentationIndex order=-time:)`
- Display five most recently changed pages
  `(:pagelist trail=RecentChanges count=5:)`

list=

The "`list=`" option allows a search to include or exclude pages according to predefined patterns set by the administrator.

- "`list=normal`" is predefined, and which excludes things like AllRecentChanges, RecentChanges, GroupHeader, GroupFooter, GroupAttributes, and the like from being displayed in the list results. Note that list=normal also excludes the current page.
- "`list=all`" over-rides a "default" list that may be set by the wiki's administrator to exclude groups such as PmWiki or Site from regular search results.
- Wiki administrators can define custom lists via the `$SearchPatterns` array (see Cookbook:SearchResults).

fmt=

The "`fmt=`" option determines how the resulting list should be displayed. PmWiki predefines several formats:

- `fmt=#bygroup` - Display pages within groups (default format)
- `fmt=#simple` - Display a simple ordered list of pages in the form Group.Name
- `fmt=#title` - Display a list of pages by page title. Use "`order=title`" to have them sorted by title (default is to order by page name).
- `fmt=#titlespaced` - Display a list of pages by page title, like above, but with spaces between the words in the title.
- `fmt=#group` - Display a list of wikigroups (without listing the pages in the groups)
- `fmt=#include` - Display the contents of each page in the list (note, this could take a very long time for long lists!)

These formats are defined by page list templates, which can be customized.

This format is not predefined by a page list template:

- `fmt=count` - Display the number of pages in the list (note the absence of the "#"). In a trail`fmt=count` counts existing and non-existing pages ; to limit count to existing pages, use : `if="exists {=$FullName}" fmt=count` (mailing list).

link=

The "`link=`" option implements "backlinks" -- i.e., it returns a list of pages with a link to the target. It's especially useful for category pages and finding related pages.

- all pages with a link to PmWiki.DocumentationIndex
  `(:pagelist link=PmWiki.DocumentationIndex:)`
- all pages with links to the current page
  `(:pagelist link={$FullName}:)`
- all pages in the "Skins" category
  `(:pagelist link=Category.Skins:)`

Note that the `link=` parameter doesn't accept multiple or negative targets and wildcard lists. For these see Cookbook:PageListMultiTargets.

Also, `link=` will ignore the directives `(:if...:), (:include...:), (:redirect...:), (:pagelist...:)`, and page text variable directives, while searching for links in a page. That means links in included pages will not be found, and links inside non-displayed conditional markup will be found. See PageTextVariables for ways to hide a link on a page while still allowing `link=` to find it.

count=

The "`count=`" option provides the ability to

- limit the pagelist to a specific number of pages
- subsets of a list
- return items from the end of a list, subsets of a list
- display pages in reverse sequence

A simple bullet list of ten most recently modified pages	`(:pagelist trail=Site.AllRecentChanges count=10 fmt=#simple:)`
Display the first ten pages of a list	`count=10 # display the first ten pages of list`
Negative numbers specify pages to be displayed from the end of the list:	`count=-10 # display last ten pages of list`

Ranges may be specified using '..', thus:	`count=1..10       # first ten pages of list`
	`count=5..10       # 5th through 10th pages of list`

Negative numbers in ranges count from the end of the list:	`count=-10..-5 # 10th from end, 9th from end, ..., 5th from end`

Omitting the start or end of the range uses the start or end of the list:	`count=10..        # skip first ten pages`
	`count=..10        # 1st through 10th page of list`
	`count=-10..       # last ten pages of list`
	`count=..-10       # all but the last nine pages`

Ranges can be reversed, indicating that the order of pages in the output should likewise be reversed:	`count=5..10       # 5th through 10th pages of list`
	`count=10..5       # same as 5..10 but in reverse sequence`
	`count=-1..1       # all pages in reverse sequence`

"Reverse sequence" here refers to the sequence*after* any sorting has taken place. Therefore the three directives to the right are equivalent:	`(:pagelist order=-name count=10:)`
	`(:pagelist order=-name count=1..10:)`
	`(:pagelist order=name count=-1..-10:)`

wrap=

The "`wrap`" option has the values, *none* and *inline*.

With "wrap=inline" and "wrap=none", the output from pagelist (markup or HTML) is directly embedded in a page's markup without any surrounding <div> class=...</div> tags.

With "wrap=inline", any surrounding <ul> is continued. Without "wrap=inline", the HTML output starts a new <ul>. This is important if you want to get a second level <ul> produced by the page list since starting a new <ul> with "**" doesn't yield a second level <ul> but <dl><dd><ul>...

"wrap=inline" likely has other effects since it suppresses the call to $FPLTemplateMarkupFunction (being MarkupToHTML by default).

class=

By default, a pagelist has the 'fpltemplate' class. The 'bygroup', 'simple', 'group' and 'title' page list formats have specific class names fplbygroup, fplsimple etc. You can set any class using the class= parameter or by setting the $FPLFormatOpt array.

request=1

With `(:pagelist [other parameters] request=1:)` you can override most pagelist parameters, by providing request parameters in the URL. For example, `(:pagelist order=name request=1:)` will normally sort the list by name. But if the page's URL contains `?order=time`, the list will be sorted by time. If the URL contains`?order=`, the list will be unordered. Note: In the URL, encode any "#"s that are in your parameters as "%23". Since this parameter gives users who don't have edit rights the ability to run a pagelist of their choosing, consider its security implications for your website before using it.

Since version 2.2.71, it is possible to explicitely allow only certain parameters that can be overridden, or to disallow some parameters to be overridden. If you need this, instead of 1, enter the parameter names.

Allow all parameters to be overridden:
`(:pagelist request=1:)`

Allow only 'order' and 'count' parameters to be overridden:
`(:pagelist request=order,count:)`

Allow all parameters to be overridden, except 'fmt' and 'trail', note the "minus" sign before each forbidden parameter:
`(:pagelist request=-fmt,-trail:)`

req=1

The `req=1` parameter requires that search terms be posted (that is, that the user presses "search" on a search form, or follows a link with additional parameters like `[[Page?q=terms&order=-name]]`) before the pagelist is executed. Note that `(:pagelist request=1 req=1:)` works mostly like `(:searchresults:)` without the lines "*Results of search for ...*" and "*X pages found out of Y pages searched*". Both "request=1" and "req=1" are needed.

When a search is performed, either via a searchbox directive, or via the search form of the skin, if the page contains a "searchresults" directive, that page will be used to display the results of the search; if the page doesn't have a "searchresults" directive, the page Site.Search will be used to display the results.

passwd=

The "`passwd`" option returns only those pages that have some sort of password attribute on them.

if=

The "`if`" option allows a condition to be specified as part of the pagelist processing, rather than from within thepage list template. Only those pages for which the condition is true are retrieved. Anything that could go within an `(:if ...:)` can be used as a condition. For example

```
  (:pagelist if="date {(ftime %GW%V {*$Name})} {=$Name}" :)
```

returns all of the pages where the name is in the same week as that of the current page.

If any arguments within the quotes could contain a space they must be quoted:

```
  (:pagelist if="date 2009-01-01..2009-12-31 '{=$:Mydate}'" :)
```

order=

The "`order=`" option allows the pages in the list to be sorted according to different criteria. Use a minus sign to indicate a reverse sort. Multiple sorting criteria can be specified using a comma, and you can create your own custom pagelist sort order:

- `order=name` - alphabetically by name (default order)
- `order=$Name` - alphabetically by name across groups
- `order=title` - alphabetically by title rather than names
- `order=time` - most recently changed pages **last**
- `order=ctime` - time of page creation (see note)
- `order=group,title` - by multiple criteria, in this instance sort by title within groups
- `order=random` - shuffle the pages into random sequence
- `order=$:pagetextvarname` - alphabetically by page text variable value (note no braces)
- `order=$pagevarname` - alphabetically by page variable value (note no braces)

Also, the `order=` option allows custom ordering functions to be written.

- Note: trail= preserves the order of the pages as they appear on the trail (unless you've specified order= explicitly or there is a default order in the page list template). So PmWiki's alphabetical default order does not apply when trail= is specified.
- Note: ctime was added to pages only from pmwiki 2.1.beta15 onwards, pages created by earlier versions don't carry a ctime attribute and can't be sorted that way.

cache=0

Pagelist has the capability to cache lists which greatly speeds up processing (when $PageListCacheDir is set). Every once in a while this caching can result in undesired results. Specifying `cache=0` disables caching.

Specifying variables as parameters

You can also specify variable values inline with the pagelist statement, and refer to the variables in the template using the `{$$variable1}` format:

```
(:pagelist fmt=#pagelist variable1="value" variable2="value2":)
```

This assumes that a site has `$EnableRelativePageVars` enabled (default since 2.2.9).

For example, in the template:

```
>>comment<<
[[#tvars]]
(:template default count=1 ParamName=Simon:)
Hi, {$$ParamName}, how are you today?
[[#tvarsend]]
>><<
```

This gives:

`(:pagelist fmt=#tvars ParamName="Sam":)`	Hi, Sam, how are you today?
`(:pagelist fmt=#tvars ParamName="Sally":)`	Hi, Sally, how are you today?
`(:pagelist fmt=#tvars:)`	Hi, Simon, how are you today?

See also `$EnableUndefinedTemplateVars`.

Examples

Include the contents of a random page from the Banners group:

```
(:pagelist group=Banners order=random count=1 fmt=#include list=normal:)
```

Display a simple list of the last ten recently changed pages:

```
(:pagelist trail=Site.AllRecentChanges count=10 fmt=#simple:)
```

Display the "top twenty" biggest cookbook pages:

```
(:pagelist group=Cookbook order=-size count=20 :)
```

The Searchbox Directive

The `(:searchbox:)` directive generally accepts the same parameters as `(:pagelist:)` and `(:input text:)` directives:

- Pagelist parameters can be added to the input text of a searchbox (or to the markup, or both)
- Input text box parameters can be added to the searchbox markup
  - An initial search string can be specified in the searchbox markup, but it must be in the form`value='search string'`. That search string is displayed in the input text and can be modified by when the search is run.
  - An optional placeholder value can be specified in the form`placeholder="Search"`. In recent browsers, this value appears gray in the search field when it is empty. Note, this attribute is valid HTML5 but if you use it in a HTML4 skin your page will not validate.
  - The size of the text input field can be specified with the size parameter, where "size=40" would specify the current default value.
    - Tip: If more than one searchbox appears on a page, adding a blank initial value like this `value=''`, to the markup for each searchbox will prevent a search string for one box from populating all of the other boxes.
- The target page for displaying searchbox results can be set with the parameter`target=*GroupName.PageName*`. The default is the current page.
- The entire searchbox form can be overridden by defining the $SearchBoxFmt variable in one's configuration file. If $SearchBoxFmt is defined, then the parameters to `(:searchbox:)` are ignored, and the content of the $SearchBoxFmt variable are used instead.

The additional parameter `label="Label"` can be used to change the label of the associated submit button:

```
(:searchbox label="Search this wiki":)
```

The Searchresults directive

The `(:searchresults:)` directive generally accepts the same parameters as `(:pagelist:)` and `(:input text:)` directives.

Customizing "Results of search for..." and "3 pages found out of..."

To change the text surrounding the search results, customize the following and add it to *local/config.php* or `$FarmD`*/local/farmconfig.php*. Note that `'en'` should be changed to the localized language.

```
XLSDV('en', array(
        'SearchFor' => 'Results of search for <em>$Needle</em>:',
        'SearchFound' =>
                '$MatchCount pages found out of $MatchSearched pages searched.'
));
```

Alternatively, adjust the 'SearchFor' and 'SearchFound' phrases in your translation pages.

The $SearchResultsFmt variable can also be set in *local/config.php* or `$FarmD`*/local/farmconfig.php*.

```
SDV($SearchResultsFmt, "<div class='wikisearch'>\$[SearchFor]
  <div class='vspace'></div>\$MatchList
  <div class='vspace'></div>\$[SearchFound]</div>");
```

You can remove the lines above and below the generated list by adding this in config.php:

```
$SearchResultsFmt = '$MatchList';
```

See Also

- Site.PageListTemplates - default pmwiki pagelist templates
- Cookbook:PagelistTemplateSamples - contributed pagelist template samples
- PageListTemplates - how to create custom pagelist templates for the fmt= option
- PagelistVariables - *local/config.php* customizations
- Cookbook:Forms - documentation for `(:input text:)` markup, which applies to `(:searchbox:)`
- CustomPagelistSortOrder - creating custom order sort functions
- Cookbook:CustomPagelistSortOrderFunctions -
- Cookbook:PageListMultiTargets -
- Cookbook:SearchResults -
- PageDirectives#attachlist - display a list of attachments
- PmWiki.Search - Targeting and customising search results

Last modified by Petko on July 30, 2016.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PageLists

toc top

PageTextVariables

toc top

Page text variables are string variables created in the wiki text of a page, and can be automatically made available for inclusion in other pages. In the default installation, PageTextVariables can only have a name containing basic Latin/Roman (ASCII) letters, digits, dash and underscore. This is a limitation for international wikis (experimental recipe for international PTV : Cookbook:InternationalPTVs).

Table of contents

Defining Page Text Variables

There are three ways to define automated Page Text Variables (more patterns can be defined if needed) :

- use a definition list - the normal pmwiki markup for a definition list will create a page text variable

Example definition list:

`:Name: Crisses`
`"{$:Name}"`
	Name
	Crisses
	"Crisses"

This creates a new variable that can be accessed by `{$:Name}` (becomes: "Crisses") in the page.

- use a simple colon delimiter in normal text

Example colon delimited:

`Address: 1313 Mockingbird Lane`
 	
`"{$:Address}"`
	Address: 1313 Mockingbird Lane
	
	"1313 Mockingbird Lane"

This creates the `{$:Address}` variable (variable markup becomes: "1313 Mockingbird Lane") in the page.

- hidden directive form - PmWiki markup that doesn't render on the page, but defines the variable

Example directive:

`(:Country: Transylvania :)`
`"{$:Country}"`
	"Transylvania "

This creates the `{$:Country}` variable (variable markup becomes: "Transylvania ") in the page.

Usage

Usage on the same page

On the same page you can resolve page text variables through the `{$:Var}` format (shown above).

Usage in headers and footers: special references

If you want a GroupHeader, GroupFooter, SideBar, etc to call on page text variable in the main page, you need to include special reference information. To explicitly reference the page text variable from the page being displayed add an asterisk to the page text variable's markup: `{*$:Address}` on the GroupFooter or GroupHeader page.

Example

`{*$:Mountain} \\`
`{*$Namespaced}`
	
	`Access Keys`

To include a page text variable *from* a header or footer see usage from other pages below.
Special references also apply to page variables and page list templates.

Usage from other pages

If you want to pull the data from another page, use the `{Group/PageName$:Var}` format.

Example:

`Suburb: Khandallah`
`(:Lake:Taupo:)`
`:Mountain:Mt Ruapehu`
 	
`->"{PmWiki/PageTextVariables$:Suburb}"`
`->"{{$FullName}$:Lake}"`
`->"{PmWiki/PageTextVariables$:Mountain}"`
	Suburb: Khandallah
	Mountain
	Mt Ruapehu
	
	"Khandallah"
	"Taupo"
	"Mt Ruapehu"

Usage from included pages

Page text variables are never included from their source page. See Usage from other pages above to refer to a page text variable on another page.

Usage with pagelists

Page lists can also access the page text variables:

Example:

`(:pagelist group=PmWiki order=$:Summary count=6 fmt=#singleline:)`	Pm Wiki, Wiki Sandbox, Patrick Michaud, Special Characters, Wiki Administrator, Wiki Farm Terminology,

And to create pagelist formats (such as those documented at Site.Page List Templates, Page Lists, Page List Templates, Page Variables. Store custom pagelists at Site.Local Templates).

Page lists can also use page text variables to select pages :

Example:

`(:pagelist group=PmWiki $:City=Paris count=8 fmt=#singleline order=-name:)`	

lists pages having '$:City' set to 'Paris'.

Example: multiple selections with spaces

`(:pagelist group=PmWiki $:City="Addis Ababa,Paris" order=-$:Version count=8 fmt=#singleline:)`	

'quotes' must surround *all* the selections.

- When using page text variables for selection or ordering, don't put the curly braces around the variable name. The curly forms do a replacement before the pagelist command is evaluated.
- Link markup within the contents of a hidden page text variable directive (as opposed to other ways of specifying PTVs) will

not be cached as a link on the page and thus won't be seen by pagelist's link= option. If you want the link to be found by link=, you need to specify the PTV using non-directive markup, or else put the link on the page even if it's hidden within a false conditional: `(:Linkme: [[PageToLink]]:) (:if false:){$:Linkme}(:ifend:)`

The page text variable value is always the text that is written in the page. It is only evaluated when the variable is printed/output to HTML. To sort by a page text variable variable, all values in all pages are the not-yet-evaluated text strings, and the pagelist order function does what it can with them. It does not process/evaluate the text at this point.

E.g. With a page name in to format "yyyyMonth" it is simpler to use a `PageVariable` calculated in `config.php`, not a `PageTextVariable`:

```
$FmtPV['$NameToYearMonth'] = 'strftime("%Y%m", strtotime($name))';
```

Then use `(:pagelist order=$NameToYearMonth:)`

An alternative is writing in the wiki page:

```
(:MonthNum:07:)
```

as the markup expression that follows won't work:

```
(:MonthNum:{(ftime fmt=%m when="{$Namespaced}")}:)
```

Testing if set or not set

=-	PTV is set (is not empty), eg `(:pagelist $:Var=- :)`
=-?*	PTV is not set (is empty), ie not one character followed by 0 or more characters, eg. `(:pagelist $:Var=-?* :)`
=*	display *all* pages, regardless of the page text variable (slow)
=-*	display *no* pages, regardless of the page text variable (slow)

Tip : `(:if ! equal "{$:PTV}" "":)` will test if PTV is empty/unset or not.

Example: Pages without a summary

```
(:pagelist group=PmWiki $:Summary=-?*
count=6  fmt=#singleline:)
```

Patrick Michaud, Pm Wiki, Recent Changes, Special Characters, Wiki Administrator, Wiki Farm Terminology,

Use page text variable in a template

Display pages by Audience page text variable.

Example:

```
>>comment<<
[[#byaudience]]
(:if ! equal '{=$:Audience}'
'{<$:Audience}':)
-<'''{=$:Audience}''':
(:ifend:)
[[{=$Name}]]
[[#byaudienceend]]
>><<
(:pagelist group=PmWiki count=10
fmt=#byaudience order=-$:Audience:)
```

visitors (intermediate) :
WebFeeds AccessKeys
authors, admins (intermediate) :
PageLists
authors, admins (advanced) :
ConditionalMarkup
authors (intermediate) :
PageVariables TableDirectives Categories Uploads IncludeOtherPages GroupHeaders

Use page text variables in conditional markup

Page text variables will be assigned/evaluated **before** any conditional markup is evaluated. This effectively means that you cannot declare a PTV within an if...else condition; and also that a PTV will have a value even if it is set within a `(:if false:)....(:if:)` condition.

Usage - from within code (developers only)

The standard `PageVar( $pagename,$varname)` function can return page text variables, but remember to include the dollar and colon like this:

```
$var=PageVar( $pagename,'$:City')
```

It works by caching all page (text) variables it finds in a page (in `$PCache`) and returns the one requested.

Page specific variables

This page describes the "variables" that are associated with pages. Page variables have the form `{$variable}`, and can be used in page markup or in certain formatting strings in PmWiki. For example, the markup "`{$Group}`" renders in this page as "PmWiki".

Note: Do not confuse these variables (set and used only in PmWiki pages) with PHP variables. Page variables can be read in PHP with the PageVar() function.

Note that these variables do not necessarily exist in the PHP code, because they have to be determined for a specific page. (However, they are usable in FmtPageName strings.)

There is also the form `{pagename$variable}`, which returns the value of the variable for another page. For example, "`{MarkupMasterIndex$Title}`" displays as "Markup Master Index".

Special references

Special referenced variables are used to specify the context of the variable when:

- the variable is included into a destination (target) page
- the variable is used in a sidebar, header, or footer

Prefixing the variable name with an asterisk (*) means the variable's value is related to the target page or main (body) page.

- `{*$PageVariablename}` - prefixed by an asterisk (*) - value reflects target page context

Without the asterisk the variable's value is that in the page from which it originates, eg source page of include, sidebar, or header or footer.

- `{$PageVariablename}` - retains value in source page context

See also `$EnableRelativePageVars`.
Special references are also used in page text variables and page list templates.

For example you can test to see if the page is part of another page

```
(:if ! name {$FullName}:)
%comment% name of this page is not the same as the page this text was sourced from
->[[{$FullName}#anchor | more ...]]
(:ifend:)
```

or refer to the main page in a sidebar, footer, or header

`This page is [[{*$FullName}]]`	This page is PmWiki.AccessKeys

Default page variables

The page variables defined for PmWiki are:

`{$Action}` - page's url action argument, as in "pdfgroup"
`{$BaseName}` - page's "base" form (stripping any prefixes or suffixes defined via `$BaseNamePatterns`) as in "PmWiki.PageVariables"
`{$DefaultGroup}` - default group name, as in "Main"
`{$DefaultName}` - name of default page, as in "HomePage" (take note also of `$PagePathFmt` for setting a homepage for a group)
`{$Description}` - page's description from the `(:description:)` markup, as in "Documentation for "variables" that are associated with pages."
`{$FullName}` - page's full name, as in "PmWiki.PageVariables"
`{$Group}` - page's group name, as in "PmWiki"
`{$Groupspaced}` - spaced group name, as in "Pm Wiki"

`{$LastModified}` - date page was edited, as in "June 29, 2016"
`{$LastModifiedBy}` - page's last editor, as in "Petko"
`{$LastModifiedHost}` - IP of page's last editor, as in ""
`{$LastModifiedSummary}` - Summary from last edit, as in "`$WikiTitle`, `$SiteAdminGroup`"
`{$LastModifiedTime}` - time page was edited in unix-style timestamp, as in "1467201112"
    This can be used (preceded by '@') in `{(ftime)}` and other date/time markups.

`{$Name}` - page name, as in "PageVariables"
`{$Namespaced}` - spaced page name, as in "Page Variables"
`{$PageUrl}` - page's url, as in "http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PageVariables"
`{$PasswdRead}` - "read" permissions for the page e.g. ""
`{$PasswdEdit}` - "edit" permissions for the page e.g. ""
`{$PasswdAttr}` - "attr" permissions for the page e.g. ""

`{$RequestedPage}` - page requested in URL, used on Site.PageNotFound. e.g. "PmWiki/AccessKeys"
`{$SiteGroup}` - default interface group name for e.g. SideBar, forms, templates, as in "Site"
`{$SiteAdminGroup}` - default administrative group name for e.g. AuthUser, Blocklist, as in "SiteAdmin"
`{$WikiTitle}` - title of the website, as in "PmWiki"
`{$Title}` - page title (may differ from Name), as in "Page specific variables"
`{$Titlespaced}` - either the page title (if defined), or the spaced page name, as in "Page specific variables"

In addition to the above, there are some page-invariant variables available through this markup:

`{$Author}` - the name of the person currently interacting with the site, as in "Petko"
`{$AuthId}` - current authenticated id, as in ""note the lower case 'd'.

`{$Version}` - PmWiki version, as in "pmwiki-2.2.99"
`{$VersionNum}` - The internal version number, as in "2002099"
`{$ScriptUrl}` - The url to the pmwiki script, as in "http://127.0.0.1:8080/pmwiki/pmwiki.php"

Page variable security ($authpage)

The form `{`*`pagename`*`$`*`variable`*`}` or some PageLists, can display the values for other pages, regardless of the password protections.

If the other pages are protected and the visitor has no read permissions, PageVariables, unlike PageTextVariables, normally display the values. While most variables do not contain sensitive information, some of them could do: $Title, $Description and those starting with $LastModified.

Administrators and module developers can redefine the sensitive page variables to respect authentications, by using the "$authpage" variable instead of "$page" in the definition. The following snippet can be added in local/config.php -- it will rewrite the default possibly sensitive definitions to the secure ones.

```
foreach($FmtPV as $k=>$v) {
  if(preg_match('/^\\$(Title(spaced)?|LastModified(By|Host|Summary|Time)?|Description)$/', $k))
    $FmtPV[$k] = str_replace('$page', '$authpage', $v);
}
```

Custom page variables

You may add custom page variables as a local customization. In a local configuration file or a recipe script, use the variable `$FmtPV`:

```
$FmtPV['$VarName'] = "'variable definition'";
$FmtPV['$CurrentSkin'] = '$GLOBALS["Skin"]';
$FmtPV['$WikiTitle'] = '$GLOBALS["WikiTitle"]';
```

Defines new Page Variable of name $CurrentSkin, which can be used in the page with`{$CurrentSkin}` (also for Conditional markup). It's necessary to use the single quotes nested inside double-quotes as shown above (preferred) or a double-quoted string nested inside single-quotes like *`'"this"'`*.

Please note that the values of the elements of `$FmtPV` are `eval()`ed so always sanitize any user input. The following is very insecure:

```
$FmtPV['$Var'] = $_REQUEST['Var']; # critically insecure, allows PHP code injection
$FmtPV['$Var'] = "'". addslashes($_REQUEST['Var']).'"'; # critically insecure, allows PHP code injection
```

See the recipe Cookbook:HttpVariables for a better way to use these variables.

See also

- Cookbook:More custom page variables
- PmWiki.Variables — about variables internal to PmWiki.
- PmWiki.MarkupMasterIndex — complete list of PmWiki markups.
- PageTextVariables — page variables automatically made available through natural page markup or explicit page markup within the wiki text of the page.
- PmWiki.Markup Expressions — markup expressions can manipulate page variables

Is there a variable like $LastModified, but which shows me the creation time?

No, but you can create one in config.php. For instance:

```
# add page variable {$PageCreationDate} in format yyyy-mm-dd
$FmtPV['$PageCreationDate'] = 'strftime("%Y-%m-%d", $page["ctime"])';
```

If you like the same format that you define in config.php with `$TimeFmt` use
```
$FmtPV['$Created'] = "strftime(\$GLOBALS['TimeFmt'], \$page['ctime'])";
```

How can I test if a variable is set and/or not empty?

Use `(:if ! equal "{$Variable}" "":) $Variable is not empty. (:ifend:)`. Note that undefined/inexistent variables appear as empty ones.

Categories: PmWiki Developer

Last modified by Petko on June 29, 2016.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PageVariables

toc top

PagelistVariables

toc top

`$EnablePageListProtect`
When set to 1 (which is the default), causes `(:pagelist:)` and `(:searchresults:)` to exclude listing any pages for which the browser does not currently have read authorization. Setting this to zero means that read-protected pages can appear in a listing, but can also speed up searches considerably (because page permissions do not need to be checked).

`$SearchPatterns`
An array of page name patterns to be required or excluded from search and pagelist results. In order to be included in a search listing or page listing, a page's name must not match any pattern that is delimited by exclamation points (!) and must match all other patterns. See Cookbook:SearchPatterns.
```
# Limit all searches to Main group
$SearchPatterns['default'][] = '/^Main\\./';
# Exclude the Main group from search results
$SearchPatterns['default'][] = '!^Main\\.!';
# Exclude RecentChanges pages from search results
$SearchPatterns['default'][] = '!\\.(All)?RecentChanges$!';
# Prevent a page from listing itself in (:pagelist:) or (:searchresults:)
$SearchPatterns['default'][] = FmtPageName('!^$FullName$!', $pagename);
```

`$SearchBoxOpt`
For example `$SearchBoxOpt ['target'] = '$DefaultGroup.Search';`

`$SearchBoxInputType`
The HTML "type" attribute for the search field. Default is "text" which is valid HTML4. If your skin uses HTML5, you can change this to "search":
```
$SearchBoxInputType = "search";
```

`$EnablePageIndex`
When set to 1, causes PmWiki to maintain a "link and word index" in `$PageIndexFile` which significantly speeds up categories, backlinks, and searches.

`$PageIndexFile`
The location of the "page index" file for `(:pagelist:)`, defaults to `$WorkDir/.pageindex`.

`$PageListCacheDir`
The name of a writable directory where PmWiki can cache results of `(:pagelist:)` directives to speed up subsequent displays of the same list. Default is empty, which disables the pagelist cache.
```
# Enable pagelist caching in work.d/
$PageListCacheDir = 'work.d/';
```

`$PageSearchForm`
The page to be used to format search results for `?action=search` (unless the current page has a "searchresults" directive in it). This variable can be an array, in which case the first page found from the array is used.
```
# Simple use of page search form in the default group
$PageSearchForm = '$DefaultGroup.Search';
# Use Search page in current group if it exists, otherwise use Site.Search
$PageSearchForm = array('$Group.Search', '[=$[$SiteGroup/Search]=]');
```

`$FPLTemplatePageFmt`
The pages to be searched for a pagelist template specified by a `fmt=#xyz` parameter. Defaults to searching the current page, Site.LocalTemplates and Site.PageListTemplates.

```
# PMWiki default setup
global $FPLTemplatePageFmt;
$FPLTemplatePageFmt = array(
    '{$FullName}',
    '{$SiteGroup}.LocalTemplates',
    '{$SiteGroup}.PageListTemplates');
```

It can be customized to look in other pages.

```
    # Search a Group.Templates page as well as the Site templates
    global $FPLTemplatePageFmt;
    $FPLTemplatePageFmt = array(
        '{$Group}.Templates',
        '{$SiteGroup}.LocalTemplates',
        '{$SiteGroup}.PageListTemplates');
```

Or declare defaults for the template array:

```
    # Search a Group.Templates page as well as the Site templates
    global $FPLTemplatePageFmt;
    SDV($FPLTemplatePageFmt, array('{$FullName}',
        '{$Group}.Templates',
        '{$SiteGroup}.LocalTemplates', '{$SiteGroup}.PageListTemplates')
       );
```

`$EnableUndefinedTemplateVars`

This variable controls how undefined {$$Variable} is processed in includes and PageList templates. If set to 0 (default), undefined {$$Variable}s are removed from the included section or template. If set to 1, undefined {$$Variable}s are displayed as is, with {$$...}. *Note that PmWiki versions 2.2.13 and earlier* ***kept*** *unset include/template variables.*

`$EnableUndefinedTemplateVars = 0;` # Delete unset raw template variables
`$EnableUndefinedTemplateVars = 1;` # Keep and print unset raw template variables

Passwords

PmWiki has built-in support for password-protecting various areas of the wiki site. Authors generally want to be able to apply passwords to individual pages or to wiki groups. Wiki Administrators can apply passwords to individual pages, to wiki groups, or to the entire site. Setting an edit password on a page or group (or the entire site) is one of the most common ways to stop spam. As with any access control system, the password protection mechanisms described here are only a small part of overall system and wiki security.

As an author editing pages...

An author will generally set 3 types of passwords:

1. to control who can see a page or group, use `read` passwords
2. to control who can edit a page or group, use `edit` passwords
3. to control who can alter the passwords used to protect a page or group, use `attr` passwords

If required most page actions can be password protected.

Protect an individual page

To set a password on an individual wiki page, add the page action

```
?action=attr
```

to the page's URL (address) to access its attributes. Using the form on the attributes page, you can set or clear the `read`, `edit`, or `attr` passwords on the page. In the form you enter the passwords as cleartext; PmWiki encrypts them for you automatically when it stores them.

Additional options:

- Leaving a field blank will leave the attribute unchanged.
- To remove a password from a page (*reverting back* to the group's or site's default), enter
  ```
  clear
  ```
- To indicate that the page can be edited *even if a group or site password is set*, enter
  ```
  @nopass
  ```
- To lock a page for everybody but the admin, enter
  ```
  @lock
  ```
- To assign the site's site-wide passwords to the `read`, `edit`, or `attr` password for the page, enter
  ```
  @_site_edit, @_site_read or @_site_upload
  ```

Protect a wiki group of pages

To set a password on a wiki group is slightly more difficult -- you just set the passwords on a special page in each group called GroupAttributes

First, you can get to the attributes page for GroupAttributes by entering a URL (address) like

```
http://example.com/pmwiki/pmwiki.php?n=GroupName.GroupAttributes?action=attr
```
Replace example.com with your domain name, and GroupName with the name of the group

Then, using the form on the attributes page, you can set or clear the `read`, `edit`, or `attr` passwords for the entire group. In the form you enter the passwords as cleartext; PmWiki encrypts them for you automatically.

Additional options:

- To remove a password from a group (*reverting back* to the site's default), enter
  ```
  clear
  ```
- To indicate that the group can be edited *even if a site password is set*, enter
  ```
  @nopass
  ```
- To lock a group for everybody but the admin, enter
  ```
  @lock
  ```
- (Beginning with Ver 2.2.3) To assign the site's site-wide passwords to the `read`, `edit`, or `attr` password for the group, enter
  ```
  @_site_edit, @_site_read or @_site_upload
  ```

Passwords

Passwords may consist of any combination of characters, except double "quotes" or 'apostrophes'. Passwords with spaces or colons must be entered using quotes, eg "foo bar" or "foo:bar". Obviously longer is better, and on some systems passwords need to have 4 or more characters.

Multiple passwords

Multiple passwords for a page, group or site are allowed. Simply enter multiple passwords separated by a space. This allows you to have a read password, a write password, and have the write password allow read/write access. In other words, if the read password is

alpha

and the edit password is

beta

then enter
```
Set new read password: alpha beta
Set new edit password: beta
```

This says that either

alpha

or

beta

can be used to read pages, but only

beta

may edit. Since PmWiki checks the passwords you've entered since the browser has been opened, entering a read password that is also a write password allows both reading and writing.

Protect the site

Passwords can be applied to the entire wiki website in *config.php*. See passwords administration for details.

administrator

As an administrator ...

You can set passwords on pages and groups exactly as described above for authors. You can also:

1. set site-wide passwords for pages and groups that do not have passwords
2. use `attr` passwords to control who is able to set passwords on pages
3. use `upload` passwords to control access to the file upload capabilities (if uploads are enabled)
4. use an `admin` password to override the passwords set for any individual page or group
5. use SiteAdmin.AuthList to view the permissions settings for pages that have permissions set.

For more information on password options available to administrators, see PasswordsAdmin.

Which password wins?

In PmWiki, page passwords override group passwords, group passwords override the *default* passwords, and the `admin` password overrides all passwords. This gives a great deal of flexibility in controlling access to wiki pages in PmWiki.

The special page SiteAdmin.AuthList is a page list of all pages with access permissions set.

Opening access to pages in protected groups/sites

Sometimes we want to "unprotect" pages in a group or site that is otherwise protected. In these cases, the special password

```
@nopass
```
is used to indicate that access should be allowed to a page without requiring a password.

For example, suppose Main.GroupAttributes has an edit password set, thus restricting the editing of all pages in Main. Now we want Main.WikiSandbox to be editable without a password. Using
```
clear
```
for the edit password for Main.WikiSandbox *doesn't unprotect the page*, because the password is being set by the group. Instead, we set the edit password for Main.WikiSandbox to the special value
```
@nopass
```
which tells PmWiki to ignore any site-wide or group-level passwords for that page.

How can I password protect all the pages and groups on my site? Do I really have to set passwords page by page, or group by group?

Administrators can set passwords for the entire site by editing the config.php file; they don't have to set passwords for each page or group. For example, to set the entire site to be editable only by those who know an "edit" password, an administrator can add a line like the following to local/config.php:

```
$DefaultPasswords['edit'] = pmcrypt('edit_password');
```

For more information about the password options that are available only to administrators, see PasswordsAdmin.

I get http error 500 "Internal Server Error" when I try to log in. What's wrong?

This can happen if the encrypted passwords are not created on the web server that hosts the PmWiki.
The PHP crypt() function changed during the PHP development, e.g. a password encrypted with PHP 5.2 can not be decrypted in PHP 5.1, but PHP 5.2 can decrypt passwords created by PHP 5.1.
This situation normally happens if you prepare everything on your local machine with the latest PHP version and you upload the passwords to a webserver which is running an older version.
The same error occurs when you add encrypted passwords to local/config.php.

Solution: Create the passwords on the system with the oldest PHP version and use them on all other systems.

How can I create private groups for users, so that each user can edit pages in their group, but no one else (other than the admin) can?

Modify the edit attribute for each group to id:username, e.g. set the edit attribute in JaneDoe.GroupAttributes to id:JaneDoe.

There is a more automatic solution, but it's probably not a good idea for most wikis. Administrators can use the AuthUser recipe and add the following few lines to their local/config.php file to set this up:
```
$group = FmtPageName('$Group', $pagename);
$DefaultPasswords['edit'] = 'id:'.$group;
include_once("$FarmD/scripts/authuser.php");
```
This automatically gives edit rights to a group to every user who has the same user name as the group name. Unfortunately it also gives edit rights to such a user who is visiting a same-named group not just for pages in that group, but for any page on the wiki that relies on the site's default edit password. This can create security holes.

How come when I switch to another wiki within a farm, I keep my same authorization?

PmWiki uses PHP sessions to keep track of authentication/authorization information, and by default PHP sets things up such that all interactions with the same server are considered part of the same session.

An easy way to fix this is to make sure each wiki is using a different cookie name for its session identifier. Near the top of one of the wiki's local/config.php files, before calling authuser or any other recipes, add a line like:
session_name('XYZSESSID');
You can pick any alphanumeric name for XYZSESSID; for example, for the cs559-1 wiki you might choose session_name('CS559SESSID');
This will keep the two wikis' sessions independent of each other.

Is it possible to test the password level for display and/or if condition? Example: * (:if WriterPassword:) (display Edit link) (:ifend:)

You can use `(:if auth edit:)`. See ConditionalMarkup.

PasswordsAdmin

PmWiki has built-in support for password-protecting various areas of the wiki site. Passwords can be applied to individual pages, to Wiki Groups, or to the entire wiki site. Note that the password protection mechanisms described here are only a small part of overall system (and wiki) security, see PmWiki.Security for more discussion of this.

Authors can use PmWiki to add passwords to individual pages and WikiGroups as described in Passwords. However, WikiAdministrators can also set passwords in *local/config.php* as described below. (Please note that one cannot set passwords reliably in per group or per page customization files. See the FAQ section for details.)

Password basics

PmWiki supports several levels of access to wiki pages, known as authorisation level:

- `read` passwords allow viewing the contents of wiki pages
- `edit` passwords control editing and modification of wiki pages (effective against spam)
- `attr` passwords control who is able to set passwords on pages (and potentially other future attributes)
- `upload` password, if uploads are enabled, controls uploading of files and attachments
- in addition all available actions can be password authorised
- `admin` password allows an administrator to override the passwords set for any individual page or group.

By default, PmWiki has the following password settings:

- The `admin` and `upload` passwords are locked by default.
- The Main and PmWiki groups have a locked `attr` password (in their respective GroupAttributes pages).
- The pages in the Site group except Site.SideBar are locked against editing; by default the Site.SideBar page requires the admin or the site-wide edit password.

An `admin` password can be used to overcome "locked" passwords, other than that, no password will allow access.

See Passwords for information about setting per-page and per-group passwords. The remainder of this page describes setting site-wide passwords from the *local/config.php* file.

Setting site-wide passwords

One of the first things an admin should do is set an `admin` password for the site. This is done via a line like the following in the *local/config.php* file:

```
$DefaultPasswords['admin'] = pmcrypt('secret_password');
```

Note that the pmcrypt() call is required for this -- PmWiki stores and processes all passwords internally as encrypted strings. See the crypt section below for details about eliminating the cleartext password from the configuration file.

To set the entire site to be editable only by those who know an "edit" password, add a line like the following to *local/config.php*:

```
$DefaultPasswords['edit'] = pmcrypt('edit_password');
```

Similarly, you can set a password for any available action, via `$DefaultPasswords['read']`, `$DefaultPasswords['edit']`, and `$DefaultPasswords['upload']` to control default `read`, `edit`, and `upload` passwords for the entire site. The default passwords are used for pages and groups which do not have passwords set, and as additional passwords for pages and groups which do have passwords set. Also, each of the `$DefaultPasswords` values may be arrays of encrypted passwords:

```
$DefaultPasswords['read'] = array(pmcrypt('alpha'), pmcrypt('beta'));
$DefaultPasswords['edit'] = pmcrypt('beta');
```

This says that either "alpha" or "beta" can be used to read pages, but only the "beta" password will allow someone to edit a page. Since PmWiki remembers any passwords entered during the current session, the "beta" password will allow both reading and writing of pages, while the "alpha" password allows reading only. A person without either password would be unable to view pages at all.

