

Wikiprint Book

Title: Wiki Processors

Subject: TechTIDE-Wiki - WikiProcessors

Version: 1

Date: 07/15/25 01:04:29

Table of Contents

| | |
|------------------------|----------|
| Wiki Processors | 3 |
| Using Processors | 3 |
| Examples | 3 |
| Available Processors | 4 |

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, such as:

- [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 and 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 |
|---|--|
| <u>Example 1:</u> Inserting raw HTML | |
| <pre>{{{ #!html <h1 style="color: grey">This is raw HTML</h1> }}}</pre> | This is raw HTML |
| <u>Example 2:</u> Highlighted Python code in a <code><div></code> 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 style="background: #ffd; border: 3px ridge;"> <p>This is an example of embedded "code" block:</p> <pre>def hello(): return "world"</pre> </div> |
| <u>Example 3:</u> 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. |
| <code>#!rtl</code> | Introduce a Right-To-Left block with appropriate CSS direction and styling. (<i>since 0.12.2</i>) |
| 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 wiki content inside a <code><div></code> element. |
| <code>#!span</code> | Wrap wiki content inside a <code></code> element. |
| <code>#!td</code> | Wrap wiki content inside a <code><td></code> element. (<i>since 0.12</i>) |
| <code>#!th</code> | Wrap 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 or better visual grouping. (<i>since 0.12</i>) |
| <code>#!table</code> | Can optionally be used for wrapping <code>#!tr</code> , <code>#!td</code> and <code>#!th</code> blocks, for specifying table attributes. One current limitation however is that tables cannot be nested. (<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 a 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 Pygments for syntax coloring.</p> <p>See TracSyntaxColoring for information about which languages are supported and how to enable support for more languages.</p> |

Since 1.1.2 the default, coding highlighting and MIME-type processors support the argument `lineno` for adding line numbering to the code block. When a value is specified, as in `lineno=3`, the numbering will start at the specified value. When used in combination with the `lineno` argument, the marks

argument is also supported for highlighting lines. A single line number, set of line numbers and range of line numbers are allowed. For example, `marks=3`, `marks=3-6`, `marks=3,5,7` and `marks=3-5,7` are all allowed. The specified values are relative to the numbered lines, so if `lineno=2` is specified to start the line numbering at 2, `marks=2` will result in the first line being highlighted.

Using the MIME type as processor, it is possible to syntax-highlight the same languages that are supported when browsing source code.

| MIME Type Processors | | | |
|--|--|--|--|
| 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> | |
| | | #!diff has a particularly nice renderer: | |
| <u>Version</u> | | | |
| <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 = myplugs.helloworld - """, + entry_points = { + 'trac.plugins': [+ 'helloworld = myplugs.helloworld', +], + }, }}}</pre> | | <pre>115 115 name='TracHelloWorld', version='1.0', 116 116 packages=find_packages(exclude= 117 entry_points = """ 118 [trac.plugins] 119 helloworld = myplugs.helloworld 120 """, 117 117 entry_points = { 118 118 'trac.plugins': [119 119 'helloworld = myplugs.helloworld', 120 120], 121 121 }, 121 122)</pre> | |

Line numbers can be added to code blocks and lines can be highlighted (since 1.1.2).

```
{{{#!python lineno=3 marks=3,9-10,16
def expand_markup(stream, ctxt=None):
    """A Genshi stream filter for expanding `genshi.Markup` events.

    Note: Expansion may not be possible if the fragment is badly
    formed, or partial.
    """

    for event in stream:
        if isinstance(event[1], Markup):
            try:
                for subevent in HTML(event[1]):
                    yield subevent
            except ParseError:
                yield event
        else:
            yield event
}}}
```

For more processor macros developed and/or contributed by users, visit the [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](#)