Code Howto Create a Wiki Plugin

Creating a Wiki Plugin is a great way to get started contributing to Tiki as plugins can stand alone and minimum PHP programming skills are needed.

Wiki plugins extend the function of wiki syntax with more specialized commands. Usually expressed in {curly brackets} plugins compact a chunk of PHP or HTML code into something that can be understood by non-programmers.

Here are all the current Wiki plugins: https://gitlab.com/tikiwiki/tiki/-/tree/master/lib/wiki-plugins

Example

Let's use and example where we want to create a plugin to allow text formatting in any font and size:

```
{FONT(size=&gt;20,face=&gt;arial)} some text {FONT}
```

If a plugin doesn't require parameters, the syntax will be:

```
{EXAMPLE()} content {EXAMPLE}
```

When tiki finds a plugin the engine will look at the plugin name and look for the file:

```
lib/wiki-plugins/wikiplugin_name.php
```

Don't be too creative with plugin names. For instance, any number in a plugin names seems to make the plugin fail. A plugin name containing only alphabetical character will be valid.

For example:

```
lib/wiki-plugins/wikiplugin_font.php
```

That file should be a PHP file defining both functions:

```
function wikiplugin_font($data,$params) {}
```

and

```
function wikiplugin_font_info() {}
```

Info function

The info function defines the plugin's parameters and is used by the PLUGINMANAGER plugin for documenting an individual plugin, so it is important to populate this function as completely and accurately as possible. In our example there are only two parameters, these will be *face* and *size* but they each have

a number of 'settings'.
For example:

wikiplugin_font_info()

```
function wikiplugin_font_info()
{
        return array(
                &#039;name&#039; =&gt; tra(&#039;Font&#039;),
                &#039;documentation&#039; =&gt; &#039;PluginFont&#039;,
                &#039;description&#039; =&gt; tra(&#039;Create a an HTML (deprecated)
font tag.&#039;),
                &#039;tags&#039; =&gt; array( &#039;basic&#039; ),
                &#039;prefs&#039; =&gt; array( &#039;wikiplugin_font&#039; ),
                                &#039;introduced&#039; =&gt; 9,
                &#039;params&#039; =&gt; array(
                        &#039;size&#039; =&gt; array(
                                &#039;required&#039; =&gt; false,
                                &#039;name&#039; =&gt; tra(&#039;Font size&#039;),
                                &#039;description&#039; =&gt; tra(&#039;Choose font size
in pixels. Enter numbers only&#039;),
                                &#039;filter&#039; =&gt; &#039;digits&#039;,
                                &#039;default&#039; =&gt; &#039;12&#039;,
                                &#039;since&#039; =&gt; &#039;9.0&#039;,
                        ),
                        &#039;face&#039; =&gt; array(
                                &#039;required&#039; =&gt; false,
                                &#039;name&#039; =&gt; tra(&#039;Font face&#039;),
                                &#039;description&#039; =&gt; tra(&#039;Choose font
face.&#039;),
                                &#039;filter&#039; =&gt; &#039;alpha&#039;,
                                &#039;default&#039; =&gt; &#039;normal&#039;,
                                &#039;since&#039; =&gt; &#039;9.0&#039;,
                        ),
                ),
        );
}
```

Individual plugins may have parameters that need additional 'settings' but the settings above ie 'required', 'name', 'description', 'filter', 'default' (but only if 'required is false) and 'since', are the minimum number that should be set for every parameter.

Body function

The function receives the plugin content in $data and the parameters in the $params associative array. The function manipulates the content and must return a string with the HTML that will replace the plugin content when rendering the wiki page (it can be just text if no HTML markup is needed).

For example:

```
function wikiplugin_example($data,$params) {

extract($params, EXTR_SKIP);

if(!isset($face)) {
```

```
return (&quot;&lt;b&gt;missing face parameter for plugin&lt;/b&gt;&lt;br/&gt;&quot;);
}
if(!isset($size)) {
return (&quot;&lt;b&gt;missing size parameter for plugin&lt;/b&gt;&lt;br/&gt;&quot;);
}
$ret = &quot;&lt;span style=&#039;font-face: $face; font-size:
$size&#039;&gt;$data&lt;/span&gt;&quot;;
return $ret;
}
```

The return of your plugin will be wiki parsed. If you do not want it to be parsed again, you must enclose the value like this

```
return &#039; '.$ret.' &#039;;
```

Special Plugins

These aren't *real* plugins but, like plugins, use a slightly different curly braces syntax. Unlike plugins, you don't "close" them. e.g., use {maketoc} instead of {maketoc()}{maketoc}. These can also appear in template (.tpl) files.

PluginBanner	Displays a banner. {banner zone=}. Set up banners via admin menu. Note that if you assign multiple banners to the same zone, missing parameter will pick a different banner in that zone each time (more or less) the page is refreshed.
PluginContent	Displays dynamic content
PluginDraw	Displays a drawing which is an image that can be modified via a Java applet. {draw name=}. name must be unique for the entire site. Requires "drawing" feature to be turned on via the admin menu. If the drawing doesn't exist, the page will display a button that will create the drawing.
PluginImg	Displays an image {img src= height= width=}. Mimics the HTML img tag.
PluginMaketoc	Displays an indented table of contents using !, !!, !!! headers. {maketoc}.
PluginRSS	Displays an RSS feed. {rss id= max=}. Set up RSS feeds via admin menu (RSS Modules).
PluginTOC	Displays a table of contents for the structure that the page belongs to.

Related docs

How to add an item on the toolbar by default (once you add a plugin, you likely want it to be easy for people to use in text areas)

For notes on the future of plugins (and modules) see Gadgets.

Related links

- Wiki Plugins
- Hello World

- Create a Plugin