Setting passwords by reference

This is an unintended feature.

Setting passwords by reference allows you to change the password for a whole set of pages as easily as you can change site-wide passwords. (Otherwise you would have to update each page's attributes individually.) Enter in the Page Attributes or Group Attributes:

@_site_MyLevel2

And in the local configuration file set the actual password with lines like this:

```
$DefaultPasswords['MyLevel2'] = array(pmcrypt('secret'), '@admins');
$DefaultPasswords['MyLevel9'] = array('$1$NuBV/Mcc$GG3J60h.TLczUTRKhoVPM.');
```

Note that passwords set by reference in a configuration file currently can not be used as a site-wide default. However, you could explicitly specify your @_site_level at the group level for every group to achieve the same effect. Once specified as a group attribute, the password applies to all pages in the group unless overridden, just like any other password.

Identity-based authorization (username/password logins, AuthUser)

Unlike many systems which have **identity-based** systems for controlling access to pages (e.g., using a separate *username* and *password* for each person), PmWiki defaults to a *password-based* system as described above. In general password-based systems are often easier to maintain because they avoid the administrative overheads of creating user accounts, recovering lost passwords, and mapping usernames to permitted actions.

However, PmWiki's *authuser.php* script augments the password-based system to allow access to pages based on a username and password combination. See AuthUser for more details on controlling access to pages based on user identity.

Security holes ...

Administrators need to carefully plan where passwords are applied to avoid opening inadvertent security holes. If your wiki is open (anyone can read and edit), this would not seem to be a concern, **except**, a malicious or confused user could apply a read password to a group and make the group completely unavailable to all other users. At the very least, even an open wiki should have a site-wide "admin" password and a site-wide "attr" password set in config.php. The *sample-config.php* file distributed with PmWiki indicates that the PmWiki and Main groups have "attr" locked by default, but if anyone creates a new group, "attr" is unlocked. Administrators must remember to set "attr" passwords for each new group (if desired) in this case. An easier solution is to include these lines in *config.php* :

```
$DefaultPasswords['admin'] = pmcrypt('youradminpassword');
$DefaultPasswords['attr'] = pmcrypt('yourattrpassword');
```

Encrypting passwords in *config.php*

One drawback to using the pmcrypt() function directly to set passwords in *config.php* is that anyone able to view the file will see the unencrypted password. For example, if *config.php* contains

```
$DefaultPasswords['admin'] = pmcrypt('mysecret');
```

then the "mysecret" password is in plain text for others to see. However, a wiki administrator can obtain and use an encrypted form of the password directly by using `?action=crypt` on any PmWiki url on the target wiki (or just jump to PasswordsAdmin?action=crypt on your own wiki). This action presents a form that generates encrypted versions of passwords for use in the *config.php* file. For example, when `?action=crypt` is given the password "`mysecret`", PmWiki will return a string like

```
$1$hMMhCdfT$mZSCh.BJOidMRn4SOUUSi1
```

The string returned from `?action=crypt` can then be placed directly into config.php, as in:

```
$DefaultPasswords['admin'] = '$1$hMMhCdfT$mZSCh.BJOidMRn4SOUUSi1';
```

Note that in the encrypted form the *pmcrypt* function and parentheses are removed, since the password is already encrypted. Also, the encrypted password must be in single quotes. In this example the password is still "`mysecret`", but somebody looking at *config.php* won't be able to see that just from looking at the encrypted form. *?action=crypt* may give you different encryptions for the same password--this is normal (and makes it harder for someone else to determine the original password).

Please note that the encrypted password should be created with ?action=crypt on the wiki that will use it. A password encrypted on one system may or may not be usable on another.

Removing passwords

To remove a site password entirely, such as the default locked password for uploads, just set it to empty:

```
$DefaultPasswords['upload'] = '';
```

You can also use the special password "@nopass" via `?action=attr` to have a non-password protected page within a password-protected group, or a non-password protected group with a site-wide default password set.

Revoking or invalidating passwords

If a password is compromised and the wiki administrator wants to quickly invalidate all uses of that password on a site, a quick solution is the following in *local/config.php*:

```
$ForbiddenPasswords = array('secret', 'tanstaafl');
if (in_array(@$_POST['authpw'], $ForbiddenPasswords))
  unset($_POST['authpw']);
```

This prevents "secret" and "tanstaafl" from ever being accepted as a valid authorization password, regardless of what pages may be using it.

See Also

- The `$HandleAuth` array, which sets the required authentication level that is necessary to perform an action.
- Cookbook:RequireAuthor

Protecting actions (example)

Each action can be password protected. Cookbook authors providing scripts with own actions can use this also, but I'll limit the example to a (by default) not protected `?action=source`. This action shows the wikisource of the actual page. Sometimes you don't want that especially to Cookbook:protect email or when using some conditional markup which should not be discovered easily or only by persons that are allowed to edit the page.

There are several solutions for that:

1. Limit "source" only to editors add the following to your*local/config.php*:

```
$HandleAuth['source'] ='edit';
```

2. For using "source" with an own password, then add:

```
$HandleAuth['source'] ='source';
$DefaultPasswords['source'] = pmcrypt('secret'); # see above
```

If you additionally want to set the password in the attributes page add:

```
$PageAttributes['passwdsource'] = "$['Set new source password']";
```

In general, adding the prefix 'passwd' to an action name in the`$PageAttributes` array indicates that you wish for the given field to be encrypted when saved to disk.

The full set of steps to add new password handling for an action such as "diff" would be:

```
# add a new (encrypted) field to the attr page
$PageAttributes['passwddiff'] = '$[Set new history password:]';

# clear the default password for 'diff'
$DefaultPasswords['diff'] = '';

# Tell PmWiki that the 'diff' password allows action 'diff'.
$HandleAuth['diff'] = 'diff';

# Tell PmWiki that a 'read' password
# (or optionally the 'edit') password
# is also sufficient to enable 'diff'.
# Of course, the 'admin' password will work too.
$AuthCascade['diff'] = 'read';    ## or 'edit'
```

There seems to be a default password. What is it?

> There isn't any valid password until you set one. Passwords admin describes how to set one.
>
> PmWiki comes "out of the box" with `$DefaultPasswords`['admin'] set to '*'. This doesn't mean the password is an asterisk, it means that default admin password has to be something that encrypts to an asterisk. Since it's impossible for the pmcrypt() function to ever return a 1-character encrypted value, the admin password is effectively locked until the admin sets one in config.php.

How do I use passwd-formatted files (like .htpasswd) for authentication?

> See AuthUser, Cookbook:HtpasswdForm or Cookbook:UserAuth2.

Is there anything I can enter in a GroupAttributes field to say 'same as the admin password'? If not, is there anything I can put into the config.php file to have the same effect?

> Enter '@lock' in GroupAttributes?action=attr to require an admin password for that group.

How do I edit protect, say, all RecentChanges pages?

> see Security#wikivandalism.

How can I read password protect all pages in a group except the HomePage using configuration files?

> As described in PmWiki.GroupCustomizations per-group or per-page configuration files should not be used for defining

passwords. The reason is that per-group (or per-page) customization files are only loaded for the current page. So, if `$DefaultPasswords['read']` is set in *local/GroupA.php*, then someone could use a page in another group to view the contents of pages in GroupA. For example, Main.WikiSandbox could contain:

(:include GroupA.SomePage:)

and because the *GroupA.php* file wasn't loaded (we're looking at Main.WikiSandbox -->*local/Main.php*), there's no read password set.

How can I password protect the creation of new pages?

See Cookbook:LimitWikiGroups, Cookbook:NewGroupWarning, Cookbook:LimitNewPagesInWikiGroups.

How do I change the password prompt screen?

If your question is about how to make changes to that page... edit Site.AuthForm. If your question is about how to change which page you are sent to when prompted for a password, you might check out the Cookbook:CustomAuthForm for help.

How do I change the prompt on the attributes (`?action=attr`) screen?

Simply create a new page at Site.AttrForm, and add the following line of code to`config.php`:

```
$PageAttrFmt = 'page:Site.AttrForm';
```

Note that this only changes the text above the password inputs on the attributes page, but doesn't change the inputs themselves - the inputs have to be dealt with separately. See Cookbook:CustomAttrForm for more info.

I get http error 500 "Internal Server Error" when I try to log in. What's wrong?

This can happen if the encrypted passwords are not created on the web server that hosts the PmWiki.
The crypt function changed during the PHP development, e.g. a password encrypted with PHP 5.2 can not be decrypted in PHP 5.1, but PHP 5.2 can decrypt passwords created by PHP 5.1.
This situation normally happens if you prepare everything on your local machine with the latest PHP version and you upload the passwords to a webserver which is running an older version.
The same error occurs when you add encrypted passwords to local/config.php.

Solution: Create the passwords on the system with the oldest PHP version and use them on all other systems.

I only want users to have to create an 'edit' password, which is automatically used for their 'upload' & 'attr' passwords (without them having to set those independently). How do I do this?

By setting `$HandleAuth` like so:

```
   $HandleAuth['upload'] = 'edit';
   // And to prevent a WikiSandbox from having it's 'attr' permissions changed
   // except by the admin (but allowing editors to change it on their own pages/group)
   if(($group=="Site") || ($group=="Main") || ($group=="Category") ||
           ($group=="SiteAdmin") || ($group=="PmWiki") ) {
$HandleAuth['attr'] = 'admin';  // for all main admin pages, set 'attr' to 'admin' password
   } else {
$HandleAuth['attr'] = 'edit';  // if you can edit, then you can set attr
   }
```

PathVariables

When dealing with file or path variables, one has to recognize the difference between working with URLs and files on disk. For example:

- The include() statements are used to include other files (on disk) into the currently running PmWiki script. Thus they require paths on the server's filesystem.
- The `$ScriptUrl` and `$PubDirUrl` variables are used to tell a*browser*, connecting via the webserver, how to execute the pmwiki script (`$ScriptUrl`) and the base url for getting files from PmWiki's pub/ directory (`$PubDirUrl`).

Note that a browser needs a URL (http://example.com/pmwiki/pub) while an include statement requires a server file path (`$FarmD`/scripts/something.php).

`$FarmD`
: The directory on the server where the farm is located (i.e., the directory containing the farm's copy of *pmwiki.php* and the *scripts/* directory). This directory is automatically determined by pmwiki.php when it runs, and can be used to distinguish the farm's *cookbook/* and *pub/* subdirectories from a field's subdirectories.

`$FarmPubDirUrl`
: is the url that refers to the `pub` directory for an entire farm. It defaults to the same value as `$PubDirUrl`.

`$PageCSSListFmt`

is an associative array which PmWiki uses to find any local css configuration files. It consists of a set of (*key*,*value*) pairs that point to the same file. The *key* is a possible path to a file on disk holding the css data, while the *value* is the coresponding URL for that same file. They keys are tested in turn, and for each named file that exists, the browser is instructed to load the corresponding URL. This allows for PMWiki to only load the css file if it exists. (Why see if a CSS exists?) The default value for this variable is:

```
$PageCSSListFmt = array(
  'pub/css/local.css' => '$PubDirUrl/css/local.css',
  'pub/css/{$Group}.css' => '$PubDirUrl/css/{$Group}.css',
  'pub/css/{$FullName}.css' => '$PubDirUrl/css/{$FullName}.css');
```

Note that the default (as of version pmwiki-2.1.beta26) makes no reference to `$FarmPubDirUrl` for css configuration files. If you wish to be able to place css configuration files in both the field's `pub` directory, and the farm's `pub` directory, you may want to add these lines to your `local/config.php` file (as described in Cookbook:SharedPages):

```
# this adds farm.css to all wikis
$PageCSSListFmt = array(
  '$FarmD/pub/css/farm.css' => '$FarmPubDirUrl/css/farm.css',
  'pub/css/local.css' => '$PubDirUrl/css/local.css',
  'pub/css/$Group.css' => '$PubDirUrl/css/$Group.css',
  'pub/css/$FullName.css' => '$PubDirUrl/css/$FullName.css');

# this enables farm css files in a similar manner to a local wiki
$PageCSSListFmt = array(
  '$FarmD/pub/css/local.css' => '$FarmPubDirUrl/css/local.css',
  '$FarmD/pub/css/$Group.css' => '$FarmPubDirUrl/css/$Group.css',
  '$FarmD/pub/css/$FullName.css' => '$FarmPubDirUrl/css/$FullName.css',
  'pub/css/local.css' => '$PubDirUrl/css/local.css',
  'pub/css/$Group.css' => '$PubDirUrl/css/$Group.css',
  'pub/css/$FullName.css' => '$PubDirUrl/css/$FullName.css');
```

Note the difference between CSS configuration files and CSS files associated with a skin. Skin files, including associated CSS, can be put in either the farm or the field `pub/skins` directory, and the program will find them.

`$PubDirUrl`

is the URL that refers to the `pub` directory. That directory contains all the files and subdirectories that must be directly accessible from a browser (e.g. CSS and HTML files). Most prominent here is the `skins` subdirectory.

The following may work for you [1]

```
  $ScriptUrl = 'http://'.$_SERVER['HTTP_HOST'].'/pmwiki/pmwiki.php';
  $PubDirUrl = 'http://'.$_SERVER['HTTP_HOST'].'/pmwiki/pub';
```

`$ScriptUrl` http://127.0.0.1:8080/pmwiki/pmwiki.php

is the URL that you want people's browsers to use when accessing PmWiki, either as a field or farm. It's used whenever PmWiki needs to generate a link to another PmWiki page or action. PmWiki is usually fairly good about "guessing" the correct value for `$ScriptUrl` on its own, but sometimes an admin needs to set it explicitly because of URL manipulations by the webserver (such as Cookbook:CleanUrls, mod_rewrite, bizarre PHP configurations, and so on).

`$SkinDir`

Set by *scripts/skins.php* to be the base url of the current skin's directory (i.e., within a 'pub/skins/' directory). This variable is typically used inside of a skin .tmpl file to provide access to .css files and graphic images associated with the skin. See security note regarding use.

`$SkinDirUrl`

Set by *scripts/skins.php* to be the base path of the current skin's directory (i.e., within a 'pub/skins/' directory). This variable is typically used inside of a skin .tmpl file to provide access to secondary files. See security note regarding use.

`$WorkDir`

This variable is a string that gives a local path to a directory where the pmwiki engine can create temporary files etc. PmWiki needs this for a variety of things, such as building merged edits, caching mailposts entries, keeping track of the last modification time of the site, other types of cache, etc. Do not confuse this variable with `$WikiDir`; the reason that both `$WorkDir` and `$WikiDir` refer by default to the directory `wiki.d/` is merely to simplify things for the administrator.

`$WikiDir`

A PageStore-object that refers to how wiki pages are stored.
This can be a simple reference to a directory (typically *wiki.d/*), or something more advanced such as a MySQL backend or a .dbm-file. Do not confuse this variable with `$WorkDir`; the reason that both `$WorkDir` and `$WikiDir` refer by default to the directory `wiki.d/` is merely to simplify things for the administrator.
To store groups of pages in subdirectories add `$WikiDir = new PageStore('wiki.d/$Group/$FullName');` to the start of your config file. [2]

`$WikiLibDirs`

An array of PageStore objects that specify where to look for pages.
By default it is set up to look in*wiki.d/* and *wikilib.d/*, but can be changed to look other places.
For example, to exclude the pages that are bundled in the PmWiki distribution, use the line below. (Note that some features such as editing and search rely on having certain pages available, so you may need to copy them to the `$WikiDir`.)
```
$WikiLibDirs = array(&$WikiDir);
```
Another example
```
  ## for any page name, use the version located in wiki.d if it exists,
  ## use the version located in wikilib2.d, if a wiki.d version does not, and
  ## the version located in wikilib.d, if neither of the above exists
 $WikiLibDirs = array(&$WikiDir,
     new PageStore('wikilib2.d/{$FullName}'),
     new PageStore('$FarmD/wikilib.d/{$FullName}'));
```
See also CustomPageStore.

`$LocalDir`
: The filesystem location of the *local/* directory, holding local customization and per group customizations files. Typically set in a WikiFarm's *farmconfig.php*. (Note that farm configuration files always occur in *`$FarmD`/local/farmconfig.php*, regardless of any setting for `$LocalDir`.)

See also

- Layout Variables for URL layout options
- Link Variables - variables that control the display of links in pages
- Edit Variables - variables used when editing pages
- Upload Variables - variables used for uploads/attachments

Last modified by ff on January 08, 2015. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PathVariables

PatrickMichaud

toc top

Patrick Michaud (Pm) is the author of PmWiki. More information about him can be found at http://www.pmichaud.com.

Last modified by on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PatrickMichaud

PerGroupCustomizations

toc top

Page redirects to GroupCustomizations

Last modified by Petko on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PerGroupCustomizations

PmWikiPhilosophy

toc top

This page describes some of the ideas that guide the design and implementation of PmWiki. Patrick Michaud doesn't claim that anything listed below is an original idea; these are just what drive the development of PmWiki. You're welcome to express your disagreement with anything listed below. PmWiki.Audiences also describes much of the reasoning behind the ideas given below.

1. Favor writers over readers
: At its heart, PmWiki is a collaborative authoring system for hyperlinked documents. It's hard enough to get people (including Pm) to contribute written material; making authors deal with HTML markup and linking issues places more obstacles to active contribution. So, PmWiki aims to make it easier to author documents, even if doing so limits the types of documents being authored.

2. Don't try to replace HTML
: PmWiki doesn't make any attempt to do everything that can be done in HTML. There are good reasons that people don't use web browsers to edit HTML--it's just not very effective. If you need to be writing lots of funky HTML in a web page, then PmWiki is not what you should be using to create it. What PmWiki does try to do is make it easy to link PmWiki to other "non-wiki" web documents, to embed PmWiki pages inside of complex web pages, and to allow other web documents to easily link to PmWiki.

This principle also follows from the "favor writers over readers" principle above--every new feature added to PmWiki requires some sort of additional markup to support it. Pretty soon the source document looks pretty ugly and we'd all be better off just writing HTML.

Another reason for avoiding arbitrary HTML is that ill-formed HTML can cause pages to stop displaying completely, and arbitrary HTML can be a security risk--more so when pages can be created anonymously. See http://www.cert.org/advisories/CA-2000-02.html for more information.

3. Avoid gratuitous features (or "creeping featurism")
: In general PmWiki features are implemented in response to specific needs, rather than because someone identifies

something that "might be useful". In any sort of useful system, it's hard to change a poorly designed feature once people have built a lot of structure based on it. (Need an example? Look at MS-DOS or Windows.) One way to avoid poor design is to resist the temptation to implement something until you have a clearer idea of how it will be used.

4. Support collaborative maintenance of public web pages

Although this wasn't at all the original intent of PmWiki, it became quickly obvious that WikiWikiWeb principles could be used to make it easier for groups to collaboratively design and maintain a public web site presence. PmWiki allows individual pages to be password protected, and a couple of local customizations makes it easy to protect large sections of PmWiki pages. Furthermore, in many ways PmWiki provides "style sheets on steroids": you can quickly change the headers, footers, and other elements on a large group of pages without ever having to touch the individual page contents. Finally, it's relatively easy to add custom markup for specialized applications.

5. Be easy to install, configure, and maintain

With a gzip-compressed file size of just around 400K, uploading PmWiki to your server is a speedy operation. Do a chmod or two, update a few settings in config.php and you should be up and running. PmWiki stores all data in flat files, so there is no need for MySQL or other utilities. Upgrading is usually a simple matter of copying the latest version's files over the files of your existing PmWiki installation. (One of the biggest reasons for the creation of PmWiki was that other wiki engines at the time required modifications to the distribution files, so admins ended up losing their customizations on every upgrade.)

Last modified by Anomen on June 28, 2012.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/PmWikiPhilosophy

toc top

RefCount

toc top

RefCount performs link reference counts on pages in the PmWiki database (i.e., counts of links between pages). Before using RefCount, it must be enabled by the wiki administrator by placing the following line in a local customization file:

```
include_once("$FarmD/scripts/refcount.php");
```

To use refcount add `?action=refcount` to the URL of any wiki page to bring up the reference count form. For example:

PmWiki.RefCount?action=refcount

The refcount form contains the following controls:

- **Show** ~ This selects which pages will appear in the output
  - all ~ Shows all references
  - missing ~ Shows only references to pages that don't exist
  - existing ~ Shows only references to pages that do exist
  - orphaned ~ Shows pages that exist but don't have any references to them. There is no way to browse to an orphaned page.
- **page names in group** ~ Selects which group(s) to the referenced pages can be in
- **referenced from pages in** ~ Selects which group(s) the referencing pages can be in
- **Display referencing pages** ~ Includes a link to the referencing page -- this can make for a very long output unless you limit the groups searched

The output is a table where each row of the table contains a page name or link reference, the number of (non-RecentChanges) pages that contain links to the page and the number of Recent Changes pages with links to the page.

Last modified by Petko on September 10, 2011.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/RefCount

toc top

Release Notes

toc top

See also: Upgrades, Change log and Road map.

Version 2.2.99 (2017-06-26)

This version fixes a bug where an incomplete text variable without a closing parenthesis like "`(:Var:Value`" could hide the remaining of the page.

A bug was fixed where previewing a page didn't show changes to be done by replace-on-save patterns (the function ReplaceOnSave was refactored). Markup rules for previewing author signatures are no longer needed and were removed.

A bug and a warning for PHP 4 installations were fixed. Two minor bugs with the `[[<<]]` line break for the responsive skin and the `$Version` variable link in the documentation were fixed.

The InterMap prefix to Wikipedia was corrected to use the secure HTTPS protocol and the documentation was updated.

Version 2.2.98 (2017-05-31)

This version adds a new skin that is better adaptable to both large and small screens, desktop and mobile devices (touchscreens). The new skin "pmwiki-responsive" is not enabled by default but available as an option, and as a base for customized copies. It requires a relatively modern browser (post-2009). The old skin is still available and enabled by default.

The Vardoc links now use MakeLink() to allow a custom LinkPage function. The function ReplaceOnSave() was refactored to allow easier calling from recipes. Markup processing functions now can access besides `$pagename`, a $markupid variable that contains the "name" of the processed markup rule, allowing a single function to process multiple markup rules. The "*.mkv" video extension was added to the list of allowed uploads.

A bug was fixed with the `(:markup:)` output where a leading space was lost. Note that the "markup" frame is now wrapped in a <pre> block with a "pre-wrap" style instead of <code>.

A number of other (minor) bugs were fixed: see ChangeLog, and the documentation was updated.

Version 2.2.97 (2017-04-07)

This version fixes a bug concerning `$ScriptUrl` when `$EnablePathInfo` is set, introduced in 2.2.96 and reported by 3 users.

Version 2.2.96 (2017-04-05)

This version fixes a severe PHP code injection vulnerability, reported by Gabriel Margiani. PmWiki versions 2.2.56 to 2.2.95 are concerned.

Only certain local customizations enable the vulnerability. Your website may be at risk if your local configuration or recipes call too early some core functions like CondAuth(), RetrievePageName() or FmtPageName(), before the `$pagename` variable is sanitized by ResolvePageName() in stdconfig.php. A specific URL launched by a malicious visitor may trigger the vulnerability.

Most recipes call core functions from a $HandleActions function, or from a Markup expression rule, these do not appear to be affected by the current exploit.

If your wiki may be at risk, it is recommended to upgrade to version 2.2.96 or most recent at the earliest opportunity. If you cannot immediately upgrade, you should place the following line in your local (farm)config.php file:

```
$pagename = preg_replace('![${}\'"\\\\]+!', '', $pagename);
```

Place this line near the top of the file but after you include scripts/xlpage-utf-8.php or other character encoding file.

This version filters the `$pagename` variable to exclude certain characters. A new variable $pagename_unfiltered is added in case a recipe requires the previous behavior. The documentation was updated.

Version 2.2.95 (2017-02-28)

This is a documentation update version.

Version 2.2.94 (2017-01-31)

This version allows webmasters to configure and use both .html and .htm extensions. The cached information about whether a page exists or not will now be cleared when that page is created or deleted. The documentation was updated.

Version 2.2.93 (2016-12-31)

This is a documentation update version.

Version 2.2.92 (2016-11-30)

This version allows administrators to disable the "nopass" password by setting `$AllowPassword` to false. The function FmtPageName() will now expand PageVariables with asterisks like `{*$FullName}`. The documentation was updated.

Version 2.2.91 (2016-09-30)

This is a documentation update version.

Version 2.2.90 (2016-08-31)

This version adds a parameter to the upload form which can improve analytics from the server logs. Two new CSS classes were added to help skin developers: `imgonly` and `imgcaption`, for standalone embedded pictures with or without a caption. A bug with the plus-links was fixed. The documentation was updated.

Version 2.2.89 (2016-07-30)

This version allows to set a default class name for simple tables. The`(:searchbox:)` directive can now have a "placeholder" attribute, and the input type can be changed from "text" to "search" for HTML5 websites. The edit form elements have now identifier attributes to allow easier styling. All core scripts will now inject CSS into the skin only if it hasn't already been defined. The vardoc.php script now recognizes and links to the documentation for the variables `$pagename`, `$Author` and `$Skin`. The documentation was updated.

Version 2.2.88 (2016-06-29)

This version fixes invalid HTML output of some WikiTrail links. The function PHSC() can now have an optional fourth argument for a safe replacement of htmlspecialchars(). A new page variable `{$SiteAdminGroup}` was added and the documentation was updated.

Version 2.2.87 (2016-05-31)

This version adds the `$HTMLTagAttr` variable to be used in the <html> tag in skins for additional attributes like "lang" or "manifest". To enable it, use it in your skin, for example:

```
<html xmlns="http://www.w3.org/1999/xhtml" $HTMLTagAttr>
```

The variable `$EnableRevUserAgent`, if set to 1, will cause the User-Agent string from browsers to be stored with each page history entry (as opposed to only storing the last user agent string). The output variable $DiffUserAgent can be used in history templates like $DiffStartFmt.

A wrong page variable in Site.UploadQuickReference was corrected, and the documentation was updated.

Version 2.2.86 (2016-04-28)

This version adds updates for PHP 7, for the PageStore() class and for the `$DefaultPasswords` default/unset definitions (no action should be needed upon upgrades). The documentation was updated.

Version 2.2.85 (2016-03-31)

This version adds Scalable Vector Graphics (*.svg, *.svgz) as allowed uploads and as embeddable picture extensions (with the html tag <img/>). The documentation was updated.

Version 2.2.84 (2016-02-21)

This version fixes "indent" and "outdent" styles for right-to-left languages. A new variable `$EnableLinkPlusTitlespaced` allows "plus links" `[[Link|+]]` to display the "Spaced Title" of the page instead the "Title". The documentation was updated.

Version 2.2.83 (2015-12-31)

This is a documentation update version.

Version 2.2.82 (2015-11-30)

This version enables stripmagic() to process arrays recursively and updates the documentation.

Version 2.2.81 (2015-10-31)

This version fixes an inconsistency with single line page text variables. International wikis enabling UTF-8 will now be able to use the CSS classes "rtl" and "ltr" to override the text direction when inserting right to left languages. The documentation was updated.

Version 2.2.80 (2015-09-30)

This version modifies the `(:searchbox:)` directive to use type="search" semantic input, and updates the documentation.

Version 2.2.79 (2015-08-27)

This version adds WikiStyles for the CSS basic colors "fuchsia", "olive", "lime", "teal", "aqua", "orange" and "gray"/"grey". New input elements "email", "url", "number", "date", and "search" can now be used in wiki forms.

Note: the "target" attribute of input forms which was added in the previous version broke the PmForm processor, and was removed until we find a solution. If you don't use PmForm and require this attribute (or others), the usual way to add it is to redefine the $InputAttrs array in your local configuration.

A new variable `$EnableROSEscape` can be set to 1 if `$ROSPatterns` and `$ROEPatterns` should not process source text wrapped with `[=...=]` or `[@...@]`. By default "replace on edit" patterns are performed even in such text.

The insMarkup() function in guiedit.js was refactored to allow custom input ids and/or custom functions to process the selected text.

The documentation was updated.

Version 2.2.78 (2015-07-21)

This version updates the $RobotPattern list with currently active user agents.~~Input forms can have a "target" attribute~~ (removed in 2.2.79). The documentation was updated.

Note, this release broke the Cookbook:PmForm module. Please do upgrade to 2.2.79 or newer if your wiki uses PmForm.

Version 2.2.77 (2015-06-19)

This version extends the `(:if attachments:)` conditional to specify file and page names. A `{$WikiTitle}` page variable was added. A MatchNames() function was introduced as a generic way to match array values the same way MatchPageNames() does currently with lists of pages -- recipe authors can use it to get a subset of attachments for example. The PageStore() class was slightly optimized when recoding pages from-to UTF-8. The documentation was updated.

Version 2.2.76 (2015-05-31)

This version improves support for arrays in form elements: setting default values and recovering values from posted forms. A new "label" argument to checkbox and radio input elements allows easy insertion of clickable text labels after the form elements. Division blocks wrapping standalone images, and standalone image captions, now receive CSS classes allowing greater control via stylesheets. The documentation was updated.

Version 2.2.75 (2015-04-26)

This version adds a pmcrypt($pass, $salt) function which can be used as a replacement for the PHP crypt() function when encrypting passwords. From PHP 5.6 on, crypt() should not be used without a $salt parameter and would raise a notice. If pmcrypt() is called with a $salt parameter it will simply call crypt() in order to check a password. If it is called without a $salt parameter, pmcrypt() will create a password hash with the password_hash() function or with crypt() depending on your installation. You can replace any calls to crypt() with pmcrypt(), notably in config.php when defining `$DefaultPasswords` entries.

Markup was added for the semantic HTML5 tags article, section, nav, header, footer, aside, address.

A bug with the uploads feature was fixed when $EnableReadOnly is set, and the documentation was updated.

Version 2.2.74 (2015-03-28)

This version allows the translation of the word "OK" in authentication forms. The documentation was updated to the latest state on pmwiki.org.

Version 2.2.73 (2015-02-28)

This release only updates the documentation to the latest state on pmwiki.org.

Version 2.2.72 (2015-01-27)

This version improves the ?action=ruleset display for markup rules potentially incompatible with PHP 5.5 when the function debug_backtrace() is not available. It restores the ability to set a custom function handling the (:markup:) demos. A variable `$AbortFunction` was added allowing administrators to override the core Abort() function. The documentation was updated.

Version 2.2.71 (2014-12-29)

This version removes the hard word wrap in `(:markup:)` wikicode examples, and instead of <pre> tags, it wraps it in <code> tags. This allows newcomers to copy and paste the code in their wikis without inserted line breaks (which often cause the markup to not work).

The release also adds back-tracing for markup rules potentially incompatible with PHP 5.5. Such rules, often added by recipes, can trigger "Deprecated: preg_replace()" warnings. To find out which recipes may trigger the warnings, enable diagnostic tools in config.php with `$EnableDiag = 1;` then open a page with the 'ruleset' action, eg. `[[HomePage?action=ruleset]]`. The PHP-5.5-incompatible rules will be flagged with filenames, line numbers and patterns. See also the pages Troubleshooting and CustomMarkup on pmwiki.org.

The variable `$DraftActionsPattern` was added, the pagelist "request" parameter can now contain a list of allowed or disallowed parameters that can be overridden by the user, the "input default source" parameter can now contain multiple pages, and a minor bug was fixed in upload.php ('strict' warning). See the updated documentation for more information.

Version 2.2.70 (2014-11-08)

This release only updates the documentation to the latest state on pmwiki.org.

Version 2.2.69 (2014-10-13)

This version fixes a bug when dates are defined as relative to other dates, eg. "2014-10-13 -3 days". The documentation was updated; note that the instructions in Site.UploadQuickReference were updated to reflect the display of the upload form in current browsers.

Version 2.2.68 (2014-09-01)

This version adds a Skins: InterMap prefix pointing to the Skins section on PmWiki.org, a "signature" markup in the edit quick reference, new WikiStyles clear, min-width and max-width and the documentation was updated.

Version 2.2.67 (2014-08-02)

This version fixes an inconsistency with input forms when values are taken from PageTextVariables. The documentation was updated to the latest state on pmwiki.org.

Version 2.2.66 (2014-07-02)

This version fixes a minor longstanding bug in the default Notification format when a page is deleted. In custom patterns, the "_" character will no longer be considered a function name. The documentation was updated.

Version 2.2.65 (2014-06-07)

This version fixes Pagelist handling of `{$$PseudoVars}` when they contain page variables. File permissions handling was improved when the current directory is owned by "root". The documentation was updated.

Version 2.2.64 (2014-05-08)

This version adds the "{(mod)}" markup expression for modulo/remainder calculations, and the "tel:" and "geo:" URI schemes which, on compatible devices like smartphones, allow the creation of links to dial telephone numbers and open map/location applications.

The $SysMergePassthru switch was added, if enabled, it allows the "Simultaneous Edits" conflict resolution to use the passthru() function instead of popen().

The documentation was updated.

Version 2.2.63 (2014-04-05)

This version allows for form elements to have custom attributes containing a dash in the attribute names and enables the attributes 'required', 'placeholder' and 'autocomplete' for HTML5 forms. A minor bug with pagelist {$$RequestVariables} appearing on some installations was fixed. The documentation was updated.

Version 2.2.62 (2014-02-28)

This version adds the variable `$EnableTableAutoValignTop` which allows to make advanced tables compatible with HTML5. For developers, a fourth argument $template was added to the Markup_e() function, and a callback template 'return' was added. The documentation was updated.

Version 2.2.61 (2014-01-31)

This version removes unnecessary snippets of code and adds the variable `$TableCellAlignFmt` which allows to make simple tables compatible with HTML5. The documentation was updated.

Version 2.2.60 (2014-01-12)

This version reverts the changes to the pmwiki.css file made in 2.2.59.

Version 2.2.59 (2014-01-11)

This version has an improvement for Blocklist when multiple text fields are posted. A bug with some nested markup conditionals was fixed. The default skin switched font sizes from points (fixed) to percents (relative). A couple of other minor bugs were fixed and the documentation was updated.

Version 2.2.58 (2013-12-25)

This version enables customization of (:input auth_form:), and fixes a couple of minor bugs. The documentation was updated.

Version 2.2.57 (2013-11-03)

This version enables the use of the Attach: link format in the (:attachlist:) directive. The documentation was updated.

Version 2.2.56 (2013-09-30)

This version aims to fix a PHP 5.5 compatibility issue with a deprecated feature of the preg_replace() function. The PageStore() class now detects and works around a bug with the iconv() function, and the documentation was updated.

Version 2.2.55 (2013-09-16)

This version adds the variable `$EnableDraftAtomicDiff`. If enabled, publishing from a draft version will clear the history of intermediate draft edits, and the published version will contain a single combined diff from the previous published version. The documentation was updated.

Version 2.2.54 (2013-08-13)

This version fixes a bug when old versions are restored from draft pages. The documentation was updated.

Version 2.2.53 (2013-07-08)

This version enables a message to be shown when a post is blocked because of too many unapproved links. The documentation was updated.

Version 2.2.52 (2013-06-08)

This version hides warnings about a deprecated feature in PHP 5.5 installations (preg_replace with /e eval flag). Three new upload extensions were added: docx, pptx and xlsx produced by recent versions of some office suites. The documentation was updated.

Version 2.2.51 (2013-05-08)

This version updates the addresses for the remote blocklists. A minor XSS vulnerability for open wikis, which was discovered today, was fixed. The documentation was updated.

Version 2.2.50 (2013-04-08)

This release only updates the documentation to the latest state on pmwiki.org.

Version 2.2.49 (2013-03-09)

This version adds an array `$UploadBlacklist` containing forbidden strings of an uploaded filename (case insensitive). Some Apache installations try to execute a file which has ".php", ".pl" or ".cgi" anywhere in the filename, for example, "test.php.txt" may be executed. To disallow such files to be uploaded via the PmWiki interface, add to config.php such a line:

```
  $UploadBlacklist = array('.php', '.pl', '.cgi', '.py', '.shtm', '.phtm', '.pcgi', '.asp', '.jsp', '.sh');
```

The documentation was updated.

Version 2.2.48 (2013-02-11)

This version fixes a bug introduced yesterday with some links.

Version 2.2.47 (2013-02-10)

This version enables tooltip titles in links to anchors in the same page, and the documentation was updated.

Version 2.2.46 (2013-01-07)

This version adds `$UploadPermAdd` and `$UploadPermSet` variables, and the documentation was updated.

If your wiki has uploads enabled, it is recommended to set the variable `$UploadPermAdd` to 0.

The `$UploadPermAdd` variable sets additional unix permissions applied to newly uploaded files, and should be 0 (recommended as of 2013). If uploaded files cannot be downloaded and displayed on the website, for example with the error 403 Forbidden, set this value to 0444 (core setting, default since 2004).

```
    $UploadPermAdd = 0; # recommended
```

The `$UploadPermSet` variable unconditionally sets the file permissions on newly uploaded files. Only advanced administrators should use it.

Version 2.2.45 (2012-12-02)

This version fixes some PHP notices appearing on some installations. The documentation was updated.

Version 2.2.44 (2012-10-21)

This version improves the display of consecutive whitespaces in page histories, and fixes the definition of PageTextVariables containing a dash. The documentation was updated.

Version 2.2.43 (2012-09-20)

This version makes it possible to use HTML attribute names that contain dashes, and removes a warning when editing and previewing Site.EditForm. The documentation was updated.

Version 2.2.42 (2012-08-20)

This version provides a workaround for cases when a wiki page contains a character nonexistent in the active encoding. The documentation was updated.

Version 2.2.41 (2012-08-12)

This version changes the internal $KeepToken separator to be compatible with more encodings. The documentation was updated.

Version 2.2.40 (2012-07-21)

This version provides a helper function replacing htmlspecialchars() and compatible with PHP 5.4. The documentation was updated.

Version 2.2.39 (2012-06-25)

This version provides a fix for links to attachments containing international characters. The documentation was updated.

Version 2.2.38 (2012-05-21)

This version fixes a "parameter count" warning which appeared on some websites.

Version 2.2.37 (2012-05-01)

This version provides a workaround for installations with broken iconv() function, while optimizing the recode function. This should fix the "Unable to retrieve edit form" problem in some wikis. Dots in sections are now better supported, PageVariables are expanded in PageList template defaults, and the documentation is updated.

Version 2.2.36 (2011-12-28)

This version fixes the recode function to try to recover Windows-1252 characters in ISO-8859-1 files. A new variable $EnableOldCharset enables the $page["=oldcharset"] entry which will be used in the future. A couple of minor bugs were fixed and the documentation was updated.

Version 2.2.35 (2011-11-11)

This release fixes a critical PHP injection vulnerability, reported today by Egidio Romano. PmWiki versions 2.2.X, 2.1.X, 2.0.X and 2.0.beta33 and newer are vulnerable. When you upgrade, please read carefully the Release notes for all PmWiki versions since yours.

If you cannot upgrade, it is recommended to disable Searches at the earliest opportunity (even if your wiki skin doesn't have a search form). Add to config.php such a line:

```
  if ($action == 'search') $action = 'browse';
```

If your old version wiki allows editing by not entirely trusted visitors, even on limited pages like a WikiSandbox, you should also disable PageLists. Add to config.php this line:

```
  $EnablePageList = 0;
```

This version has an important change for international wikis: the XLPage() function no longer loads encoding scripts such as xlpage-utf-8.php. When you upgrade, you need to include those scripts from config.php, before calling XLPage():

```
  include_once("scripts/xlpage-utf-8.php"); # if your wiki uses UTF-8
  XLPage('bg','PmWikiBg.XLPage');
```

All links can now have tooltip titles. Previously, only images and external links could have tooltip titles, now this feature is enabled for internal links. To set a tooltip title, add it in quotes after the link address:

```
  [[Main.HomePage"This is a tooltip title"]]
  [[Main.HomePage"This is a tooltip title"|Home]]
  [[http://www.pmwiki.org"Home of PmWiki"]]
  Attach:image.jpg"Tooltip title of the image"
```

The following new upload extensions were added: svg, xcf, ogg, flac, ogv, mp4, webm, odg, epub. A couple of minor optimizations were added (MarkupExpressions and rendering of page history) and the documentation was updated.

