Wikiprint Book

Title: Trac Macros

Subject: YAM - Yet Another Mailer - WikiMacros

Version: 12

Date: 12/06/2016 04:02:31 AM

Table of Contents

Trac Macros

Trac macros are plugins to extend the Trac engine with custom 'functions' written in Python. A macro inserts dynamic HTML data in any context supporting WikiFormatting.

Another kind of macros are WikiProcessors. They typically deal with alternate markup formats and representation of larger blocks of information (like source code highlighting).

Using Macros

Macro calls are enclosed in two *square brackets*. Like Python functions, macros can also have arguments, a comma separated list within parentheses.

Getting Detailed Help

The list of available macros and the full help can be obtained using the MacroList macro, as seen below.

A brief list can be obtained via `[[MacroList(*)]]` or `[[?]]`.

Detailed help on a specific macro can be obtained by passing it as an argument to MacroList, e.g. `[[MacroList(MacroList)]]`, or, more conveniently, by appending a question mark (?) to the macro's name, like in `[[MacroList?]]`.

Example

A list of 3 most recently changed wiki pages starting with 'Trac':

Wiki Markup	Display
`[[RecentChanges(Trac,3)]]`	**Nov 7, 2016** • TracWiki (diff) • TracInstall (diff) • TracModWSGI (diff)

`[[RecentChanges?(Trac,3)]]`	**`[[RecentChanges]]`**
	
	List all pages that have recently been modified, ordered by the time they were last modified.
	
	This macro accepts two ordered arguments and a named argument. The named argument can be placed in any position within the argument list.
	
	The first parameter is a prefix string: if provided, only pages with names that start with the prefix are included in the resulting list. If this parameter is omitted, all pages are included in the list.
	
	The second parameter is the maximum number of pages to include in the list.
	
	The `group` parameter determines how the list is presented:
	
	`group=date`
	The pages are presented in bulleted lists that are grouped by date (default).
	`group=none`
	The pages are presented in a single bulleted list.
	
	Tip: if you only want to specify a maximum number of entries and don't want to filter by prefix, specify an empty first parameter, e.g. `[[RecentChanges(,10,group=none)]]`.

`[[?]]`	**`[[Image]]`** Embed an image in wiki-formatted text. The first argument is the file ? **`[[InterTrac]]`** Provide a list of known InterTrac prefixes. **`[[InterWiki]]`** Provide a description list for the known InterWiki prefixes. **`[[KnownMimeTypes]]`** List all known mime-types which can be used as WikiProcessors. Can be ? etc.

Available Macros

Note that the following list will only contain the macro documentation if you've not enabled `-OO` optimizations, or not set the `PythonOptimize` option for mod_python.

`[[BlogList]]`

A macro to display list of posts and extracts outside (or inside) the Blog module - most commonly Wiki pages.

All arguments are optional:

```
[[BlogList]]
```

Available named arguments:

- `recent=` - max. number of posts
- `category=` - a category
- `author=` - an author
- `period=` - time period of the format YYYY/MM
- `heading=` - a heading for the list
- `format=` - type of display (see below for details)
- `max_size=` - max. number of characters to render for each post
- `meta=` - use `=off` to hide date, author and categories (default 'on')

Example showing some available named arguments:

```
[[BlogList(recent=5, max_size=250, period=2007/12, author=osimons, format=float, heading=Some Trac Posts)]]
```

The arguments for criteria are 'AND'-based, so the above example will render at most 5 posts by 'osimons' in December 2007.

There is no heading unless specified.

Without restriction on recent number of posts, it will use the number currently active in the Blog module as default for 'float' and 'full' rendering, but for rendering of 'inline' list it will render all found as default unless restricted. Additionally for 'float' and 'full' it will truncate content if it is larger than a max_size (if set).

The `format=` keyword argument supports rendering these formats:

`format=inline`	Renders an unordered list in the normal text flow (default).
`format=float`	A floating box out on the side of the page with slightly more detail.
`format=full`	Full rendering like on period, category and author listings inside blog.

The arguments can appear in any order.

Posts are rendered sorted by newest first for all modes.

`[[ChangeLog]]`

Write repository change log to output.

