The `(:include:)` directive makes it possible to insert (or "transclude") the contents of other pages into the current wiki page. All of the include directives below perform a straight text inclusion. In particular, any page links in the included text are assumed to link to pages in the current group if not otherwise qualified.

Syntax

The basic syntax is

- (:include PageName:)

with pagename includes the full page from the same group.

- {Group/PageName$:PTVar}

includes a named variable from a page, Group and PageName are options

The full syntax is

- (:include FullName#fromanchor#toanchor lines=*123* self=*0* basepage=*abc* variable=*value* :)

includes a page according to the parameters supplied. Parameters are optional.

Parameters

The directive can have multiple Name parameters with or without anchors, and multiple template variable parameters.

Named pages

```
(:include PageName:)
(:include Group.PageName:)
(:include Page1 Page2 Group1.Page3 Group2.Page4:)
```
Includes the entire text of another page into the current page. Multiple pages may be specified, but only the first available is included.

You can use the above feature to *display an error message if an include fails*. Create a page, eg. Site.IncludeFailed containing the error message. You can use any page name. Then, in your include markup, append this page at the end of the page list:

```
(:include Page1 Page2 Page3 Site.IncludeFailed:)
```

A slightly more complex approach is outlined at the talk page.

Anchors

`(:include PageName#from#to:)`	include lines from *PageName* between the `[[#from]]` and `[[#to]]` anchors
`(:include PageName#from#:)`	include all lines after `[[#from]]` to the end of the page
`(:include PageName##to:)`	include all lines from the start of the page to `[[#to]]`
`(:include PageName#from:)`	include everything between `[[#from]]` and the next anchor
`(:include PageName#:)`	include everything from the top of the page to the first anchor

Note: do not put whitespace between "#from" "#to"

Note: text on the same line as a closing anchor but preceding the closing anchor will **NOT** be included in the text. Example Belo
`some text on the last line [[#end]]`

The above, when included via `(:include PageName#start:)` will have the text on the first line but not the text on the last line.

`(:include Page1 Page2 #from#to:)`
Include from the first available of Page1, Page2 between the `[[#from]]` and `[[#to]]`
Note: put whitespace between "Page2" and "#from#to". The same anchors "#from#to" should be in both pages. If proper anchors are missing in the first available of Page1, Page2 the whole contents of the page is included.
This does not seem to work in 2.2 betas. See Cookbook:IncludeSection for a fix.

`(:include Page1#from1#to1 Page2#from2#to2:)`
Include the first from the first available of Page1 (between the `[[#from1]]` and `[[#to1]]`) or Page2 (between the `[[#from2]]` and `[[#to2]]`)

Note: Previous versions of PmWiki allowed whitespace between `#from` and `#to` anchors even though it was not designed to. Newer versions do not allow whitespace anymore. To re-enable this "exploited misbehavior" put this into your config.php or farmconfig.php

```
Markup('includeanchors', '<include', '/(\\(:include.*?#\\w+)\\s+(#\\w+)/', '$1$2');
```

Lines=

```
(:include PageName lines=10:)
(:include PageName lines=5..10:)
(:include PageName lines=5..:)
```
Include the first 10 lines, lines 5-10, or lines 5 and up from *PageName*. A "line" in this context refers to a line of source. *Thus a line may be a paragraph that wraps over several lines on the screen, or a completely blank line.*

`(:include Page1 Page2 Page3 lines=1..5:)`
Include the first five lines from the first available of Page1, Page2, or Page3. (To include lines from a list of pages, use a separate include for each.)

Self=

`(:include PageName self=0:)`
The parameter `self` can be `0` or `1`. It tells the include directive if it is allowed to include the current page. This is useful if PageName is a variable like `{$Name}` and you want to prevent the directive from including the current page.

Page text variables

`{Group/PageName$:Var}`
Includes definition list values from an (optional) page as page text variables. These are defined using a definition list (`:item:description`), simple colon delimiter (`item:description`), or special markup ((`:item:description:`)).

Basepage=

`(:include PageName basepage=BasePageName:)`
Include PageName, but treat all relative links and page variables on *PageName*` as relative to *BasePageName*.

If `basepage=` is provided all relative links and page variables are interpreted relative to basepage. So, if one creates `TemplateName` as

```
Name: {$:Name}
Address: {$:Address}
```

then the directive


```
(:include TemplateName basepage=PageName:)
```

will retrieve the contents of `TemplateName`, treating any page variables and links as being relative to `PageName`. In particular, the values for `{$:Name}` and `{$:Address}` will be taken from `PageName`, but things like `{$Title}` and `{$LastModifiedBy}` would also work here.