Version 2.2.34 (2011-10-10)

This version resets the timestamps of the default pages Site(Admin).AuthUser which are expected in case of upgrades from the versions 2.1.*. Core MarkupExpressions which manipulate strings should now work better with international characters. The documentation was updated to its latest state from pmwiki.org.

Version 2.2.33 (2011-09-23)

This version fixes a security bug introduced in 2.2.32 which left the groups Site and SiteAdmin open for reading and editing because the pages Site.GroupAttributes and SiteAdmin.GroupAttributes didn't have all necessary attributes.

All wikis running 2.2.32 should upgrade. If you cannot immediately upgrade, you can set the attributes from your wiki:

- open the attributes page [[SiteAdmin.GroupAttributes?action=attr]] and set a "read" and an "edit" password, `@lock` is recommended.
- open the attributes page [[Site.GroupAttributes?action=attr]] and set an "edit" password, `@lock` is recommended. Do not set a "read" password here.

The release also fixes the refcount.php script to produce valid HTML, and updates intermap.txt entries PITS: and Wikipedia: to point to their current locations.

Version 2.2.32 (2011-09-18)

This is the first version shipping with the core documentation in the UTF-8 encoding. PmWiki will automatically convert it on the fly for wikis using an older encoding.

It is recommended that all **new** PmWiki installations enable UTF-8. Migration of *existing* wikis from an older encoding to UTF-8 shouldn't be rushed: it is not trivial and will be documented in the future.

A required HTML xmlns attribute was added to the print skin template. The history rendering is now faster when many lines are added or removed.

Note: Due to a manipulation error, a version 2.2.31 was created before it was ready for a release.

Version 2.2.30 (2011-08-13)

This version fixes a $Charset definition in international iso-8859-*.php files. This will help for a future transition to UTF-8.

A variable $EnableRangeMatchUTF8 was added, set it to 1 to enable range matches of pagenames in UTF-8 like [A-D]. Previously the range matches were always enabled in UTF-8, but we found out that on some installations this feature breaks all pagelists, even those without range matches. In case the feature worked for you, you can re-enable it.

Version 2.2.29 (2011-07-24)

This release fixes Attach links that were broken with the Path fix in 2.2.28 earlier today.

Version 2.2.28 (2011-07-24)

This release fixes 2 potential XSS vulnerabilities and a bug with Path: links.

Version 2.2.27 (2011-06-19)

This release fixes a validation bug on pages after a redirection. A new block WikiStyle `%justify%` was added, allowing left and right aligned text. The page history now accepts a URL parameter `?nodiff=1` which hides the rendering of edit differences, showing only timestamps, authors, summaries and "Restore" links; it allows to restore a vandalized page with a huge contents or history which otherwise would break the memory or time limits of the server.

Version 2.2.26 (2011-05-21)

This release fixes a redundant removal of link hashes from WikiTrails, and updates the documentation to the most recent version from PmWiki.org.

Version 2.2.25 (2011-03-22)

This release only updates the documentation to the latest state on pmwiki.org.

Version 2.2.24 (2011-02-15)

This version reverts the way existing PageVariables are processed, like version 2.2.21 or earlier, but it adds a special variable $authpage which can be used in PageVar definitions. It is the same as the $page array, but exists only if the visitor has read permissions. For example, an administrator can set to config.php:

```
$FmtPV['$LastModifiedSummary'] = '@$authpage["csum"]'; # instead of '@$page["csum"]'
```

Then, the edit summary metadata will only be available if the user has read permissions.

Version 2.2.23 (2011-01-25)

This version sets the default value of `$EnablePageVarAuth` to 0 until we investigate a reported problem with authentication.

Version 2.2.22 (2011-01-16)

This version adds the variable `$EnableXLPageScriptLoad` which, if set to 0, will prevent authors to load scripts from XLPage and to accidentally change the encoding of the wiki. If you use it, make sure you include the required files, eg. xlpage-utf-8.php from local config files.

PageVariables should now respect authentications: without read permissions, the title, description, change summary, author of a protected page are unavailable. PageVariables that are computed without reading the page are still available (eg. $Group, $Namespaced, `$Version` etc.). Administrators can revert the previous behavior by adding to config.php such a line:

```
$EnablePageVarAuth = 0;
```

Version 2.2.21 (2010-12-14)

Due to a mis-configuration of a local svn repository, some of the changes intended for 2.2.20 didn't make it in the correct branch. This release corrects this.

Version 2.2.20 (2010-12-14)

This version fixes a potential XSS vulnerability, reported today. An AuthUser bug with excluding users from authgroups was fixed. A new InterMap prefix PmL10n: was added, it leads to the Localization section on PmWiki.org and should help the work of translators. A couple of other minor bugs were fixed and the documentation was updated.

Version 2.2.19 (2010-11-10)

This is a documentation-update release.

Version 2.2.18 (2010-09-04)

This version fixes 3 minor bugs, and updates the documentation.

Version 2.2.17 (2010-06-20)

This version adds a variable $PostConfig containing functions and scripts to be loaded after stdconfig.php. Tabindex was added as a valid form field attribute. Protected downloads now respect existing browser caches. AuthUser now allows more flexible cookbook recipe integration. A couple of bugs were fixed and the documentation was updated.

Version 2.2.16 (2010-05-10)

This version fixes a bug with parsing html attributes which could allow XSS injection. Wikis allowing unprotected editing are encouraged to upgrade.

A bug with the "center" button of the GUI edit toolbar was corrected.

The "exists" conditional now accepts wildcards, for example:

```
  (:if exists Main.*:)There are pages in the Main group (:if:)
```

The documentation was updated.

Version 2.2.15 (2010-03-27)

This version adds some minor bugfixes and optimizations notably a bug with `(:template none:)` introduced in the last version 2.2.14.

Version 2.2.14 (2010-02-27)

This release corrects inline styles for WikiTrail links. Undefined include/template`{$$variables}` are now removed from the included section, like Page(Text)Variables, and can be used in conditional expressions. If needed, this change can be reverted by adding to config.php such a line:

```
  $EnableUndefinedTemplateVars = 1; # keep and display unset {$$variables}
```

PageList templates now accept the sections `!first` and `!last` for markup to appear for every page in list *except* the first or last one.

"Title" attributes were added to external links. You can have tooltip titles on external links, including InterMap and attachments, by adding the link title in double quotes after the URL:

```
  [[http://www.pmwiki.org"Home of PmWiki"| External link]]
```

For international wikis, PmWiki now automatically translates the titles of technical pages like GroupAttributes or RecentChanges -- just define these strings as usual in XLPage, for example, in French:

```
  'AllRecentChanges' => 'Tous les changements récents',
```

Some minor optimizations were done and the documentation was updated.

Version 2.2.13 (2010-02-21)

This release fixes a bug with `$DiffKeepNum` introduced in 2.2.10 -- the count of revisions was incorrect and a page could drop more revisions than it should.

The page history layout was modified with a rough consensus in the community. The history now defaults to "source" view with word-level highlighting of the differences. Authors can see the changes in rendered output by clicking on the link "Show changes to output". Admins can switch back the default by adding such a line to config.php:

```
$DiffShow['source'] = (@$_REQUEST['source']=='y')?'y':'n';
```

To disable word-level highlighting and show plain text changes:

```
$EnableDiffInline = 0;
```

In the page history rendering, a few minor bugs were fixed and the code was slightly optimized.

The documentation was updated.

Version 2.2.12 (2010-02-17)

This release adds simple word-level highlighting of differences in the page history, when "Show changes to markup" is selected. To enable the feature, add to config.php such a line:

```
$EnableDiffInline = 1;
```

This feature is like what the InlineDiff recipe provides, but not exactly the same, and the implementation is simpler. It is enabled on PmWiki.org and can be improved -- your comments are welcome.

Version 2.2.11 (2010-02-14)

This release adds two new table directives for header cells, (:head:) and (:headnr:). They work the same way as (:cell:) and (:cellnr:) except that create <th> instead of <td> html tags.

The pagerev.php script was refactored into separate functions to allow easier integration of recipes displaying the page history.

A couple of minor bugs were fixed and the documentation was updated.

Version 2.2.9, 2.2.10 (2010-01-17)

Most important in this release is the official change of `$EnableRelativePageVars` to 1. The change is about how {$Variable} in included pages is understood by PmWiki.

- When `$EnableRelativePageVars` is set to 0, {$Name} displays the name of the currently browsed page. Even if {$Name} is in an included page, it will display the name of the browsed page.
- When `$EnableRelativePageVars` is set to 1, {$Name} displays the name of the physical page where it written. If {$Name} is in an included page, it will display the name of the included page.
- {*$Name} always displays the name of the currently browsed page, regardless of`$EnableRelativePageVars`.

So, if your wiki relies on page variables from included pages, and doesn't have `$EnableRelativePageVars` set to 1, after upgrading to 2.2.9, you can revert to the previous behavior by adding to config.php such a line:

```
$EnableRelativePageVars = 0;
```

More information about page variables can be found at:

```
http://www.pmwiki.org/wiki/PmWiki/PageVariables
```

This release adds a new variable `$EnablePageTitlePriority` which defines how to treat multiple (:title..:) directives. If set to 1, the first title directive will be used, and if a page defines a title, directives from included pages cannot override it. PmWiki default is 0, for years, the last title directive was used (it could come from an included page or GroupFooter).

This release also adds a new variable `$DiffKeepNum`, specifying the minimum number (default 20) of edits that will be kept even if some of them are older than the limit of `$DiffKeepDays`.

A number of bugs were fixed and the documentation was updated.

Version 2.2.8 (2009-12-07)

This release fixes another PHP 5.3 compatibility issue with conditional markup. The Author field now handles apostrophes correctly. The documentation was updated.

Version 2.2.7 (2009-11-08)

This release fixes most PHP 5.3 compatibility issues. Unfortunately some specific builds for Windows may still have problems, which are unrelated to PmWiki. Notably, on Windows, all passwords need to be 4 characters or longer.

Upload names with spaces are now correctly quoted. The documentation was updated.

Version 2.2.6 (2009-10-04)

With this release it is now possible to display recently uploaded files to the RecentChanges pages -- if you have been using the RecentUploadsLog recipe, please uninstall it and follow the instructions at http://www.pmwiki.org/wiki/Cookbook/RecentUploadsLog.

The release also introduces `$MakeUploadNamePatterns` to allow custom filename normalization for attachements. It is now possible to replace $PageListFilters and $FPLTemplateFunctions with custom functions. Notify should now work in safe_mode. Some bugs were fixed, among which one with conditional markup with dates. The documentation was updated.

Version 2.2.5 (2009-08-25)

This release adds a new markup for Pagelist templates, `(:template none:)` which allows a message to be set when the search found no pages. The FPLTemplate() function was broken into configurable sub-parts to allow development hooks. A number of bugs were fixed, and the documentation was updated.

Version 2.2.4 (2009-07-16)

This release fixes a bug introduced earlier today with HTML entities in XLPages.

Version 2.2.3 (2009-07-16)

This release fixes six potential XSS vulnerabilities, reported by Michael Engelke. The vulnerabilities may affect wikis open for editing and may allow the injection of external JavaScripts in their pages. Public open wikis should upgrade.

A new variable `$EnableUploadGroupAuth` was added; if set to 1, it allows password-protected uploads to be checked against the Group password.

It is now possible to use `@_site_edit, @_site_read, @_site_admin` or `@_site_upload` global passwords in GroupAttributes pages.

A number of other small bugs were fixed, and the documentation was updated.

Version 2.2.2 (2009-06-21)

The major news in this release is a fix of an AuthUser vulnerability.

The vulnerability affects only wikis that (1) rely on the AuthUser core module for User:Password authentication, -AND- (2) where the PHP installation runs with the variable "magic_quotes_gpc" disabled.

All PmWiki 2.1.x versions from pmwiki-2.1.beta6 on, all 2.2.betaX, 2.2.0, and 2.2.1 are affected.

The PmWiki SiteAnalyzer can detect if your wiki needs to upgrade:
`http://www.pmwiki.org/wiki/PmWiki/SiteAnalyzer`

If your wiki is vulnerable, you should do one of the following at the earliest opportunity:

- Upgrade to a version of PmWiki at least 2.2.2 or greater.
- Turn on magic_quotes_gpc in the php.ini file or in a .htaccess file.

Alternatively, you can temporarily disable AuthUser until you upgrade.

Note that even if your wiki does not have the AuthUser vulnerability at the moment, you are strongly encouraged to upgrade to PmWiki version 2.2.2 or later, as some future configuration of your hosting server might put you at risk.

This release also comes with minor updates in the local documentation; fixes were applied for international wikis - notably global variables in xlpage-utf-8.php and a new variable `$EnableNotifySubjectEncode`, which allows e-mail clients to correctly display the Subject header; and a number of other small bugs were fixed.

Version 2.2.1 (2009-03-28)

This release comes with an updated local documentation; wiki trails now work cross-group; guiedit.php now produces valid HTML, and other small bugs were fixed. We also added `$EnableRedirectQuiet`, which allows redirects to take place without any mention of "redirected from page ....".

Version 2.2.0 (2009-01-18)

This is a summary of changes from 2.1.x to 2.2.0.

- Several pages that were formerly in the Site.* group are now in a separate SiteAdmin.* group, which is read-restricted by default. The affected pages include Site.AuthUser, Site.AuthList, Site.NotifyList, Site.Blocklist, and Site.ApprovedUrls . If upgrading from an earlier version of PmWiki, PmWiki will prompt to automatically copy these pages to their new location if needed. If a site wishes to continue using the old Site.* group for these pages, simply set

`$SiteAdminGroup` = `$SiteGroup`;

when carrying out this upgrade inspect your config files for lines such as
`$BlocklistDownload`['Site.Blocklist-PmWiki'] = array('format' => 'pmwiki');
as you you may wish to fix then, eg
`$BlocklistDownload`[`$SiteAdminGroup` . '.Blocklist-PmWiki'] = array('format' => 'pmwiki');

- Important Change in Passwords in PmWiki 2.2 indicating that the group can be edited even if a site password is set will be done by `"@nopass"` prior it was done by `"nopass"`
When migrating a wiki you will have to manually modify the permission or by a script replace in all the page concerned `passwdread=nopass:` by `passwdread=@nopass` (see PITS:00961) --isidor

- PmWiki now ships with WikiWords entirely disabled by default. To re-enable them, set either `$LinkWikiWords` or `$EnableWikiWords` to 1. To get the 2.1 behavior where WikiWords are spaced and parsed but don't form links, use the following:
`$EnableWikiWords` = 1;
`$LinkWikiWords` = 0;

- It's now easy to disable the rule that causes lines with leading spaces to be treated as preformatted text -- simply set `$EnableWSPre`=0; to disable this rule.

  Important: There is ongoing discussion that the leading whitespace rule may be disabled *by default* in a future versions of PmWiki. If you want to make sure that the rule will continue to work in future upgrades, set `$EnableWSPre`=1; in *local/config.php*.

- The `$ROSPatterns` variable has changed somewhat -- replacement strings are no longer automatically passed through FmtPageName() prior to substitution (i.e., it must now be done explicitly).

- Page variables and page links inside of `(:include:)` pages are now treated as relative to the included page, instead of the currently browsed page. In short, the idea is that links and page variables should be evaluated with respect to the page in which they are written, as opposed to the page in which they appear. This seems to be more in line with what authors expect. There are a number of important ramifications of this change:

  - We now have a new `{*$var}` form of page variable, which always refers to "the currently displayed page". Pages such as Site.PageActions and Site.EditForm that are designed to work on "the currently browsed page" should generally switch to using `{*$FullName}` instead of `{$FullName}`.

  - The $EnableRelativePageLinks and `$EnableRelativePageVars` settings control the treatment of links and page variables in included pages. However, to minimize disruption to existing sites, `$EnableRelativePageVars` defaults to **disabled**. This will give existing sites an opportunity to convert any absolute `{$var}` references to be `{*$var}` instead.

  - Eventually `$EnableRelativePageVars` will be enabled by default, so we highly recommend setting `$EnableRelativePageVars = 1;` in *local/config.php* to see how a site will react to the new interpretation. Administrators should especially check any customized versions of the following:
    Site.PageActions
    Site.EditForm
    Site.PageNotFound
    SideBar pages with ?action= links for the current page
    `$GroupHeaderFmt`, `$GroupFooterFmt`
    Page lists that refer to the current group or page, etc in sidebars, headers, and footers

  - The `(:include:)` directive now has a `basepage=` option whereby an author can explicitly specify the page upon which relative links and page variables should be based. If no basepage= option is specified, the included page is assumed to be the base.

- Sites that want to retain the pre-2.2 behavior of `(:include:)` and other items can set `$Transition['version'] = 2001900;` to automatically retain the 2.1.x defaults.

- Text inserted via `(:include:)` can contain "immediate substitutions" of the form `{$$option}` -- these are substituted with the value of any options provided to the include directive.

- PmWiki now recognizes when it is being accessed via "https:" and switches its internal links appropriately. This can be overridden by explicitly setting `$ScriptUrl` and `$PubDirUrl`.

- A new `$EnableLinkPageRelative` option allows PmWiki to generate relative urls for page links instead of absolute urls.

- Draft handling capabilities have been greatly improved. When `$EnableDrafts` is set, then the "Save" button is relabeled to "Publish" and a "Save draft" button appears. In addition, an `$EnablePublishAttr` configuration variable adds a new "publish" authorization level to distinguish editing from publishing. See PmWiki:Drafts for more details.

- There is a new `{$:var}` "page text variable" available that is able to grab text excerpts out of markup content. For example, `{SomePage$:Xyz}` will be replaced by a definition of "Xyz" in SomePage. Page text variables can be defined using definition markup, a line beginning with the variable name and a colon, or a special directive form (that doesn't display anything on output):

```
:Xyz: some value           # definition list form
Xyz: some value            # colon form
(:Xyz: some value:)        # directive form
```

- The `(:pagelist:)` command can now filter pages based on the contents of page variables and/or page text variables. For example, the following directive displays only those pages that have an "Xyz" page text variable with "some value":

```
(:pagelist $:Xyz="some value":)
```

  Wildcards also work here, thus the following pagelist command lists pages where the page's title starts with the letter "a":

```
(:pagelist $Title=A* :)
```

- The if= option to `(:pagelist)` can be used to filter pages based on conditional markup:

  `(:pagelist if="auth upload {=$FullName}":)` pages with upload permission
  `(:pagelist if="date today.. {=$Name}":)` pages with names that are dates later than today

- Spaces no longer separate wildcard patterns -- use commas. (Most people have been doing this already.)

- Because page variables are now "relative", the `{$PageCount}`, `{$GroupCount}`, `{$GroupPageCount}` variables used in pagelist templates are now `{$$PageCount}`, `{$$GroupCount}`, `{$$GroupPageCount}`.

- One can now use `{$$option}` in a pagelist template to obtain the value of any 'option=' provided to the `(:pagelist:)` command.

- The `(:pagelist:)` directive no longer accepts parameters from urls or forms by default. In order to have it accept such parameters (which was the default in 2.1 and earlier), add a `request=1` option to the `(:pagelist:)` directive.

- The `count=` option to pagelists now accepts negative values to count from the end of the list. Thus `count=5` returns the the first five pages in the list, and `count=-5` returns the last five pages in the list. In addition, ranges of pages may be specified, as in `count=10..19` or `count=-10..-5`.

- Pagelist templates may have special `(:template first ...:)` and `(:template last ...:)` sections to specify output for the first or last page in the list or a group. There's also a `(:template defaults ...:)` to allow a template to specify default options.

- PmWiki comes with an ability to cache the results of certain `(:pagelist:)` directives, to speed up processing on subsequent visits to the page. To enable this feature, set `$PageListCacheDir` to the name of a writable directory (e.g., *work.d/*).

- The `(:if ...:)` conditional markup now also understands `(:elseif ...:)` and `(:else:)`. In addition, markup can nest conditionals by placing digits after if/elseif/else, as in `(:if1 ...)`, `(:elseif1 ...:)`, `(:else1:)`, etc.

- The `(:if date ...:)` conditional markup can now perform date comparisons for dates other than the current date and time.

- WikiTrails can now specify #anchor identifiers to use only sections of pages as a trail.

- A new `(:if ontrail ...:)` condition allows testing if a page is listed on a trail.

- The extensions .odt, .ods, and .odp (from OpenOffice.org) are now recognized as valid attachment types by default.

- A new blocklist capability has been added to the core distribution. It allows blocking of posts based on IP address, phrase, or regular expression, and can also make use of publicly available standard blocklists. See PmWiki.Blocklist for details.

- There is a new SiteAdmin.AuthList page that can display a summary of all password and permissions settings for pages on a site. This page is restricted to administrators by default.

- There are new `{$PasswdRead}`, `{$PasswdEdit}`, etc. variables that display the current password settings for a page (assuming the browser has attr permissions or whatever permissions are set in $PasswdVarAuth).

- Forms creation via the `(:input:)` markup has been internally refactored somewhat (and may still undergo some changes prior to 2.2.0 release). The new `(:input select ...:)` markup can be used to create select boxes, and `(:input default ...:)` can be used to set default control values, including for radio buttons and checkboxes.

- The `(:input textarea:)` markup now can take values from other sources, including page text variables from other pages.

- Specifying `focus=1` on an `(:input:)` control causes that control to receive the input focus when a page is loaded. If a page has multiple controls requesting the focus, then the first control with the lowest value of `focus=` "wins".

- PmWiki now provides a *scripts/creole.php* module to enable Creole standard markup. To enable this, add `include_once('scripts/creole.php')` to a local customization file.

- PmWiki adds a new `{(...)}` *markup expression* capability, which allows various simple string and data processing (e.g., formatting of dates and times). This is extensible so that recipe authors and system administrators can easily add custom expression operators.

- It's now possible to configure PmWiki to automatically create Category pages whenever a page is saved with category links and the corresponding category doesn't already exist. Pages are created only if the author has appropriate write permissions into the group. To enable this behavior, add the following to *local/config.php*:

  ```
  $AutoCreate['/^Category\\./'] = array('ctime' => $Now);
  ```

- Sites with wikiwords enabled can now set `$WikiWordCount`['WikiWord'] to -1 to indicate that 'WikiWord' should not be spaced according to `$SpaceWikiWords`.

- WikiWords that follow # or & are no longer treated as WikiWords.

- Links to non-existent group home pages (e.g., `[[Group.]]` and `[[Group/]]`) will now go to the first valid entry of `$PagePathFmt`, instead of being hardcoded to "Group.Group". For example, to set PmWiki to default group home pages to `$DefaultName`, use

  ```
  $PagePathFmt = array('{$Group}.$1', '$1.{$DefaultName}', '$1.$1');
  ```

- PmWiki now provides a $CurrentTimeISO and $TimeISOFmt variables, for specifying dates in ISO format.

- Cookbook authors can use the internal PmWiki function UpdatePage (temporarily documented at DebuggingForCookbookAuthors) to change page text while preserving history/diff information, updating page revision numbers, updating RecentChanges pages, sending email notifications, etc.

- Skin templates are now required to have <!--HTMLHeader--> and <!--HTMLFooter--> directives. Setting $EnableSkinDiag causes PmWiki to return an error if this isn't the case for a loaded skin. Skins that explicitly do not want HTMLHeader or HTMLFooter sections can use <!--NoHTMLHeader--> and <!--NoHTMLFooter--> to suppress the warning.

- Added a new "pre" wikistyle for preformatted text blocks.

- The xlpage-utf-8.php script now understands how to space UTF-8 wikiwords.

- Searches on utf-8 site are now case-insensitive for utf-8 characters.

- Many Abort() calls now provide a link to pages on pmwiki.org that can explain the problem in more detail and provide troubleshooting assistance.

- PmWiki no longer reports "?cannot acquire lockfile" if the visitor is simply browsing pages or performing other read-only actions.

- The $EnableReadOnly configuration variable can be set to signal PmWiki that it is to run in "read-only" mode (e.g., for distribution on read-only media). Attempts to perform actions that write to the disk are either ignored or raise an error via Abort().

- Including authuser.php no longer automatically calls ResolvePageName().

- Authentication using Active Directory is now simplified. In Site.AuthUser or the $AuthUser variable, set "ldap://name.of.ad.server/" with no additional path information (see PmWiki.AuthUser for more details).

- Pages are now saved with a "charset=" attribute to identify the character set in effect when the page was saved.

- The phpdiff.php algorithm has been optimized to be smarter about finding smaller diffs.

- Removed the (deprecated) "#wikileft h1" and "#wikileft h5" styles from the pmwiki default skin.

- The mailposts.php and compat1x.php scripts have been removed from the distribution.

Version 2.1.27 (2006-12-11)

This version backports from 2.2.0-beta a bugfix for `$TableRowIndexMax` and also support for the `{*$Variable}` markup.

Version 2.1.26 (2006-09-11)

This version fixes a bug in feeds.php that would cause feed entries to be mixed up.

Version 2.1.25 (2006-09-08)

This release fixes a bug in authuser.php introduced by the 2.1.24 release.

The skin template code has also been extended to allow `<!--XMLHeader-->` and `<!--XMLFooter-->` as aliases for `<!--HTMLHeader-->` and `<!--HTMLFooter-->`.

Version 2.1.24 (2006-09-06)

This release makes some improvements and fixes to the AuthUser capability.

A bug in authuser.php that had trouble dealing with non-array values in $AuthUser has been fixed.

It is now possible to specify group memberships from *local/config.php* (remember that such entries must come *before* including the *authuser.php* script):

```
# alice and bob's passwords
$AuthUser['alice'] = crypt('alicepassword');
$AuthUser['bob'] = crypt('bobpassword');

# members of the @writers and @admins groups
$AuthUser['@writers'] = array('alice',  'bob');
$AuthUser['@admins'] = array('alice', 'dave');

# carol is a member of @editors and @writers
$AuthUser['carol'] = array('@editors', '@writers');
```

AuthUser can now read from Apache-formatted .htgroup files. The location of the .htgroup file can be done either in *local/config.php* or Site.AuthUser

```
# local/config.php:
$AuthUser['htgroup'] = '/path/to/.htgroup';

# Site.AuthUser
htgroup: /path/to/.htgroup
```

Versions 2.1.21, 2.1.22, 2.1.23 (2006-09-05, 2006-09-06)

This release closes a potential security vulnerability for sites that are running with 'register_globals' set to on. Details of the vulnerability will be forthcoming on the mailing list and site.

Sites that are running with PHP 'register_globals' and 'allow_url_fopen' set to 'On' should upgrade to this release at the earliest opportunity. If upgrading isn't an option, contact Pm for a patch to older versions.

There is now a tool available to analyze PmWiki sites for security and other configuration settings, see PmWiki:SiteAnalyzer.

Version 2.1.23 also corrects a bug that prevented PmWiki from being able to read pagefiles created by versions of PmWiki before 0.5.6.

Version 2.1.20 (2006-09-04)

More minor bugfixes:

- Corrected a bug with WikiWord references appearing in the `(:attachlist:)` markup.
- Restore ability to remove/override PmWiki's default CSS settings.

Version 2.1.19 (2006-08-30)

This release provides a number of very minor bugfixes and enhancements:

- Fixed a bug in the pageindex code that was causing it to not regenerate as quickly as it should.
- Fixed image/object/embed handling in wikistyles to better support the Cookbook:Flash recipe.
- Fixed a bug with wikistyles and input form tags.

The next release(s) may have a number of substantial code enhancements and changes, so this release simply closes out a few items before introducing those changes.

Version 2.1.18 (2006-08-28)

This release closes a potential cross-site scripting vulnerability that could allow authors to inject Javascript code through the various table markups.

The release also adds a new `(:input image:)` markup to generate image input tags in forms.

Finally, this release corrects a problem with `?action=print` failing to properly set the `{$Action}` page variable.

Version 2.1.17 (2006-08-26)

This release fixes a long-standing bug with `$EnableIMSCaching` (PITS:00573), whereby login/logout operations wouldn't invalidate browser caches, causing some people to see versions of a page prior to the login/logout taking place.

The new IMS caching code maintains a "imstime" cookie in the visitor's browser that keeps track of the time of last login, logout, author name change, or site modification. This cookie is then used to determine the proper response to browser requests containing If-Modified-Since headers. (Previously only the time of the last site modification was available.)

Browsers which do not accept cookies will effectively act as though IMS caching is disabled.

Version 2.1.16 (2006-08-26)

This release makes some improvements to skin handling -- primarily this improves the capability of relocating skin files to other locations, and to provide the ability for recipes to insert items at the *end* of HTML output.

This release introduces a `<!--HTMLFooter-->` directive into skin templates, which allows recipes and local customizations to insert output near the end of a document using a `$HTMLFooterFmt` array from PHP.

Also, the `<!--HeaderText-->` directive, which inserts the contents of `$HTMLHeaderFmt` into the output, has now been renamed to `<!--HTMLHeader-->`. PmWiki will continue to recognize `<!--HeaderText-->` to preserve compatibility with existing skins, but `<!--HTMLHeader-->` is preferred.

A new `$SkinLibDirs` array has been introduced which allows the source locations and urls for skins to be specified from a customization file. By default `$SkinLibDirs` is set as

```
  $SkinLibDirs = array("./pub/skins/\ $Skin"      => " $PubDirUrl/skins/\ $Skin",
                 " $FarmD/pub/skins/\ $Skin" => " $FarmPubDirUrl/skins/\ $Skin");
```

The keys (on the left) indicate the places to look for a "skin .tmpl file" in the filesystem, while the values (on the right) indicate the url location of the "skin css file". Modifying the value of `$SkinLibDirs` allows a skin .tmpl file to be located anywhere on the filesystem.

As far as I can see, none of the changes introduced by this release should have any sort of negative impact on existing sites, so it should be safe to upgrade. (If I'm wrong, please let me know.)

Version 2.1.15 (2006-08-25)

This release includes a number of feature enhancements and code cleanups as reported or requested by administrators.

First, AuthUser's LDAP authentication system now allows the use of a `?filter` parameter, consistent with urls used for mod_auth_ldap authorization in Apache. See the newly updated LDAP section of the AuthUser documentation for more details.

A chicken-and-egg problem with the `@_site_*` authorization groups has been resolved. It's now possible to have a page's read authorization refer to things such as `_site_edit`.

Also, the RetrieveAuthPage() function -- used for retrieving pages only if the visitor is authorized to do so -- now recognizes a special level parameter of 'ALWAYS', which means to always authorize access regardless of the browser or visitors current permissions. This may be useful for allowing certain operations to take place from within trusted scripts without having to grant full authorization to the browser.

Hardcoded instances of the *local/* directory now use a customizable `$LocalDir` variable. This variable controls where PmWiki looks for *local/config.php* and per-group customization files. It may be useful for some Wiki Farm contexts. Note that this does not change or affect the location of *`$FarmD`/local/farmconfig.php*.

Some minor internal changes have been made to *scripts/wikistyles.php* to better accommodate the wikipublisher recipe. It's probably better if we don't try to explain them. :-)

Version 2.1.13, 2.1.14 (2006-08-15, 2006-08-16)

This release fixes a bug in handling numeric passwords, and also allows ldaps:// authentication sources.

Version 2.1.12 (2006-08-07)

This version introduces the ability to nest divs and tables. The standard `(:table:)` and `(:div:)` markups are still available, except that a `(:div:)` may contain a `(:table:)` and vice-versa.

As in previous versions of PmWiki, the `(:div:)` markup automatically closes any previous `(:div:)`. However, there are now `(:div1:)`, `(:div2:)`, etc. markups (and the corresponding `(:div1end:)`, `(:div2end:)`, ...) which can be used to uniquely distinguish divs for nesting purposes.

To restore PmWiki's previous "non-nested" div behavior, set $Transition['nodivnest'] = 1; in a local customization file.

Other changes in this release:

- Add a `(:noaction:)` directive to suppress display of page actions.
- Allow anchor tags to contain colons, hyphens, and dots.
- Add "white-space" as an allowed wikistyle.
- Other minor bug fixes and typographical corrections.

Version 2.1.11 (2006-06-09)

This is a minor update that prevents `%define=%` wikistyles from generating empty paragraphs in the HTML output. Prior to this release, markup lines containing only wikistyle definitions would often generate empty paragraphs (<p></p>), this release changes things so that a markup line beginning with `%define=` and containing only wikistyle definitions will not initiate a new paragraph.

Version 2.1.10 (2006-06-03)

Version 2.1.4 introduced an `{$Action}` page variable that would contain the current `?action=` value. Unfortunately, this page variable conflicted with a pre-existing `$Action` global variable that was being used by skins to display a human-friendly form of the current action. Since there's not really a clean way to resolve this, I've decided to keep `{$Action}` as a page variable with the current action value (as introduced in 2.1.4), and change the global for skins to be $ActionTitle. This will require updating skins to use $ActionTitle instead of $Action. I apologize for the conflict.

This release adds a Site.LocalTemplates page for the `fmt=#xyz` option in pagelist and search results. The list of pages to be searched can be customized via the `$FPLTemplatePageFmt` variable. The `fmt=#xyz` option will now also search the current page for a matching template before searching Site.LocalTemplates and Site.PageListTemplates.

The 'pmwiki' skin now places a <span> around the "Recent Changes" link in the header to make it somewhat easier to style.

Version 2.1.9 (2006-06-02)

This release fixes a long-standing and difficult-to-find bug with the handling of `[[~Author]]` links.

Version 2.1.8 (2006-06-01)

This release simply changes the $NotifyListFmt variable to be `$NotifyListPageFmt` (more descriptive), and adds a `$NotifyList` array that can be used to specify notification entries from a configuration file.

Version 2.1.7 (2006-05-31)

This release introduces a variety of improvements and bugfixes.

Vspace paragraphs are now divs: Version 2.1.7 changes the way that PmWiki handles vertical space in output (the infamous `<p class='vspace></p>` sequence). Instead of using paragraphs, PmWiki now generates `<div class='vspace'></div>` for vertical space sequences. In addition, PmWiki is able to collapse the vspace <div> with any subsequent paragraph tags, such that a sequence like

```
<div class='vspace></div><p>...paragraph text...</p>
```

is automatically converted to

```
<p class='vspace'>...paragraph text...</p>
```

This allows for better control over paragraph spacing. It is expected that this change in vspace handling will not have any detrimental effects on existing sites. Sites that have set custom values for `$HTMLVSpace` will continue to use the custom value. A site that wants to restore PmWiki's earlier handling of vspace can do so by adding the following to *local/config.php*:

```
$HTMLVSpace = "<p class='vspace'></p>";
```

Improved email notifications of changes: Version 2.1.7 incorporates a *notify.php* script that provides improved capabilities for sending email notifications in response to page changes. This script is intended to replace the previous MailPosts capability, which is now deprecated (but will continue to be supported in PmWiki 2.1.x). Details and instructions for using notify.php are in the PmWiki.Notify page.

Added 'group home page' syntax: A group name followed by only a dot or slash is automatically treated as a reference to the group's home page, whatever it happens to be. This simplifies some pagelist templates as well as a number of other items. In particular, group links in pagelist output now points to the correct locations (instead of being a page in the current group).

Several bugs and vulnerabilities have been fixed:
- The default width of edit forms is now more appropriate for Internet Explorer.
- Authentication failure messages from LDAP are now suppressed.
- Some cross-site scripting vulnerabilities in uploads and page links have been corrected (courtesy Moritz Naumann, http://moritz-naumann.com).
- A problem with invalid pagenames resulting in redirect loops has been corrected.

Version 2.1.6 (2006-05-22)

The primary improvement in this release is the addition of a pagename argument to the `(:if auth:)` conditional markup. Thus one can display markup based on a visitor's authorization to a page other than the current one. For example, to test for edit privileges to Main.WikiSandbox, one would use `(:if auth edit Main.WikiSandbox:)`. As before, if the pagename is omitted the directive tests authorization to the current page.

This release also restores the ability to have hyphens in InterMap link names.

Lastly, the release closes a potential cross-site scripting vulnerability in the WikiTrail markup, and provides some small performance improvements.

Version 2.1.4, 2.1.5 (2006-03-29)

This release fixes a few more bugs:
- Pagelist-based feeds using ?action=rss work again.
- Multi-term searches with special characters is fixed.

The release also adds a couple of items:
- There is now an `{$Action}` page variable.
- Usernames and passwords submitted to authuser.php can contain quotes.
- The `(:attachlist:)` command now uses a natural case sort.

Version 2.1.3 (2006-03-17)

This release fixes a bug that prevents the`lines=` option from working on sites running PHP 5.1.1 or later. It also re-fixes a bug involving empty passwords and LDAP authentication.

Version 2.1.2 (2006-03-16)

This release fixes a bug with handling "nopass" passwords. It also makes some speed improvements to large web feeds, and fixes a couple of minor HTML tag mismatches.

Version 2.1.1 (2006-03-13)

This release primarily fixes a bug with passwords containing multiple authorization groups, and in the process slightly liberalized the formatting of "@group" and "id:name" handling. This release also adds a new mechanism for managing and displaying FAQ pages.

Version 2.1.0 (2006-03-12)

This set of release notes is fairly lengthy, as it chronicles all of the changes since 2.0.13 (four months of development). A lot remains the same, but some changes warrant extra care when upgrading from a 2.0.x version to 2.1.0 (thus the major revision number change). As always, questions and issues can be mailed to the pmwiki-users mailing list.

Here's the list:

- WikiWords are now disabled by default. To enable them, set " `$LinkWikiWords` = 1;" in a local customization file. As of 2.1.beta2, you can now leave WikiWords enabled but have links to non-existent pages display without decoration -- to do this, place the following lines in *pub/css/local.css*:

```
span.wikiword a.createlink { display:none; }
span.wikiword a.createlinktext
  { border-bottom:none; text-decoration:none; color:inherit; }
```

- The `(:pagelist:)` code has been substantially revised. Pagelist formatting can now be specified using markup, and several defaults are available from Site.PageListTemplates. Also, several built-in pagelist formatting functions (FPLSimple, FPLByGroup, FPLGroup) are now removed in favor of the template code. The FPLByGroup function can be restored by setting $Transition['fplbygroup']=1; . **Remark:** Check to see if your page Site.PageListTemplates is not passwordprotected for viewing, otherwise the resulting pagelist will not be shown.

- `(:pagelist:)` now also understands wildcards in`group=` and `name=` arguments, as well as excluding specific names and groups.

- `(:pagelist:)` now has an "order=random" option.
- `(:searchbox:)` now accepts "group=", "link=", "list=", etc. options to be passed along to the search results. It also accepts a "target=" option that identifies the page on which to send the search query.
- `?action=search` will display the contents of the current page if it contains a `(:searchresults:)` directive, otherwise it uses the content of the page identified by `$PageSearchForm` (default is the search page for the current language translation).
- PmWiki no longer maintains a ".linkindex" file -- it now has a ".pageindex" file that contains not only a table of links, but also words used in each page (to speed up term searches). The maintenance of the .pageindex file can be disabled by setting `$PageIndexFile`='';
- The `$EnablePageListProtect` variable now defaults to true, so that read-only pages appear in pagelists only if the visitor has read authorization. Note that this can also slow down some `(:pagelist:)` and search commands, so if the site doesn't have any read-only pages or if you aren't worried with cloaking read-only pages from searchlists, it might be worth setting `$EnablePageListProtect`=0; .
- Whitespace indentation rules now exist and are enabled by default. Any line that begins with whitespace and aligns with a previous list item is considered to be "within" that list item. Text folds and wraps as normal, and the `(:linebreaks:)` directive is honored. To turn off whitespace indentation, use `DisableMarkup('^ws');`.
- A single blank line after a `!!Heading` is silently ignored.
- The `(:redirect:)` directive is now a true markup, and can be embedded inside conditional markups or includes. It also allows redirecting to an anchor in a page, such as `(:redirect PageName#anchor:)`. A new `from=` option allows the redirect to take place only from pages that match the given wildcard specification. The `status=` option allows a 301, 302, 303, or 307 HTTP status code to be returned.
- The built-in authorization function has gone through some substantial internal changes, however these changes should be fully backward compatible so that it doesn't impact any existing sites. (If it *does* cause a problem, please let me know so I can investigate why!) The password prompts are now specified by an admin-customizable Site.AuthForm page. In addition, the authorization function no longer creates PHP sessions for visitors that aren't being authenticated.
- The authuser.php has likewise been substantially updated. The new version should have complete backwards compatibility with previous authuser.php settings, but this version also offers the ability to configure authentication resources and authorization groups through the Site.AuthUser page. Note that by default the Site.AuthUser page can only be edited using the admin password.
- The `$EnableSessionPasswords` variable can be used to control whether passwords are held in PHP sessions. (This does not affect user authentication via AuthUser, however.)
- The `$Author` variable now defaults to `$AuthId` if not otherwise set by a script or cookie.
- The Site.SideBar page now defaults its edit password to the sitewide edit password (in `$DefaultPasswords`['edit']).
- PmWiki now supports a "draft edit" mode, enabled by `$EnableDrafts` = 1. This creates a "Save as draft" button that will save a page under a "-Draft" suffix, for intermediate edits.
- There is now an ?action=login action available.
- A potential security vulnerability for sites running PHP 5 with register_globals enabled has been fixed.
- The `[[PageName |+]]` markup is now available by default; this creates a link to PageName and uses that page's title as the link text.
- What used to be "markup variables" are now "page variables". These are always specified using the `{$`*`variable`*`}` syntax, and can be used in markup and in $...Fmt strings. In addition, one can request a value for a specific page by placing the pagename in front of the variable, as in `{`*`pagename`*`$`*`variable`*`}`.
- The *scripts/rss.php* script is now *scripts/feeds.php*, and is a complete redesign for web feed generation. The new version supports UTF-8 and other encodings, can generate Atom 1.0 (`?action=atom`), Dublin Core Metadata (`?action=dc`) output, and enclosures for podcasting. It also allows feeds to be generated from trails, groups, categories, and backlinks, and provides options (same as pagelists) for sorting and filtering the contents of the feed. Most sites can simply switch to using `include_once("scripts/feeds.php");` instead of the previous *rss.php* include. The *rss.php* file has been removed from the distribution (but still works with PmWiki 2.1 for those sites that wish to continue using it).
- InterMap entries can now come from a Site.InterMap page as well as the *local/localmap.txt* and *local/farmmap.txt* files. The format of these files has changed slightly, in that the InterMap name should now have a colon after it (previously the colon was omitted).

