

This page describes the skin template files (.tmpl) that are used to create PmWiki *skins*, and how PmWiki uses them. As described on 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 it is worth mentioning what those steps are:

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 it is 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.

1. When `$PageTemplateFmt` is blank (as it should be), PmWiki gathers the names of all candidate skins. It starts with any and [\\$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 use

- To attempt to find a skin, PmWiki first consults the `$SkinLibDirs` variable to know where to look. Skins consist of server-side files and images) that will be requested by the user's browser when they look at a skinned PmWiki page. `$SkinLibDirs` is an array where the corresponding value is a URL that points to the public client-side resources used by the skin. The default value of `$SkinLibDirs` is `./pub/skins/fo`.

So, using the above definitions, PmWiki would try to find the skin 'foo' by looking for a directory called `./pub/skins/fo` (the 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 look for a `skin.php` file in that directory.

- Once a valid skin directory has been found, PmWiki starts processing the files in that directory, looking for a `.php` skin file. 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 that 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:


```
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.

- 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 name `skin.tmpl` in the skin 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.

- 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.
- The directive `<!--HTMLHeader-->`, which goes somewhere in the `<head>` section of the HTML document.
- 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.

There is a long list of variables available for substitution in pages; some of the most useful include:

<code>\$PageUrl</code>	the url of the current page
<code>\$ScriptUrl</code>	the base url to the pmwiki.php script
<code>\$Title</code>	the page's title (e.g., "`SkinTemplates")
<code>\$Titlespaced</code>	the page's title with spaces (e.g., "Skin Templates")
<code>\$Group</code>	the name of the current group (e.g., "`PmWiki")
<code>\$FullName</code>	the page's full name (e.g., "`PmWiki.SkinTemplates")
<code>\$LastModified</code>	the page's last modification time
<code>\$PageLogoUrl</code>	the url of a site logo
<code>\$WikiTitle</code>	the site's title
<code>\$SkinDirUrl</code>	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.

If multiple pages are listed in the directive, then only the first available page is used. Thus `<!--wiki:$Group.SomePage Site.SomePage-->` doesn't. To always display `Site.SomePage`, even if `$Group.SomePage` exists, use two consecutive `<!--wiki: . . .-->` directives.

The `<!--wiki: . . .-->` directive only displays pages for which the browser has read permissions. The `<!--page: . . .-->` directive does not.

```
<!--file:somefile.txt-->
```

The directive `<!--file:somefile.txt-->` outputs the contents of another file (on the local filesystem) at the point of the directive. Like the `<!--wiki: . . .-->` directive above, `$`-substitutions are available to be able to output files based on the current page.

```
<!--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 and the text following the function name. It allows only one argument, which has to be splitted then. `<!--function:SomeFunction arg1 arg2 arg3-->` generates output.

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)");
```

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 [Cookbook: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](#).

Copyright: simon - 18/07/2010