The ChangeLog macro writes a log of the last changes of a repository at a given path. Following variants are possible to use:

```
1. [[ChangeLog([reponame:]path)]]
2. [[ChangeLog([reponame:]path@rev)]]
3. [[ChangeLog([reponame:]path@rev, limit)]]
4. [[ChangeLog([reponame:]path@from-to)]]
5. [[ChangeLog([reponame:]path, limit, rev)]]
```

i. Default repository is used if reponame is left out. To show the last five changes of the default repository:

```
[[ChangeLog(/)]]
```

To show the last five changes of the trunk folder in a named repo:

```
[[ChangeLog(otherrepo:/trunk)]]
```

ii. The ending revision can be set. To show the last five changes up to revision 99:

```
[[ChangeLog(otherrepo:/trunk@99)]]
```

iii. The limit can be set by an optional parameter. To show the last 10 changes, up to revision 99:

```
[[ChangeLog(otherrepo:/trunk@99, 10)]]
```

iv. A range of revisions can be logged.

```
[[ChangeLog(otherrepo:/trunk@90-99)]]
```

To lists all changes:

```
[[ChangeLog(otherrepo:/trunk@1-HEAD)]]
```

HEAD can be left out:

```
[[ChangeLog(otherrepo:/trunk@1-)]]
```

v. For backwards compatibility, revision can be stated as a third parameter:

```
[[ChangeLog(otherrepo:/trunk, 10, 99)]]
```

limit and rev may be keyword arguments.

```
[[ChangeLog(otherrepo:/trunk, limit=10, rev=99)]]
```

[[Embed]]

Macro produces html code for embedding flash content from certain service, by it's 'key' and content id. It also can embed simple SWF by it's URL.

Syntax and examples:

```
 [[Embed(youtube=emYqURahUKI)]]
[[Embed(vimeo=3840952,w=400,h=300)]]
[[Embed(swf=http://media.nadprof.org/flash/rudy/flowers/flowers.swf,w=500,h=400)]]
[[Embed(flv=video.flv,purl=/jwplayer.swf,w=500,h=400)]]
```

Available keys:

- *youtube*: video from YouTube ?http://youtube.com
- *vimeo*: video from Vimeo ?http://vimeo.com
- *swf*: SWF by URL
- *flv*: insert JW player with flv file (?http://www.longtailvideo.com/players/jw-flv-player/)

Optional parameters:

- *w* and *h*: width and height of embedded flash object

`[[Image]]`

Embed an image in wiki-formatted text.

The first argument is the file specification. The file specification may reference attachments in three ways:

- `module:id:file`, where module can be either **wiki** or **ticket**, to refer to the attachment named *file* of the specified wiki page or ticket.
- `id:file`: same as above, but id is either a ticket shorthand or a Wiki page name.
- `file` to refer to a local attachment named 'file'. This only works from within that wiki page or a ticket.

Also, the file specification may refer to repository files, using the `source:file` syntax (`source:file@rev` works also).

Files can also be accessed with a direct URLs; `/file` for a project-relative, `//file` for a server-relative, or `http://server/file` for absolute location of the file. The ?rfc2397 `data` URL scheme is also supported if the URL is enclosed in quotes.

The remaining arguments are optional and allow configuring the attributes and style of the rendered `<img>` element:

- digits and unit are interpreted as the size (ex. 120px, 25%) for the image
- `right`, `left`, `center`, `top`, `bottom` and `middle` are interpreted as the alignment for the image (alternatively, the first three can be specified using `align=...` and the last three using `valign=...`)
- `link=some TracLinks...` replaces the link to the image source by the one specified using a TracLinks. If no value is specified, the link is simply removed.
- `inline` specifies that the content generated be an inline XHTML element. By default, inline content is not generated, therefore images won't be rendered in section headings and other one-line content.
- `nolink` means without link to image source (deprecated, use `link=`)

  `key=value` style are interpreted as HTML attributes or CSS style indications for the image. Valid keys are:
  - align, valign, border, width, height, alt, title, longdesc, class, margin, margin-(left,right,top,bottom), id and usemap
  - `border`, `margin`, and `margin-*` can only be a single number (units are pixels).
  - `margin` is superseded by `center` which uses auto margins