Basepage usage

The primary purpose of basepage is to allow the include of pages in a way that mimics the 2.1.x behavior where page variables and links are interpreted relative to the currently displayed page. This is done with:

```
(:include SomeOtherPage basepage='' :)
  -or-

(:include SomeOtherPage basepage={*$FullName} :)
```

It also allows GroupHeader and GroupFooter to have their page variables and links be relative to the currently displayed page (instead of GroupHeader and GroupFooter):

```
  ## PmWiki default $GroupHeaderFmt setting $GroupHeaderFmt = '(:include
{$Group}.GroupHeader self=0 basepage={*$FullName}:)(:nl:)';
```

Otherwise, using IncludeOtherPages inside of a GroupHeader would display 'GroupHeader' and not the name of the currently displayed page.

The basepage= parameter is general enough that it can also be used as a templating engine, so that we can grab a template page containing variables that are then filled in with values from another page:

```
(:include TemplatePage basepage=DataPage :)
```

And, of course, a single TemplatePage can actually contain multiple templates delimited by anchors, so that we end up with a syntax eerily similar[1] to pagelist-templates:

```
    (:include TemplatePage#abc basepage=DataPage :)
```

So then TemplatePage can use a syntax like:

```
    [[#abc]]
    ...template stuff here...
    [[#abcend]]
```

and it's possible to display TemplatePage as a template without it being interpreted... same as we do for Site.PageListTemplates.

[1] Okay, maybe it's not so eerie, given that the pagelist template code actually uses the same function as (:include:) to grab its templates. But it's still a useful parallel.

Specifying variables as parameters

You can also specify variable values inline with the include statement, and refer to the variables in the template using the `{$$variable1}` format:

```
(:include TemplatePage variable1="value" variable2="value2":)
```

This assumes that a site has `$EnableRelativePageVars` enabled, which is recommended in PmWiki 2.2.0 -- but was disabled by default in version 2.2.8 and earlier.

For example, on my included page ("template") I might have this:

```
[[#ivars]]
Hi, {$$Name}, how are you today?
[[#ivarsend]]
```

Hi, {$$Name}, how are you today?

Then, including that section above (that section is available via the section `{$FullName}#ivars`) you get this type of behavior:

```
(:include {$FullName}#ivars Name=Sam:)
```

Hi, Sam, how are you today?

If a value contains spaces, quote it:

```
(:include {$FullName}#ivars Name="my
friend":)
```

Hi, my friend, how are you today?

See also $EnableUndefinedTemplateVars.

See Also

- Page text variables Page variables automatically made available through natural or explicit page markup
- Cookbook:IncludeUrl

Styling Note

By default, Included pages or lines cannot be distinguished from other text on the page. To provide a visual indication that this text is special, you can apply Wiki Styles. For example:

```
%define=leftborder border-left="2px solid #88f" margin-left="2px"
padding="1px 0 3px 10px"%
What is PmWiki?
>>leftborder<< (:include PmWiki.PmWiki lines=1..4:)
>><<
''Have a very nice day!''
```

What is PmWiki?

> PmWiki is a wiki-based system for collaborative creation and maintenance of websites.
>
> PmWiki pages look and act like normal web pages, except they have an "Edit" link that makes it easy to modify existing pages

Have a very nice day!

Parameter References

Any parameters supplied to an include statement (whether they are keywords or not) are accessible inside the included page as a special `{$$...}` variable of the same name. This feature can be used to provide extra information to use when displaying the included page.

Notes

- You can also say `(:include My/Page#myanchor lines=4:)` which starts from, and includes, the line with the anchor `[[#myanchor]]` for four lines.

Notes about use with conditional markup

The `(:include ...:)` markup is processed after conditional markup is evaluated.
Therefor you can include a page or page section as part of a condition, like

`(:if some condition:)(:include SomePage#section:)(:if:)`

But `(:include SomePage#section:)` doesn't look to see if `[[#section]]` is part of a conditional, like

`(:if some condition:)[[#section]]...[[#sectionend]](:ifend:)`

`(:include SomePage#section:)` will ignore such a condition.

When testing variables in included pages the context of the page (source or target) can be useful. See special references for details.

What's the maximum number of includes that can exist in a page?

My site seems to stop including after 48 includes. (`$MaxIncludes`)

By default, PmWiki places a limit of 50 include directives for any given page, to prevent runaway infinite loops and other situations that might eat up server resources. (Two of these are GroupHeader and GroupFooter.) The limit can be modified by the wiki administrator via the `$MaxIncludes` variable.

Is there any way to include from a group of pages without specifying by exact name, e.g. between Anchor X and Y from all pages named IFClass-* ?

This can be achieved using page lists.

There appears to be a viewing issue when the included page contains the (:title:) directive.

In a default installation, the *last* title in the page overrides previous ones so you can place your (:title :) directive at the bottom of the page, after any includes. See also `$EnablePageTitlePriority`.

Copyright: Petko - 13/01/2011