- We can now provide better control of robot (webcrawler) interactions with a site to reduce server load and bandwidth. The $RobotPattern variable is used to detect robots based on the user-agent string, and any actions not listed in the $RobotActions array will return a 403 Forbidden response to robots. In addition, setting $EnableRobotCloakActions will eliminate any forbidden ?action= values from page links returned to robots, which will reduce bandwidth loads from robots even further (PITS:00563).

- Non-existent page handling has been improved; whenever a browser hits a non-existent page, PmWiki returns the contents of Site.PageNotFound and a 404 ("Not Found") status code.

- Page links that have "?action=" in their query arguments are now treated as "existing page" links even if the page does not exist.

- The PmWiki default skin now adds rel='nofollow' to various action links.

- Some of the CSS styles in the PmWiki default skin have been changed for better presentation.

- The gui edit buttons have transparent (instead of white) borders so they integrate better into skins.

- The `$EnableIMSCaching` variable is now much smarter, it can detect changes in local customization files as well as pages.

- WikiStyles can now make percentage specifications by using "pct" to mean "%".

- Class attributes in WikiStyle shortcuts are now cumulative, so that `%class1 class2%` results in `class='class1 class2'` instead of just `class='class2'` in the output.

- A problem with the `(:include PageName#from#:)` markup not working has been fixed (PITS:00560).

- Viewing a GroupHeader or GroupFooter page no longer displays the contents twice.

- It's now easier to share pages among multiple sites (e.g., WikiFarms), see Cookbook:SharedPages (PITS:00459).

- A problem with nested apostrophe markups has been fixed (PITS:00590).

- PmWiki is now smarter about not surrounding block HTML tags with <p>...</p> tags.

- If an `[[#anchor]]` is used more than once in a page, only the first generates an actual anchor (to preserve XHTML validity).

- There are now `(:if equal ...:)` and `(:if exists pagename:)` conditional markups.

- Compound conditional markup expressions are now possible -- e.g.`(:if [ group PmWiki && ! name PmWiki ] :)`.

- Added an $InputValues array that can supply default values for certain form controls (PITS:00566).

- The default setting of `$UploadUrlFmt` is now based on `$PubDirUrl` instead of `$ScriptUrl`.

- The $text global variable has been removed (use $_GET['text'], $_POST['text'], or $_REQUEST['text']).

- A possible problem with url-encoding of attachments with non-ASCII characters has been addressed (PITS:00588).

- Page actions in non-existent pages no longer display with non-existent link decorations.

- A README.txt file has been added, and several documentation files are now available through the docs/ directory.

- PmWiki is no longer available through CVS on sourceforge.net. It is now available via SVN on pmwiki.org, at svn://pmwiki.org/pmwiki/tags/latest . For more details, see PmWiki:Subversion.

- The $NewlineXXX variable (deprecated in 2.0.0) has been removed.

- There is experimental support for server-side caching of pages that take a long time to render; this is currently an unsupported feature and may be removed in future releases.

Wiki administrators should note that from this release on PmWiki defaults to having WikiWords disabled.

To make sure WikiWords are enabled, use `$LinkWikiWords = 1;` in the *local/config.php* file.

Bugs and other requests can be reported to the PmWiki Issue Tracking System at http://www.pmwiki.org/wiki/PITS/PITS. Any help in testing, development, and/or documentation is greatly appreciated..

Release Notes archive - notes for versions older than 2.1.0.

Last modified by Petko on June 26, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/ReleaseNotes

Requirements

toc top

Prerequisites for running the PmWiki wiki engine:

1. PHP 4.3 or later
   - PHP 5.3 or later is recommended
   - for PHP 5.5 to 7.1 compatibility use the current version of PmWiki
2. Some sort of webserver that can run PHP scripts.

PmWiki has been reported to work with the following OS/webserver combinations:

- Apache 1.3 or 2.0, on roughly anything (Unix, Linux, Windows, and Mac OS X)
  - Apache 2.4 or later is recommended
- lighttpd (1.4.19 php-fastcgi ssl) on Linux
- nginx (0.8.47) on Windows
- Microsoft Internet Information Server, on Windows
- Linux + LiteSpeedWeb Server Standard Edition
- appWeb (a small, php-enabled webserver) executing on a Linksys NSLU2 Network Storage Link device

PmWiki has been reported not to work on:

- Mac OS before Mac OS X because there's no PHP available
- Specific Release Candidate builds of PHP 5.3 for Windows may not work correctly with passwords

The Standalone recipe provides a special, bare-bones webserver application that can be used to run PmWiki in places where another webserver isn't available. PmWiki can also be run from a USB drive.

Last modified by mfwolff on June 07, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Requirements

Search

toc top

This page uses custom searches.
For regular searches, view another page.

PmWiki provides a basic search function. While it is not powered by a "search engine", it can be tweaked to produce results that are *targeted* and *customized.*

Targeted searches

Searches can be targeted to restrict the search to certain pages. For example, a search can be restricted based on groups, where, for instance, "group=PmWiki" searches only the PmWiki group, and "group=-PmWiki" searches only pages that are not in the PmWiki group. In addition to groups, searches can be restricted based on page names ("name="), wiki trails ("trail="), backlinks ("link=") and other criteria (e.g. "list=normal") and capped at a maximum number ("count="). For documentation about each of these parameters, see page lists.

Customized display

The display of search results can be customized to control the format, content and order of the returned results.

fmt=
: select format and content by specifying a pagelist template that determines layout, such as list styles, and page elements, such as title and description.

order=
: allows results to be sorted according to different criteria, such as name and title. For documentation about each of these parameters, see page lists.

For examples of pagelist template formats see Site.Page List Templates, Site.Local Templates, and Cookbook:Pagelist Template Samples.

The `(:pagelist request=1 req=1:)` directives can be used instead of `(:searchresults:)` to remove the "Results of search for" message. Neither of these directives work for the `(:searchresults:)` or `(:searchbox:)` directives.

req=1
: disables the pagelist until search results are returned.

request=1
: see pagelists

This can be used in many more cases than the default pmwiki search. Data from pages with PTVs, etc can be searched, filtered, and reordered. Note that the default ordering is of text strings, ie. 1, 10, 2, 3 and not the numeric value 1, 2, 3, 10, but a custom pagelist sort function (see the cookbook) can return any order required.

Anyone, anywhere

Readers can create targeted and customized search results simply by typing the relevant parameters , e.g. "group=PmWiki", into search boxes together with their search string. **Authors** can predefine such targeted and customized searches by incorporating the parameters into pages using the `(:searchbox:)` and `(:searchresults:)` directives (documented at PageLists).

`(:searchresults:)` can be customized by editing page `Site.Search`.

See also

- `$PageSearchForm`
- `$SearchBoxOpt`
- `$SearchPatterns`

Try it: this page generates custom searches

Any search that is run from this page will automatically generate pre-defined sets of search results that***target*** different clusters of pages (documentation, cookbook and PITS, if available); use ***customized*** formats, content and ordering; and ***reveal*** the specific parameters used to generate each search result. Whether you use the search box below, or the regular search box that appears at the top of this page, any search that you run from this page will provide the customized results.

Search

Last modified by RDJones on November 29, 2012. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Search

Security

toc top

Aspects of PmWiki security are found on the following pages:

Pages distributed in a PmWiki release:

- Page history History of previous edits to a page
- Passwords General use of passwords and login
- Passwords Admin More password options for the administrator
- AuthUser Authorization system that uses usernames and passwords
- Url Approvals Require approval of Url links
- Site Analyzer
- Blocklist Blocking IP addresses, phrases, and expressions to counteract spam and vandalism.
- Notify How to receive email messages whenever pages are changed on the whole wiki site, individual groups or selected watchlists of pages
- Security variables variables crucial for site security

Cookbook pages

- Security recipes from the Cookbook
- Cookbook:HtpasswdForm Form based management of users and passwords using .htpasswd/.htgroup files
- Cookbook:Secure attachments Protecting uploaded attachments
- Cookbook:Web server security Making the server more secure with .htaccess
- Cookbook:Farm security Making Farm installations secure
- Cookbook:DeObMail Hide e-mail address
- Cookbook:Spam filters Automatic blocking of some spambots
- Cookbook:Audit images Check to see what images have been uploaded to your wiki.
- Cookbook:Private groups Create and secure private groups on a public wiki
- Cookbook:Only one login Only allow 1 login at the same time for a username
- Cookbook:Recipe check Check for new versions of recipes on pmwiki.org
- Cookbook:Session guard Protects against Session Theft
- Cookbook:TrackChanges Ways to more easily detect and verify all recent edits
- Cookbook:SwitchToSSLMode One approach to forcing https instead of http

How do I report a possible security vulnerability of PmWiki?

Pm wrote about this in a post to pmwiki-users from September 2006. In a nutshell he differentiates two cases:

1. The possible vulnerability isn't already known publicly: In this case please contact us by private mail.
2. The possible vulnerability is already known publicly: In this case feel free to discuss the vulnerability in public (e.g. on pmwiki-users or in the PITS).

See his post mentioned above for details and rationals.

What about the botnet security advisory at http://isc.sans.org/diary.php?storyid=1672?

Sites that are running with PHP's *register_globals* setting set to "On" and versions of PmWiki prior to 2.1.21 may be vulnerable to a botnet exploit that is taking advantage of a bug in PHP. The vulnerability can be closed by turning *register_globals* off, upgrading to PmWiki 2.1.21 or later, or upgrading to PHP versions 4.4.3 or 5.1.4.
In addition, there is a test at PmWiki:SiteAnalyzer that can be used to determine if your site is vulnerable.

Wiki Vandalism and Spam

Assumptions

you are using a Blocklist and Url approvals.

You don't want to resort to password protecting the entire wiki, that's not the point after all.
Ideally these protections will be invoked in`config.php`

How do I stop pages being deleted, eg password protect a page from deletion?

Use Cookbook:DeleteAction and password protect the page deletion action by adding `$DefaultPasswords['delete'] = '*';` to `config.php` or password protect the action with `$HandleAuth['delete'] = 'edit';`
or `$HandleAuth['delete'] = 'admin';` to require the edit or admin password respectively.

How do I stop pages being replaced with an empty (all spaces) page?

Add `block: /^\s*$/` to your blocklist.

how do I stop pages being completely replaced by an inane comment such as *excellent site*, *great information*, where the content cannot be blocked?

Try using the newer automatic blocklists that pull information and IP addresses about known wiki defacers.

(OR) Try using Cookbook:Captchas or Cookbook:Captcha (note these are different).

(OR) Set an edit password, but make it publicly available on the Site.AuthForm template.

How do I password protect the creation of new groups?

See Cookbook:Limit Wiki Groups

How do I password protect the creation of new pages?

See Cookbook:Limit new pages in Wiki Groups

How do I take a whitelist approach where users from known or trusted IP addresses can edit, and others require a password?

Put these lines to local/config.php:
```
## Allow passwordless editing from own turf, pass for others.
if ($action=='edit'
 && !preg_match("/^90\\.68\\./", $_SERVER['REMOTE_ADDR']) )
 { $DefaultPasswords['edit'] = pmcrypt('foobar'); }
```
Replace 90.68. with the preferred network prefix and foobar with the default password for others.

For a single IP, you may use
```
if($_SERVER['REMOTE_ADDR'] == '127.0.0.1') { # your IP address here
 $_POST['authpw'] = 'xxx';                  # the admin password
}
```

Please note the security issues : this means that you have your admin passwords in clear in config.php and someone with access to the filesystem can read them (for example a technician of your hosting provider) ; your IP address may change from time to time (unless you have a fixed IP contract with your ISP). When that happens, someone with your old IP address will be logged in automatically as admin on your wiki. It is extremely unlikely to become a problem, but you should know it is possible ; if you are behind a router, all other devices which pass through that router will have the same IP address for PmWiki - your wifi phone, your wife's netbook, a neighbour using your wifi connection, etc. All these people become admins of your wiki. Again, you should evaluate if this is a security risk ; In some cases, your ISP will route your traffic through the same proxy as other people. In such a case, thousands of people may have the same IP address.

See also Cookbook:AuthDNS & Cookbook:PersistentLogin

How do I password protect page actions?

See Passwords for setting in config.php
```
 $HandleAuth['pageactionname'] = 'pageactionname'; # along with :
 $DefaultPasswords['pageactionname'] = pmcrypt('secret phrase');
```
or
```
 $HandleAuth['pageactionname'] = 'anotherpageactionname';
```

How do I moderate all postings?

Enable PmWiki.Drafts
- Set `$EnableDrafts`, this relabels the "Save" button to "Publish" and a "Save draft" button appears.
- Set `$EnablePublishAttr`, this adds a new "publish" authorization level to distinguish editing from publishing.

How do I make a read only wiki?

In config.php set an "edit" password.

How do I restrict access to uploaded attachments?

See
- instructions for denying public access to the uploads directory
- see Cookbook:Secure attachments

How do I hide the IP addresses in the "diff" pages?

If the user fills an author name, the IP address is not displayed. To require an author name, set in config.php such a line:

```
$EnablePostAuthorRequired = 1;
```

The IP address can also be seen in a tooltip title when the mouse cursor is over the author name. To disable the tooltip, set in config.php:

```
$DiffStartFmt =
  "<div class='diffbox'><div class='difftime'><a name='diff\$DiffGMT'
href='#diff\$DiffGMT'>\$DiffTime</a>
   \$[by] <span class='diffauthor'>\$DiffAuthor</span> - \$DiffChangeSum</div>";
```

How do I stop some Apache installations executing a file which has ".php", ".pl" or ".cgi" anywhere in the filename

Use `$UploadBlacklist`

How do I stop random people from viewing the ?action=source (wiki markup) of my pages? I have `(:if auth edit:)` text that I don't want the world to see.

`$HandleAuth['source'] = 'edit';` or `$HandleAuth['source'] = 'admin';`

SecurityVariables

toc top

`$AllowPassword`
: This variable contains the special "nopass" password which was used in the past to leave pages or groups accessible without a password. Recent PmWiki versions use "@nopass" instead. If your wiki is old and/or may contain pages with the "nopass" password, you should not change it. If that variable is empty or set to false, PmWiki will not check if pages have a special "allowed password".

`$DefaultPasswords`
: Specifies default passwords for user admin or actions (edit, read, upload). See PasswordsAdmin#settingsitewidepasswords.

`$EnablePostAttrClearSession`
: A switch to control whether or not changing a page's attributes causes any existing passwords to be forgotten. The default is that changing attributes forgets any passwords entered; this can be changed by setting `$EnablePostAttrClearSession` to zero.

`$EnableSessionPasswords`
: Control whether passwords are saved as part of the session. If set to zero, then session passwords are never saved (although any AuthUser authentications are still remembered).

`$SessionEncode`
: Function to use to encode sensitive information in sessions. Set this to NULL if you want to not use any encoding. (See also `$SessionDecode` below.)

`$SessionDecode`
: Function to reverse the decoding given by `$SessionEncode` above. Set this to NULL if sensitive session values are not encoded.

`$HandleAuth`
: This sets the required authentication Level that is necessary to perform an action. When using the following example in your `config.php` you need to be authenticated as editor in order to view the page history:

```
    $HandleAuth['diff'] = 'edit';
```

`$PageAttributes`
: Set the string shown on the attributes page when entering a password for an action.

`$AuthLDAPBindDN`
: For sites using AuthUser with LDAP authentication, this specifies the distinguished name (DN) to be used to bind to the LDAP server to check identity.

`$AuthLDAPBindPassword`
: For AuthUser with LDAP authentication, this specifies the password used for binding (in conjunction with `$AuthLDAPBindDN` above).

`$EnablePublishAttr`
: Adds a new "publish" authorization level to distinguish editing of drafts from publishing - See `$EnableDrafts`.

`$EnablePageVarAuth`
: In PmWiki versions 2.2.22 and 2.2.23 this variable should be set to 0. In 2.2.24 it will no longer be used.

See also:

- Security
- `$EnablePageListProtect`

Last modified by Petko on December 02, 2016. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/SecurityVariables

SimultaneousEdits

toc top

PmWiki has support for handling the case where multiple authors attempt to edit the same page nearly simultaneously. Here's the basic scenario for systems where simultaneous edits are *not* handled:

- Alice starts to edit a page.
- Before Alice saves her edits, Bob requests an edit of the same page, and receives the page text prior to Alice's edits.
- Bob finishes with his edits and hits "save".
- Alice finishes editing her page, hits "save", and since she was working from a version of the page from before Bob had made his changes, she wipes out Bob's edits in the process.

PmWiki's simultaneous edit feature detects when this occurs, and instead of saving Alice's edits PmWiki presents Alice with a message that someone else changed the page while she was editing it. Furthermore, Bob's changes are merged into Alice's copy of the page, with any conflicts highlighted by <<<<<<< and >>>>>>>. Alice can then fix things as appropriate and save the updated page, or, if Alice is lazy, she can just hit "save" a second time and leave it to someone else to fix.

The simultaneous edits feature is also invoked whenever someone requests a page preview; thus if a page changes while previewing a page the author gets notification and can see the merged results.

How can I test/experiment with this feature?

1. Open up two browser windows and select the same page to be edited in each window (e.g., try WikiSandbox?action=edit).
2. In one browser window, make some changes to the page and then save those changes.
3. In the second browser window, make some different changes to the same page and hit "save". Since the page changed after the edit form was loaded into the second window, there's a potential edit conflict and you'll receive the "edit conflict message".
4. You can make any adjustments in the second window, and press "Save" again to save the changes.

Notice

Some server environments such as Windows and PHP running in safe_mode are unable to use the simultaneous edits capability distributed with PmWiki. See Cookbook:SimultaneousEdits for a solution for these environments.

Last modified by simon on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/SimultaneousEdits

SitePageActions

toc top

authors (basic)

The Site.PageActions page is used as the source of the default wiki commands shown in the default PmWiki skin at the top right of the page. It displays as follows:

- View
- Edit
- History
- Attach
- Print (group)
- Backlinks
- *Logout*
- pdf page group

Note that there are many other available actions from the Cookbook, and PmWiki diagnostics and scripts.

This page gives a brief explanation of how Site.PageActions are displayed and formatted, and pointers to where more information can be found.

Below is what is shipped as Site.PageActions with PmWiki version 2.2:

```
* %item rel=nofollow class=browse    accesskey='$[ak_view]'%      [[{*$FullName}                | $[View]
]]
* %item rel=nofollow class=edit      accesskey='$[ak_edit]'%      [[{*$FullName}?action=edit    |
$[Edit] ]]
* %item rel=nofollow class=diff      accesskey='$[ak_history]'%   [[{*$FullName}?action=diff    |
$[History] ]]
(:if auth upload:)
* %item rel=nofollow class=upload    accesskey='$[ak_attach]'%    [[{*$FullName}?action=upload |
$[Attach] ]]
```

```
(:ifend:)
* %item rel=nofollow class=print     accesskey='$[ak_print]'%     [[{*$FullName}?action=print  |
$[Print] ]]
(:if group Site,SiteAdmin,Cookbook,Profiles,PmWiki*:) (:comment delete if and ifend to enable
backlinks:)
* %item rel=nofollow class=backlinks accesskey='$[ak_backlinks]'% [[{*$Name}?action=search&q=link=
{*$FullName} | $[Backlinks] ]]
(:ifend:)
(:if enabled AuthPw:)
* %item rel=nofollow class=logout    accesskey="$[ak_logout]"%''  [-[[{*$FullName}?action=logout |
$[Logout] ]]-]''
(:ifend:)
```

To start with, we'll look at just the first line, and take it apart. This will also give us a good handle on how most of the other lines work.

List

Each line is an item in an unordered list, marked up by an unindented '*'. You can find out more about lists on the Basic Editing page.

PmWiki will normally display an unordered list as a set of bulleted items, but they can appear differently depending on the context and styles they are displayed in. This difference in display is generally controlled by CSS defined in the Skin: for the PageActions links, the list items are displayed inline.

Style

Following the '*', on the line we have `%item ... %` which is a WikiStyle. It is used to control the properties of a given output element, like its size or color. By default they apply to the text between them and the end of the line or a closing `%%`, whichever is sooner. So, for example, one can enter `"this %blue%text%% is blue"` and it will appear as "this text is blue".

In this case the WikiStyle starts with the word `item`, and that says to apply the given style to the entire list item as opposed to just the text that follows. In particular, it causes PmWiki to generate HTML of

```
<li class='edit'>...</li>
```

instead of

```
<li><span class='edit'>...</span></li>
```

Setting the class attribute of the list item allows CSS properties to be applied to the item that corresponds to the current action. For example, to have the current action display with a background color of blue, a wiki administrator can do:

```
$HTMLStylesFmt[]= ' .{$Action} { background-color: blue; }';
```

Then if the current action is 'edit' (as in "?action=edit"), the list item corresponding to the edit action will be drawn with a blue background.

The other property inside the `%item ... %` WikiStyle is the accesskey='' statement. AccessKeys are keyboard shortcuts for tasks that would otherwise require a mouse. They can be attached to links or to form elements and the WikiStyle will use whichever it finds first on the line. In this case they will attach to the link `[[{*$FullName} | $[View] ]]`.

Accesskey

An accesskey can be defined in a number of locations, but essentially it is a phrase translation following the model used for internationalizations. PmWiki's accesskey defaults are defined in `scripts/prefs.php`, but can be overridden in lots of different places, including skins, language translation pages (XLPage), and even per-browser preferences (see Site.Preferences).

The `$[...]` markup defines phrase translations, used for internationalizations (and access keys, as noted above). In the first line of Site.PageActions it is used in both `$[ak_view]` and `$[View]`. Essentially `$[View]` tells PmWiki to substitute the current translation of "View". If no translation is defined for "View", then PmWiki just uses the phrase inside the brackets.

You can most easily see this working in the other languages sections of PmWiki. For example, atPmWikiDe/PmWikiDe you'll notice that the default "View", "Edit", "History", and "Print" actions are displayed as "Artikel", "Bearbeiten", "Historie", and "Druckansicht". This is because the PmWikiDe group is loading in a set of translations from PmWikiDe.XLPage

That page defines things like

```
'View' => 'Artikel'
'Edit' => 'Bearbeiten'
'History' => 'Historie'
'Print' => 'Druckansicht'
```

which says that things like `$[View]` and `$[Edit]` should be replaced by "Artikel" and "Bearbeiten".

This makes it very easy for PmWiki to support multiple languages, since a recipe author can simply put any translatable prompts or phrases inside of `$[...]`, and leave it to others to actually build the translation tables (either locally or on pmwiki.org for others to use). More information about `$[...]` is available at Internationalizations.

Link

All that leaves on the first line to be explained is the link itself: `[[{*$FullName} | $[View] ]]`. Links are not complex, but this one is using both the internationalization feature and a Page Variable. The `$[View]` has already been explained and it shows up in the link text section of link markup, so that, if viewed in English, the link will appear as View.

The link target section contains the `{*$FullName}` variable. This variable expands to the full name of the page on which it is being displayed, including the group and page names. For simple browsing, this is good enough, because viewing a page is the default action to perform on a page. Later lines use link targets like `{*$FullName}?action=edit` which says to go to the currently displayed page and start editing it.

If

This explains what all of the '*' lines are about. That only leaves the `(:if auth upload:)` and `(:ifend:)` lines, and they go together. The first starts some Conditional Markup and the second ends it. The `(:if test :)` markup only lets the following text be displayed if the test succeeds. The text that conditionally displayed ends at the next `(:if...:)` statement so an empty `(:ifend:)` is a convenient way to end the conditional block. The particular test being used here is `auth upload` which is only true if the current user is authorized to upload files to the wiki. Thus, the conditional block says to only display a link to perform an upload if the user is actually allowed to upload.

Depending on the security and permissions model on a given site, its not unusual to see many more conditional markups that test if, for example, a user has editing rights to the current page. More information on all the different conditions can be found at the Conditional Markup page, and a general index of all the PmWiki documentation can be found at Documentation Index.

Hopefully this bit of documentation has answered your questions about the Site.PageActions page. If not, you may wish to consult the helpful people on one of the PmWiki Mailing Lists.

Group PageActions

Note that any Group can have a PageActions page, not just **Site**. If a page named Group.PageActions exists, it will be used, otherwise, Site.PageActions, much like for the SideBar pages.

Last modified by Petko on May 01, 2012. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/SitePageActions

SitePreferences

toc top

The page Site.Preferences contains customisable browser preference settings. These include access keys (keyboard shortcuts to certain actions like edit, history, browse) and settings of the Site.EditForm (width and height of the edit textarea) as well as the name of the edit form in use.

A different page than Site.Preferences can be chosen by making a copy of that page under a new name, customising it, and setting a cookie which will point to this page for the browser being used, through

```
   ?setprefs=SomeGroup.CustomPreferences
```

SomeGroup.CustomPreferences being the name of the new customised preference page.

Notes and Comments

Note that in order to enable parsing of Site.Preferences by default, a line like the following needs to be added to local/config.php:

```
 XLPage('prefs', "Site.Preferences");
```

Last modified by OliverBetz on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/SitePreferences

SkinTemplates

toc top

This page describes the skin template files (.tmpl) that are used to create PmWiki *skins*, and how PmWiki uses them. As described in the skins page, a skin is a collection of files that specifies the layout for PmWiki pages. Each skin must include a template file that provides the skeleton for displaying a PmWiki page.

Finding and Processing Templates

When you set the value of the `$Skin` variable in a configuration file like `local/config.php`, like this

```
     ## Use the Foo Skin.
     $Skin = 'foo';
```

it tells PmWiki to search for a skin of that name, and use it. The usual result of the search is for PmWiki to load a template file from the appropriate skin directory. In this example, that would probably be the file `pub/skins/foo/foo.tmpl`.

The actual processing that PmWiki goes through to find a template file is important for those who are making complex skins, so its worth mentioning what those steps are:

1. When `$PageTemplateFmt` is blank (as it should be), PmWiki gathers the names of all candidate skins. It starts with any action-specific skin that is specified in `$ActionSkin[$action]`. Thus, if the current action is 'login', and `$ActionSkin['login']` is `'Bar'`, then PmWiki will look for a skin named 'Bar'.

2. If no skin has been found yet, it looks for the skin(s) named in the `$Skin` variable (which is allowed to be an array) and uses the first skin it can find. If it gets to the end of the list without finding a skin, it issues an error.

3. To attempt to find a skin, PmWiki first consults the `$SkinLibDirs` variable to know where to look. Skins consist of server-side files that are loaded by PmWiki (such as .php and .tmpl files) and client-side files (such as .css files and images) that will be requested by the user's browser when they look at a skinned PmWiki page. `$SkinLibDirs` is an array of key/value pairs. The key is a directory to look in for the server-side files, while the corresponding value is a URL that points to the public client-side resources used by the skin. The default value of `$SkinLibDirs` is:

```
    $SkinLibDirs = array(
      "./pub/skins/\$Skin"      => "$PubDirUrl/skins/\$Skin",
      "$FarmD/pub/skins/\$Skin" => "$FarmPubDirUrl/skins/\$Skin");
```

   So, using the above definitions, PmWiki would try to find the skin 'foo' by looking for a directory called `./pub/skins/foo` and then for `$FarmD/pub/skins/foo` (with the value of `$FarmD` replaced by the root server directory for Farm files). The first such directory that was found would be assumed to contain the skin it was looking for. It would then set `$SkinDir` to the name of this directory and `$SkinDirUrl` to the corresponding URL.

Security Note

The default value for `$SkinLibDirs` has server-side and client-side files stored in the same publicly-accessible directory. That is, `$SkinDir` and `$SkinDirUrl` point to the same place. This is done for convenience (both for the skin user, and the skin writer), but it is not necessary.

It has the side effect that its possible to construct a URL (like this one) that will let you look at the contents of the the `.tmpl` or `.php` files that a skin uses. This is usually not an issue as skin files should not contain any sensitive information.

Still, a purist might want to move their `.tmpl` and `.php` files out of the directories that are accessible as URLs, and modify their `$SkinLibDirs` array to reflect this.

4. Once a valid skin directory has been found, PmWiki starts processing the files in that directory, looking for a `php` skin file to run. It first looks for one with the same name as the skin. So, if the skin is 'foo', it looks for `foo.php`. If no such file is found, it then checks for a file named `skin.php`. If one of these `.php` files is found, PmWiki loads and runs it. This allows a skin to define custom markup, or custom configuration parameters. It also allows a skin to choose between which of several different `.tmpl` files to load.

   To specify which `.tmpl` file to load, simply call `LoadPageTemplate()` inside the skin `.php` file, with the name of the `.tmpl` file to be loaded:

```
    LoadPageTemplate( $pagename, "$SkinDir/xyz.tmpl");
```

   For example, a skin might specify a special template to be used if the action is 'print':

```
    if ($GLOBALS['action'] == 'print')
      LoadPageTemplate($pagename, "$SkinDir/print.tmpl");
```

   When the action is something else, PmWiki will fall back to loading the default `.tmpl` file instead.

5. If no appropriate `.php` file is found, or if that file doesn't load a template, then PmWiki falls back to looking for a template with the same name as the skin, or, failing that, any `.tmpl` file at all, so long as its the only one in the directory. If it finds one, it will load and process it. If not, it will issue an error.

Template file format

A template file is basically an HTML file that also contains variable substitutions (indicated by '$') and special directives embedded in HTML comments. The following special directives are *required* in the template file.

1. The directive `<!--PageText-->` belongs to the <body> section of the HTML document, and tells PmWiki where the main content of each wiki page should be placed.
2. The directive `<!--HTMLHeader-->`, which goes somewhere in the <head> section of the HTML document.
3. The directive `<!--HTMLFooter-->` directive, which typically goes before the final </body> tag and is used by some recipes to insert things at the end of the HTML document. *Prior to PmWiki 2.2.0 the `<!--HTMLFooter-->` directive was optional.*

When PmWiki displays a page, it replaces the directives and variable substitutions with the values appropriate to the current page. For example, the `<!--PageText-->` directive is replaced with the page's contents, while any instances of $PageUrl are replaced with the url (address) of the current page.

Note: your skin template shouldn't have a <meta/> tag specifying the charset (encoding), as PmWiki adds this tag when needed.

There is a long list of variables available for substitution in pages; some of the most useful include:

```
    $PageUrl         the url of the current page
```

```
$ScriptUrl       the base url to the pmwiki.php script
$Title           the page's title (e.g., "`SkinTemplates")
$Titlespaced     the page's title with spaces (e.g., "Skin Templates")
$Group           the name of the current group  (e.g., "`PmWiki")
$FullName        the page's full name (e.g., "`PmWiki.SkinTemplates")
$LastModified    the page's last modification time
$PageLogoUrl     the url of a site logo
$WikiTitle       the site's title
$SkinDirUrl      the url of the skin's folder
```

This last variable, `$SkinDirUrl`, is particularly useful in templates as it allows the skin designer to refer to other files (such as images or style sheets) in the skin folder without having to know the exact url.

The template is not limited to using the variables listed here; nearly any PHP global variable that begins with a capital letter can be used in a skin template. Page variables can also be used in templates.

Skin directives

Besides the required `<!--PageText-->` and `<!--HTMLHeader-->` directives, PmWiki provides other built-in directives for generating page output. It's not necessary to use any of these directives, but they can often add capabilities to a skin

`<!--wiki:Main.SomePage-->`
`<!--page:Main.SomePage-->`
: The `<!--wiki:Main.SomePage-->` directive outputs the contents of Main.SomePage. $-substitutions are allowed in directives, thus a directive like `<!--wiki:$Group.SomePage-->` will include "SomePage" of the current group.

  If multiple pages are listed in the directive, then only the first available page is used. Thus `<!--wiki:$Group.SomePage Site.SomePage-->` will display the contents of SomePage in the current group if it exists, and Site.SomePage if it doesn't. To always display Site.SomePage, even if $Group.SomePage exists, use two consecutive `<!--wiki:...-->` directives.

  The `<!--wiki:...-->` directive only displays pages for which the browser has read permissions. The `<!--page:...-->` directive displays pages even if the browser doesn't have read permission.

`<!--file:somefile.txt-->`
: The directive `<!--file:somefile.txt-->` outputs the contents of another file (on the local filesystem) at the point of the directive. If the file to be included is a .php script, then the PHP script is executed and its output is sent to the browser. Like the `<!--wiki:...-->` directive above, $-substitutions are available to be able to output files based on the current page name or group.

`<!--markup:...-->`
: The markup directive processes any text that follows the colon as wiki markup and displays that in the output.

`<!--function:SomeFunction args-->`
: This directive calls a PHP function named "SomeFunction", passing the *current page's name* as first argument, and the optional *text following the function name* as second argument. PHP functions called in this manner are typically defined in a local customization file. Args allows only one argument, which has to be split in your function. `<!--function:SomeFunction arg1 arg2 arg3-->` will call `SomeFunction($pagename, "arg1 arg2 arg3")` when the skin is processed. However variables can be used (like $LastModifiedBy).

Page sections

A template file can designate "sections" that are included or excluded from the output based on page directives or other criteria. A section always begins with `<!--Page...Fmt-->` and continues to the next section, the end of the template file, or `<!--/Page...Fmt-->`. For example, a template can specify a `<!--PageLeftFmt-->` section that is excluded from the output whenever the `(:noleft:)` directive is encountered in the page's contents. PmWiki's predefined sections (and their corresponding page directives) are:

```
<!--PageHeaderFmt-->          (:noheader:)
<!--PageFooterFmt-->          (:nofooter:)
<!--PageTitleFmt-->           (:notitle:)
<!--PageLeftFmt-->            (:noleft:)
<!--PageRightFmt-->           (:noright:)
<!--PageActionFmt-->          (:noaction:)
```

Skin designers can define custom sections and markups, but currently all section names in the template must begin with "Page" and end with "Fmt". As mentioned you also have to define the corresponding markup (for example in your config.php) like this:

```
    Markup('noxyz', 'directives',  '/\\(:noxyz:\\)/ei',
        "SetTmplDisplay('PageXYZFmt',0)");
```

And, better, compatible with PHP version 5.5, for PmWiki 2.2.58+ :

```
    Markup('noxyz', 'directives',  '/\\(:noxyz:\\)/i',
    "HideXYZ");
  function HideXYZ() {
    SetTmplDisplay('PageXYZFmt',0);
  }
```

See also: the recipe Skins:TestPageDirectives can help you test your skins with combinations of the above directives.

Internationalization (i18n)

Skins can also be internationalized by using `$[...]` substitutions. Any string placed inside of `$[...]` is treated as a "translatable phrase", and the phrase is looked up in the current translation tables for a corresponding output phrase. If a translation is available, then the translated phrase is substituted at that point, otherwise the original phrase is left intact.

For example, the substitution `$[Edit]` will display the current translation of "Edit" if it is known, otherwise it displays "Edit". Thus, the same template can be used for multiple languages, displaying "Editer" when French translations are loaded, "Bearbeiten" when German translations are loaded, and "Edit" when no translation is available.

How do I customize the CSS styling of my PmWiki layout?

> See Skins for how to change the default PmWiki skin. See also Skins, where you will find pre-made templates you can use to customize the appearance of your site. You can also create a file called *local.css* in the *pub/css/* directory and add CSS selectors there (this file gets automatically loaded if it exists). Or, styles can be added directly into a local customization file by using something like:

```
$HTMLStylesFmt[] = '.foo { color:blue; }';
```

Where can the mentioned "translation table" be found for adding translated phrases?

> See Internationalizations.

Is it possible to have the edit form in full page width, with no sidebar?

> If the sidebar is marked with `<!--PageLeftFmt-->`, adding `(:noleft:)` to Site.EditForm will hide it when a page is edited.

Can I easily hide the Home Page title from the homepage?

> Yes, you can use in the wiki page either `(:title Some other title:)` to change it or `(:notitle:)` to hide it.

Is it possible to hide the Search-Bar in the default PmWiki Skin?

> Yes, please see Cookbook:HideSearchBar.

Skins

What's a skin?

A skin changes the look and feel of a PmWiki page, Group of pages, or the entire wiki. To see this try some skins out using the links below.

- BeeblebroxNetGila
- JHSkin
- Amber
- Adapt
- Monobook
- Simple
- PmWiki (default)

As you see, all skins show the same page contents, but the other elements such as the sidebar, header, and footer, have changed. For example, different skins may display the sidebar on the left, on the right, or even not at all. Some skins have action links and features that others do not, especially if they were designed to take advantage of particular cookbook recipes.

So, a skin is just the set of files that determine how pages are displayed in PmWiki. Skins are stored as subfolders of *pub/skins/*. For example you might create the *example* skin in *pub/skins/example/*. Each skin typically has one or more of the following kinds of files:

- A template file, such as *skin.tmpl* or *example.tmpl*. The template is written in HTML or XHTML, and is the skeleton for the skin. It contains special markers that tell PmWiki where to insert the page's contents.
- CSS stylesheet files, which can control the skin's appearance, such as*pmwiki.css* or *example.css*.
- Image files, for decorating a page with images.
- PHP files, such as *skin.php* or *example.php*. These let skins provide extra customization setting or capabilities that HTML and CSS alone cannot.
- Documentation files for the administrator, usually with names like*readme.txt*, *install.txt* or *skinname.txt*. These usually give

you information about any special installation steps or nifty features the skin has.

Where do I get skins?

Skins are available in the Skins collection. The skins in the collection have been contributed by many PmWiki administrators for all to use, and typically have their own set of customization possibilities. When you find a skin you like, follow the link to download the skin package. You can also make your own skin.

How do I use or install a skin?

Most skin packages are .zip, .tgz, or .tar.gz files. You should be able to unpack these with most archiving software.

1. Unpack the skin to *pub/skins/* inside your pmwiki folder. Most well-designed skin packages will create a subfolder in *pub/skins/* named after the skin.
   - If the skin did not make a folder of its own, create one and move the skin files to it.

2. Open up your *local/config.php* file, and add a line like
   `$Skin = 'example';`

where *example* is the name of the skin's folder.

Reload a page from your wiki in the browser, and you should be able to see the difference.

If you'd like to let your site's visitors choose one skin from a selection of skins you've installed, look at the Skin Change recipe. (That's what we used for the demo above.)

How can I modify an existing skin?

There are a number of ways to further customize the appearance of a skin, including
- adding statements to /local/config.php that are compatible with your chosen skin;
- adding css files to /pub/css/, such as local.css (for your entire wiki) and MyGroup.css (for MyGroup); and
- directly editing the skin's files.