Examples:

```
[[Image(photo.jpg)]]               # simplest
[[Image(photo.jpg, 120px)]]        # with image width size
[[Image(photo.jpg, right)]]        # aligned by keyword
[[Image(photo.jpg, nolink)]]       # without link to source
[[Image(photo.jpg, align=right)]]  # aligned by attribute
```

You can use an image from a wiki page, ticket or other module.

```
[[Image(OtherPage:foo.bmp)]]    # from a wiki page
[[Image(base/sub:bar.bmp)]]     # from hierarchical wiki page
[[Image(#3:baz.bmp)]]           # from another ticket
[[Image(ticket:36:boo.jpg)]]    # from another ticket (long form)
[[Image(source:/img/bee.jpg)]]  # from the repository
[[Image(htdocs:foo/bar.png)]]   # from project htdocs dir
[[Image(shared:foo/bar.png)]]   # from shared htdocs dir (since 1.0.2)
```

Adapted from the Image.py macro created by Shun-ichi Goto <gotoh@?>

`[[Include]]`

A macro to include other resources in wiki pages. More documentation to follow.

`[[InterTrac]]`

Provide a list of known InterTrac prefixes.

`[[InterWiki]]`

Provide a description list for the known InterWiki prefixes.

`[[KnownMimeTypes]]`

List all known mime-types which can be used as WikiProcessors.

Can be given an optional argument which is interpreted as mime-type filter.

`[[LastVoted]]`

Show most recently voted resources.

`[[ListTagged]]`

List tagged resources.

Usage:

```
[[ListTagged(<query>[[,format=<format>],cols=<columns>])]]
```

format

result list presentation; supported values:

`compact`	comma-separated inline list of "linked-description"
`oldlist` (default)	" * linked-id description (tags)" list
`table`	table... (see corresponding column option too)
`short` or other value	bulleted list of "linked-description"

cols

columns for 'table' format using a "|"-separated list of column names (order matters); supported columns: realm, id, description, tags

See tags documentation for the query syntax.

`[[MacroList]]`

Display a list of all installed Wiki macros, including documentation if available.

Optionally, the name of a specific macro can be provided as an argument. In that case, only the documentation for that macro will be rendered.

Note that this macro will not be able to display the documentation of macros if the `PythonOptimize` option is enabled for mod_python!

`[[PageOutline]]`

Display a structural outline of the current wiki page, each item in the outline being a link to the corresponding heading.

This macro accepts four optional parameters:

- The first is a number or range that allows configuring the minimum and maximum level of headings that should be included in the outline. For example, specifying "1" here will result in only the top-level headings being included in the outline. Specifying "2-3" will make the outline include all headings of level 2 and 3, as a nested list. The default is to include all heading levels.
- The second parameter can be used to specify a custom title (the default is no title).
- The third parameter selects the style of the outline. This can be either `inline` or `pullout` (the latter being the default). The `inline` style renders the outline as normal part of the content, while `pullout` causes the outline to be rendered in a box that is by default floated to the right side of the other content.
- The fourth parameter specifies whether the outline is numbered or not. It can be either `numbered` or `unnumbered` (the former being the default). This parameter only has an effect in `inline` style.

`[[RecentChanges]]`

List all pages that have recently been modified, ordered by the time they were last modified.

This macro accepts two ordered arguments and a named argument. The named argument can be placed in any position within the argument list.

The first parameter is a prefix string: if provided, only pages with names that start with the prefix are included in the resulting list. If this parameter is omitted, all pages are included in the list.

The second parameter is the maximum number of pages to include in the list.

The `group` parameter determines how the list is presented:

`group=date`
: The pages are presented in bulleted lists that are grouped by date (default).

`group=none`
: The pages are presented in a single bulleted list.

Tip: if you only want to specify a maximum number of entries and don't want to filter by prefix, specify an empty first parameter, e.g. `[[RecentChanges(,10,group=none)]]`.

`[[RecentTopics]]`

Lists all topics, that have been recently active, grouping them by the day they were lastly active. Accepts two parameters: First one is a forum ID. If provided, only topics in that forum are included in the resulting list. Otherwise topics from all forums are listed. Second parameter is a number. I. e. specifying 5 will result in only the five most recently active topics to be included in the list.

