From EULER Project

PmWiki: 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[]`.

PSS($string)

The PSS() function 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
> ```

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.

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(approve sites) 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.

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 a ???. 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 1Ff3w34HASH...") 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;
$DefaultPasswords['publish'] = crypt('secret');
include_once("$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)

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.

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 `ReadPage($pagename);` 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*.

`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.