If the skin is updated regularly, you probably will want to avoid editing the files in the skin's folder. Check the skin's page in the Cookbook for specific suggestions.

If you want to modify the default pmwiki or print skins included with the PmWiki distribution, you should copy the pub/skins/pmwiki/ and pub/skins/print/ directories to another name and then use those skins instead of the default ones. While the name of the skin.tmpl and skin.css files don't usually matter, the optional skin.php file MUST match the name of the skin.

How can I make a skin?

The best way to make your first skin is to modify a copy of PmWiki's default skin.

1. Make a copy of the folder *pub/skins/pmwiki* and name it whatever your new skin should be named.
2. In your *local/config.php* file, set `$Skin` to be the name of your new skin.
3. Modify the template and CSS files to suit you.
4. Test your new skin.
5. Repeat steps 3 and 4 until you're happy with the results.

The reason we recommend starting with the default PmWiki skin is that it's quite a simple skin, much more so than many of the skins you'll find in Skins. The starting point is the template (.tmpl) file, which provides the overall layout of the page. Inside of the template file are a number of special substitutions and directives that provide places for PmWiki to insert the data relevant to the current page being displayed. Skin Templates describes the format and directives in more detail. There are also skin guidelines available on pmwiki.org.

It's beyond the scope of this page to explain how to write HTML (hypertext markup language), XHTML (extensible HTML, which is a bit newer) or CSS (cascading style sheets), but there are many good tutorials on the web for all three of them. One caution: if you run into an HTML tutorial that explains about how to use <font> or <blink> tags, or spacer gifs, it's at least five years out of date, so skip it and find another one.

You should test your skin on a variety of browsers -- ideally as many as you can, on as many different platforms as you can -- but at minimum you should be testing on Internet Explorer 8, Firefox 3, and Chrome, since those are the most common, and have different bugs, it is also useful to test on Opera and Safari. Don't forget to do things like resize windows and change text size during your testing.

Print Skins

By default your new skin will use the standard /pub/skins/print/ skin.

To over-ride this add the following to local/config.php:

```
    $ActionSkin['print'] = 'yourprintskin';
```

This says to use 'yourprintskin' for ?action=print instead of the default.

Tools that you'll need

There are good examples of all these programs available for free.

HTML and CSS editor(s). There are two types of editors: graphical (WYSIWYG, or "what you see is what you get"), and hand-coding or programmer's editors. Graphical editors are less intimidating to novices, but you won't learn as much, or know your code as intimately as you will by using a hand-coding editor. Whichever you choose, get one that has syntax highlighting for the code, because it will help you spot mistakes. Also, live preview features are not that helpful when writing a PmWiki skin, because PmWiki does stuff that the live preview won't, such as substitute values for variables and insert sidebar content.

Test wiki. You don't want to be wreaking havoc on your skin while visitors can see your site. It's a better idea to set up a test wiki, either on your real webserver or on your own machine. Linux or MacOS computer owners may have webservers and PHP already running on their machines, but Windows users often don't. If that describes you, then you might want to take a look at the Cookbook:Standalone recipe, which runs PmWiki without needing a complex webserver, or Cookbook:InstallOnIIS. Or, you can find many local server packages which install a webserver, PHP, and other stuff (e.g. MySQL), all configured to work together. Try to get a package that has the same software and versions as used on your live setup, since then there will be less to go wrong when the site goes live.

FTP client to transfer files to your webserver. You probably had one of these already.

Color picker. Your editor might include one, or you could pick up a standalone application. Extremely helpful for creating and saving color palettes.

See also

- PmWiki Installation   Obtaining and installing PmWiki
- SkinTemplates   Skin templates (.tmpl files)
- Skins
- Skin Guidelines
- Cookbook:Standalone

How do I change the Wiki's default name in the upper left corner of the Main Page?

Put the following config.php

```
$WikiTitle = 'My Wiki Site';
```

The *docs/sample-config.php* file has an example of changing the title.

How can I embed PmWiki pages inside a web page?

Source them through a PHP page, or place them in a frame.

How do I change the font or background color of the hints block on the Edit Page?

Add a CSS style to pub/css/local.css: `.quickref {background:...; color:... }`. The hints are provided by the Site.EditQuickReference page, which is in the PmWiki or Site wikigroup. Edit that page, and change the "bgcolor" or specify the font "color" to get the contrast you need.

SpecialCharacters

toc top

When creating pages it's common to use commercial trademarks, copyright, umlaut, and other non-keyboard symbols. therefore it's important that you have the means to input these special characters.

ISO Standard codes

PmWiki supports the HTML special character listings by the w3c. W3C Page of Special Character codes ISO standard.

Here are some samples:

```
&#169; | &#188; | &#189; | &#174; | &#181; | &#168;
```

© | ¼ | ½ | ® | µ | ¨

```
&#198; | 32&#176; | Un&#239;ted St&#228;tes | &#182; | &#165;Yen | PmWiki&#8482;
```

Æ | 32° | Unïted Stätes | ¶ | ¥Yen | PmWiki™

For a nice table with all available special characters, see List of Unicode characters at Wikipedia.

Other ways to do it:

Character Map

Find the "Character Map" utility in your computer's System Tools folder. Click the symbol you're interested in, and note the keystroke information at the bottom of the box. You execute these by *holding "Alt"* while keying the numbers *on the numerical keypad* of your keyboard (not the numbers across the top of the board).

&#169; = Alt+0169 = © | &#174; = Alt+0174 = ® | &#176; = Alt+0176 = ° (degrees).

Paste

- Use Word or another desktop application to create your text with the special characters that you want. Copy and paste the text to the wiki page you're editing or creating.
- Find an instance of a special character in an online document; copy and paste the character to your wiki page: ©

There's a list of special characters at PmWiki:SpecialCharactersList. There's another illustration at PmWiki:Characters

Table directives

toc top

There are six directives for table processing. All must be at the beginning of a line to have any effect.

`(:table [attr...]:)`

Generates a new HTML <table> tag with the attributes provided in*attr...*. Closes the previous table, if any. Valid attributes and values are:

- border *(a positive integer)*
- bordercolor *(a color name or hex number; doesn't display in all browsers)*
- cellspacing *(a positive integer indicating the space between cells)*
- cellpadding *(a positive integer indicating the interior border of a cell)*
- width *(a positive integer or percent)*
- bgcolor *(a color name or hex number)*
- align *(left, center or right)*
- summary *(does not display; used primarily to help visually disabled people navigate)*

`(:cellnr [attr...]:), (:cell [attr...]:), (:headnr [attr...]:), (:head [attr...]:)`

- The `(:head:)` directive opens a new "header cell" of the table (creates `<th>` tag in HTML).
- The `(:cell:)` directive opens a new "regular cell" of the table (creates `<td>` tag in HTML).
- The directives `(:headnr:)` and `(:cellnr:)` open a new cell on a new row in the table.

These directives close any previous cell and/or row. Note, the`(:head:)` and `(:headnr:)` directives exist from PmWiki version 2.2.11 or newer.

Valid attributes and values are:

- align *(left, center or right)*
- valign *(top, middle or bottom)* * *default is "top", see note below*
- colspan *(a positive integer)*
- rowspan *(a positive integer)*
- bgcolor *(a color name or hex number)*
- width *(a positive integer or percent)*
- class *(a CSS class of the cell)*
- style *(custom CSS styles of the cell)*

`(:tableend:)`

Closes the previous table cell and closes off any table. Generates </th>, </td>, </tr>, and </table> tags as needed.

* valign attribute

If not already set, PMWiki will automatically include the attribute valign='top' with all`(:cell[nr]:)` and `(:head[nr]:)`. Pm said "Table Directives were created for layout purposes and in that case it makes the most sense for each cell (column) to have its content at the top of the row. The attribute is placed in each cell and not in the row because certain browsers didn't recognize valign='top' in the row tag.

See `$EnableTableAutoValignTop` on how to disable the automatic insertion of the attribute.

Notes

For the table, cell, and cellnr tags the author can specify any attributes that would be valid in the HTML <table> or <td> tags. Thus you can specify rowspan, colspan, etc. arguments to build arbitrary tables. However, it's not possible to nest a `(:table:)` inside of a `(:cell:)` or `(:cellnr:)` -- the next paragraph explains why.

Many are likely to ask why we didn't just use the standard HTML table markup (<table>, <tr>, <td>, <th>) instead of creating a new markup, and allowing nested tables as a result. There are two answers: first, the HTML table markup is very ugly for naive authors (see PmWiki.Audiences and PmWikiPhilosophy #2), and second, it'd be very easy for authors to create tables that are incorrect HTML and that display incorrectly (or not at all) on some browsers. Even seasoned web professionals sometimes get the table markup wrong, so it's a bit unrealistic to expect the average author to always get it right, or to be able to read arbitrary HTML table markup that someone else has created.

> *Common comment:* Surely, the average or naive author would not be writing HTML directly, but using a tool, such as FrontPage, or even MSWord, to generate the HTML. This would be a lot simpler than learning even the simplest PmWiki markups.
>
> *Pm's Response:* And once the HTML has been generated and posted, how is someone else going to edit or modify the table if they don't have the original FrontPage or MSWord file used to create it? Remember that we're talking about *collaborative* authoring. The HTML that those packages generate is among the hardest to read and edit of all!

It's difficult to write the code needed to make PmWiki understand and fix arbitrary table markup, so PmWiki uses the simplified version above. Still, this version is able to handle most table requirements (with the possible exception of nested tables).

And, this is not to say that nested HTML tables are impossible in PmWiki --they just can't be easily created by wiki authors using the default wiki markup. A site administrator can of course create header/footer HTML code and other local customizations that make use of nested tables.

Example 1. A table using table directive markup.

"`&nbsp;`" is a non-breaking space in html. Place it in a cell if a cell is to be empty or the border of the cell will not be drawn properly.

```
(:table border=1 cellpadding=5 cellspacing=0:)
(:head:) a1
(:cell:) b1
(:cell:) c1
(:cell:) d1
(:headnr:) a2
(:cell:) b2
(:cell:) c2
(:cell:) &nbsp;
(:tableend:)
```

a1	b1	c1	d1
a2	b2	c2	

In HTML, this is the same as

```
<table border='1' cellpadding='5' cellspacing='0'>
  <tr>
    <th>a1</th>
    <td>b1</td>
    <td>c1</td>
    <td>d1</td>
  </tr>
  <tr>
    <th>a2</th>
    <td>b2</td>
    <td>c2</td>
    <td>&nbsp;</td>
  </tr>
</table>
```

Floating Table with bulleted navigation list

What if you wanted to create a nice little table like a table of contents in a page like this? In this example, the table is floating right and contains some links in a bulleted list. This is a nice demonstration of how it's possible to build a little table of contents in the page, which might navigate to other pages just within the same wiki group. Note that having a bulleted list *won't work in a ordinary table* - it only works inside an table created with table directives such as the example code used here.

```
(:table border=1 width=30% align=right bgcolor=#cccc99 cellspacing=0 :)
(:cellnr:)
'''Navigation Links'''
(:cellnr:)
*[[Tables]]
*[[Table directives]]
(:tableend:)
```

Navigation Links

- Tables
- Table directives

```
(:table border=1 width=30% align=right bgcolor=#cccc99 cellspacing=0 :)
(:cellnr colspan=2 align=center:)
'''Navigation Links'''
(:cellnr align=center:)
[[Tables]]
(:cell align=center:)
[[Table directives]]
(:tableend:)
```

Navigation Links	
Tables	Table directives

Looking at the markup here, notice that we have used a #cccc99 hex color for the table background. Also, the`(:cellnr:)` markup creates a new row, a new cell and closes the row at the end.

You could take this concept a little further: since you might want each page in the group to contain the same table of contents, you can make ONE table like the above and put it in its own page. Then use an include on any of your pages and bring in the table. The float (align) property will be honored in each page where it's included.

Can I define table headers using the table directive markup?

Yes, use `(:head:)` or `(:headnr:)` with PmWiki version 2.2.11 or newer. See also Cookbook:AdvancedTableDirectives.

Is it possible to do nested tables?

Yes, if you nest simple tables inside advanced tables. See also Cookbook:AdvancedTableDirectives.

Is it possible to add background images to tables and table cells?

Yes, see Cookbook:BackgroundImages.

Is it possible to apply styles to the elements of the table, like an ID to the table row, or a class/style to the TD?

Yes, see $WikiStyleApply.

Is it possible to automatically generate columns or rows in tables, i.e. without having to do a lot of counting?

Yes, this is possible with the Cookbook:CreateColumns recipe - it allows you to specify a certain number of columns, and/or to specify a certain number of items per column. Plus, someone has provided some similar markup on the TableDirectives-Talk page.

Last modified by Petko on February 28, 2014.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/TableDirectives

toc top

Tables

toc top

Table basics

PmWiki has two types of table markup; the markup described in this page is useful for creating*simple* tables with lots of small cells, while table directive markups help with larger scale tables. For more possibilities with table formatting see Cookbook:Rowspan in simple tables and Cookbook:Formatting tables.

Tables are created via use of double pipe characters:`||`. Lines beginning with this markup denote rows in a table or a formatting line. Within table row lines the double-pipe is used to delimit cells. In the examples below a border is added for illustration (the default is no border).

The first line in the markup contains formatting commands for the table. It only has double pipe characters at the start of the line.

Basic table

```
|| border=1
|| cell 1 || cell 2 || cell 3 ||
|| cell 1 || cell 2 ||
```

cell 1	cell 2	cell 3
cell 1	cell 2	

Header cells can be created by placing ! as the first character of a cell. Note that these are *table headers*, not *headings*, so it doesn't extend to !!, !!!, etc.

Table headers

```
|| border=1
||! cell 1 ||! cell 2 ||! cell 3 ||
```

cell 1	cell 2	cell 3

```
|| cell 1  ||  cell 2 ||  cell 3 ||
```

cell 1	cell 2	cell 3

A table can have a caption, indicated by `||!caption!||`. Any caption must appear prior to other rows of the table.

Table caption

```
|| border=1
||! A special table !||
||! cell 1 ||! cell 2 ||! cell 3 ||
|| cell 1  ||  cell 2 ||  cell 3 ||
```

A special table

cell 1	**cell 2**	**cell 3**
cell 1	cell 2	cell 3

Formatting cell contents

Cell contents may be aligned left, centered, or aligned right.

- To left-align contents, place the cell contents next to the leading `||`.
- To center contents, add a space before and after the cell contents.
- To right-align contents, place a space before the cell contents and leave the cell contents next to the trailing `|`.

Cell alignments

```
|| border=1 width=100%
||!cell 1      ||! cell 2  ||!       cell 3||
||left-aligned || centered || right-aligned||
```

cell 1	**cell 2**	**cell 3**
left-aligned	centered	right-aligned

Default cell alignments

```
|| border=1 width=100%
||!cell default||!cell left ||
||default-aligned||left-aligned ||
```

cell default	**cell left**
default-aligned	left-aligned

Note that header and detail cells have different default alignments.

To get a cell to span multiple columns, follow the cell with empty cells. (At present there is no markup for spanning rows.)

Column spanning

```
|| border=1 width=100%
|| |||| right column ||
|| || middle column ||||
|| left column ||||||
|| left column || middle column || right column ||
```

		right column
	middle column	
left column		
left column	middle column	right column

Table attributes

Any line that begins with `||` but doesn't have a closing `||` sets the *table attributes* for any tables that follow. These attributes can control the size and position of the table, borders, background color, and cell spacing. (In fact these are just standard HTML attributes that are placed in the <table> tag.)

Use the `width=` attribute to set a table's width, using either a percentage value, an absolute size, or*.

See also `$SimpleTableDefaultClassName`.

Table width

```
|| border=1 width=100%
|| cell 1 || cell 2 || cell 3 ||
|| c1 || cellcellcellcell2 || cell 3 ||
```

cell 1	cell 2	cell 3
c1	cellcellcellcell2	cell 3

The `border=` attribute sets the size of a table's borders.

Bordered table

```
|| border=10 width=70%
||!cell 1      ||! cell 2  ||!       cell
3||
||left-aligned || centered || right-
aligned||
```

cell 1	cell 2	cell 3
left-aligned	centered	right-aligned

Borderless table

```
|| border=0 width=70%
||!cell 1      ||! cell 2  ||!       cell
3||
||left-aligned || centered || right-
aligned||
```

cell 1	cell 2	cell 3
left-aligned	centered	right-aligned

Use `align=center`, `align=left`, and `align=right` to center, left, or right align a table. Note that `align=left` and `align=right` create a *floating table*, such that text wraps around the table.

Table alignment: center

```
|| border=1 align=center width=50%
||!cell 1      ||! cell 2  ||!       cell 3||
||left-aligned || centered || right-aligned||
Notice how text does not wrap with a table using "align=center".
```

cell 1	cell 2	cell 3
left-aligned	centered	right-aligned

Notice how text does not wrap with a table using "align=center".

Table alignment: left

```
|| border=1 align=left width=50%
||!cell 1      ||! cell 2  ||!       cell 3||
||left-aligned || centered || right-aligned||
Notice how text wraps to the right of a table using "align=left".
```

cell 1	cell 2	cell 3
left-aligned	centered	right-aligned

Notice how text wraps to the right of a table using "align=left".

Table alignment: right

```
|| border=1 align=right width=50%
||!cell 1      ||! cell 2  ||!       cell 3||
||left-aligned || centered || right-aligned||
Notice how text wraps to the left of a table using "align=right".
```

Notice how text wraps to the left of a table using "align=right".

cell 1	cell 2	cell 3
left-aligned	centered	right-aligned

Note: to get a table to align left (but not "float left") requires CSS, as in

```
    ||style="margin-left:0px;"
```

The `bgcolor=` attribute sets the background color for a table. At present there is no way to specify the color of individual rows or cells in this type of table (but see Cookbook:FormattingTables).

```
|| border=1 align=center bgcolor=yellow
width=70%
||!cell 1    ||! cell 2 ||!     cell 3||
||left-align || center  || right-align||
```

cell 1	cell 2	cell 3
left-align	center	right-align

How do I create a basic table?

Tables are created via use of the double pipe character: `||`. Lines beginning with this markup denote rows in a table; within such lines the double-pipe is used to delimit cells. In the examples below a border is added for illustration (the default is no border).

Basic table

```
     || border=1 rules=rows frame=hsides
     || cell 1 || cell 2 || cell 3 ||
     || cell 1 || cell 2 || cell 3 ||
```

cell 1	cell 2	cell 3
cell 1	cell 2	cell 3

How do I create cell headers?

Header cells can be created by placing ! as the first character of a cell. Note that these are *table headers*, not *headings*, so it doesn't extend to !!, !!!, etc.

Table headers

```
|| border=1 rules=cols frame=vsides
||! cell 1 ||! cell 2 ||! cell 3 ||
|| cell 1  ||  cell 2 ||  cell 3 ||
```

cell 1	cell 2	cell 3
cell 1	cell 2	cell 3

How do I obtain a table with thin lines and more distance to the content?

"Thin lines" is tricky and browser dependent, but the following works for Firefox and IE (Nov. 2009):

Thin lines and cell padding

```
||border="1" style="border-
collapse:collapse" cellpadding="5"
width=66%
||!Header ||! Header || '''Header'''||
||cells   ||  with   ||      padding||
||        ||         ||             ||
```

Header	Header	Header
cells	with	padding

How do I create an advanced table?

See table directives

My tables are by default centered. When I try to use '||align=left' they don't align left as expected.

Use ||style="margin-left:0px;" instead.

How can I specify the width of columns?

You can define the widths via custom styles, see Cookbook:FormattingTables and `$TableCellAttrFmt`. Add in config.php : `$TableCellAttrFmt = 'class=col$TableCellCount';`
And add in pub/css/local.css :
```
table.column td.col1 { width: 120px; }
table.column td.col3 { width: 40px; }
```

How can I display a double pipe "||" in cell text using basic table markup?

Escape it with `[=||=]` to display || unchanged.

How do I apply styles to the elements of the table, like an ID to the table row, or a class/style to the TD?

See $WikiStyleApply.

TextFormattingRules

This page provides a more complete list of some of the markup sequences available in PmWiki. Note that it's easy to create and edit pages without using any of the markups below, but *if* you ever need them, they're here.

To experiment with the rules, please edit the Wiki Sandbox.

Table of contents

Paragraphs

To create paragraphs, simply enter text. Use a blank line to start a new paragraph.

Words on two lines in a row will **wrap and fill** as needed (the normal XHTML behavior). To turn off the automatic filling, use the `(:linebreaks:)` directive above the paragraph.

- Use `\` (single backslash) at the end of a line to join the current line to the next one.
- Use `\\` (two backslashes) at the end of a line to force a line break.
- Use `\\\` (three backslashes) at the end of a line to force 2 line breaks.
- Use `[[<<]]` to force a line break that will clear floating elements.

Indented Paragraphs *(Quotes)*

Arrows (`->`) at the beginning of a paragraph can be used to produce an indented paragraph. More hyphens at the beginning (`--->`) produce larger indents.

```
->Four score and seven years ago our fathers placed upon this continent a new nation, conceived in
liberty and dedicated to the proposition that all men are created equal.
```

Four score and seven years ago our fathers placed upon this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal.

Inverted Arrows (`-<`) at the beginning of a paragraph can be used to produce a paragraph with a hanging indent. Adding hyphens at the beginning (`---<`) causes all the text to indent.

```
-<Four score and seven years ago our fathers placed upon this continent a new nation, conceived in
liberty and dedicated to the proposition that all men are created equal.
```

Four score and seven years ago our fathers placed upon this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal.

```
--<Four score and seven years ago our fathers placed upon this continent a new nation, conceived in
liberty and dedicated to the proposition that all men are created equal.  And that food would be good
too.
```

Four score and seven years ago our fathers placed upon this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal. And that food would be good too.

Blocks of text to which `(:linebreaks:)` has been applied can be indented by preceding the first line of the block with indention arrows (`->`) and aligning subsequent lines under the first. An unindented line stops the block indentation. See Cookbook:Markup Tricks for an example.

Bulleted and Numbered Lists

Bullet lists are made by placing asterisks at the beginning of the line. Numbered lists are made by placing number-signs (#) at the beginning of the line. More asterisks/number-signs increases the level of bullet:

```
* First-level list item
** Second-level list item
### Order this
#### And this (optional)
### Then this
** Another second-level item
* A first-level item: cooking
## Prepare the experiment
### Unwrap the pop-tart
### Insert the pop-tart into the toaster
## Begin cooking the pop tart
## Stand back
```

- First-level list item
    - Second-level list item
        1. Order this
            1. And this (optional)
        2. Then this
    - Another second-level item
- A first-level item: cooking
    1. Prepare the experiment
        1. Unwrap the pop-tart
        2. Insert the pop-tart into the toaster
    2. Begin cooking the pop tart
    3. Stand back

```
# A list is terminated
by the first line that is not a list.
# Also terminate a list using the escape sequence [@[==]@]
[==]
# Continue a list item by lining
  up the text with leading whitespace.
# Use a forced linebreak \\
  to force a newline in your list item.
```

1. A list is terminated

by the first line that is not a list.

1. Also terminate a list using the escape sequence `[==]`

1. Continue a list item by lining up the text with leading whitespace.
2. Use a forced linebreak
   to force a newline in your list item.

```
## Text between list items can cause numbering to restart
## %item value=3% this can be dealt with
```

1. Text between list items can cause numbering to restart

3. this can be dealt with

Also see: PmWiki:ListStyles, Cookbook:WikiStylesPlus.

Definition Lists

Definition lists are made by placing colons at the left margin (and between each term and definition):

Powerful new* feature
When you define terms using this markup PmWiki will recognize them as PageTextVariables that you can use on any page or PageList.
* Added in PmWiki version 2.2.0

```
:term:definition of term
```

term
: definition of term

Whitespace Rules

Whitespace indentation in lists. Any line that begins with whitespace *and aligns* with a previous list item (whether bulleted, numbers or definitional) is considered to be "within" that list item. Text folds and wraps as normal, and the `(:linebreaks:)` directive is honored.

```
# First-level item\\
  Whitespace used to continue item on a new line
# Another first-level item
  # Whitespace combined with a single # to create a new item one level deeper
```

1. First-level item
   Whitespace used to continue item on a new line
2. Another first-level item
   1. Whitespace combined with a single # to create a new item one level deeper

This rule also apply on definition lists, but only the number of leading colons is significant for the following whitespace indented lines.

```
:Item: Definition text
 dispatched on several
 lines
::SubItem: Same kind
  of multiline
  definition
```

Item
: Definition text dispatched on several lines
  SubItem
  : Same kind of multiline definition

Otherwise, lines that begin with whitespace are treated as*preformatted text*, using a monospace font and not generating linebreaks except where explicitly indicated in the markup. Note to administrators: Starting with version 2.2.0-beta41, this feature can be modified using `$EnableWSPre`. (Another way to create preformatted text blocks is by using the [@...@] markup.)

Horizontal Line

Four or more dashes (----) at the beginning of a line produce a horizontal line.

Emphasis and character formatting

- Enclose text in doubled single-quotes (''text''), i.e.,*two apostrophes*, for emphasis (usually *italics*)
- Enclose text in tripled single-quotes ('''text'''), i.e.*three apostrophes*, for strong (usually **bold**)
- Enclose text in five single-quotes ('''''text'''''), or triples within doubles (*five apostrophes*), for strong emphasis (usually ***bold italics***)
- Enclose text in doubled at-signs (@@text@@) for `monospace` text
- Use [+large+] for large text, [++larger++] for larger, [-small-] for small text, and [--smaller--] for smaller.
- Emphasis can be used multiple times within a line, but cannot span across markup line boundaries (i.e., you can't put a paragraph break in the middle of bold text).
- '~italic~' and '*bold*' are available if enabled in config.php

Other styling

```
'+big+', '-small-', '^super^', '_sub_',

{+insert or underscore+},

{-delete or strikethrough or strikeout-}
```

big, small, super, sub,

insert or underscore,

~~delete or strikethrough or strikeout~~

- `` `WikiWord `` WikiWord neutralisation

See also Wiki Styles for advanced text formatting options.

References

- Use words and phrases in double brackets (e.g., [[text formatting rules]]) to create links to other pages on this wiki.
- On some PmWiki installations, capitalized words joined together (e.g., WikiWords) can also be used to make references to other pages without needing the double-brackets.
- Precede URLs with "`http:`", "`ftp:`", "`gopher:`", "`mailto:`", or "`news:`" to create links automatically, as in http://www.pmichaud.com/toast.
- URLs ending with `.gif`, `.jpg`, or `.png` are displayed as images in the page
- Links with arbitrary text can be created as either [[*target* | *text*]] or [[*text* -> *target*]]. *Text* can be an image URL, in which case the image becomes the link to the remote *url* or *WikiWord*.
- Anchor targets within pages (#-links) can be created using `[[#target]]`.

See Links for details.

Headings

Headings are made by placing an exclamation mark (!) at the left margin. More exclamation marks increase the level of heading. For example,

```
!! Level 2 Heading
!!! Level 3 Heading
!!!! Level 4 Heading
!!!!! Level 5 Heading
```

Level 2 Heading

Level 3 Heading

Level 4 Heading

Level 5 Heading

Note that level 1 heading is already used as page title (at least in the PmWiki skin), so you should start with level 2 headings to create well formed, search engine optimized web pages.

See Cookbook:Numbered Headers for numbered headings.

Escape sequence

Anything placed between [= and =] is not interpreted by PmWiki, but paragraphs are reformatted. This makes it possible to turn off special formatting interpretations and neutralise WikiWords that are not links (even easier is to use a tick ` in front, like `WikiWord).

For preformatted text blocks, use the [@...@] markup. It does neither reformat paragraphs nor process wiki markup:

```
[@
Code goes here like [[PmWiki.PmWiki]]
'$CurrentTime $[by] $AuthorLink:  [=$ChangeSummary=]'; #just some code
@]
```

```
Code goes here like [[PmWiki.PmWiki]]
'$CurrentTime $[by] $AuthorLink:  [=$ChangeSummary=]'; #just some code
```

The multiline `[@...@]` is a block markup, and in order to change the styling of these preformatted text blocks, you need to apply a "block" WikiStyle.

```
%block blue%[@
  The font color of
  this text is blue
@]
```

```
  The font color of
  this text is blue
```

It is also useful to use `[= =]` within other wiki structures, as this enables the inclusion of new lines in text values. The example below shows how to include a multi-line value in a hidden form field.

```
(:input hidden message "[=Line1
Line2=]":)
```

Comments

`(:comment Some information:)` can be very kind to subsequent authors, especially around complicated bits of markup.

Special Characters

When creating pages it's common to use commercial trademarks, copyright, umlaut, and other non-keyboard symbols. therefore it's important that you have the means to input these special characters.

ISO Standard codes

PmWiki supports the HTML special character listings by the w3c. W3C Page of Special Character codes ISO standard.

Here are some samples:

```
&#169; | &#188; | &#189; | &#174; | &#181; | &#168;
```

© | ¼ | ½ | ® | µ | ¨

```
&#198; | 32&#176; | Un&#239;ted St&#228;tes | &#182; | &#165;Yen | PmWiki&#8482;
```

Æ | 32° | Unïted Stätes | ¶ | ¥Yen | PmWiki™

For a nice table with all available special characters, see List of Unicode characters at Wikipedia.

Other ways to do it:

Character Map

Find the "Character Map" utility in your computer's System Tools folder. Click the symbol you're interested in, and note the keystroke information at the bottom of the box. You execute these by *holding "Alt"* while keying the numbers *on the numerical keypad* of your keyboard (not the numbers across the top of the board).

&#169; = Alt+0169 = © | &#174; = Alt+0174 = ® | &#176; = Alt+0176 = ° (degrees).

Paste

- Use Word or another desktop application to create your text with the special characters that you want. Copy and paste the text to the wiki page you're editing or creating.
- Find an instance of a special character in an online document; copy and paste the character to your wiki page: ©

There's a list of special characters at PmWiki:SpecialCharactersList. There's another illustration at PmWiki:Characters

Tables

Tables are defined by enclosing cells with '||'. A cell with leading and trailing spaces is centered; a cell with leading spaces is right-aligned; all other cells are left-aligned. An empty cell will cause the previous cell to span multiple columns. (There is currently no mechanism for spanning multiple rows.) A line beginning with '||' specifies the table attributes for subsequent tables. A '!' as the first character in a cell provides emphasis that can be used to provide headings.

```
||border=1 width=50%
||!Table||!Heading||!Example||
||!Left   || Center || Right||
||A       ||!  a B   ||     C||
||        || single ||      ||
||        || multi span   ||||
```

Table	**Heading**	**Example**
Left	Center	Right
A	**a B**	C
	single	
	multi span	

See Table Directives for advanced tables.

Can't find it here?

See Markup Master Index.

Last modified by StefCT on June 24, 2014. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/TextFormattingRules

Troubleshooting

toc top

PmWiki is pretty robust and can automatically adapt to a very wide variety of environments. However, sometimes things don't go as we expect, so we're cataloging common errors and their fixes here.

Troubleshooting Frequently Asked Questions

Note: This page on pmwiki.org is probably not the best place to post questions. Consider seeking assistance from the pmwiki-users mailing list, or post your question on the PmWiki:Questions page.

My wiki displays warnings "Deprecated: preg_replace(): The /e modifier is deprecated, use preg_replace_callback instead".

This is caused by a change in PHP version 5.5 for the preg_replace() function. PmWiki no longer relies on the deprecated feature since version 2.2.56 (it is recommended to upgrade to the latest version) but many recipes do. Note that even if the warning points to a line in pmwiki.php, the problem comes from a local configuration or recipe.

Recipes and Skins are currently being updated for PHP 5.5. Check if there are more recent versions published by their maintainers on the Cookbook. If you update your PmWiki and recipes, and still see the warnings, here is how to find out which recipes cause them:

For PmWiki version 2.2.71 or newer, in config.php, enable diagnostic tools:
```
$EnableDiag = 1;
```
Then visit your wiki with the action 'ruleset', for example http://www.pmwiki.org/wiki/PmWiki/PmWiki?action=ruleset or follow a link like `[[HomePage?action=ruleset]]`. This page will list all markup rules; those potentially incompatible with PHP 5.5 will be flagged with filenames, line numbers and search patterns triggering the warning.

If the ?action=ruleset page shows no flagged rules, it is possible that either your recipes call the preg_replace() function directly, or they define various search-replace patterns in incompatible ways. In these cases, your warning should display the file name and line number causing problems, if not, here is how to track it. In config.php disable all recipes: included files from the cookbook directory, or a custom skin, or any line containing "Patterns". You can insert # at the beginning of a line to disable it. Then test the wiki: if you have disabled everything, the warning message should disappear.

Next, re-enable your customizations one after another, every time testing the wiki. If at some point the warnings re-appear, you'll know that the customization you just enabled is not compatible with PHP 5.5.

You can contact the authors of the broken recipes and (kindly) ask them to update their recipes for PHP 5.5 - recent PmWiki versions add new helper functions which make it easy, see CustomMarkup. If you cannot have the recipes fixed by their authors, tell us and we'll try to fix them.

Note that many hosting providers allow you to run different versions of PHP. See the documentation of your hosting plan to learn how to enable a PHP version earlier than 5.5.

Finally, it is possible to suppress these warnings in PHP 5.5, by setting this line at the beginning of config.php:
```
error_reporting(E_ALL & ~E_NOTICE & ~E_DEPRECATED);
```
This should be a temporary solution, left only until your recipes are fixed.

After a PHP upgrade, some of the pages on my wiki are completely blank, empty, some have blank or missing sections, but the sidebar and the action links are visible.

This can be caused by a change in PHP 5.4 which affects the function `htmlspecialchars()`.

The easiest temporary fix would be in your `php.ini`, or in `.user.ini` to change the `default_charset` directive to an 8-bit charset, for example cp1252:

```
default_charset = "Windows-1252"
```

Or, this may sometimes work in `pmwiki/local/config.php`:

```
ini_set("default_charset", "Windows-1252");
```

A more permanent fix would be to upgrade your installation to a more recent PmWiki version, your recipes, and in your own recipes or modules replace all calls to `htmlspecialchars()` with `PHSC()`, a PmWiki helper function for such cases.

The "blank" pages come from the fact that in PHP 5.4 the default encoding switched from an 8-bit encoding to variable-bit validated UTF-8, and that an incorrect UTF-8 string will be rejected. If your wiki uses an 8-bit encoding, it is virtually certain that it is not valid UTF-8. Worse, even if you do use UTF-8 some browsers may submit invalid bits. So the PHSC() function always pretends that it converts an 8-bit encoding where all bits are allowed.

Why am I seeing strange errors after upgrading?

Make sure all of the files were updated, in particular *pmwiki.php*.

This question sometimes arises when an administrator hasn't followed the advice, which used to be less prominent, on the installation and initial setup tasks pages and has renamed *pmwiki.php* instead of creating an *index.php* wrapper script. If you have renamed *pmwiki.php* to *index.php*, then the upgrade procedure won't have updated your *index.php* file. Delete the old version and create a wrapper script so it won't happen again.

Sometimes an FTP or other copy program will fail to transfer all of the files properly. One way to check for this is by comparing file sizes.

Be sure all of the files in the *wikilib.d/* directory were also upgraded. Sometimes it's a good idea to simply delete the *wikilib.d/* directory before upgrading. (Local copies of pages are stored in *wiki.d/* and not *wikilib.d/*.)

Make sure that the file permissions are correct. The official files have a restricted set of permissions that might not match your site's needs.

If you use a custom pattern for `$GroupPattern` make sure that it includes Site (`$SiteGroup`) and since PMWiki 2.2 also SiteAdmin (`$SiteAdminGroup`). Otherwise migration may fail (e.g. missing SiteAdmin for PMWiki 2.2 and later) and/or login does not work.
Additionally Main (`$DefaultGroup`) should be included too.

I'm suddenly getting messages like "`Warning: fopen(wiki.d/.flock): failed to open stream: Permission denied...`" and "`Cannot acquire lockfile`"... what's wrong?

Something (or someone) has changed the permissions on the *wiki.d/.flock* file or the *wiki.d/* directory such that the webserver is no longer able to write the lockfile. The normal solution is to simply delete the *.flock* file from the *wiki.d/* directory -- PmWiki will then create a new one. Also be sure to check the permissions on the *wiki.d/* directory itself. (One can easily check and modify permissions of the *wiki.d/* directory in FileZilla (open-source FTP app) by right-clicking on the file > File attributes)

My links in the sidebar seem to be pointing to non-existent pages, even though I know I created the pages. Where are the pages?

Links in the sidebar normally need to be qualified by a WikiGroup in order to work properly (use [[Group.Page]] instead of [[Page]]).
Also: Make sure you type Side**B**ar with a capital B.

Why am I seeing "`PHP Warning: Cannot modify header information - headers already sent by ...`" messages at the top of my page.

If this is the first or only error message you're seeing, it's usually an indication that there are blank lines, spaces, or other characters before the `<?php` or after the `?>` in a local customization files such as `config.php`. Double-check the file and make sure there is nothing before the initial `<?php`. It's often easiest and safest to eliminate any closing `?>` altogether. On Windows, it may be, but shouldn't be, necessary to use a hex editor to convert LFCR line endings to LF line endings in the `local\config.php` file.

When you save the file, the encoding/charset should be either cp1252/Windows1252 or UTF-8 without Byte Order Mark. NotePad++ is an editor that can do this.

When you transfer the files, tell your FTP manager to use text mode transfer, or, if that doesn't help, binary mode transfer.

If the warning is appearing after some other warning or error message, then resolve the other error and this warning may go away.

How do I make a PHP Warning about `function.session-write-close` go away?

If you are seeing an error similar to this

```
Warning: session_write_close() [function.session-write-close]:
open(/some/filesystem/path/to/a/directory/sess_[...]) failed: No such file
or directory (2) in /your/filesystem/path/to/pmwiki.php on line NNN
```

PmWiki sometimes does session-tracking using PHP's session-handling functions. For session-tracking to work, some information needs to be written in a directory on the server. That directory needs to exist and be writable by the webserver software. For this example, the webserver software is configured to write sessions in this directory

```
/some/filesystem/path/to/a/directory/
```

but the directory doesn't exist. The solution is to do at least one of these:

- **Create the directory** and make sure it's writable by the webserver software
- Provide a session_save_path value that points to a directory that is writable by the server, e.g. in config.php:

`session_save_path('/home/someuser/tmp/sessions');` # unix-type OS
`session_save_path('C:/server/tmp/sessions');` # Windows

Why is PmWiki prompting me multiple times for a password I've already entered?

This could happen like out of nowhere if your hosting provider upgrades to PHP version 5.3, and you run an older PmWiki release. Recent PmWiki releases fix this problem.

Alternatively, this may be an indication that the browser isn't accepting cookies, or that PHP's session handling functions on the server aren't properly configured. If the browser is accepting cookies, then try setting `$EnableDiag`=1; in *local/config.php*, run PmWiki using `?action=phpinfo`, and verify that sessions are enabled and that the session.save_path has a reasonable value. Note that several versions of PHP under Windows require that a session_save_path be explicitly set (this can be done in the *local/config.php* file). You might also try setting session.auto_start to 1 in your php.ini.

See also the question I have to log in twicebelow.

I edited *config.php*, but when I look at my wiki pages, all I see is "`Parse error: parse error, unexpected T_VARIABLE in` *`somefile`* `on line` *`number`*."

You've made a mistake in writing the PHP that goes into the *config.php* file. The most common mistake that causes the T_VARIABLE error is forgetting the semi-colon (;) at the end of a line that you added. The line number and file named are where you should look for the mistake.

Searches and pagelists stopped working after I upgraded -- no errors are reported, but links to other pages do not appear (or do not appear as they should) -- what gives?

Be sure all of the files in the *wikilib.d/* directory were also upgraded. In particular, it sounds as if the Site.PageListTemplates page is either missing (if no links are displayed) or is an old version (if the links do not appear as they should). Also make sure that read-permissions (attr) are set for the pages Site.PageListTemplates and Site.Search.

Some of my posts are coming back with "403 Forbidden" or "406 Not Acceptable" errors, or "Internal Server Error". This happens with some posts but not others.

Your webserver probably has mod_security enabled. The mod_security "feature" scans all incoming posts for forbidden words or phrases that might indicate someone is trying to hack the system, and if any of them are present then Apache returns the 403 Forbidden or 406 Not Acceptable error. Common phrases that tend to trigger mod_security include "curl ", "wget", "file(", and "system(", although there are many others (depending on the configuration, percent signs, html tags, international characters).