`[[Screenshot]]`

Allows embed screenshot image in wiki page. First mandatory argument is ID of the screenshot. Number or image attributes can be specified next:

- `align` - Specifies image alignment in wiki page. Possible values are: `left`, `right` and `center`.
- `alt` - Alternative description of image.
- `border` - Sets image border of specified width in pixels.
- `class` - Class of image for CSS styling.
- `description` - Brief description under the image. Accepts several variables (see bellow).
- `format` - Format of returned image or screenshot behind link.
- `height` - Height of image. Set to 0 if you want original image height.
- `id` - ID of image for CSS styling.
- `longdesc` - Detailed description of image.
- `title` - Title of image.
- `usemap` - Image map for clickable images.
- `width` - Width of image. Set to 0 if you want original image width.

Attribute `description` displays several variables:

- `$id` - ID of image.
- `$name` - Name of image.
- `$author` - User name who uploaded image.
- `$time` - Time when image was uploaded.
- `$file` - File name of image.
- `$description` - Detailed description of image.
- `$width` - Original width of image.
- `$height` - Original height of image.
- `$tags` - Comma separated list of screenshot tags.
- `$components` - Comma separated list of screenshot components.
- `$versions` - Comma separated list of screenshot versions.

Example:

```
[[Screenshot(2,width=400,height=300,description=The $name by $author: $description,align=left)]]
```

`[[ScreenshotsList]]`

Displays list of all available screenshots on wiki page. Accepts one argument which is template for list items formatting. Possible variables in this template are:

- `$id` - ID of image.
- `$name` - Name of image.
- `$author` - User name who uploaded image.
- `$time` - Time when image was uploaded.
- `$file` - File name of image.
- `$description` - Detailed description of image.
- `$width` - Original width of image.
- `$height` - Original height of image.
- `$tags` - Comma separated list of screenshot tags.
- `$components` - Comma separated list of screenshot components.
- `$versions` - Comma separated list of screenshot versions.

Example:

```
 [[ScreenshotsList($name - $description ($widthx$height))]]
```

`[[TOC]]`

Generate a table of contents for the current page or a set of pages.

If no arguments are given, a table of contents is generated for the current page, with the top-level title stripped:

```
[[TOC]]
```

To generate a table of contents for a set of pages, simply pass them as comma separated arguments to the TOC macro, e.g. as in

```
[[TOC(TracGuide, TracInstall, TracUpgrade, TracIni, TracAdmin, TracBackup,
     TracLogging, TracPermissions, TracWiki, WikiFormatting, TracBrowser,
     TracRoadmap, TracChangeset, TracTickets, TracReports, TracQuery,
     TracTimeline, TracRss, TracNotification)]]
```

A wildcard `*` can be used to fetch a sorted list of all pages starting with the preceding pagename stub:

```
[[TOC(Trac*, WikiFormatting, WikiMacros)]]
```

The following *control* arguments change the default behaviour of the TOC macro:

Argument	Description
`heading=<x>`	Override the default heading of "Table of Contents"
`noheading`	Suppress display of the heading.
`depth=<n>`	Display headings of *subsequent* pages to a maximum depth of **<n>**.
`inline`	Display TOC inline rather than as a side-bar.
`sectionindex`	Only display the page name and title of each page in the wiki section.
`titleindex`	Only display the page name and title of each page, similar to TitleIndex.
`notitle`	Supress display of page title.
`reverse`	Display TOC sorted in reversed order. *(Since 11.0.0.4)*

For `titleindex` argument, an empty pagelist will evaluate to all pages:

```
[[TOC(titleindex, notitle, heading=All pages)]]
```

The `sectionindex` argument allows a title index to be generated for all pages in a given section of the wiki. A section is defined by wiki page name, using `/` as a section level delimiter (like directories in a file system). Giving `/` or `*` as the page name produces the same result as `titleindex` (title of all pages). If a page name ends with a `/`, only children of this page will be processed. Otherwise, the page given in the argument is also included, if it exists. For `sectionindex` argument, an empty pagelist will evaluate to all page below the same parent as the current page:

```
[[TOC(sectionindex, notitle, heading=This section pages)]]
```

`[[TagCloud]]`

Display a tag cloud.

Show a tag cloud for all tags on resources matching query.

