

# 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 `<span>` 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

```
{{{
#!html
<h1 style="color: grey">This is raw HTML</h1>
}}}
```

**This is raw HTML**

Example 2: Highlighted Python code in a `<div>` block with custom style

```
{{{#!div style="background: #ffd; border: 3px ridge"
This is an example of embedded "code" block:
```

This is an example of embedded "code" block:

```
{{{
#!python
```

```
def hello():
    return "world"
```

This is raw HTML

```

def hello():
    return "world"
}}}
}}}

```

Example 3: Searching tickets from a wiki page, by keywords.

```

{{{
#!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>
}}}}

```

## Available Processors

The following processors are included in the Trac distribution:

### #!default

Present the text verbatim in a preformatted text block. This is the same as specifying *no* processor name (and no #!)

### #!comment

Do not process the text in this section (i.e. contents exist only in the plain text - not in the rendered page).

### HTML related

#### #!html

Insert custom HTML in a wiki page.

#### #!htmlcomment

Insert an HTML comment in a wiki page (*since 0.12*).

Note that `#!html` blocks have to be *self-contained*, 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.

#### #!div

Wrap an arbitrary Wiki content inside a `<div>` element (*since 0.11*).

#### #!span

Wrap an arbitrary Wiki content inside a `<span>` element (*since 0.11*).

#### #!td

Wrap an arbitrary Wiki content inside a `<td>` element (*since 0.12*)

#### #!th

Wrap an arbitrary Wiki content inside a `<th>` element (*since 0.12*)

#### #!tr

Can optionally be used for wrapping `#!td` and `#!th` blocks, either for specifying row attributes of better visual grouping (*since 0.12*)

See [WikiHtml](#) for example usage and more details about these processors.

### Other Markups

`#!rst`

Trac support for Restructured Text. See [WikiRestructuredText](#).

`#!textile`

Supported if [?Textile](#) is installed. See [?a Textile reference](#).

### Code Highlighting Support

`#!c`

`#!cpp (C++)`

`#!python`

`#!perl`

`#!ruby`

`#!php`

`#!asp`

`#!java`

`#!js (Javascript)`

`#!sql`

`#!xml (XML or HTML)`

`#!sh (Bourne/Bash shell)`

`etc.`

Trac includes processors to provide inline syntax highlighting for source code in various languages.

Trac relies on external software packages for syntax coloring, like [?Pygments](#).

See [TracSyntaxColoring](#) for information about which languages are supported and how to enable support for more languages.

### 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:

```
{{{
#!text/html
<h1>text</h1>
}}}
```

The result will be syntax highlighted HTML code:

```
<h1>text</h1>
```

The same is valid for all other [mime types supported](#).

`#!diff` has a particularly nice renderer:

```
{{{
#!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',
+        ],
+    },
+    ),
}}}
```

## • Version

```
115 115     name='TracHelloWorld', version='1.0',
```

```
116 116     packages=find_packages(exclude=['*.tests*']
```

```
117         entry_points = ""
```

```
118             [trac.plugins]
```

```
119                 helloworld = myplugins.helloworld
```

```
120             "",
```

```
117     entry_points = {
```

```
118         'trac.plugins': [
```

```
119             'helloworld = myplugins.helloworld',
```

```
120         ],
```

```
121     },
```

```
121 122 )
```

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](#)