Since mod_security intercepts the requests and sends the "forbidden" message before PmWiki ever gets a chance to run, it's not a bug in PmWiki, and there's little that PmWiki can do about it. Instead, one has to alter the webserver configuration to disable mod_security or reconfigure it to allow whatever word it is forbidding. Some sites may be able to disable mod_security by placing `SecFilterEngine off` in a *.htaccess* file.

I get the following message when attempting to upload an image, what do I do?

```
Warning: move_uploaded_file(): SAFE MODE Restriction in effect. The script whose uid is 1929 is not
allowed to access /home/onscolre/public_html/pmwikiuploads/Photos owned by uid 33 in
/home/onscolre/public_html/pmwiki/scripts/upload.php on line 198

PmWiki can't process your request

?cannot move uploaded file to
/home/onscolre/public_html/pmwikiuploads/Photos/FoundationPupilsIn1958.jpeg

We are sorry for any inconvenience.
```

Your server is configured with PHP Safe Mode enabled. Configure your wiki to use a site-wide uploads prefix, then create the *uploads/* directory manually and set 777 permissions on it (rather than letting PmWiki create the directory).

I'm starting to see "Division by zero error in pmwiki.php..." on my site. What's wrong?

It's a bug in PmWiki that occurs only with the tables markup and only for versions of PHP >= 4.4.6 or >= 5.2.0. Often it seems to occur "out of nowhere" because the server administrator has upgraded PHP. Try upgrading to a later version of PmWiki to remove the error, or try setting the following in *local/config.php*:

```
    $TableRowIndexMax = 1;
```

I have to log in twice (two times) (2 times). -or- My password is not being required even though it should. -or- I changed the password but the old password is still active. -or- My config.php password is not over-riding my farmconfig.php password.

It could happen if (farm)config.php, or an included recipe, directly calls the functions CondAuth(), or RetrieveAuthPage(), PageTextVar(), PageVar() and possibly others, before defining all passwords and before including AuthUser (if required).

The order of config.php is very significant.

When editing an existing page, The "Save" causes a no-response of your server (not a blank page, no response at all, an endless connexion try). To get back the hand, it is necessary to request for another page (by clicking on its link in the menu for instance). And horror!, the ...?action=edit is then inhibited, it becomes impossible to edit any page.

When the editing of a page is initiated a file names `.flock` is created in the `wiki.d` repository. As long as this file exists it is impossible to edit any page. This file denotes an edition in progress and is automatically destroyed when leaving successfully an edit action by "Save". In case of a crash of the editing, this file is not destroyed. The remedy is, with an FTP client parameterized to show hidden files, to remove the `.flock` file. And all get back OK. This behavior is typically caused by a bug which provokes (directly or indirectly), an endless loop in a recipe concerned by the edited page.

I get the error "Data Mismatch - Locking FAILED!"

This is probably not a PmWiki error. PmWiki cannot create a lock file due to an underlying file system problem. For example the disk quota has been exceeded (e.g. by an error log file or file uploads), or there are problems with file system permissions.

Last modified by simon on February 11, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Troubleshooting

UTF-8

toc top

Summary: Enabling UTF-8 Unicode language encoding in your wiki.

UTF-8 supports all languages and alphabets, including Asian languages and their character depth. It is a widely supported and flexible character encoding.

It's fairly simple to enable UTF-8 on your wiki pages. Current PmWiki versions have the UTF-8 file which is enabled by default in the sample-config.php.

Enabling UTF-8 on a new wiki

If you start a new wiki in any language with the latest PmWiki version, it is highly recommended to enable UTF-8. In the future, PmWiki will change to use the UTF-8 encoding by default, so if you already use it, you will not need a complex "migration" to UTF-8 later.

To enable UTF-8 for a new wiki, add this line near the beginning of config.php (the docs/sample-config.php file has this line already):

```
include_once("scripts/xlpage-utf-8.php");
```

This line should come *before* a call to the XLPage() function in international wikis.

Save your config.php file encoded as UTF-8 (NO BOM). That allows entry of UTF-8 encoded characters in it. Make sure your editor does support this, and test by adding some non-ANSI UTF-8 characters, to see them in the text editor [1].

With UTF-8 thus enabled you also got use of classes *rtl* and *ltr*, which offer setting of the text direction to right-to-left, or left-to-right. This is useful for inclusion of right-to-left scripts like Arabic, Farsi (Persian), Hebrew, Urdu and others.

Enabling UTF-8 on existing wikis

Currently, this is possible *only if your group and page* names*, as well as upload names, don't contain international characters* The names of wiki pages are used as file names, and we don't have yet an easy way to rename the disk files.

If your wiki doesn't have international page/file names, first upgrade to the latest PmWiki version. To enable UTF-8, add these lines near the beginning of config.php:

```
include_once("scripts/xlpage-utf-8.php");
$DefaultPageCharset = array(''=>'ISO-8859-1'); # see below
```

These lines should come *before* a call to the XLPage() function in international wikis.

The `$DefaultPageCharset` line is there to fix and correctly handle some pages with missing or wrong attributes, created by older PmWiki versions.

- Most wikis in European languages are likely to be in the ISO-8859-1 encoding and should use:
  `$DefaultPageCharset = array(''=>'ISO-8859-1');`
- Wikis in Czech and Hungarian language are likely to be in the ISO-8859-2 encoding, they should use this line instead:
  `$DefaultPageCharset = array(''=>'ISO-8859-2', 'ISO-8859-1'=>'ISO-8859-2');`
- Wikis in Turkish language are likely to be in the ISO-8859-9 encoding, they should use this line instead:
  `$DefaultPageCharset = array(''=>'ISO-8859-9', 'ISO-8859-1'=>'ISO-8859-9');`

You should also delete the file `wiki.d/.pageindex`. This file contains a cache of links and words from your pages and is used for searches and pagelists. PmWiki will rebuild it automatically with the new encoding.

Support for RTL right-to-left languages

Languages like Arabic, Hebrew, Farsi (Persian), Urdu and others are written in script flowing from right to left. Classes *rtl* and *ltr* can be used to specify direction of text independently of the general text direction within a page, for example:

```
>>rtl<<
يتدفق هذا النص من اليمين إلى اليسار
>>ltr<<
This text flows left to right.
>><<
```

يتدفق هذا النص من اليمين إلى اليسار
This text flows left to right.

To set text direction for a wiki generally to RTL, you could add to config.php a line like:

```
  $HTMLStylesFmt['rtl'] = " body { direction:rtl; }"
```

but the skin you use may need other modifications, for instance to swap the search box and the page actions to the other side etc.

Some skins have full support for RTL, see for instance Amber.

Notes

- You need to save your config.php file in the UTF-8 encoding, and "Without Byte Order Mark (BOM)". See Character encoding of config.php.

- This page concerns the most recent versions of PmWiki. See Cookbook:UTF-8 for tips on older versions.

- In the case your pmwiki installation displays wrong encoding, or save an UTF-8 page to an other encoding without explanation, you can double check your custom .htaccess settings at the root of your served pages.

Upgrades

PmWiki is designed to make it easy to upgrade the PmWiki software without affecting your existing data files or installation. For most upgrades, you simply copy the files in the new release over your existing installation.

Note for PmWiki 1.0 sites: Upgrading from 1.0.x to 2.0 requires more than simply copying the 2.0 software over the 1.0 installation. See Upgrading From PmWiki 1 for more details.

Contents

Generic instructions

1. Read the release notes

Please read carefully the ReleaseNotes before performing an upgrade, about the changes between your previous version and the new one. See if there are any significant changes or preparation tasks that must be handled before performing the upgrade.

2. Backup

It's *always* a good idea to have a backup copy of your existing PmWiki installation before starting. You can copy the entire directory containing your existing installation, or you can just make copies of the *wiki.d/* directory and any other local customization files you may have created (e.g., *config.php*, *localmap.txt*, etc.).

3. Download and extract

Download the version of PmWiki that you want from the download page.

Extract the tar image using `tar -xvzf tgzfile`, where *tgzfile* is the tar file you downloaded above. This will create a `pmwiki-x.y.z` directory with the new version of the software.

4. Copy

Copy the files in `pmwiki-x.y.z` over the files of your existing PmWiki installation. For example, if your existing PmWiki installation is in a directory called *pmwiki*, then one way to copy the new files over the existing ones is to enter the command:

```
cp -a pmwiki-x.y.z/. pmwiki
```

Note that BSD systems will not have the -a option as a command-line argument for *cp*, but that's okay, since it's just shorthand for *cp -dpR*, so use that instead of *-a*.

Some environments have an alias established for *cp* that enable interactive prompts before overwriting a file. To work around this specify the absolute path to *cp*, such as */bin/cp*.

On (some) FreeBSD servers and Mac OS X systems you need to use

```
cp -Rpv pmwiki-x.y.z/. pmwiki
```

5. Update customisations and recipes

That's it! Your base PmWiki installation is complete.

Now use the PmWiki:Site Analyzer to determine which recipes could be updated to the most recent version.

Unless you have made customizations to the *pmwiki.php* script or to the files in *scripts/*, your PmWiki installation should continue to run correctly! (Changes to these files are <u>not</u> recommended).

(Local customizations should go in *local/config.php*, *pub/css*, and *pub/skins/*yourskinname)

Note: Additional tips can be found on the PmWiki:Troubleshooting page.

Upgrading from version 2.1.27 to 2.2.0

Between the stable versions 2.1.27 and 2.2.0 there are a number of additions. Some of them may need changes to local config files or to wiki pages, and they are outlined here. For the full list of changes see the release notes.

If you are upgrading from a 2.2.beta version, your wiki may already include these features.

- Some pages that were formerly in the Site.* group are now in a separate read-protected SiteAdmin.* group: Site.AuthUser, Site.AuthList, Site.NotifyList, Site.Blocklist, and Site.ApprovedUrls. If upgrading from an earlier version, PmWiki will prompt to automatically copy these pages to their new location if needed. If a site wishes to continue using the old Site.* group for these pages, simply set to config.php `$SiteAdminGroup = $SiteGroup;`

- To authorize reading or editing in protected areas, the former password `"nopass"` should now be written as `"@nopass"`.

- WikiWords are now disabled by default. To re-enable them, set either `$LinkWikiWords` or `$EnableWikiWords` to 1.

- The `$ROSPatterns` variable has changed -- replacement strings are no longer passed through FmtPageName() i.e., it must now be done explicitly.

- Page links inside included pages, sidebars, headers or footers are now treated as relative to the page where they are written, instead of the page where they appear. For example, in Site.SideBar, always set the group in a wikilink like `[[Main/HomePage]]` or with a page variable `[[{*$Group}/HomePage]]`, because a link `[[HomePage]]` will point to a page Site.HomePage.

- PageLists
  - Spaces no longer separate wildcard patterns -- use commas.
  - `{$PageCount}, {$GroupCount}, {$GroupPageCount}` variables used in pagelist templates are now `{$$PageCount}, {$$GroupCount}, {$$GroupPageCount}`.
  - The directive no longer accepts parameters from urls by default. In order to have it accept such parameters (which was the default in 2.1 and earlier), add a `request=1` option to the `(:pagelist:)` directive.

- Skin templates are now required to have <!--HTMLHeader--> and <!--HTMLFooter--> directives.

- Authentication using Active Directory is now simplified, see PmWiki.AuthUser.

Upgrading from version 2.2.0 to the latest version

Note: this page may have a more recent version, see PmWiki:Upgrades.

Some additions since version 2.2.0 may need changes to local config files or to wiki pages, and they are outlined here. For the full list of changes see release notes and change log.

- Version 2.2.10: `$EnableRelativePageVars` was changed to enabled by default, and it affects PageVariables from included pages, sidebars, headers and footers.
  - The form `{*$var}` refers to "the currently browsed page" while `{$var}` without an asterisk refers to "the physical page where the PageVar is written".
  - Pages that are designed to work on "the currently browsed page" should switch to using `{*$FullName}` instead of `{$FullName}`. Administrators should especially check any customized versions of Site.PageActions, Site.EditForm, Site.PageNotFound, SideBar pages, `$GroupHeaderFmt`, `$GroupFooterFmt`, Page lists in sidebars, headers, and footers. See Special references.
  - If your wiki heavily relies on the previous behavior, you can revert to it, see `$EnableRelativePageVars`.

- Version 2.2.35: Important change for international wikis: the XLPage() function no longer loads encoding scripts such as xlpage-utf-8.php. When you upgrade, you need to include those scripts from config.php, before the call to XLPage():
```
include_once("scripts/xlpage-utf-8.php"); # if your wiki uses UTF-8
XLPage('bg','PmWikiBg.XLPage');
```

FAQ

How can I determine what version of PmWiki I'm running now?

> See version - Determining and displaying the current version of PmWiki (pmwiki-2.2.99).

How can I test a new version of PmWiki on my wiki without changing the prior version used by visitors?

> The easy way to do this is to install the new version in a separate directory, and for the new version set (in local/config.php):

```
$WikiLibDirs = array(&$WikiDir,
  new PageStore('/path/to/existing/wiki.d/{$FullName}'),
  new PageStore('wikilib.d/{$FullName}'));
```

> This lets you test the new version using existing page content without impacting the existing site or risking modification of the pages. (Of course, any recipes or local customizations have to be installed in the new version as well.)

Then, once you're comfortable that the new version seems to work as well as the old, it's safe to upgrade the old version (and one knows of any configuration or page changes that need to be made).

UpgradingFromPmWiki1

This page gives suggestions for upgrading an existing PmWiki 1.x installation to use PmWiki 2.0. In this page we assume that a site administrator already has a site running using PmWiki version 1.x or earlier in a somewhat standard configuration, and wants to upgrade to the 2.0 software.

> **Important note:** The normal PmWiki upgrade procedure (i.e., copy the new software over the existing one) won't work for moving from 1.x to 2.0. Either start over with a new installation, or use some of the conversion scenarios listed below.

As always, questions and requests for assistance can be posed to pmwiki-users. Errors or problems with the methods below can be corrected here, or posted to the PmWiki Issue Tracking System.

Conversion

Because of the substantial redesign of PmWiki for 2.0, converting an existing site to 2.0 is likely to cause a wiki administrator a fair amount of apprehension. The approach given here allows the administrator to install, configure, and test PmWiki 2.0 on an existing set of pages without risking an existing 1.x installation.

> It shall be noted that the compatibility script being used by this method was removed in PmWiki 2.2.0beta43. You need to install PmWiki 2.2.0beta42 to carry out the migration procedure, and **then** upgrade to the latest pmwiki version.

Step 0: It's always a very good idea to back up your existing PmWiki 1.x installation before doing anything else -- especially save the files in the *local/* and *wiki.d/* directories.

Step 1: Install PmWiki 2.0 into a new directory away from the existing 1.x installation.

Step 2: Briefly test the PmWiki 2.0 installation and make sure it is working properly -- i.e., edit and save a couple of pages. Then, remove the pages you created (you can just remove the files from PmWiki 2.0's *wiki.d/* directory, or remove the *wiki.d/* directory altogether).

Step 3: Add the following lines to the *local/config.php* file in the 2.0 installation, replacing "`/path/to/pmwiki1/wiki.d`" below with the location of your PmWiki 1.x installation's *wiki.d/* directory on disk.

```
include_once("$FarmD/scripts/compat1x.php");
UseV1WikiD("/path/to/pmwiki1/wiki.d");
```

For example, my 2.0 test conversion uses:

```
include_once("$FarmD/scripts/compat1x.php");
UseV1WikiD("/home/pmichaud/pmwiki/wiki.d");
```

Step 4: After making the above change, all of your existing pages should appear in the new 2.0 installation. Furthermore, if you "edit page" on any of the existing pages, you should see that any PmWiki 1.x markups (links, etc.) have been converted to the new markup syntax.

Any pages edited/saved by the 2.0 wiki installation are kept separate from the pages in the previous installation. Thus you can safely experiment with editing and changing pages in the new site without affecting the existing 1.x site.

Step 5: Once you see that your existing pages are available in the 2.0 installation, you can then begin going through the remaining initial setup tasks for the 2.0 site to enable any local customizations you may want for your site. Many local customizations (e.g. page layout templates) remain the same between 1.x and 2.0, others such as custom markup or cookbook recipes need to be converted to 2.0 as well.

Note: WikiWord links are disabled by default since Pmwiki version 2.1 beta2. So you may either enable WikiWord links by setting `$LinkWikiWords = 1;` in config.php, or convert your existing WikiWord links manually to bracketed links. To find those WikiWord links easier you can highlight them by setting in config.php

```
$HTMLStylesFmt['wikiword'] = "
span.wikiword { background:yellow; }
";
```

Step 6: Continue configuring the new installation just as if you were setting up a new PmWiki site. If you find PmWiki 1.x markups that aren't converted or convert incorrectly, be sure to enter a new PITS issue so that we can improve the conversion script.

Step 7: If you're comfortable with the conversion and want to go ahead and convert all of the 1.x pages into 2.0 format, change the `UseV1WikiD(...)` call in *local/config.php* above to `ConvertV1WikiD(...)` instead, as in:

```
    include_once("$FarmD/scripts/compat1x.php");
    ConvertV1WikiD("/path/to/pmwiki1/wiki.d");
```

Running the pmwiki.php script will then bring up some forms to allow you to bulk migrate some or all of your 1.x pages to 2.0 format. After you've converted pages, you can then just eliminate these two lines from the configuration and your PmWiki 2.0 site will be running standalone.

If you have local customisations that require you to specify $Compat1x['/match/'] = 'replace'; entries so they are correctly converted, make sure these are defined *before* the call to ConvertV1WikiD.

Note that there's nothing that requires you to convert all of the pages or get rid of the 1.x *wiki.d/* directory -- PmWiki works just fine with it in place. And it's good to have a backup.

Step 8: Once you're comfortable that the PmWiki 2.0 site will meet your needs, you can then discontinue the 1.x site and just start using the 2.0 site. Or, if you decide that 2.0 isn't for you, then the 1.x site is still intact and can continue to be used.

Step 9: If your previous site had an *uploads/* directory, you'll probably want to copy it or move it into the new location.

External link
- Fix Links in wiki pages from version 0.4.23 to 2.2.1 with UEDIT

Last modified by simon on August 02, 2012. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/UpgradingFromPmWiki1

UploadVariables

toc top

See also: Uploads, Uploads admin.

`$EnableUpload`
: The upload.php script is automatically included from stdconfig.php if the `$EnableUpload` variable is true in config.php. Note that one may still need to set an upload password before users can upload (see UploadsAdmin).

`$UploadBlacklist`
: This array contains forbidden strings for an uploaded file (case insensitive). Some installations with the Apache server will try to execute a file which name contains ".php", ".pl" or ".cgi" even if it is not the last part of the filename. For example, a file named "test.php.txt" may be executed. To disallow such files to be uploaded, add to config.php such a line:

```
$UploadBlacklist = array('.php', '.pl', '.cgi'); # disallow common script files
```

`$UploadPermAdd`
: This variable sets additional unix permissions applied to newly uploaded files, and should be 0 (recommended as of 2013). If uploaded files cannot be downloaded and displayed on the website, for example with the error 403 Forbidden, set this value to 0444 (core setting, default since 2004).

```
$UploadPermAdd = 0; # recommended
```

`$UploadPermSet`
: This variable sets unix permissions unconditionally applied to newly uploaded files, for example `0604`. **Danger!** Do not use this variable unless you know what you're doing! If you make a mistake, uploaded files may be impossible to edit or delete via the FTP/SSH account (in that case, Cookbook:Attachtable may be used) or to be downloaded and displayed on the website. Note that file permissions may differ on different systems - if you copy or move your PmWiki installation, you may have to change it.

`$UploadDir`
: The directory where uploads are to be stored. Defaults to *uploads/* in the pmwiki directory, but can be set to any location on the server. This directory must be writable by the webserver process if uploading is to occur.

`$UploadUrlFmt`
: The url of the directory given by `$UploadDir`. By default, `$UploadUrlFmt` is derived from `$PubDirUrl` and `$UploadDir`.

`$IMapLinkFmt`['Attach:']
: The format of the upload link displayed when an attachment exists. Can be changed with such a line in config.php:

```
$IMapLinkFmt['Attach:'] = "<a class='attachlink' href='\$LinkUrl'>\$LinkText</a>";
```

`$LinkUploadCreateFmt`
: The format of the upload link displayed when an attachment not present. Can be changed with such a line in config.php:

```
$LinkUploadCreateFmt = "<a class='createlinktext' href='\$LinkUpload'>\$LinkText</a>
<a class='createlink' href='\$LinkUpload'>&nbsp;&Delta;</a>";
```

`$UploadPrefixFmt`
: Sets the prefix for uploaded files to allow attachments to be organized other than by groups. Defaults to `'/$Group'` (uploads are organized per-group), but can be set to other values for sitewide or per-page attachments.

```
$UploadPrefixFmt = '/$Group/$Name';     # per-page attachments
$UploadPrefixFmt = '';                  # sitewide attachments
```

It is recommended to have the `$UploadPrefixFmt` variable defined in config.php, the same for all pages in the wiki, and not in group/page local configuration files. Otherwise you *will* be unable to link to attachments in other wikigroups.

`$EnableDirectDownload`
: When set to 1 (the default), links to attachments bypass PmWiki and come directly from the webserver. Setting `$EnableDirectDownload`=0; causes requests for attachments to be obtained via `?action=download`. This allows PmWiki to protect attachments using a page's read permissions, but also increases the load on the server. Don't forget to protect your directory /uploads/ with a .htaccess file (Order Deny,Allow / Deny from all).

`$EnableUploadGroupAuth`
: Set `$EnableUploadGroupAuth = 1;` to authenticate downloads with the group password. This could be used together with `$EnableDirectDownload = 0;`. Note: `$EnableUploadGroupAuth` should not be enabled if your wiki uses per-page attachments.

`$EnableUploadVersions`
: When set to 1 (default is 0), uploading a file to a location where a file of the same name already exists causes the old version to be renamed to `file.ext,timestamp` (instead of being overwritten). `timestamp` is a Unix-style timestamp.

`$EnableUploadOverwrite`
: When set to 1 (the default), determines if overwriting previously uploaded files is allowed.

`$UploadNameChars`
: The set of characters allowed in upload names. Defaults to `"-\w. "`, which means alphanumerics, hyphens, underscores, dots, and spaces can be used in upload names, and everything else will be stripped.
`$UploadNameChars` = "-\\w. !"; # allow dash, letters, digits, dots, spaces and exclamations
`$UploadNameChars` = "-\\w. \\x80-\\xff"; # allow Unicode
Note: Not all characters can be used in file names, because of various limitations in protocols or operating systems, file systems and server software, or conflict with PmWiki markup:
- `+?:@#%!=/` have special meanings in URL addresses,
- `|\^`[]?:@#%/` may be impossible to save on some systems,
- `<>"|\^`(){}[]#%` may conflict with PmWiki markups,

so it is strongly recommended to only enable those if you know what you're doing.

`$MakeUploadNamePatterns`
: An array of regular expression replacements that is used to normalize the filename of an attached file. First, everything but `$UploadNameChars` will be stripped, then the file extension will be converted to lowercase. Administrators can override these replacements with a custom definition (the full array needs to be defined). Currently the default sequence is:

```
$MakeUploadNamePatterns = array(
"/[^$UploadNameChars]/" => '',          # strip all not-allowed characters
'/\\.[^.]*$/e' => 'strtolower("$0")',   # convert extension to lowercase
'/^[^[:alnum:]_]+/' => '',              # strip initial spaces, dashes, dots
'/[^[:alnum:]_]+$/' => ''));            # strip trailing spaces, dashes, dots
```

`$UploadDirQuota`
: Overall size limit for all uploads.

```
$UploadDirQuota = 100*1024;          # limit uploads to 100KiB
$UploadDirQuota = 1000*1024;         # limit uploads to 1000KiB
$UploadDirQuota = 1024*1024;         # limit uploads to 1MiB
$UploadDirQuota = 25*1024*1024;      # limit uploads to 25MiB
$UploadDirQuota = 2*1024*1024*1024;  # limit uploads to 2GiB
```

`$UploadPrefixQuota`
: Overall size limit for one directory containing uploads. This directory is usually`uploads/GroupName` (one for every WikiGroup), or `uploads/Group/PageName` (one for every page), depending on the variable `$UploadPrefixFmt`.

`$UploadMaxSize`
: Maximum size for uploading files, 50000 octets (bytes) by default.

`$UploadExtSize`
: Maximum size per extension, overriding the default in `$UploadMaxSize`.

```
$UploadExtSize['zip'] = 2*1024*1024; # allow up to 2MiB for zip files
```

Uploads

PmWiki can be configured to allow authors to upload and store files and images (known as attaching them). These attachments may then be referenced from any page.

> *Note: PmWiki is distributed with uploads disabled by default. See Uploads Admin for information about how to enable and configure the upload feature.*

Note2: Uploads can be configured site-wide, by-group, or by-page; see Uploads Admin for details. This determines whether all uploads go in one directory for the site, an individual directory for each group, or an individual directory for each page. The default is to organize uploads by group.

`Attach:` Syntax

To add or link to an attachment, an author edits a page to include the markup "`Attach:`" followed by a name of an attachment (e.g., "`Attach:resume.pdf`"). When the page is displayed, the `Attach:` markup becomes one of the following:

- A link to the named attachment (if uploaded, ie already in the upload directory)
- A link to a form whereby the author can specify a file to be uploaded and used as the new attachment (if not yet uploaded, ie not in the upload directory)
- If the attachment is an image file with an extension such as .gif, .jpeg, or .png, it is displayed as an image.

The behaviour of links can be modified to
- prevent an image attachment from displaying as an image, place it in double brackets (e.g.,`[[Attach:image.jpg]]`).
- have a link to an attachment appear without the "`Attach:`" at the beginning of the link, use `[[(Attach:)file.ext]]`.

Attachments on other pages and groups

To link to an uploaded attachment (image or file) from another group, you simply refer the group itself (make sure "Groupname" has the dot in it).
    `Attach:Groupname./file_name.ext` (note the dot after the groupname)
If PmWiki is configured with an individual directory per page use
    `Attach:Pagename/file_name.ext` (Pagename is in the same WikiGroup)
    `Attach:Groupname.Pagename/file_name.ext`

Names with spaces

To link to a filename with spaces in it use the bracket link notation, eg

```
[[Attach:a filename with spaces.txt]]
```

"Embedding in the page" an image with spaces is not supported: just upload the images with names without spaces, and use the markup `Attach:image.jpg`.

The following workaround is possible, but is unsupported and not recommended:

```
[[#blank | Attach:image space.jpeg]]
```

International characters in file names

See UploadsAdmin and `$UploadNameChars`.

Listing Uploaded Files On A Page

To list files that have been uploaded, use the markup: `(:attachlist:)`

This will list attachments to the current group or page, depending whether attachments are organised per group or per page; each instance includes a link to the attachment for viewing or downloading. A list of attachments is also shown as part of the uploads page form.

Upload Form / Upload Replacement

One can go directly to the upload form by appending "?action=upload" to the URI for any page that has file uploads enabled by the Wiki Administrator. Replace a file by simply uploading a new version of the file with the same name.
- Be sure to clear your browser cache after replacing an upload. Otherwise, it may appear that the original upload is still on the server.

If you put `$EnableUploadVersions=1;` in your `local/config.php`, the old versions of the same files are renamed and not removed.

Type and Size Restrictions

For security reasons, the upload feature is disabled when PmWiki is first installed. When enabled uploads are restricted as to the types and sizes of files that may be uploaded to the server (see Uploads Admin). PmWiki's default configuration limits file sizes to 50 kilobytes and file extensions to common types such as ".gif", ".jpeg", ".doc", ".txt", and ".pdf".

In addition, the administrator can configure the system to require an `upload` password--see Passwords and Passwords Admin.

By default the upload allows the following extensions. Note that by default, it is possible to upload files with no extensions.

```
  gif, jpg, jpeg, png, bmp, ico, wbmp, svg, svgz, xcf,       # images

  mp3, au, wav, ogg, flac,                                   # audio
  ogv, mp4, webm, mpg, mpeg, wmf, mov, qt, avi,              # video
  zip, 7z, gz, tgz, rpm, hqx, sit,                              # archives
  odt, ods, odp, odg, doc, docx, ppt, pptx, xls, mdb, rtf,      # Office
  exe,                                                          # executables
  pdf, psd, ps, ai, eps,                                        # Adobe
  htm, html, css, fla, swf,                                     # web stuff
  txt, tex, dvi,                                                # text files
  epub, kml, kmz, (files with no extension)                     # misc
```

Removal

At present uploaded files can only be deleted from the server by the wiki administrator. Any uploads-authorized user may overwrite an existing file by uploading another of the same name and extension to the same location.

The administrator may remove an uploaded file by accessing the server via ftp (or via a control panel, if the host offers such a feature). The recipe Cookbook:Attachtable allows the deletion of the files from the wiki.

When I upload a file, how do I make the link look like "file.doc" instead of " Attach:file.doc Δ"?

> Use parentheses, as in `[[(Attach:)file.doc]]`. There is also a configuration change that can eliminate the `Attach:` -- see Cookbook:AttachLinks.

Why can't I upload files of size more than 50kB to my newly installed PmWiki?

> Out of the box PmWiki limits the size of files to be uploaded to 50kB. Add
> `$UploadMaxSize = 1000000; # limit upload file size to 1 megabyte`
> to your *config.php* to increase limit to 1MB (for example). See UploadsAdmin for how to further customize limits. Note that both PHP and webservers also place their own limits on the size of uploaded files.

Why does my upload exit unexpectedly with "Incomplete file received"?

> You may be running out of space in a 'scratch' area, used either by PmWiki or by PHP. On *nix, check that you have sufficient free space in /tmp and /var/tmp.

How do I make it so that the upload link still allows one to make another upload (if someone wants to replace the old version of a file with a newer version, for example). Currently you only get the upload link when there is no file in the upload directory.

> Use the Attach page action, and click on the delta symbol (Δ) shown against each of files listed. If you can't see the attach action either uploads are not enabled, you are not authorized to upload, or the attach action has been commented out or is missing. See also available actions.

How do I hide the "Attach:" for all attachments

> See Cookbook:AttachLinks, note that this does not currently work for `[[Attach:my file.ext]]`.

How can I link a file that have a 4-letter file extension such like 'abc.pptx'?

> See Cookbook:Upload Types

How can I prevent others from using the url's of my images on their site

> See Cookbook:Prevent Hotlinking

How can I display a file that lacks a correct extension? (e.g. you are using Cookbook:LinkIcons)

> A file can be displayed by addition of a "false" extension to the URL. For example, if the url is `http://example.com/dox/mydoc`, add a fake query string on the end with the desired extension (e.g., `http://example.com/dox/mydoc?format=.docx`). If query strings are unsuitable, a fragment identifier should work, e.g. `http://example.com/dox/mydoc#.docx`.

Last modified by Petko on March 31, 2016.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Uploads

toc top

Uploads Administration

toc top

PmWiki includes a script called *upload.php* that allows users to upload files to the wiki server using a web browser. Uploaded files (also called *attachments*) can then be easily accessed using markup within wiki pages. This page describes how to install and configure the upload feature.

Some notes about security

PmWiki takes a somewhat, but justifiable, paranoid stance when it comes to the uploads feature. Thus, the default settings for uploads tend to try to restrict the feature as much as possible:

- The upload function is disabled by default
- Even if you enable it, the function is password locked by default
- Even if you remove the password, you're restricted to uploading files with certain names, extensions, and sizes

- The characters that may appear in upload filenames are (default) alphanumerics, hyphen, underscore, dot, and space (see also here).
- The maximum upload size is small (50K by default)

This way the potential damage is limited until/unless the wiki administrator explicitly relaxes the restrictions.

Keep in mind that letting users (anonymously!) upload files to your web server does entail some amount of risk. The *upload.php* script has been designed to reduce the hazards, but wiki administrators should be aware that the potential for vulnerabilities exist, and that misconfiguration of the upload utility could lead to unwanted consequences.

By default, authorized users are able to overwrite files that have already been uploaded, without the possibility of restoring the previous version of the file. If you want to disallow users from being able to overwrite files that have already been uploaded, add the following line to *config.php*:

```
$EnableUploadOverwrite = 0;
```

Alternatively, an administrator can keep older versions of uploads.

An administrator can also configure PmWiki so the password mechanism controls access to uploaded files.

Basic installation

The *upload.php* script is automatically included from *stdconfig.php* if the `$EnableUpload` variable is true in *config.php*. In addition, *config.php* can set the `$UploadDir` and `$UploadUrlFmt` variables to specify the local directory where uploaded files should be stored, and the URL that can be used to access that directory. By default, `$UploadDir` and `$UploadUrlFmt` assume that uploads will be stored in a directory called *uploads/* within the current directory (usually the one containing *pmwiki.php*). In addition, *config.php* should also set a default upload password (see PasswordsAdmin).

Thus, a basic *config.php* for uploads might look like:

```
<?php if (!defined('PmWiki')) exit();
##  Enable uploads and set a site-wide default upload password.
$EnableUpload = 1;
$UploadPermAdd = 0;
$DefaultPasswords['upload'] = pmcrypt('secret');
```

If you have edit passwords and wish to allow all users with edit rights to upload, instead of `$DefaultPasswords`['upload'], you can set `$HandleAuth['upload'] = 'edit';` in config.php.

Important: do NOT create the uploads directory yet! See the next paragraph.

You may also need to explicitly set which filesystem directory will hold uploads and provide a URL that corresponds to that directory like:

```
$UploadDir = "/home/foobar/public_html/uploads";
$UploadUrlFmt = "http://example.com/~foobar/uploads";
```

Note: In most installations, you don't need to define or change these variables, usually PmWiki can detect them (and if you do, uploads may simply not work).

Upload directory configuration

Uploads can be configured *site-wide*, *by-group* (default), or *by-page* by changing `$UploadPrefixFmt` in `config.php`. This determines whether all uploads go in one directory for the site, an individual directory for each group, or an individual directory for each page. The default is to organize upload by group.
*It is recommended that the `$UploadPrefixFmt` variable defined in config.php is the same for all pages in the wiki, and not different in group or page local configuration files. Otherwise you **will** be unable to link to attachments in other wikigroups.*

Single upload directory

For site-wide uploads, use

```
$UploadPrefixFmt = '';
```

Per page upload directories

To organize uploads by page, use:

```
$UploadPrefixFmt = '/$Group/$Name';
```

You may prefer uploads attached per-page rather than per-group or per-site if you plan to have many files attached to individual pages. This setting simplifies the management of picture galleries for example. (In a page, you can always link to attachments to other pages.)

The upload directory

For the upload feature to work properly, the directory given by $UploadDir must be writable by the web server process, and it usually must be in a location that is accessible to the web somewhere (e.g., in a subdirectory of *public_html*). Executing PmWiki with uploads enabled will prompt you with the set of steps required to create the uploads directory on your server (it differs from one server to the next). *Note that you are likely to be required to explicitly create writable group- or page-specific subdirectories as well!*

Uploading a file

Once the upload feature is enabled, users can access the upload form by adding "`?action=upload`" to the end of a normal PmWiki URL. The user will be prompted for an upload password similar to the way other pages ask for passwords (see Passwords and PasswordsAdmin for information about setting passwords on pages, groups, and the entire site).

Another way to access the upload form is to insert the markup "`Attach:filename.ext`" into an existing page, where `filename.ext` is the name of a new file to be uploaded. When the page is displayed, a '?-link' will be added to the end of the markup to take the author to the upload page. (See Uploads for syntax variations.)

By default, PmWiki will organize the uploaded files into separate subdirectories for each group. This can be changed by modifying the `$UploadPrefixFmt` variable. See Cookbook:UploadGroups for details.

Versioning Uploaded Files

PmWiki does not manage versioning of uploaded files by default. However, by setting `$EnableUploadVersions`=1; an administrator can have older versions of uploads preserved in the uploads directory along with the most recent version.

Upload restrictions

Restricting uploaded files for groups and pages

Uploads can be enabled only for specific groups or pages by using a group customization. Simply set `$EnableUpload=1;` for those groups or pages where uploading is to be enabled; alternately, set `$EnableUpload=1;` in the config.php file and then set `$EnableUpload=0;` in the per-group or per-page customization files where uploads are to be disabled.

Restricting total upload size for a group or the whole wiki

Uploads can be restricted to an overall size limit for groups. In the group configuration file (i.e., local/Group.php), add the line

> `$UploadPrefixQuota` = 1000000; # limit group uploads to 1000KB (1MB)

This will limit the total size of uploads for that group to 1000KB --any upload that pushes the total over the limit will be rejected with an error message. This value defaults to zero (unlimited).

Uploads can also be restricted to an overall size limit for all uploads. Add the line

> `$UploadDirQuota` = 10000000; # limit total uploads to 10000KB (10MB)

This will limit the total size of uploads for the whole wiki to 10000KB --any upload that pushes the total over the limit will be rejected with an error message. This value defaults to zero (unlimited).

Restricting uploaded files type and size

The upload script performs a number of verifications on an uploaded file before storing it in the upload directory. The basic verifications are described below.

filenames
: the name for the uploaded file can contain only letters, digits, underscores, hyphens, spaces, and periods, and the name must begin and end with a letter or digit.

file extension
: only files with approved extensions such as "`.gif`", "`.jpeg`", "`.doc`", etc. are allowed to be uploaded to the web server. This is vitally important for server security, since the web server might attempt to execute or specially process files with extensions like "`.php`", "`.cgi`", etc.

file size
: By default all uploads are limited to 50K bytes, as specified by the `$UploadMaxSize` variable. Thus, to limit all uploads to 100KB, simply specify a new value for `$UploadMaxSize` in *config.php*:

```
$UploadMaxSize = 100000;
```

However, the default maximum file size can also be specified for each type of file uploaded. Thus, an administrator can restrict "`.gif`" and "`.jpeg`" files to 20K, "`.doc`" files to 200K, and all others to the size given by `$UploadMaxSize`. The `$UploadExtSize` array is used to determine which file extensions are valid and the maximum upload size (in bytes) for each file type. For example:

```
    $UploadExtSize['gif'] = 20000; # limit .gif files to 20KB
```

Disabling file upload by file type

Setting an entry to zero disables file uploads of that type altogether:

```
    $UploadExtSize['zip'] = 0;  # disallow .zip files
    $UploadExtSize[''] = 0;     # disallow files with no extension
```

You can limit which types of files are uploadable by disabling all defaults and specifying only desired types. Setting the variable `$UploadMaxSize` to zero will disable all default file types. Individual file types may then be enabled by setting their maximum size with the variable `$UploadExtSize`.

```
    # turns off all upload extensions
    $UploadMaxSize = 0;

    # enable only these file types for uploading
    $aSize=100000; // 100 KB file size limitation
    $UploadExtSize['jpg' ] = $aSize;
    $UploadExtSize['gif' ] = $aSize;
    $UploadExtSize['png' ] = $aSize;
```

Note: Files with multiple extensions

Some installations with the Apache server will try to execute a file which name contains ".php", ".pl" or ".cgi" even if it isn't the last part of the filename. For example, a file named "test.php.txt" may be executed. To disallow such files to be uploaded, add to config.php such a line:

```
  $UploadBlacklist = array('.php', '.pl', '.cgi');
```

Adding new file types to permitted uploads

To add a new extension to the list of allowed upload types, add a line like the following to alocal customization file:

```
    $UploadExts['ext'] = 'content-type';
```

where *ext* is the extension to be added, and*content-type* is the " MIME type", or content-type (which you may find here or on the lower part of this page) to be used for files with that extension. For example, to add the '`dxf`' extension with a Content-Type of '`image/x-dxf`', place the line

```
    $UploadExts['dxf'] = 'image/x-dxf';
```

Each entry in $UploadExts needs to be the extension and the mime-type associated with that extension, thus:

```
    $UploadExts = array(
      'gif' => 'image/gif',
      'jpeg' => 'image/jpeg',
      'jpg' => 'image/jpeg',
      'png' => 'image/png',
      'xxx' => 'yyyy/zzz'
    );
```

For the types that PmWiki already knows about it's not necessary to repeat them here (the*upload.php* script adds PmWiki's defaults to whatever the administrator supplies). See also Cookbook:UploadTypes for additional types.