Usage:

```
[[TagCloud(<query>[,caseless_sort=<bool>][,mincount=<n>])]]
```

caseless_sort
: Whether the tag cloud should be sorted case-sensitive.

mincount
: Optional integer threshold to hide tags with smaller count.

See tags documentation for the query syntax.

`[[TitleIndex]]`

Insert an alphabetic list of all wiki pages into the output.

Accepts a prefix string as parameter: if provided, only pages with names that start with the prefix are included in the resulting list. If this parameter is omitted, all pages are listed. If the prefix is specified, a second argument of value `hideprefix` can be given as well, in order to remove that prefix from the output.

Alternate `format` and `depth` named parameters can be specified:

- `format=compact`: The pages are displayed as comma-separated links.
- `format=group`: The list of pages will be structured in groups according to common prefix. This format also supports a `min=n` argument, where `n` is the minimal number of pages for a group.
- `format=hierarchy`: The list of pages will be structured according to the page name path hierarchy. This format also supports a `min=n` argument, where higher `n` flatten the display hierarchy
- `depth=n`: limit the depth of the pages to list. If set to 0, only toplevel pages will be shown, if set to 1, only immediate children pages will be shown, etc. If not set, or set to -1, all pages in the hierarchy will be shown.
- `include=page1:page*2`: include only pages that match an item in the colon-separated list of pages. If the list is empty, or if no `include` argument is given, include all pages.
- `exclude=page1:page*2`: exclude pages that match an item in the colon- separated list of pages.

The `include` and `exclude` lists accept shell-style patterns.

`[[TopVoted]]`

Show listing of voted resources ordered by total score.

`[[TracAdminHelp]]`

Display help for trac-admin commands.

Examples:

```
[[TracAdminHelp]]               # all commands
[[TracAdminHelp(wiki)]]         # all wiki commands
[[TracAdminHelp(wiki export)]]  # the "wiki export" command
[[TracAdminHelp(upgrade)]]      # the upgrade command
```

`[[TracGuideToc]]`

Display a table of content for the Trac guide.

This macro shows a quick and dirty way to make a table-of-contents for the Help/Guide. The table of contents will contain the Trac* and WikiFormatting pages, and can't be customized. See the ?TocMacro for a more customizable table of contents.

`[[TracIni]]`

Produce documentation for the Trac configuration file.

Typically, this will be used in the TracIni page. Optional arguments are a configuration section filter, and a configuration option name filter: only the configuration options whose section and name start with the filters are output.

`[[TranslatedPages]]`

Macro to show the translated pages list.

Simply calling that macro in a page adds a menu linking to all available translations of a page.

A language page (usually TracLanguages) must provide the language codes as a table with following entries:

```
||<language code>||<language name>||<english name>||<description>||
```

The description contains the text displayed above language links in that language (usually a variant of 'Other languages'). A table title line starting with `||=` is not parsed.

The Macro accepts arguments as well:

- **revision=<num>** to specify the version of the base page when last translated, a negative revision indicates that a page needs updating in the status overview table
- **outdated=<text>** mark the page as outdated with given comment
- **silent** don't output empty chapter for show options when nothing is shown
- **showoutdated** to show all pages, where revision does not match base revision
- **showmissing** to show all pages, where translation is missing
- **showproblems** to show all pages which have problems
- **showuntranslated** to show all untranslated pages
- **showstatus** to show one big status table
- **lang=<code>** to restrict output of show outdated, status or missing to a specific language
- **label_outdated** label to display when using the showoutdated option

`[[ViewTopic]]`

Displays content of a discussion topic. Unless argument passed, it tries to find the topic with the same name as the current wiki page. If a name is passed, displays that topic.

`[[VisitCounter]]`

Macro displays how many times was wiki page displayed.

This macro accepts up to tree parameters. First parameter is wiki page name which visit count you want to display. If no parameters specified current page visit count is displayed. Second parameter determines if displaying of macro should update specified page visit count. Accepted values of this parameter are: True, False, true, false, 1, 0. Default value is true. Third parameter specifies number of digits for visit count display. If its value is 0 then visit count is displayed as simple text. Default value is 4.

Examples:

```
 [[VisitCounter(WikiStart)]]
[[VisitCounter(WikiStart, True)]]
[[VisitCounter(WikiStart, True, 3)]]
```

`[[VoteList]]`

Show listing of most recent votes for a resource.

