

Wiki Processors

Processors are [WikiMacros](#) designed to provide alternative markup formats for the [Wiki engine](#). Processors can be thought of as *macro functions to process user-edited text*.

Wiki processors can be used in any Wiki text throughout Trac, for various different purposes, like:

- [syntax highlighting](#) or for rendering text verbatim,
- rendering [Wiki markup inside a context](#), like inside `<div>` blocks or `` or within `<td>` or `<th>` table cells,
- using an alternative markup syntax, like [raw HTML](#) and [Restructured Text](#), or [textile](#)

Using Processors

To use a processor on a block of text, first delimit the lines using a Wiki *code block*:

```
{{{
The lines
that should be processed...
}}}
```

Immediately after the `{{{` or on the line just below, add `#!` followed by the *processor name*.

```
{{{
#!processorname
The lines
that should be processed...
}}}
```

This is the "shebang" notation, familiar to most UNIX users.

Besides their content, some Wiki processors can also accept *parameters*, which are then given as `key=value` pairs after the processor name, on the same line. If `value` has to contain space, as it's often the case for the `style` parameter, a quoted string can be used (`key="value with space"`).

As some processors are meant to process Wiki markup, it's quite possible to *nest* processor blocks. You may want to indent the content of nested blocks for increased clarity, this extra indentation will be ignored when processing the content.

Examples

Wiki Markup	Display
Example 1: Inserting raw HTML	
<pre>{{{ #!html <hl style="color: grey">This is raw HTML</hl> }}}</pre>	This is raw HTML
Example 2: Highlighted Python code in a <div> block with custom style	
<pre>{{{#!div style="background: #ffd; border: 3px ridge" This is an example of embedded "code" block: {{{ #!python def hello(): return 'world' }}} }}}</pre>	<div>This is an example of embedded "code" block:</div> <pre>def hello(): return 'world'</pre>
Example 3: Searching tickets from a wiki page, by keywords.	
<pre>{{{ #!html <form action="/query" method="get"><div> <input type="text" name="keywords" value="" size="30"/> <input type="submit" value="Search by Keywords"/> <!-- To control what fields show up use hidden fields <input type="hidden" name="col" value="id"/> <input type="hidden" name="col" value="summary"/> <input type="hidden" name="col" value="status"/> <input type="hidden" name="col" value="milestone"/> <input type="hidden" name="col" value="version"/> <input type="hidden" name="col" value="owner"/> <input type="hidden" name="col" value="priority"/> <input type="hidden" name="col" value="component"/> --> </div></form> }}}</pre>	

Available Processors

The following processors are included in the Trac distribution:

<code>#!default</code>	Present the text verbatim in a preformatted text block. This is the same as specifying <i>no</i> processor name (and no <code>#!</code>)
<code>#!comment</code>	Do not process the text in this section (i.e. contents exist only in the plain text - not in the rendered page).
HTML related	
<code>#!html</code>	Insert custom HTML in a wiki page.
<code>#!htmlcomment</code>	Insert an HTML comment in a wiki page (<i>since 0.12</i>).
	Note that <code>#!html</code> blocks have to be <i>self-contained</i> , i.e. you can't start an HTML element in one block and close it later in a second block. Use the following processors for achieving a similar effect.
<code>#!div</code>	Wrap an arbitrary Wiki content inside a <code><div></code> element (<i>since 0.11</i>).
<code>#!span</code>	Wrap an arbitrary Wiki content inside a <code></code> element (<i>since 0.11</i>).
<code>#!td</code>	Wrap an arbitrary Wiki content inside a <code><td></code> element (<i>since 0.12</i>)
<code>#!th</code>	Wrap an arbitrary Wiki content inside a <code><th></code> element (<i>since 0.12</i>)
<code>#!tr</code>	Can optionally be used for wrapping <code>#!td</code> and <code>#!th</code> blocks, either for specifying row attributes of better visual grouping (<i>since 0.12</i>)
	See WikiHtml for example usage and more details about these processors.
Other Markups	
<code>#!rst</code>	Trac support for Restructured Text. See WikiRestructuredText .
<code>#!textile</code>	Supported if Textile is installed. See Textile reference .
Code Highlighting Support	
<code>#!c</code> <code>#!cpp (C++)</code> <code>#!python</code> <code>#!perl</code> <code>#!ruby</code> <code>#!php</code> <code>#!asp</code> <code>#!java</code> <code>#!js (Javascript)</code> <code>#!sql</code> <code>#!xml (XML or HTML)</code> <code>#!sh (Bourne/Bash shell)</code> <code>etc.</code>	<p>Trac includes processors to provide inline syntax highlighting for source code in various languages.</p> <p>Trac relies on external software packages for syntax coloring, like Pygments.</p> <p>See TracSyntaxColoring for information about which languages are supported and how to enable support for more languages.</p>
MIME Type Processors	
Using the MIME type as processor, it is possible to syntax-highlight the same languages that are supported when browsing source code.	
Some examples:	The result will be syntax highlighted HTML code:
<pre> {{{ #!text/html <h1>text</h1> }}} </pre>	<pre> <h1>text</h1> </pre> <p>The same is valid for all other mime types supported.</p>
	<code>#!diff</code> has a particularly nice renderer:
	Version
<pre> {{{ #!diff --- Version 55 +++ Version 56 @@ -115,8 +115,9 @@ name='TracHelloWorld', version='1.0', packages=find_packages(exclude=['*.tests*']), - entry_points = "" - [trac.plugins] - helloworld = myplugins.helloworld - """ + entry_points = { + 'trac.plugins': [+ 'helloworld = myplugins.helloworld', +], + } +) }}} </pre>	<pre> 115 name='TracHelloWorld', 115 version='1.0', 116 packages=find_packages(exclude=["*.tests*"]), 117 entry_points = "" 118 [trac.plugins] 119 helloworld = 120 myplugins.helloworld 117 """ 118 entry_points = { 119 'trac.plugins': [120 'helloworld = 121 myplugins.helloworld', 122], 121 }, 122) </pre>

For more processor macros developed and/or contributed by users, visit:

- [■ProcessorBazaar](#)
- [■MacroBazaar](#)
- [■Trac Hacks](#) community site

Developing processors is no different from Wiki macros. In fact they work the same way, only the usage syntax differs. See [WikiMacros#DevelopingCustomMacros](#) for more information.

See also: [WikiMacros](#), [WikiHtml](#), [WikiRestructuredText](#), [TracSyntaxColoring](#), [WikiFormatting](#), [TracGuide](#)