Other file size limits

There are other factors involved that affect upload file sizes. In Apache 2.0, there is a ` LimitRequestBody directive that controls the maximum size of anything that is posted (including file uploads). Apache has this defaulted to unlimited size. However, some Linux distributions (e.g., Red Hat Linux) limit postings to 512K so this may need to be changed or increased. (Normally these settings are in an *httpd.conf* configuration file or in a file in*/etc/httpd/conf.d*.)

Problem noted on Red Hat 8.0/9.0 with Apache 2.0.x, the error "Requested content-length of 670955 is larger than the configured limit of 524288" was occurring under Apache and a "Page not found" would appear in the browser. Trying the above settings made no change with PHP, but on Red Hat 8.0/9.0 there is an additional PHP config file, /etc/httpd/conf.d/php.conf, and increasing the number on the line "LimitRequestBody 524288" solves the issue.

PHP itself has two limits on file uploads (usually located in `/etc/php.ini`). The first is the `upload_max_filesize` parameter, which is set to 2MB by default. The second is `post_max_size`, which is set to 6MB by default.

With the variables in place--PmWiki's maximum file size, Apache's request-size limits, and the PHP file size parameters, the maximum uploaded file size will be the smallest of the three variables.

Password protecting uploaded files

Setting a read password for pages (and groups) will prevent an attached file from being seen or accessed through the page, but to prevent direct access to the file location (the uploads/ directory) one can do the following:

- In local/config.php set `$EnableDirectDownload=0;`
- If you use per-group upload directories (PmWiki default, see `$UploadPrefixFmt`), add to config.php `$EnableUploadGroupAuth = 1;`
- Deny public access to the uploads/ directory through moving it out of the html/ or public_html/ directory tree, or through a .htaccess file.

See Cookbook:Secure attachments

Other notes

- If uploads doesn't seem to work, make sure that your PHP installation allows uploads. The *php.ini* file (usually */etc/php.ini* or */usr/local/lib/php.ini*) should have

```
file_uploads = On
```

- Another source of error in the *php.ini* file is a not defined *upload_tmp_dir*. Just set this variable to your temp directory, e.g.

```
upload_tmp_dir = /tmp
```

Note that if you change this values, httpd must generally be restarted. Another way to check if uploads are allowed by the server is to set `$EnableDiag` to 1 in *config.php*, and set ?action=phpinfo on a URL. The "`file_uploads`" variable must have a value of 1 (if it says "`no value`", that means it's off).

How do I disable uploading of a certain type of file?

Here's an example of what to add to your *local/config.php* file to disable uploading of .zip files, or of files with no extension:

```
$UploadExtSize['zip'] = 0;  # Disallow uploading .zip files
$UploadExtSize[''] = 0;     # Disallow files with no extension
```

How do I attach uploads to individual pages or the entire site, instead of organizing them by wiki group?

Use the `$UploadPrefixFmt` variable (see also the Cookbook:UploadGroups recipe).

```
$UploadPrefixFmt = '/$FullName'; # per-page, in Group.Name directories
$UploadPrefixFmt = '/$Group/$Name'; # per-page, in Group directories with Name subdirectories
$UploadPrefixFmt = ''; # site-wide
```

For `$UploadDirQuota` - can you provide some units and numbers? Is the specification in bytes or bits? What is the number for 100K? 1 Meg? 1 Gig? 1 Terabyte?

Units are in bytes.

```
$UploadDirQuota = 100*1024;         # limit uploads to 100KiB
$UploadDirQuota = 1000*1024;        # limit uploads to 1000KiB
$UploadDirQuota = 1024*1024;        # limit uploads to 1MiB
$UploadDirQuota = 25*1024*1024;     # limit uploads to 25MiB
$UploadDirQuota = 2*1024*1024*1024; # limit uploads to 2GiB
```

Is there a way to allow file names with Unicode or additional characters?

Yes, see `$UploadNameChars`

Where is the list of attachments stored?

It is generated on the fly by the
markup.

Last modified by simon on June 06, 2015. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/UploadsAdmin

UrlApprovals

toc top

This page explains how to discourage "link spamming" on your wiki site using PmWiki's *urlapprove.php* script. This script is already included in PmWiki files, but not activated by default.

Using *urlapprove.php*

Occasionally spammers may try to add large number of (sometimes hidden) URLs to pages because they think it will improve their search engine rankings -- which it probably won't. The *urlapprove.php* script works against these spammers' purpose by

- requiring approval of links to Internet sites before a link to them are created in the wiki, and
- allowing you to limit the number of unapproved links that may be added to a page.

To enable *urlapprove.php*, add the following line to a configuration file:

```
include_once("$FarmD/scripts/urlapprove.php");
```

By default, unapproved links display what ever should be displayed normally (the URL or a text), but not linked and next to it a link (approve links). A click on the link will approve all unapproved URLs on the page,but *only* if you are authorized to edit the *SiteAdmin.ApprovedUrls* page. You may also pre-approve sites by by adding them directly to the SiteAdmin.ApprovedUrls page.

Limiting unapproved urls per page

You can limit the number of unapproved links per page. If the limit is exceeded, the page cannot be saved. This is useful because spammers like to write long link lists, which is rare for normal authors.

Example: To set the limit to 5 unapproved links, add the following line to a configuration file:

```
$UnapprovedLinkCountMax = 5;
include_once('scripts/urlapprove.php');
```

Note that $UnapprovedLinkCountMax must be set *before* including the *urlapprove.php* script.

Handling of Unapproved Links

You can also change the disapproval message defined in the $UnapprovedLinkFmt variable, for example:

```
include_once('scripts/urlapprove.php');
$UnapprovedLinkFmt =
 "[$[Link requires approval]]<a class='apprlink'
  href='\$PageUrl?action=approvesites'>$[(approve)]</a>";
```

"Link requires approval" is whatever you want to see in place of the unapproved link and "(approve)" is the blue text. Using this feature may prove usefull if you want to always hide the unapproved link.

If you wish to totally forbid unapproved links you can use

```
$UnapprovedLinkFmt = "<b>external link not allowed</b>";
```

SideBar caveat

Please note that in general you need to go to the sidebar page in order to approve links in the sidebar. The reason for this is that the approve mechanism only approves links on the *current* page.

Initial setup

After initial setup all existing links become unapproved. You need to visit your pages and approve all links, where needed. See AllRecentChanges for a list of all pages that were created on your wiki.

Password approval of URLs

To approve external links, an author needs permissions to edit the page SiteAdmin.ApprovedUrls.

Technical tips

URL Whitelist

Urls can also be approved by adding them to a "white list", defined in the variable `$WhiteUrlPatterns`, which is set in the *local/config.php* file.
To add multiples urls, use the separator `|` (vertical bar). For example:

```
$WhiteUrlPatterns =
  "(http://example.com|http://example.net|http://example.org)";
```

To add all urls from, say New Zealand and Australia, use:

```
$WhiteUrlPatterns[] = 'http://[^/]+\\.nz';
$WhiteUrlPatterns[] = 'http://[^/]+\\.au';
```

Change Approved URLs page name

If you want to change the default name of*SiteAdmin.ApprovedUrls*, set the following in*local/config.php*:

```
$ApprovedUrlPagesFmt = array('OtherGroup.OtherName');
```

Previewing the unapproved URL

To see what link is to be approved without editing the page a tool tip can be displayed when the cursor hovers over the(approve links) link that displays the URL. e.g. Example.

Add the following setting in your*local/config.php*:

```
$UnapprovedLinkFmt =
  "\$LinkText<a class='apprlink' href='\$PageUrl?action=approvesites'
   title='\$LinkUrl'>$[(approve links)]</a>";
```

Some browsers show only the link and not the tooltip title. In this case, you can use the following code to see the unapproved link at the end of the tooltip :

```
$UnapprovedLinkFmt =
  "\$LinkText<a class='apprlink' href='\$PageUrl?action=approvesites&XES_url=\$LinkUrl'
   title='\$LinkUrl'>$[(approve sites)]</a>";
```

About rel='nofollow'

By default, PmWiki creates external links that are not followed by search engines. Here are release notes from pmwiki-2.0.beta20 (30-Jan-2005):

> *First, the `$UrlLinkFmt` variable has been modified so that links to external urls automatically have a rel='nofollow' attribute added to them, to help combat wiki spam as described in http://googleblog.blogspot.com/2005/01/preventing-comment-spam.html. Site administrators can customize `$UrlLinkFmt` and $UnapprovedLinkFmt to supply or omit rel='nofollow' as appropriate.*

See Also

- Blocklist - Blocking postings based on content or IP address
- Security - Securing your PmWIki

Last modified by Peter Bowers on September 10, 2011.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/UrlApprovals

toc top

Variables

toc top

This page documents the PHP variables available in PmWiki for local customizations. Much of this documentation is still incomplete but people are working on it now. Feel free to add placeholders for variables you want to have documented if you don't know what the variable does.

The variables documentation is divided into several pages:

- Basic Variables - core variables
- Debug Variables - variables useful for debugging
- Edit Variables - variables used when editing pages
- I18n Variables - variables used for internationalization (i18n)
- Layout Variables - variables that control page layout
- Link Variables - variables that control the display of links in pages
- Other Variables - Variables not yet classified
- Pagelist Variables - variables used with page lists and search results
- Path Variables - variables used to specify various locations on the server
- Security Variables - variables crucial for site security
- Upload Variables - Variables used for uploads/attachments

The following functions are also controlled by several variables:

- Blocklist - Blocking IP addresses, phrases, and expressions to counteract spam and vandalism.
- Notify - How to receive email messages whenever pages are changed on the whole wiki site, individual groups or selected

List of documented PHP variables

Variable	Documented in
`$AbortFunction`	DebugVariables
`$ActionSkin`	LayoutVariables
`$AllowPassword`	SecurityVariables
`$AsSpacedFunction`	BasicVariables
`$AuthId`	BasicVariables
`$AuthLDAPBindDN`	SecurityVariables
`$AuthLDAPBindPassword`	SecurityVariables
`$Author`	BasicVariables
`$AuthorGroup`	BasicVariables
`$AuthPw`	BasicVariables
`$AutoCreate`	EditVariables
`$BaseName`	BasicVariables
`$BaseNamePatterns`	BasicVariables
`$BlockedMessagesFmt`	Blocklist
`$BlocklistActions`	Blocklist
`$BlocklistDownload`	Blocklist
`$BlocklistDownloadFmt`	Blocklist
`$BlocklistDownloadRefresh`	Blocklist
`$BlocklistMessageFmt`	Blocklist

watchlists of pages

The following variables are used in page markup.

- Page Variables - variables that are associated with pages
- Page TextVariables - Page variables automatically made available through natural or explicit page markup

An complete index of documented PHP variables is given below.

In general, variables with names ending in 'Fmt' (such as $PageLayoutFmt) have their values processed for $-variable substitutions prior to being output. Thus strings such as `{$Name}` and `{$PageUrl}` are replaced with the name and URL of the page when the string is printed.

Note: The automatic variable index and link generation is done by scripts/vardoc.php using `$VarPagesFmt` to find the pages containing trails of pages with the variable documentation.

There is a slight discrepancy between index generation and link generation: The index generation finds lines starting with a colon followed by "$" and an uppercase word. In contrast, the automatic link generation works only with WikiWords (`$WikiWordPattern`) preceded by "$". Therefore all "non WikiWord" variables are shown as link only in the list below, but not elsewhere in PmWiki, as `$Author`, `$Version` and $XL.

See Also

- Functions - How some of the functions in pmwiki.php work

Categories: PmWiki Developer

Last modified by Petko on January 27, 2015. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/Variables

Version

toc top

This wiki installation is running versionpmwiki-2.2.99, version number 2002099.

Obtaining the PmWiki version

Use the `{$Version}` page variable to display the current version of PmWiki.

See the SiteAdmin.Status page for the current version and version number.

For example

`$BlocklistPages`	Blocklist
`$CategoryGroup`	BasicVariables
`$CookiePrefix`	BasicVariables
`$DefaultGroup`	BasicVariables
`$DefaultName`	BasicVariables
`$DefaultPage`	BasicVariables
`$DefaultPageCharset`	I18nVariables
`$DefaultPageTextFmt`	EditVariables
`$DefaultPasswords`	SecurityVariables
`$DeleteKeyPattern`	EditVariables
`$DiffKeepDays`	EditVariables
`$DiffKeepNum`	EditVariables
`$DraftActionsPattern`	EditVariables
`$DraftRecentChangesFmt`	LayoutVariables
`$DraftSuffix`	EditVariables
`$EditFunctions`	EditVariables
`$EditRedirectFmt`	EditVariables
`$EditTemplatesFmt`	EditVariables
`$EnableBlocklist`	Blocklist
`$EnableBlocklistImmediate`	Blocklist
`$EnableDiag`	DebugVariables
`$EnableDiffInline`	LayoutVariables
`$EnableDirectDownload`	UploadVariables
`$EnableDraftAtomicDiff`	EditVariables
`$EnableDrafts`	EditVariables
`$EnableFixedUrlRedirect`	LayoutVariables
`$EnableGUIButtons`	EditVariables
`$EnableIMSCaching`	DebugVariables
`$EnableLinkPageRelative`	LinkVariables
`$EnableLinkPlusTitlespaced`	LinkVariables
`$EnableLocalConfig`	BasicVariables
`$EnableNotify`	Notify
`$EnableNotifySubjectEncode`	Notify
`$EnablePageIndex`	PagelistVariables
`$EnablePageListProtect`	PagelistVariables
`$EnablePageTitlePriority`	LayoutVariables
`$EnablePageVarAuth`	SecurityVariables
`$EnablePathInfo`	LayoutVariables
`$EnablePGCust`	BasicVariables
`$EnablePostAttrClearSession`	SecurityVariables
`$EnablePostAuthorRequired`	EditVariables
`$EnablePublishAttr`	SecurityVariables
`$EnableRedirect`	BasicVariables
`$EnableRedirectQuiet`	LinkVariables
`$EnableRelativePageVars`	BasicVariables
`$EnableRevUserAgent`	EditVariables
`$EnableROSEscape`	EditVariables
`$EnableSessionPasswords`	SecurityVariables
`$EnableStdConfig`	BasicVariables
`$EnableStopWatch`	DebugVariables
`$EnableTableAutoValignTop`	LayoutVariables
`$EnableUndefinedTemplateVars`	PagelistVariables
`$EnableUpload`	UploadVariables
`$EnableUploadGroupAuth`	UploadVariables
`$EnableUploadOverwrite`	UploadVariables
`$EnableUploadVersions`	UploadVariables
`$EnableWhyBlocked`	Blocklist
`$EnableWikiWords`	BasicVariables
`$EnableWSPre`	BasicVariables
`$EnableXLPageScriptLoad`	I18nVariables
`$FarmD`	PathVariables
`$FarmPubDirUrl`	PathVariables
`$FmtP`	OtherVariables
`$FmtPV`	OtherVariables
`$FmtV`	OtherVariables
`$FPLTemplatePageFmt`	PagelistVariables
`$FTimeFmt`	BasicVariables

$GroupFooterFmt	LayoutVariables
$GroupHeaderFmt	LayoutVariables
$GroupPattern	BasicVariables
$GroupPrintFooterFmt	LayoutVariables
$GroupPrintHeaderFmt	LayoutVariables
$GUIButtons	EditVariables
$HandleAuth	SecurityVariables
$HandleEditFmt	EditVariables
$HTMLFooterFmt	LayoutVariables
$HTMLHeaderFmt	LayoutVariables
$HTMLPNewline	LayoutVariables
$HTMLStylesFmt	LayoutVariables
$HTMLTagAttr	LayoutVariables
$HTMLVSpace	LayoutVariables
$IMapLinkFmt	LinkVariables
$InterMapFiles	LinkVariables
$IsPagePosted	EditVariables
$LinkPageCreateFmt	LinkVariables
$LinkPageCreateSpaceFmt	LinkVariables
$LinkPageExistsFmt	LinkVariables
$LinkPageSelfFmt	LinkVariables
$LinkUploadCreateFmt	UploadVariables
$LinkWikiWords	BasicVariables
$LocalDir	PathVariables
$LogoutCookies	BasicVariables
$LogoutRedirectFmt	BasicVariables
$MakePageNameFunction	LinkVariables
$MakePageNamePatterns	LinkVariables
$MakePageNameSplitPattern	LinkVariables
$MakeUploadNamePatterns	UploadVariables
$MaxIncludes	LayoutVariables
$MaxPageTextVars	OtherVariables
$MessagesFmt	LayoutVariables
$MetaRobots	LayoutVariables
$NamePattern	BasicVariables
$NotifyBodyFmt	Notify
$NotifyDelay	Notify
$NotifyFile	Notify
$NotifyFrom	Notify
$NotifyHeaders	Notify
$NotifyItemFmt	Notify
$NotifyList	Notify
$NotifyListPageFmt	Notify
$NotifyParameters	Notify
$NotifySquelch	Notify
$NotifySubjectFmt	Notify
$NotifyTimeFmt	Notify
$PageAttributes	SecurityVariables
$PageCacheDir	OtherVariables
$PageCSSListFmt	PathVariables
$PageEditFmt	EditVariables
$PageEditForm	EditVariables
$PageIndexFile	PagelistVariables
$PageListCacheDir	PagelistVariables
$PageLogoUrl	LayoutVariables
$pagename	BasicVariables
$PageNotFoundHeaderFmt	LayoutVariables
$PagePathFmt	BasicVariables
$PageRedirectFmt	LayoutVariables
$PageSearchForm	PagelistVariables
$PubDirUrl	PathVariables
$QualifyPatterns	LinkVariables
$RCLinesMax	LayoutVariables
$RecentChangesFmt	LayoutVariables
$RecentUploadsFmt	LayoutVariables
$ROEPatterns	EditVariables
$ROSPatterns	EditVariables

$ScriptUrl	PathVariables
$SearchBoxInputType	PagelistVariables
$SearchBoxOpt	PagelistVariables
$SearchPatterns	PagelistVariables
$SessionDecode	SecurityVariables
$SessionEncode	SecurityVariables
$SimpleTableDefaultClassName	LayoutVariables
$SiteAdminGroup	BasicVariables
$SiteGroup	BasicVariables
$Skin	BasicVariables
$SkinDir	PathVariables
$SkinDirUrl	PathVariables
$SkinLibDirs	LayoutVariables
$SpaceWikiWords	BasicVariables
$TableCellAlignFmt	LayoutVariables
$TableCellAttrFmt	LayoutVariables
$TableRowAttrFmt	LayoutVariables
$TableRowIndexMax	LayoutVariables
$TimeFmt	BasicVariables
$UploadBlacklist	UploadVariables
$UploadDir	UploadVariables
$UploadDirQuota	UploadVariables
$UploadExtSize	UploadVariables
$UploadMaxSize	UploadVariables
$UploadNameChars	UploadVariables
$UploadPermAdd	UploadVariables
$UploadPermSet	UploadVariables
$UploadPrefixFmt	UploadVariables
$UploadPrefixQuota	UploadVariables
$UploadUrlFmt	UploadVariables
$UrlLinkFmt	LinkVariables
$VarPagesFmt	I18nVariables
$Version	BasicVariables
$VersionNum	BasicVariables
$WikiDir	PathVariables
$WikiLibDirs	PathVariables
$WikiStyle	LayoutVariables
$WikiStyleApply	LayoutVariables
$WikiTitle	LayoutVariables
$WikiWordCount	LinkVariables
$WikiWordCountMax	LinkVariables
$WikiWordPattern	BasicVariables
$WorkDir	PathVariables
$XLLangs	I18nVariables

```
This wiki installation is running PmWiki {$Version}, version number {$VersionNum}.
-<The default group is {$DefaultGroup}.
-<The default name is {$DefaultName}.
-<The site group is {$SiteGroup}
```

This wiki installation is running PmWiki pmwiki-2.2.99, version number 2002099.
The default group is Main.
The default name is HomePage.
The site group is Site

See also basic variables.

The script `version.php` contains the declaration of the version number. The file is located on scripts/version.php, relative to PmWiki installation path.

Obtaining recipe versions

The Site Analyzer can be used to display the current version of Cookbook recipes. If you are the administrator, cookbook recipe guidelines state that the first couple of lines of a recipe file should contain the recipe version:

```
  $RecipeInfo['RecipeName']['Version'] = '2017-06-02'; # dates YYYY-MM-DD prefered
```

See also Cookbook:RecipeCheck

WebFeeds

toc top

Web feeds are a convenient mechanism to let visitors be notified of changes to a site. Instead of repeatedly checking RecentChanges every day to see what is new, a visitor can use a news aggregator to quickly see what pages of interest have changed on a site. Web feeds are commonly recognized by terms such as RSS, Atom, and *web syndication*. They are also the foundation for podcasting.

In its simplest form, web feeds in PmWiki are built on WikiTrails. Using a feed action such as `?action=rss` or `?action=atom` on a trail generates a web feed (often called a "channel") where each page on the trail is an item in the feed. Since the RecentChanges and Site.AllRecentChanges pages are effectively trails, one can easily get an RSS feed for a group or site by simply adding `?action=rss` to the url for a RecentChanges page. For example, to get the site feed for pmwiki.org, one would use

> http://pmwiki.org/wiki/Site/AllRecentChanges?action=rss

Authors can also create custom feeds by simply creating a wiki trail of the pages they want included in the feed. Feeds can also be generated from groups, categories, and backlinks, and the order and number of items in the feed can be changed using options in the feed url. Thus, one can obtain a feed for the *Skins* category (sorted with most recent items first) by using

> http://pmwiki.org/wiki/Category/Skins?action=rss&order=-time

PmWiki is able to generate feeds in many formats, including RSS 2.0 (`?action=rss`), Atom 1.0 (`?action=atom`), and RSS 1.0 (`?action=rdf`). In addition, although it is not normally considered a web feed, PmWiki can generate metadata information using the Dublin Core Metadata extensions (`?action=dc`).

How to read a PmWiki syndicated feed

1. You'll need a news aggregator, which is a piece of software designed to read news feeds. Many different news aggregators are available. Some run on your own computer, either on their own or as plugins for email clients, web browsers, or newsreaders. Others are web applications that you can use from any Internet-connected computer. Some are in between (technically web applications, but ones designed to run on your computer, not some remote server). Get one that you like.
2. Subscribe to the WikiTrail you desire by supplying the feed url to the aggregator. The feed url will be the name of a trail page with `?action=rss` or `?action=atom` added to the end of the url.

Feed options

Add any of the following options to the end of a PmWiki web feed url to change its output (basically any pagelist option is available for web feeds):

?count=*n*
: Limit feed to *n* items (default 10)

?order=-time
: Display most recently changed items first (default: the order of the trail, or by name; in RecentChanges pages the trail is already ordered by -time)

?trail=*page*
: Obtain items from trail on *page* (default: the trail on the current page)

?group=*group*
: Limit feed to pages in *group*

?name=*name*
: Limit feed to pages with specific *name*

?link=*page*
: Create feed from pages linked to *page*

?list=normal
: Exclude things like RecentChanges, AllRecentChanges, etc.

authors (intermediate)

Configure PmWiki for feeds

This section describes how to syndicate portions of a wiki to appear in a web feed. It does not describe how to display a web feed within a wiki page -- for that, see Cookbook:RssFeedDisplay.

To enable web feed generation for a site, add one or more of the following to a local customization file:

```
if ($action == 'rss') include_once("$FarmD/scripts/feeds.php");
if ($action == 'atom') include_once("$FarmD/scripts/feeds.php");
if ($action == 'rdf') include_once("$FarmD/scripts/feeds.php");
if ($action == 'dc') include_once("$FarmD/scripts/feeds.php");
```

or you can combine multiple feeds into a single expression using "||" to separate each feed type. For example, if you want to enable RSS and Atom feeds you would use

```
    if ($action == 'rss'  ||
        $action == 'atom' ||
        $action == 'rdf'  ||
        $action == 'dc') include_once("$FarmD/scripts/feeds.php");
```

Configure feed content

Web feeds are highly configurable, new elements can be easily added to feeds via the $FeedFmt array. Elements in $FeedFmt look like

```
$FeedFmt['atom']['feed']['rights'] = 'All Rights Reserved';
```

where the first index corresponds to the action (?action=atom), the second index indicates a per-feed or per-item element, and the third index is the name of the element being generated. The above setting would therefore generate a "<rights>All Rights Reserved</rights>" in the feed for ?action=atom.

If the value of an entry begins with a '<', then feeds.php doesn't automatically add the tag around it. Elements can also be callable functions which are called to generate the appropriate output. See RSS specification or other feed specifications for what feed content you can use.

You can also change an existing element rather than add a new one. You can use the following lines to ensure that changes made to the wiki will be picked up by some RSS readers that wouldn't otherwise "notice" a page has been changed:

```
    # Change the link URL when an item is edited.
    $FeedFmt['rss']['item']['link'] = '{$PageUrl}?when=$ItemISOTime';
    $FeedFmt['atom']['item']['link'] =
       "<link rel=\"alternate\" href=\"{\$PageUrl}?when=\$ItemISOTime\" />\n";
```

See Also

- Cookbook:FeedLinks - Add HTML <head> links for auto-discovery of your feeds.
- WikiTrails
- Wikipedia:Web_feed, Wikipedia:Web_syndication, Wikipedia:RSS, Wikipedia:Atom_%28standard%29

How do I include text from the page (whole page, or first X characters) in the feed body? (note: markup NOT digested)

```
        function MarkupExcerpt( $pagename) {
          $page = RetrieveAuthPage( $pagename, 'read', false);
          return substr(@$page['text'], 0, 200);
        }

         $FmtPV['$MarkupExcerpt'] = 'MarkupExcerpt($pn)';
        $FeedFmt['rss']['item']['description'] = '$MarkupExcerpt';
```

Does this mean if I want to include the time in the rss title and "summary" to rss body I call `$FeedFmt` twice like so:

```
         $FeedFmt['rss']['item']['description'] = '$LastSummary';
         $FeedFmt['rss']['item']['title'] = '{$Group} / {$Title} @ $ItemISOTime';
```

From mailing list Feb 13,2007, a response by Pm: Yes

How can I use the RSS <enclosure> tag for podcasting?

For podcasting of mp3 files, simply attach an mp3 file to the page with the same name as the page (i.e., for a page named Podcast.Episode4, one would attach to that page a file named "Episode4.mp3"). The file is automatically picked up by ?action=rss and used as an enclosure.

The set of potential enclosures is given by the $RSSEnclosureFmt array, thus

```
    $RSSEnclosureFmt = array('{$Name}.mp3', '{$Name}.wma', '{$Name}.ogg');
```

allows podcasting in mp3, wma, and ogg formats.

How to add "summary" to the title in a rss feed (ie. with `?action=rss`)?

Add this line in you `local/config.php`

```
    $FeedFmt['rss']['item']['title'] = '{$Group} / {$Title} : $LastModifiedSummary';
```

How to add "description" to the title in an rss feed, and summary to the body?

Add these lines to your `local/config.php`

```
$FeedFmt['rss']['item']['title'] = '{$Group} / {$Title} : {$Description}';
$FeedFmt['rss']['item']['description'] = '$LastModifiedSummary';
```

NOTES:

- you need to replicate these lines for each type (atom, rdf, dc) of feed you provide.
- the RSS **`description`**-tag is not equivalent to the pmWiki *`$Description`* variable, despite the confusing similarity.

Some of my password-protected pages aren't appearing in the feed... how do I work around this?

From a similar question on the newsgroup, Pm's reply:

The last time I checked, RSS and other syndication protocols didn't really have a well-established interface or mechanism for performing access control (i.e., authentication). As far as I know this is still the case.

PmWiki's WebFeeds capability is built on top of pagelists, so it could simply be that the `$EnablePageListProtect` option is preventing the updated pages from appearing in the feed. You might try setting `$EnablePageListProtect`=0; and see if the password-protected pages start appearing in the RSS feed.

The "downside" to setting `$EnablePageListProtect` to zero is that anyone doing a search on your site will see the existence of the pages in the locked section. They won't be able to read any of them, but they'll know they are there!

You could also set `$EnablePageListProtect` to zero only if ?action=rss:

```
    if ($action == 'rss')  $EnablePageListProtect = 0;
```

This limits the ability to see the protected pages to RSS feeds; normal pagelists and searches wouldn't see them.

Lastly, it's also possible to configure the webfeeds to obtain the authentication information from the url directly, as in:

```
    .../Site/AllRecentChanges?action=rss&authpw=secret
```

The big downside to this is that the cleartext password will end up traveling across the net with every RSS request, and may end up being recorded in Apache's access logs.

How to add feed image?

Add the following to *local/config.php* (this example is for `?action=rss`):

```
     $FeedFmt['rss']['feed']['image'] =
     " <title>Logo title</title>
      <link>http://example.com/</link>
      <url>http://example.com/images/logo.gif</url>
      <width>120</width>
      <height>60</height>";
```

Do not forget NOT to start with a '<' as there would be no <image> tag around this... See here.

How do I insert RSS news feeds into PmWiki pages?

See Cookbook:RssFeedDisplay.

How can I specify default feed options in a configuration file instead of always placing them in the url?

For example, if you want `?action=rss` to default to `?action=rss&group=News&order=-time&count=10`, try the following in a local customization file:

```
        if ($action == 'rss')
          SDVA($_REQUEST, array(
            'group' => 'News',
            'order' => '-time',
            'count' => 10));
```

Are there ways to let people easily subscribe to a feed?

On some browsers (Mozilla Firefox), the visitor can see an orange RSS icon in the address bar, and subscribe to the feed by clicking on it. To enable the RSS icon, add this to config.php :

```
$HTMLHeaderFmt['feedlinks'] = '<link rel="alternate" type="application/rss+xml"
  title="$WikiTitle" href="$ScriptUrl?n=Site.AllRecentChanges&amp;action=rss" />
<link rel="alternate" type="application/atom+xml" title="$WikiTitle"
  href="$ScriptUrl?n=Site.AllRecentChanges&amp;action=atom" />';
```

You can also add such a link, for example in your SideBar,`[[Site.AllRecentChanges?action=atom | Subscribe to feed]]`.

Can I create an RSS feed for individual page histories?

See Cookbook:PageFeed.

How do I create a custom FeedPage similar to RecentChanges or AllRecentChanges, but with only certain groups or pages recorded?

See Cookbook:CustomRecentChanges. In a nutshell, you'll declare a `$RecentChangesFmt` variable with your dedicated FeedPage, and then wrap it in a condition of your choice. For example:

```
if (PageVar($pagename, '$Group')!='ForbiddenGroup') {
  $RecentChangesFmt['Site.MyFeedPage'] =
    '* [[{$FullName}]]  . . . $CurrentTime $[by] $AuthorLink: [=$ChangeSummary=]';
}
```

How can I update my RSS feed to show every edit for pages on that feed, not just new pages added to the feed?

Add unique guid links for each edit to your to config.php file (see PITS entry):

```
$FeedFmt['rss']['item']['guid'] = '{$PageUrl}?guid=$ItemISOTime';
```

Alternatively, you can create the option for edit monitoring by adding a qualifier for RSS links. This allows the user to choose between default *new pages* RSS feeds and *new edits* RSS feeds (pmwiki.org has this option enabled).

```
## For new pages updates: http://example.com/wiki/HomePage?action=rss
## For edits updates: http://example.com/wiki/HomePage?action=rss&edits=1
if(@$_REQUEST['edits'] && $action == 'rss')
  $FeedFmt['rss']['item']['guid'] = '{$PageUrl}?guid=$ItemISOTime';
```

Last modified by mfwolff on July 07, 2016. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WebFeeds

WikiAdministrator

toc top

A Wiki Administrator is a person (or persons) who installs, configures, and administers a PmWiki system for authors and site visitors. PmWiki has been designed to make the installation and initial setup tasks as easy as possible for people who do not have a lot of knowledge about HTML, PHP, or even web server software. At the same time, PmWiki is designed to be flexible enough so that someone with just a little bit of knowledge about HTML and PHP can customize PmWiki to their specific needs.

See the PmWiki.documentation index for pages about administering PmWiki, administration tasks, security, and PmWiki.audiences for more details of PmWiki's target audiences.

Last modified by Petko on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiAdministrator

WikiFarmTerminology

toc top

administrators (intermediate) There are many ways to configure PmWiki:WikiFarms, and some of the documentation uses different terminology to describe the same things. This page attempts to explain the terminology.

For terms not related to farms, see Glossary.

Why is this page needed?

- to provide a place to find the preferred terminology with definitions
- to explain where the term "farm" came from
- to list various terms that have been deprecated but still exist in the docs
- to suggest alternate terms for the deprecated ones

The origins of WikiFarms

The term WikiFarm is based on the computing term "server farm", which is a collection of servers that use a common infrastructure. A wiki farm is nothing more than multiple wikis that share the same installation of the PmWiki software.

Some recipe and documentation authors, however, began writing about WikiFarms using agricultural terms such as "field", "farmer", "barn", "crop", and "tractor". In some cases these terms made the documentation more confusing. It is suggested that documentation authors avoid the agricultural terms, as tempting as they may be, and keep in mind that a wiki in a wiki farm is like a server in a server farm.

Wikis and components in a WikiFarm

All of the wikis in a farm are more or less the same, except the "home wiki" is a wiki that is located in the same directory as the PmWiki software. The home wiki needs special consideration because it holds the components that are shared by or affect the operation of all the wikis in the farm. In particular:

- the *scripts/* directory
- the *cookbook/* directory
- the *pub/* directory

- the *wikilib.d/* directory

It is possible to move the PmWiki software outside of the web document tree, but the *pub/* directory needs to be in a web-servable directory (one that can be accessed by a URL).

Authors writing about complex farm setups often have difficulty describing the components and their locations. However, it is probably not necessary or desirable to coin new terms for the components and their locations.

Suggested terms

WikiFarm
: An installation where one copy of PmWiki is configured to run multiple wikis. Analogous to the computing phrase "server farm". The wikis in a farm can be configured farm-wide (using the farm's *local/farmconfig.php*) or individually (using the wiki's *local/config.php*).

Wiki
: A site with it's own URL and *wiki.d/* directory. All of the wikis in a wiki farm are simply called wikis.

Home wiki
: A wiki in a farm that's located in the same directory as the PmWiki software and therefore shares the farm's *cookbook/* and *pub/* directories. If you start with a stand-alone installation and add a wiki, the original wiki becomes a home wiki.

Farm-wide
: Something available to or affecting all wikis in the farm. Typically this means modifying the *farmconfig.php* file or the contents of the farm's cookbook/ or pub/ directories.

Local
: Something available to or affecting a specific wiki. Typically this means modifying the wiki's *local/config.php* file or the contents of the wiki's *cookbook/* or *pub/* directories.

PmWiki ~~engine~~
: The software that makes PmWiki work, as opposed to the content of the wiki that readers see.

PmWiki ~~installation~~ directory
: The directory PmWiki is installed to. It contains pmwiki.php and its subdirectory scripts/, which is used by all the wikis in the WikiFarm. ~~If you do a standard, single install of PmWiki, it goes into this directory~~.

Ambiguous terms

Installation directory
: Installation of what? Some authors have used this to mean the directory that contains most of the shared components on a wiki farm. Others use it to mean a directory that has a complete standalone installation of PmWiki that is not part of a farm. Use PmWiki directory instead.

PmWiki installation
: This is sometimes used to indicate a process, sometimes used to mean a single wiki in a farm, and sometimes refers only to the shared components of a farm.

Deprecated terms that should not be used

These terms still exist in the documentation (pending revisions), and will live forever in the PmWiki-Users list archive.

farm directory
: The directory in which the home wiki lives or a directory where the shared components are stored. Use PmWiki directory instead.

field
: Any wiki in a farm which is **not** the home wiki.

farm administrator
: An administrator who has access to all of the wikis in a farm, particularly the home wiki. Use administrator instead.

field administrator
: An administrator who has access to one or more wikis in a farm, but **not** the home wiki. Use administrator instead.

barn
: The place where common components are stored. Use PmWiki directory instead.

crop
: Packaged content and customizations that can be added to a wiki. See Cookbook:ListOfBundles for similar ideas. Use component bundles instead.

tractor
: The PmWiki engine or *pmwiki.php* itself. Use PmWiki instead.

Categories: WikiFarms

Last modified by Scott Connard on September 10, 2011.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiFarmTerminology

toc top

WikiFarms

toc top

Also see: Cookbook:Farm Setup By Example, Cookbook:Wiki Farm Alternative

A WikiFarm is a collection of two or more wikis running on the same web server and sharing a set of common components. The term is based on the computing phrase "server farm".

This page provides some background information about WikiFarms and describes how to turn a "normal" configuration into a farm by adding a wiki. (Click here to go directly to instructions on configuring a farm.) There are many ways to configure wiki farms; this page describes only one, in an effort to make it as simple as possible for the administrator who is creating a farm for the first time.

This page will discuss 3 ways to organize content:
1. Use WikiGroups
2. Use independent wiki sites with a shared code base (a "farm")
3. Use independent wiki sites with a complete PmWiki installation per site

Choosing between separate wiki-sites and WikiGroups

Why use WikiGroups?

When you divide content between independently installed wikis (i.e., with their own separate URL), it is difficult (but not impossible) to provide services that require access to more than one wiki. For example, the PmWiki search function can only search within one wiki. Using a farm as a way of subdividing related content is generally a bad idea. A much better way to subdivide content is to use WikiGroups.

Why use separate wiki-sites?

When content is largely unrelated and there will be little or no need for sharing the data between the sites, it makes sense to divide the wikis into independently installed sites.

Choosing between separate, independent installations of PmWiki and a WikiFarm

Once you have decided that you need a separate wiki (with its own URL), you have two basic choices:
1. Do a complete installation of PmWiki in a new directory. This gives you two totally independent wikis that are completely self-contained. This is **not** a wiki farm.
2. Create a wiki farm using an existing wiki as the "home wiki" where most of the shared PmWiki components will live.

The primary motivation for using a wiki farm is to reduce the amount of administrative work involved in managing several wikis. In a farm, most of the PmWiki code is stored in one place and is shared by all the wikis. An administrator can (for example) upgrade to a new version of PmWiki on every wiki in the farm by simply updating the shared components in a single location.

From a reader's point of view, there is no difference between separate, self-contained installations of PmWiki and separate wikis within a WikiFarm: each wiki in a farm is completely independent, and appears as a separate web site. Each wiki in a farm:
- has its own URL, and the URLs can be in different domains
- can have its own look and feel by using different skins
- can have its own add-ons or "recipes" from the Cookbook
- can have its own administrator responsible for local configuration

Why to use independent, self-contained installations of PmWiki

- it is not a wiki farm, and requires no additional administrative knowledge - it's just two installations
- if you decide to move one of the wikis to another server, you can simply copy the wiki directory structure to the second server, and it will work (assuming there is a web server and PHP in place).
- you can run different versions of PmWiki on each wiki (good for testing new versions)
- no matter how badly you mess up one installation, it doesn't affect the other

Why to use a WikiFarm

- you can upgrade all wikis in your farm by simply upgrading your home wiki
- recipes can be shared across all wikis
- portions of your configuration can be shared across wikis
- most code is stored in one location and shared by all wikis in the farm

I still can't decide if I need a farm ...

The good news is that you don't have to decide in advance. In fact, the recommended procedure is to first do a "normal" or single installation of PmWiki. Use it for a while. Create pages and edit them. Get to know how to add recipes. Be sure to try out WikiGroups (they may be all you need).

If you choose to create a wiki farm, then read on ...

Creating/Configuring a WikiFarm

Prerequisites

Before you create a farm, make sure that:
- you have a working installation of PmWiki ready to become the home wiki for your farm
- all of the wikis in your farm will be on the same web server

- each wiki will have a unique URL, such as http://www.example.com/wiki1/, http://www.example.com/wiki2/, http://another.example.com/wiki1/ and so on.

Creating the home wiki

You do have a working installation of PmWiki at this point, don't you? That's good, because your existing wiki is about to become the home wiki of your farm.

In the directory that contains your existing wiki, create the file *local/farmconfig.php*. This file is used to hold any local customizations that apply across the whole farm. For example, you could assign an admin password in *farmconfig.php* that will be used by all of the wikis in your farm.