Macros from around the world

The ?Trac Hacks site provides a wide collection of macros and other Trac plugins contributed by the Trac community. If you're looking for new macros, or have written one that you'd like to share with the world, please don't hesitate to visit that site.

Developing Custom Macros

Macros, like Trac itself, are written in the ?Python programming language and are developed as part of TracPlugins.

For more information about developing macros, see the ?development resources on the main project site.

Here are 2 simple examples showing how to create a Macro with Trac 0.11.

Also, have a look at ?Timestamp.py for an example that shows the difference between old style and new style macros and at the ?macros/README which provides a little more insight about the transition.

Macro without arguments

To test the following code, you should saved it in a `timestamp_sample.py` file located in the TracEnvironment's `plugins/` directory.

```
from datetime import datetime
# Note: since Trac 0.11, datetime objects are used internally

from genshi.builder import tag

from trac.util.datefmt import format_datetime, utc
from trac.wiki.macros import WikiMacroBase

class TimeStampMacro(WikiMacroBase):
   """Inserts the current time (in seconds) into the wiki page."""

   revision = "$Rev$"
   url = "$URL$"

   def expand_macro(self, formatter, name, text):
       t = datetime.now(utc)
       return tag.b(format_datetime(t, '%c'))
```

Macro with arguments

To test the following code, you should saved it in a `helloworld_sample.py` file located in the TracEnvironment's `plugins/` directory.

```
from genshi.core import Markup

from trac.wiki.macros import WikiMacroBase

class HelloWorldMacro(WikiMacroBase):
   """Simple HelloWorld macro.

   Note that the name of the class is meaningful:
    - it must end with "Macro"
    - what comes before "Macro" ends up being the macro name

   The documentation of the class (i.e. what you're reading)
   will become the documentation of the macro, as shown by
   the !MacroList macro (usually used in the WikiMacros page).
   """
```

```
    revision = "$Rev$"
    url = "$URL$"

    def expand_macro(self, formatter, name, text, args):
        """Return some output that will be displayed in the Wiki content.

        `name` is the actual name of the macro (no surprise, here it'll be
        `'HelloWorld'`),
        `text` is the text enclosed in parenthesis at the call of the macro.
          Note that if there are ''no'' parenthesis (like in, e.g.
          [[HelloWorld]]), then `text` is `None`.
        `args` are the arguments passed when HelloWorld is called using a
        `#!HelloWorld` code block.
        """
        return 'Hello World, text = %s, args = %s' % \
            (Markup.escape(text), Markup.escape(repr(args)))
```

Note that `expand_macro` optionally takes a 4th parameter *`args`*. When the macro is called as a WikiProcessor, it's also possible to pass `key=value` processor parameters. If given, those are stored in a dictionary and passed in this extra `args` parameter. On the contrary, when called as a macro, `args` is `None`. (*since 0.12*).

For example, when writing:

```
{{{#!HelloWorld style="polite" -silent verbose
<Hello World!>
}}}

{{{#!HelloWorld
<Hello World!>
}}}

[[HelloWorld(<Hello World!>)]]
```

One should get:

```
Hello World, text = <Hello World!> , args = {'style': u'polite', 'silent': False, 'verbose': True}
Hello World, text = <Hello World!> , args = {}
Hello World, text = <Hello World!> , args = None
```

Note that the return value of `expand_macro` is **not** HTML escaped. Depending on the expected result, you should escape it by yourself (using `return Markup.escape(result)`) or, if this is indeed HTML, wrap it in a Markup object (`return Markup(result)`) with `Markup` coming from Genshi, (`from genshi.core import Markup`).

You can also recursively use a wiki Formatter (`from trac.wiki import Formatter`) to process the `text` as wiki markup, for example by doing:

```
from genshi.core import Markup
from trac.wiki.macros import WikiMacroBase
from trac.wiki import Formatter
import StringIO

class HelloWorldMacro(WikiMacroBase):
 def expand_macro(self, formatter, name, text, args):
   text = "whatever '''wiki''' markup you want, even containing other macros"
   # Convert Wiki markup to HTML, new style
   out = StringIO.StringIO()
   Formatter(self.env, formatter.context).format(text, out)
   return Markup(out.getvalue())
```