If the URL used to access your existing wiki is http://www.example.com/pmwiki/ then a minimal *farmconfig.php* file would look like this:

```
<?php if (!defined('PmWiki')) exit();
$FarmPubDirUrl = 'http://www.example.com/pmwiki/pub';
```

This loads the variable `$FarmPubDirUrl` with the URL location of your home wiki's *pub/* directory. All of the wikis in your farm share this *pub/* directory. The *pub/* directory holds skin definitions and GUI-edit buttons to be shared by all the wikis in the farm.

Amazing as it may sound, this completes all of the changes you need to make in order to turn your existing wiki into the home wiki of your farm.

Creating an additional wiki in your farm

1. Create a directory to hold the new wiki. This directory must be web-accessible, just like the directory that holds your home wiki.

2. Create a file called *index.php* in the directory with the following contents:

   ```
   <?php include_once('path/to/pmwiki.php');
   ```

   This allows your new wiki to share the PmWiki code stored in your home wiki. The *`path/to/pmwiki.php`* is the file path to *pmwiki.php* in your home wiki. Use an absolute file path (`/home/username/pmwiki/pmwiki.php`) or a relative file path (`../pmwiki/pmwiki.php`). Do not use a url path - there should not be an '`http://`' in it anywhere. For a web server running under Windows, you need to use a complete file path as in `C:/Apache Group/Apache2 /www/mynewwiki/`.

3. Open a web browser and browse the URL of the new wiki. This will be a web address starting with '`http://`'. PmWiki will attempt to automatically create a writable *wiki.d/* directory where the wiki's pages will be stored. If you see an error message, follow the instructions. If you choose the option for a "slightly more secure installation" be sure to execute both commands.

Your new wiki is now set up, and your farm now contains 2 wikis. To add more wikis, just repeat these 3 steps.

Customization

Each wiki in a farm inherits the settings stored in *farmconfig.php*. Do any customization that you want to apply farm-wide (to all the wikis) in *farmconfig.php*.

Create a *local/* directory within each wiki's directory to hold local customizations that apply only to that wiki. You should at least create the local/config.php file with a new title, like so :

```
<?php if (!defined('PmWiki')) exit();
  ## Title of your farmed wiki
  $WikiTitle = 'New Wiki';
```

Farm-wide customizations are processed before the individual wiki local customizations.

The PmWiki variable *`$FarmD`* points to the directory in which pmwiki.php is installed, and your home wiki, and it is used as a prefix to allow the other wikis to share PmWiki components. For example:

- *`$FarmD`/scripts/* points to the shared *scripts/* directory
- *`$FarmD`/pub/* points to the shared *pub/* directory
- *`$FarmD`/cookbook/* points to the shared *cookbook/* directory

Any Cookbook scripts you include in farmconfig.php must be included with a line such as:

```
include_once("$FarmD/cookbook/scriptfile.php");
```

Note the double quotes - single quotes may work for per farm inclusions, but they will not work for `$FarmD`.

Password use/authorization on farm wikis:

How come when I switch to another wiki within a farm, I keep my same authorization?

PmWiki uses PHP sessions to keep track of authentication/authorization information, and by default PHP sets things up such that all interactions with the same server are considered part of the same session.

An easy way to fix this is to make sure each wiki is using a different cookie name for its session identifier. Near the top of one of the wiki's local/config.php files, before calling authuser or any other recipes, add a line like:

```
    session_name('XYZSESSID');
```

You can pick any alphanumeric name for XYZSESSID; for example, for the cs559-1 wiki you might choose

```
    session_name('CS559SESSID');
```

This will keep the two wikis' sessions independent of each other.

Notes

- The terminology used to describe wiki farms is not used consistently. See WikiFarmTerminology for more info.
- It is important to remember that not all of the recipes in the Cookbook have been written for or tested with farms. Be sure to look for instructions on how to use a recipe on a farm.
- There are many, many more things you can do with farms. Some are described on PmWiki:WikiFarmsAdvanced which also contains links to step-by-step examples of setting up a farm.

Categories: WikiFarms

Last modified by Peter Bowers on June 03, 2015. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiFarms

WikiGroup

toc top

PmWiki pages are organized into groups of related pages. This feature was added to PmWiki to allow authors to create their own *wiki spaces* of specialized content on their own, without having to become, or rely on, wiki administrators. See Pm's post to the pmwiki-users mailing list.

By default, page links are between pages of the same group; to create a link to a page in another group, add the name of the other group and a dot or slash to the page name. For example, links to Main/WikiSandbox could be written as:

`* [[Main.WikiSandbox]]`	• Main.WikiSandbox
`* [[Main/WikiSandbox]]`	• WikiSandbox
`* [[(Main.Wiki)Sandbox]]`	• Sandbox
`* [[Main.WikiSandbox | link text]]`	• link text
`* [[Main.WikiSandbox | +]]`	• Wiki Sandbox

To link to the default home page of a group, the name of the page can be omitted, like this:

`* [[Main.]]`	• Main.
`* [[Main/]]`	• Main

Creating groups

Creating a new group is as easy as creating new pages; simply edit an existing page to include a link to the new group's default home page (or any page in the new group) then click on the '?' to edit the page. As a rule, group names must start with a letter (but this can be changed by the wiki administrator by adding

```
    $GroupPattern = '[[:upper:]\\d][\\w]*(?:-\\w+)*';
```

in config.php).

For example, to make a default page in the group Foo, create a link to [[Foo/]] (or [[Foo.]]). To make a page called Bar in the group Foo, create a link to [[Foo/Bar]] and follow the link to edit that page.

Groups in a standard PmWiki distribution

- Main: The default group. On many wikis, it contains most of the author-contributed content. Main.HomePage and Main.WikiSandbox come pre-installed.
- PmWiki: An edit-protected group that contains PmWiki documentation and help pages.
- Site: Holds a variety of utility and configuration pages used by PmWiki, including SideBar, Search, Preferences, Templates, and AllRecentChanges.
- SiteAdmin: Holds a number of password protected administration and configuration pages used by PmWiki, including ApprovedUrls, and Blocklist

- To list all the groups in a site, try searching for " fmt=group".
- To list all the pages in a group, try searching for " GroupName/".

Special Pages in a Group

By default, the *Recent Changes* page of each group shows only the pages that have changed within that group; the *Site.All Recent Changes* page shows all pages that have changed in all groups.

Each group can also have *Group Header* or *Group Footer* pages that contain text to be automatically prepended or appended to every page in the group. A group can also have a *Group Attributes* page that defines attributes (read and edit passwords) shared by all pages within the group.

Each page can also have its own individual read/edit password that overrides the group passwords (see Passwords).

Finally, wiki administrators can set local customizations on a per-group basis--see Group Customizations.

Group's default page

The default "start page" for a group is a page whose name can be:

1. the same as the group (Foo/Foo)
2. HomePage (Foo/HomePage)
3. a name that the administrator has assigned to the {$DefaultName} variable in the configuration ([farm]config.php) file.

Note, on this site, the value of {$DefaultName} is *HomePage* and, thus, the default home page would be Foo/HomePage.

You can usefully change the default *search* order for an entered page name by setting the variable `$PagePathFmt` in `config.php`, eg

```
      $PagePathFmt = array('$Group.$1', '$1.$DefaultName', '$1.$1', '$DefaultGroup.$1', 'Profiles.$1');
```

where "$1" is the name of the page entered.

If you are setting `$DefaultName` in order to make a start page for your groups, you will need to also define `$PagePathFmt` (see above) to get consistent use of this functionality. The simplest setting would be this:

```
      $PagePathFmt = array('$Group.$1', '$1.$DefaultName');
```

Note that the order of the definitions of these variables (`$DefaultName` and `$PagePathFmt`) is important - it must occur before any call to ResolvePageName() and it (therefore) it cannot occur in a per-page or per-group customization script.

As noted above, when linking to the default home page, authors can omit the page name and simply identify the group followed by a forward slash ([[Foo/]]).

Note the forward slash is required to ensure that the link unambiguously points to the identified group. If the slash is omitted, the link can end up being interpreted as pointing to an existing (or new) page in the current group (if the group, or its default home page, do not exist).

Subgroups? Subpages?

No, PmWiki does not have subpages. Pm's reasons for not having subgroups are described at PmWiki:Hierarchical Groups, but it comes down to not having a good page linking syntax. If you create a link or pagename like `[[A.B.C]]` PmWiki doesn't think of "B.C" as being in group "A", it instead thinks of "C" as being in group "AB", which is a separate group from "A". Wiki administrators can look at Cookbook:Subgroup Markup and Cookbook:Include With Edit for recipes that may be of some help with developing subgroups or subpages.

Restricting the creation of new groups

You can set PmWiki's `$GroupPattern` variable to only accept the group names you want to define. For example, to limit pages to the "PmWiki", "Main", "Profiles", and "Example" groups, add the following to local/config.php:

```
      $GroupPattern = '(?:Site|SiteAdmin|PmWiki|Main|Profiles|Example)';
```

With this setting, only the listed groups will be considered valid WikiGroups. You can add more groups to the list by placing additional group names separated by pipes (|).

See other solutions to this at Cookbook:Limit Wiki Groups and Cookbook:New Group Warning.

How can I get rid of the 'Main' group in urls for pages pointing to Main?

> See Cookbook:Get Rid Of Main.

How can I limit the creation of new groups?

> See Cookbook:Limit Wiki Groups.

Why doesn't [[St. Giles and St. James]] work as a link? (It doesn't display anything.)

Because it contains periods, and destroys PmWiki's file structure, which saves pages as Group.PageName. Adding those periods disrupts this format. Links may only contain words. If you need a link precisely as shown, the page must be named eg StGilesAndStJames then you can use the (:title:) directive to have the page's title appear with periods (:title St. Giles and St. James:). (Although in US grammar the period is often omitted and in UK grammar the period must be omitted for contractions like St).

How can I delete a wiki group?

Normally you can't, as this requires an admin with server-side access to delete the file that makes up the group's RecentChanges page. But there is an option method of making it possible to delete RecentChanges pages from within the wiki if the admin enables the code found on Cookbook:RecentChanges Deletion.

How can I delete a wiki group's Group.RecentChanges page?

Normally you can't, as this requires an admin with server-side access to delete a file. But there is an optional method of making it possible to delete RecentChanges pages from within the wiki if the admin enables the code found on Cookbook:RecentChanges Deletion.

Can I delete a wiki group inside wiki.d folder on the server to eliminate the group?

Yes, if you delete all files named YourGroup.*, the pages from that group will be removed from the wiki. Note that the documentation (group PmWiki) and the site configuration (groups Site and SiteAdmin) that exist in the default installation, are located in wikilib.d and not in wiki.d, and some recipes provide files located in a wikilib.d subdirectory in the cookbook directory. (You shouldn't delete the groups Site and SiteAdmin, required for normal function.)

How can I list all pages in a WikiGroup?

In a wiki page use `(:pagelist group=GroupName list=all:)` or in a search box type `GroupName/ list=all`.

Last modified by Petko on November 23, 2015. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiGroup

WikiGroups

toc top

Page redirects to WikiGroup

Last modified by simon on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiGroups

WikiPage

toc top

A WikiPage is simply the basic building block of a WikiWikiWeb that contains text and images. See WikiStructures and WikiWikiWeb for more information.

Wiki pages can have an edit template to predefine initial content, see Cookbook:Edit Templates

Wiki pages are stored in individual flat files, see Page File Format and Flat File Advantages.

Last modified by Petko on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiPage

WikiSandbox

toc top

Page redirects to Main.WikiSandbox

Last modified by Petko on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiSandbox

WikiStructure

toc top

Authors have a range of options to choose from when organizing a collection of wiki pages. Used in combination, these give a lot of flexibility. An effective wiki will use all of these to optimize

- content
- navigation

These are the two most important aspects of a website.

Wiki Word
: The most powerful organizing principle is the author's choice of page names. When a search returns a list of pages, their names need to be clear enough to guide a visitor to the right place.
Providing a network of links to other points in the wiki, with or without wiki words, is the primary means of navigating a wiki.

Wiki Page
: A page with text (and images), where the text can contain for instance WikiWords that automatically becomes a link to another WikiPage.

Wiki Group
: PmWiki requires every page to be a member of a group. A group is like a wiki within a wiki; it can have its own

presentation look, security controls and navigation aids. With default configuration, WikiWords are only searched inside the current group, and you use either `OtherGroup/MyWikiWord` or `OtherGroup.MyWikiWord` to refer to pages in other groups (see Links).

Wiki Trails
: A collection of pages, either in the same group or across multiple groups, can be designated as a trail. A visitor can move from stop to stop by clicking on *next* and *previous* links.

Categories
: Individual wiki pages can also be grouped by having tags and links to a common "category" page; we say that any pages that link to a common page are in a "category" defined by that page. PmWiki uses the `[[!category]]` markup as a shorthand to place a page into a category with other pages containing the same markup.
The shortcoming of categories is that categories do not distinguish between the declaration of a category ([[!structure]]) and the link to a category ([[Category/Structure]]).

Page text variables
: A newer and more powerful concept than Categories, pages can use one of more page text variables to store page attributes. These can the be used in page lists.

Page lists
: Page lists provide a powerful means of presenting lists of relevant pages, or selection of data from within a page. Lists are template based and are highly customizable.

Include other pages
: The capability to include parts of other pages also provides a flexible means of sharing content between pages.

Search
: Being able to search is a fundamental requirement of a website. In PmWiki search, like pagelists is both powerful and highly customizable.

Last modified by Petko on September 10, 2011. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiStructure

WikiStyleExamples

toc top

See also Wiki Styles Plus and Wiki style colors.

PmWiki uses WikiStyles for styling text with color and other attributes. PmWiki 2.0 introduced the ability to control the styling further and to even place styles on blocks.

A style is specified within a pair of %-signs and styles the text that follows, as in:

`This text is %color=red% red, %color=blue% blue, %% and normal (black).`	This text is red, blue, and normal (black).

There are a wide number of available style properties, borrowed primarily from HTML and CSS. In addition, an author can define a style "shortcut" by using the `define=` property. For example, to define a style of `%red%`, one can use:

`%define=mystyle color=red%`
`Here is some %mystyle% red text created using a style shortcut.`
	Here is some red text created using a style shortcut.

Shortcuts can be combined with other styles, including other shortcuts:

`%define=lovelyred color=red%`
`%define=likegrapefruit bgcolor=yellow%`
 	
`%red% This text is red, %red bgcolor=#ccc% red on a grey background, and %lovelyred likegrapefruit% red on a yellow background.`
	This text is red, red on a grey background, and red on a yellow background.

So far, this is all basically the same as what was available in PmWiki 1.0. PmWiki 2.0 includes the capability to style blocks, by using the `apply=` style property. Specifying `apply=block` in a WikiStyle will cause that style to be applied to the entire block, instead of just the text that follows:

```
This entire block %apply=block bgcolor=yellow% has a yellow background, even though the `WikiStyle
appears in the middle of the line.  %bgcolor=pink% Other inline (non-block) WikiStyles can appear in the
middle of the line,%% as before.
```

This entire block has a yellow background, even though the WikiStyle appears in the middle of the line. Other inline (non-block) WikiStyles can appear in the middle of the line, as before.

This means it's now possible to do right-aligned and centered text:

```
%block text-align=right% The text of this paragraph is right-aligned.

%block text-align=center% The text of this paragraph is centered.
```

The text of this paragraph is right-aligned.

The text of this paragraph is centered.

In fact, PmWiki predefines `%right%` and `%center%` style shortcuts so that you can do this more simply:

```
%right% This is right-aligned.

%center% This is centered.
```

This is right-aligned.

This is centered.

Authors can define their own custom styles:

```
%define=Pm block bgcolor=#fdf%
%define=goofy center bgcolor=#dfd border='3px dotted green'%
%define=rediguana right bgcolor=#ffffcc border='1px dotted red' padding=5px%
%define=strike text-decoration=line-through%

%Pm% Any text that is on a light purple background is a comment from "Pm".

%goofy% Here's some text from Goofy.

%rediguana% bla bla by rediguana!

%goofy%Hello, I am %strike%upset%% %strike%disheartened%% happy to meet you.
```

Any text that is on a light purple background is a comment from "Pm".

Here's some text from Goofy.

bla bla by rediguana!

Hello, I am ~~upset disheartened~~ happy to meet you.

Styles can be applied to almost any kind of block:

```
* %block bgcolor=yellow% Here is a list
item
* Here's another list item

* Here's more of a list

# A new list
```

- Here is a list item
- Here's another list item

- Here's more of a list

1. A new list

In particular, this means that outlines are now possible using the predefined `%ROMAN%`, `%roman%`, `%ALPHA%`, and `%alpha%` list-block styles. The style has to be specified on the first item in the list (and we may develop an alternate syntax for this sort of ordered list):

```
# %ROMAN% Top level
## %ALPHA% second-level
## second-level
## second-level
### third-level
### third-level
## second-level
### third-level
#### %alpha% fourth-level
##### %roman% fifth-level
##### fifth-level
#### fourth-level
# top-level
# top-level
```

```
I. Top level
    A. second-level
    B. second-level
    C. second-level
        1. third-level
        2. third-level
    D. second-level
        1. third-level
            a. fourth-level
                i. fifth-level
               ii. fifth-level
            b. fourth-level
II. top-level
III. top-level
```

Wiki styles can be combined with CSS stylesheets to do this automatically -- see Cookbook:OutlineLists.

Q & A

How do I get a block of preformatted text?

Use something similar to this (assuming you want markup within the block to be interpreted as wiki markup and URIs to be recognized).

```
>>white-space=pre<<
This block of text is ''preformatted'', see all   the   white-space
and   linebreaks
are preserved. Links such as [[wiki styles]] etc still work.
>><<
```

This block of text is *preformatted*, see all   the   white-space
and   linebreaks
are preserved. Links such as wiki styles etc still work.

How do I get a block of preformatted text with a colored background and a border?

Use something similar to this (note that wiki markup etc is not recognized within the block):

```
%block bgcolor=#f0f9ff border='1px solid gray' padding=5px%[@
ip access-list extended example-acl
remark ** This is an example acl **
deny ip any host 10.0.0.1
permit ip any any
@]
```

```
ip access-list extended example-acl
remark ** This is an example acl **
deny ip any host 10.0.0.1
permit ip any any
```

How do I get a block of text (including wiki markup) with a colored background and a border?

```
>>teal background-color:silver
border:'medium dotted green'<<
Hello world
* bullet
# number
>><<
```

Hello world
- bullet
1. number

How do I get a block of text (including wiki markup) with a border that is indented on the left and does NOT extend all the way to right? I'm not interested in having later text to the right as would occur with lfloat...

You can use the `indent width=50pct` wikistyle.

```
Before indention...
>>frame indent width=50pct<<
Hello world
* bullet
# number
>><<
... after indention!
```

Before indention...

Hello world
- bullet
1. number

... after indention!

Last modified by Petko on November 19, 2011.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiStyleExamples

toc top

WikiStyles

toc top

Contents

WikiStyle basics

WikiStyles allow authors to modify the color and other styling attributes of the contents of a page. A WikiStyle is written using percent-signs, as in `%red%` or `%bgcolor=lightblue%`.

WikiStyle attributes

The style attributes recognized within a WikiStyle specification are:

------------ CSS -------------		--HTML--
`bgcolor`	`height` *	`accesskey`
`background-color`	`list-style`	`align`
`border` [1]	`margin` [1]	`class`
`color`	`padding` [1]	`hspace`
`background-color`	`text-align`	`id`
`border`	`text-decoration`	`target`
`display`	`white-space`	`rel`
`float`	`width` *	`vspace`
`clear`		`value`
`font-size`		
`font-family`		
`font-weight`		
`font-style`		
`Special: define, apply`		

The attributes in the first two columns correspond to the *cascading style sheet* (CSS) properties of the same name. The attributes in the last column apply only to specific items:

- `class=` and `id=` assign a CSS class or identifier to an HTML element
- `target=name` opens links that follow in a browser window called "name"
- `rel=name` in a link identifies the relationship of a target page
- `accesskey=x` uses 'x' as a shortcut key for the link that follows
- `value=9` sets the number of the current ordered list item

  - The width and height attributes have asterisks because they are handled specially for <img .../> tags. If used by themselves (i.e., without anything providing an "apply=" parameter to the WikiStyle), then they set the 'width=' and 'height=' attributes of any <img ... /> tags that follow. Otherwise, they set the 'width:' and 'height:' properties of the element being styled.

  1. margin, padding, and border can be suffixed by -left, -right, -top, and -bottom

WikiStyles versus CSS styles

WikiStyles, as written in the wiki page, are not exactly CSS styles or CSS classes. WikiStyles allow authors to use both pre-defined by the administrator CSS classes, and to define new combinations of styles, without any need to edit/update local CSS files on the server.

Note that PmWiki allows advanced authors to use of`class=` and `style=` in tables and division blocks, but these are raw HTML attributes, and not WikiStyles, knowledge of CSS is required to use them.

Text color and font

The most basic use of WikiStyles is to change text attributes such as color, background color, and font. PmWiki defines several WikiStyles for changing the text color to %black%, %white%, %red%, %yellow%, %blue%, %gray% (%grey%), %silver%, %maroon%, %green%, %navy%, %fuchsia%, %olive%, %lime%, %teal%, %aqua%, %orange% and %purple%.

`The basket contains %red% apples, %blue% blueberries, %purple% eggplant, %green% limes, %% and more.`	The basket contains apples, blueberries, eggplant, limes, and more.

For colors other than the predefined colors, use the`%color=...%` WikiStyle. (Note: RGB colors (#rrggbb) should always be specified with lowercase letters to avoid WikiWord conflicts.)

`I'd like to have some %color=#ff7f00% tangerines%%, too!`	I'd like to have some tangerines, too!

To change the background color, use`%bgcolor=...%` as a WikiStyle:

`This sentence contains %bgcolor=green yellow% yellow text on a green background.`	This sentence contains yellow text on a green background.

See WikiStyle Colors for more color help.

Text justification

WikiStyles are used to control the text justification

`%center% This text is centered.`
 	
`%right% Right justified.`
	This text is centered.
	
	Right justified.

and to create floating text:

```
%rfloat% This text floats to the right

%rframe% floats to the right with a frame

Lorem ipsum dolor sit amet, consectetuer
sadipscing elitr
```

This text floats to the right

floats to the right with a frame

Lorem ipsum dolor sit amet, consectetuer sadipscing elitr

Scope

WikiStyles can also specify a *scope*; with no scope, the style is applied to any text that follows up to the next WikiStyle specification or the end of the paragraph, whichever comes first. The **apply=** attribute and its shortcuts allow to change the scope as follows:

apply attribute	shortcut	style applies to...
`%apply=img ...%`	-	all images that *follow* until another style applied
`%apply=p ...%`	`%p ...%`	the current paragraph
`%apply=pre ...%`	-	the current preformatted text
`%apply=list ...%`	`%list ...%`	the current list
`%apply=item ...%`	`%item ...%`	the current list item
`%apply=div ...%`	-	the current div
`%apply=block ...%`	`%block ...%`	to the current block, whether it's a paragraph, list, list item, heading, or division.

Thus, **`%p color=blue%`** is the same as **`%apply=p color=blue%`**, and **`%list ROMAN%`** is the same as **`%apply=list list-style=upper-roman%`**.

Some predefined style shortcuts also make use of apply, thus **`%right%`** is a shortcut for **`%text-align=right apply=block%`**.

Example: Apply a style to a paragraph:

```
%p bgcolor=#ffeeee% The WikiStyle specification at the beginning of this line applies to the entire
paragraph, even if there are %blue% other WikiStyle specifications %% in the middle of the paragraph.
```

The WikiStyle specification at the beginning of this line applies to the entire paragraph, even if there are other WikiStyle specifications in the middle of the paragraph.

Caveat: An applied WikiStyle will only take effect if it's on the line that starts the thing it's supposed to modify. In other words, a WikiStyle in the third markup line of a paragraph can't change the attributes of the paragraph:

```
after the first line of the paragraph,
we try to %apply=p color=blue% change
color.
This does't work because the style comes
after the first line of the paragraph.
```

after the first line of the paragraph, we try to change color. This does't work because the style comes after the first line of the paragraph.

```
However, this %apply=p color=red% paragraph
''will'' be in red because its block style
does
occur in the first line of its text.
```

However, this paragraph *will* be in red because its block style does occur in the first line of its text.

```
* Here's a list item
* %list red% Oops, too late to affect the
list!
```

- Here's a list item
- Oops, too late to affect the list!

If you want to break a list in two, you need to have a line not part of the list between, that is a line that has any content other than space and newlines, otherwise PmWiki considers the vertical space part of the previous list item. You can have an non-breaking space, or the escaped null character:

```
* %list red% first item
* second item
&nbsp;
* %apply=list bgcolor=lightgreen% second
list - first item
* second list - second item
[==]
```

- first item
- second item

- second list - first item
- second list - second item
- third list - first item
- third list - second item

```
* %list class=mambo% third list - first item
* third list - second item
```

Larger blocks

The `>>WikiStyle<<` block can be used to apply a WikiStyle to a large block of items. The style is applied until the next `>><<` is encountered.

```
>>blue font-style:italic bgcolor=#ffffcc<<
Everything after the above line is styled
with blue italic text,

This includes
    preformatted %red%text%%
* lists
-> indented items
>><<
```

Everything after the above line is styled with blue italic text,

This includes

```
    preformatted text
```

- *lists*

  indented items

Note, the `(:div style="..." class="...":)` directive does not work the same way as `>>WikiStyle<<`, it can only contain the regular HTML style and class attributes.

HTML "class" and "style" attributes for tables and divisions

WikiStyles are only the commands between `%...%` percent signs.

Tables, table directives and (:div:) division blocks allow advanced authors to incorporate the HTML/CSS attributes `class=` and `style=`. Note that these attributes are not WikiStyles, knowledge of CSS is required to use them.

```
(:table style="font-style:italic;
color:green; border:1px solid blue;
background-color:#ffffcc":)
(:cellnr:)
Everything after the above line is styled
with green italic text,

This includes
    preformatted text
* lists
-> indented items
(:tableend:)
```

Everything after the above line is styled with green italic text,

This includes

```
    preformatted text
```

- *lists*

  indented items

Note, the `(:div style="..." class="...":)` directive does not work the same way as `>>style<<`, as mentioned above, it can only contain the HTML style and class attributes.

Custom style shortcuts

The `define=` attribute can be used to assign a shorthand name to any WikiStyle specification. This shorthand name can then be reused in later WikiStyle specifications.

```
%define=box block bgcolor=#ddddff
border="2px dotted blue"%

%box% [@some sort of text@]

%box font-weight=bold color=green% [@some
sort of text@]
```

```
some sort of text
```

```
some sort of text
```

Tip: It's often a good idea to put common style definitions into Group Header pages so that they can be shared among multiple pages in a group. Or, the wiki administrator can predefine styles site-wide as a local customization (see Custom WikiStyles).

Tip: Use custom style definitions to associate meanings with text instead of just colors. For example, if warnings are to be displayed as green text, set `%define=warn green%` and then use `%warn%` instead of `%green%` in the document. Then, if you later decide that warnings should be styled differently, it's much easier to change the (one) definition than many occurrences of `%green%` in the text.

Tip: Any undefined WikiStyle is automatically treated as a request for a class, thus `%pre%` is the same as saying `%class=pre%`.

Predefined style shortcuts

PmWiki defines a number of style shortcuts.

- Text colors: black, white, red, yellow, blue, gray (grey), silver, maroon, green, navy, purple, fuchsia, olive, lime, teal, aqua, orange (shortcut for `%color=...%`)
- Justification: `%center%` and `%right%`
- Images and boxes
  - Floating left or right: `%rfloat%` and `%lfloat%`
  - Framed items: `%frame%`, `%rframe%`, and `%lframe%`
  - Thumbnail sizing: `%thumb%`
- Open link in new window: `%newwin%` (shortcut for `%target=_blank%`)
- Comments: `%comment%` (shortcut for `%display=none%`)
- Ordered lists: `%decimal%`, `%roman%`, `%ROMAN%`, `%alpha%`, `%ALPHA%` (see also Cookbook:OutlineLists)

Enabling Styles

Styles not listed above can be enabled by a PmWiki Administrator by modifying the local/config.php file. For instance to enable the "line-height" style attribute add the following line to the local/config.php file:

```
$WikiStyleCSS[] = 'line-height';
```

Defining scope for other HTML elements

You can add additional HTML elements to `$WikiStyleApply` to apply WikiStyles to other HTML elements. For example to allow styling on anchor tags:

```
$WikiStyleApply['link'] = 'a';
```

Examples

WikiStyle Examples contains a number of examples of ways to use WikiStyles in pages.

Known Issues

- Percents in style definitions (like: `%block width=50% %`) require the use of "pct" instead of "%". PmWiki will convert the "pct" into "%" so that it becomes valid CSS.
- If you specify multiple values for an attribute, like `border="2px solid blue"` make sure you place the values in quotes.
- Be sure to use lowercase letters for red-green-blue hex colors, `%color=#aa3333%` will work, `%color=#AA3333%` may not.

See Also

- Custom WikiStyles Predefined PmWiki styles & adding custom wiki styles
- PmWiki:List Styles
- WikiStylesPlus

Last modified by Peter Kay on April 06, 2017.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiStyles

toc top

WikiTrails

toc top

The WikiTrails feature allows wiki authors to create "trails" through sequences of pages in the wiki. You simply specify pages and their order on a "trail index", and then place the navigation markup on the pages that you will be navigating.

(Don't confuse the pagelist directive with WikiTrails - they are different animals as explained in the Q and A below.)

Table of contents

Trail types

PmWiki defines 2 trail markups, specifying a trail index link:

- `<<|[[Trail Index Page]]|>>` displays as "<< PreviousPage | Trail Index Page | NextPage >>".

- `<|[[Trail Index Page]]|>` displays as "< PreviousPage | Trail Index Page | NextPage >", except the appropriate arrow is omitted at the beginning and end of the trail.

and for a trail path:

- `^|[[TrailIndexPage]]|^`

Markup is most often added to a group header or group footer.

Trail index page link markup

The trail index page link has the same markup as a standard link, this means for example you can specify:

- `<|[[TrailIndexPage | +]]|>`
- `<<|[[TrailIndexPage | A description]]|>>`

Trail index page links can be restricted by anchors (links to a specific location within a page), this means you can have more than one trail on a page, or start a trail from a specific location in a page.

- `<|[[Trail Index Page(#trailstart#trailend)]]|>`

Creating a trail

Before you can use a trail through a set of pages, you have to create a "trail index" on a separate page, which we will call the "trail index page". On that trail index page, you simply create a numbered, bulleted, or definition list of links. (So every numbered or bulleted list of links implicitly creates a trail.)

It is important that each page name (link) be the first item following each bullet; any text or formatting in front of the page name link will exclude it from the trail.
If you want to format your trail (list), you can include a CSS.

An example trail index page might contain the list:

- Installation how to install
- The customisation page
- PmWiki some other text PmWiki Philosophy (The latter won't be in the trail because it is preceded by text)
- Yet some other text. PmWiki.WikiStyles (This won't be in the trail because it follows text)
- Uploads (This won't be in the trail because it is preceded by the %center% style.)

- Some text (This won't be in the trail because it is not a link)
- PageLists Listing pages by multiple criteria with templated output
- http://pmwiki.org (This won't be in the trail because it is not a page link)
  - PmWiki:InterMap (This won't be in the trail because it is an InterMap link)
- Cookbook:Cookbook (This won't be in the trail because it is an InterMap link)

PmWiki philosophy
    Design notes (The first link in this definition list will, and the second link won't, be in the trail defined by (definition list))

- Security (This won't be in the trail because its preceded by a (hidden) anchor)
- Links (This won't be in the trail because its preceded by a (hidden) %newwin% style)
- *Troubleshooting* (This won't be in the trail because its preceded by (hidden) *italic* style markup)

The list above creates the following "wikitrail", displayed using a pagelist:

```
(:pagelist trail={$FullName}#trailstart#trailend fmt={$FullName}#traillist:)
```

> Pm Wiki.Installation < > Pm Wiki.Local Customizations < > Pm Wiki.Pm Wiki < > Pm Wiki.Page Lists < > Pm Wiki.Pm Wiki Philosophy <

Observations

1. In general, indentation levels in the page list don't matter -- trails are a linear sequence of pages.
2. A page is part of the trail only if the page link immediately follows the list markup.
3. The list itself can be delineated by the use of anchors, allowing for multiple lists on a page, or for some list items to be excluded.

Using the trail

What makes a trail "work" is adding *trail markup* on the pages in the trail (i.e. the pages that are listed in the bullet/numbered list on the trail index page).

To build a trail, add *trail markup* like `<<|[[TrailIndexPage]]|>>` to a page, where TrailIndexPage is the page, described above, containing the bulleted list of pages in the trail. PmWiki will display the trail markup with links to any previous and next pages in the trail.

The trail markup can be placed anywhere in a group header or footer, or on a page. A page can contain multiple trail markups. If you are adding a trail to every page in a group, consider setting the trail markup in the Group Header or Group Footer pages instead of on every individual page in your group.

Path trail

`^|[[TrailIndexPage]]|^` treats the list levels as a hierarchy and displays the "path" to reach the current page (i.e., a "breadcrumb" trail). In the example trail above, the markup `^|TrailIndexPage|^` on `TrailPage4` would display as "TrailIndexPage | TrailPage2 | TrailPage4". and for a trail path

Wiki administrators can change the trail separator of the "path" trail (`^|[[TrailIndexPage]]|^`) from the default | by setting the variable `$TrailPathSep` in the *config.php* file. For instance `$TrailPathSep = ' > ';` will output "TrailIndexPage > TrailPage2 > TrailPage4".

Circular trails

Typically, a trail is a linear list with a first and a last page. However, the trail can be made "circular" by repeating the first page as the last item in the trail index:

```
* [[TrailPage1]]
* [[TrailPage2]]
...
* [[TrailPageN]]
* [[TrailPage1]]
```

If the trail index page is intended to be read by others, the last item can be made invisible inside an `(:if false:)` block:

```
* [[TrailPage1]]
* [[TrailPage2]]
...
* [[TrailPageN]]
(:if false:)
* [[TrailPage1]]
(:ifend:)
```

Cross Group Trails

Before version 2.2.1, if your trail contains pages in different groups, it should use full [[Group.Name]] links instead of just [[Name]].

Other notes

- There is no space between `<|` and `[[link]]` and `|>`; same for the other trail markups.
- Note that non-existing pages will appear in the WikiTrail as links.
- Conditional markup supports the ontrail query.
- Page lists provides the trail= parameter.

Trail style

PmWiki encapsulates the trail with a *`wikitrail`* css class. This allows the wiki trail to be customised by defining CSS for the *wikitrail* in the *`local.css`* file.

Trail in page lists

Trails from a single page can only be displayed using the pagelist trail parameter. For example

`(:pagelist trail=PmWiki/WikiTrails#trailstart fmt=PmWiki.WikiTrails#traillist order=random,$Name count=3:)`	> Pm Wiki.Pm Wiki Philosophy < > Pm Wiki.Installation < > Pm Wiki.Local Customizations <

A simple example of a WikiTrail

1) On the TrailIndexPage:

```
* [[MyTrailPage1]]
* [[MyTrailPage2]]
* [[MyTrailPage3]]
```

2) On the pages MyTrailPage1, 2, and 3:

```
<<|[[TrailIndexPage]]|>>
```

Questions

What's the difference between a PageList and a WikiTrail?

> The pagelist directive dynamically generates a list of pages. There are many ways to generate the list, including using a WikiTrail as the source. The pagelist directive then displays the pages that match the criteria using an optional template - for example displaying each page name on a separate line as a link or including the entire content. The pagelist directive currently does not have built-in navigation markup that you can put on the pages in the list. By contrast, WikiTrails are simply specified via links on an "index" page and you *can* put previous-next navigation markup on each page. The two serve very different purposes. WikiTrails are useful for specifying the pages in web feeds, for creating a "tour" through a predefined set of pages, and many other things.

Last modified by mfwolff on March 09, 2015. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiTrails

WikiWikiWeb

toc top

WikiWikiWeb is an "open-editing" system where the emphasis is on the *authoring* and *collaboration* of documents rather than the simple browsing or viewing of them. The name "wiki" is based on the Hawaiian term "wiki wiki", meaning "quick" or "super-fast".

The basic concept of a WikiWikiWeb (or "wiki") is that (almost) anyone can edit any page. While at first this sounds like a recipe for complete anarchy, the truth is that sites using this system have developed surprisingly complex and rich communities for online collaboration and communication. Yes, it's possible for someone to go and destroy everything on a page, but it doesn't seem to happen often. And, many systems (including this one) have built-in mechanisms to restore content that has been defaced or destroyed.

The point of the system is to simply make it as quick, easy *and* rewarding *as possible to create or edit online content.*

Using any standard Web browser, a person can edit (almost) any page on the system using relatively simple text formatting rules. Creating a link to a new or existing page simply involves putting the word or phrase that will be your link text inside [[double square brackets]] to reference and serve as a title for the target page. In the process of creating the link you're *creating the new page*, if it doesn't already exist. On some sites (depending on the configuration of PmWiki), a link can also be created by entering a WikiWord -- a word consisting of two or more capitalized words joined together.

It's not necessary to learn all of the formatting rules; others will often come in and reformat things for you. After all, anyone can edit! **You** can see some of the recent changes that others have posted to this site.

To learn more about adding pages to this Wiki site, see basic editing, then try editing pages in the WikiSandbox.

If you want to learn more about the WikiWikiWeb concept, try some of these Web sites:

- Wiki:WikiWikiWeb -- The original WikiWikiWeb
- Meatball:WhyWikiWorks -- how and why Wiki works
- Meatball:SoftSecurity -- how open editing can result in good Web sites
- Wiki on CommunityWiki
- WikiFeatures -- for info on features in wikis and how to use them
- Wikipedia:Wikipedia:Why_Wikipedia_is_so_great -- how and why the biggest wiki in the world made a comprehensive free-content encyclopedia
- Wikivoyage:Wiki -- another introduction to wikis, on another exemplary site

If you want to learn more about PmWiki see:

- Audiences Patrick Michaud's comments regarding the "audiences" for which PmWiki was designed
- DesignNotes Some of the features and notes about PmWiki's design decisions
- Documentation Index PmWiki documentation index
- PmWikiPhilosophy This page describes some of the ideas that guide the design and implementation of PmWiki
- Security Resources for securing your PmWiki installation

Or, send email to Patrick Michaud at pmichaud@pobox.com.

Last modified by Simon on June 19, 2017. toc top
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiWikiWeb

WikiWord

toc top

A **WikiWord** is a set of two or more words run together, where the first letter of each word is capitalized. This syntax is also sometimes referred to as "mixed case" or "camel case". Other descriptions of WikiWords are available from Wiki:WikiWord and Wikipedia:WikiWord.

Usage as page titles

WikiWords are used as **page titles** in a wiki-based system.

Usage as links

In some wikis (depending on the configuration of PmWiki), a valid **link** can be created by writing it as WikiWord. In such PmWiki installations, WikiWords surrounded by [=...=] or preceded by a backquote (`) are not turned into links:

```
LikeThis compared to `LikeThis or even [=LikeThis=]
```

LikeThis compared to LikeThis or even LikeThis

See Links for information about PmWiki's rules for forming links and forming page titles.

Enabling WikiWord links

WikiWord links are disabled by default since Pmwiki version 2.1 beta2. To enable WikiWord links you need to set in *config.php*

```
$EnableWikiWords = 1;
```

See also `$LinkWikiWords` and `$SpaceWikiWords`.

WikiWord links to non-existent pages without decoration

If you want to display links to non-existent pages without decoration, place the following lines in *pub/css/local.css*:

```
span.wikiword a.createlink { display:none; }
span.wikiword a.createlinktext { border-bottom:none; text-decoration:none; color:inherit; }
```

Finding WikiWord links

If you upgraded from an earlier version and want to convert WikiWord links to standard links, the following will help to find those WikiWord links easier by highlighting them. Set in *config.php*:

```
$HTMLStylesFmt['wikiword'] = "span.wikiword { background:yellow; }";
```

Disabling certain WikiWords links

The variable `$WikiWordCount` controls WikiWord conversion on a per word basis.

Last modified by Petko on October 18, 2016.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiWord

toc top

WikiWords

toc top

Page redirects to WikiWord

Last modified by simon on September 10, 2011.
Original URL: http://127.0.0.1:8080/pmwiki/pmwiki.php/PmWiki/WikiWords

toc top