From EULER Project

PmWiki: Conditional Markup

Using the (:if:) Directive

The `(:if:)` directive allows portions of a page to be included or excluded from rendering. The generic forms of the `(:if:)` directive are

```
(:if cond param:) body (:ifend:)
(:if cond param:) body (:else:) body (:ifend:)
(:if cond param:) body (:elseif cond param:) body (:ifend:)
```
see [1] [2]
where "cond" names a condition to be tested, and "param" is a parameter or other argument to the condition.

Markup Shortcut

You can also use an abbreviated form of `(:ifend:)`, `(:if:)` for brevity:

```
(:if cond1:) cond1 is true (:if:)
(:if cond1:) cond1 is true (:if cond2:) cond2 is true (:if:)
```
The latter is identical to :

```
(:if cond1:) cond1 is true (:ifend:)(:if cond2:) cond2 is true (:ifend:)
```

Built-in Conditions

The built-in conditions include:

`(:if name PAGENAME:)` - current page is named "`PAGENAME`"

`(:if group GROUPNAME:)` - current group is named "`GROUPNAME`"

`(:if auth LEVEL PAGENAME:)` - viewer is authorized - meaning "what they are allowed to do" - matches a "`LEVEL`" where `LEVEL` can be: `read`, `edit`, `upload`, `attr` or `admin`; `PAGENAME` is optional.

`(:if authid:)` - current viewer is authenticated - meaning they have proven who they are via login - to use this the wiki must include recipe AuthUser or others which set the `$AuthId` variable.

`(:if enabled InvalidLogin:)` - username and password not authenticated

`(:if true:)` - always include text

`(:if false:)` - always exclude text (same as a comment)

`(:if attachments:)` - current page has one or more attachments

`(:if date DATE VALUE:)` - `DATE` may be year-month. year-month-day is optional.

Evaluates to true if `VALUE` is within `DATE` ("now" or "today" is assumed for VALUE. VALUE may be omitted, as in the following examples.) (Note that `VALUE` can be a recognizable date via strtotime() whereas DATE [or DATE1 and DATE2 below] have a more fixed format which explicitly must exclude spaces. Any spaces in DATE1 or DATE2 cause unpredictable results.)

`(:if date DATE..:)` - true if current date is `DATE` or later (unlimited)
`(:if date DATE1..DATE2:)` - true if current date is in range `DATE1` to `DATE2` (inclusive)

dates are in standard(approve sites) format yyyy-mm-dd or yyyymmdd or yyyymmddhhmm (but see comment above on format of VALUE)
Note the ".." cannot have leading or trailing spaces.

`(:if enabled VAR:)` - true if PHP VAR is true
`(:if enabled AuthPw:)` - true if user has entered any password during the current browser session.

- This does not mean the user has entered the correct password, just that they entered one.

`(:if equal STRING1 STRING2:)` - true if `STRING1` equals `STRING2`, use quotes if the string or string variable contains spaces, eg `"MY STRING"`
`(:if match REG_EXPRESSION:)` - true if current page name matches the regular expression
`(:if exists PAGENAME:)` - true if the page *pagename* exists
`(:if action ACTION:)` - true if the action ?action=ACTION (edit, print, ...) is currently permitted. To test what the current action being requested is, use `(:if equal {$Action} ACTION:)`.
`(:if ontrail WikiTrailPage ThisPage:)` - true if ThisPage is in a list used as a trail on WikiTrailPage

The name and group conditionals will work even for an included page, as the "name" and "group" conditionals always check the currently displayed page, as opposed to the page that the markup appears in.

Negated Conditions

Negated forms of conditions also work:

`(:if !attachments:)` - this page has no attachments

`(:if ! name PAGENAME:)`
`(:if name -PAGENAME :)` current page is NOT named "`PAGENAME`"
`(:if name !PAGENAME :)`

Nesting Conditions

Conditions may be nested (from 2.2.beta 66).

Nested `(:if:)` works the same way as nested `(:div:)`. To have nested conditionals you need to number the if, and the matching else/ifend:

```
(:if cond1:)
  cond1 is true
  (:if2 cond2:)
     cond1 and cond2 are true
  (:else2:)
     cond1 is true, cond2 is not
  (:if2end:)
(:else:)
  cond1 is false, cond2 testing was skipped
(:ifend:)
```

Spaces were added for better readability.

Using wildcard placeholders

The character `*` can be used as a wildcard to represent any character, zero, one, or multiple times.
The character `?` can be used as a wildcard to represent any character exactly once.
Wildcard characters (`*` and `?`) can be used with the *name* and *group* conditional markups, thus:

`(:if name PmCal.2005* :)`	- current page is in group PmCal and begins with 2005
`(:if group PmWiki* :)`	- current page is in group PmWiki or a group beginning with PmWiki
`(:if name Profiles.*,-Profiles.Profiles :)`	- current page is in group `Profiles` but not `Profiles.Profiles`

Using page text variables, page variables and markup expressions

Page text variables (PTVs), page variables (PVs) and markup expressions can be used in conditional markup. They will be assigned/evaluated before the condition(s).

Use with page list templates

Conditional markup is used extensively with page list templates.

Use with page variables:

= current item
< previous item
> next item
Conditionals used to structure pagelist output:

```
(:if equal {<$Group}:) deprecated in favour of (:template first:) At beginning of list
  (:if equal {>$Group}:) deprecated in favour of (:template last:) At end of list
  (:if ! equal {=$Group} {<$Group}:) deprecated in favour of (:template first {=$Group}:) First i
  (:if ! equal {=$Group} {>$Group}:) deprecated in favour of  (:template last {=$Group}:) Last it
```

Combining conditions

Conditions (as previously defined) may be combined into more complex conditional expressions using one of these three equivalent forms:

```
(:if expr EXPRESSION :)
(:if [ EXPRESSION ] :)
(:if ( EXPRESSION ) :)
```

Conditions are combined into expressions with boolean operators and brackets. In the next table, A and B are either regular conditions or (round-)bracketed sub-expressions of regular conditions:

Expression	Operator	Result

`A and B`	And	TRUE if both A and B are TRUE.
`A or B`	Or	TRUE if either A or B is TRUE.
`A xor B`	Xor	TRUE if either A or B is TRUE, but not both.
`! A`	Not	TRUE if A is not TRUE.
`A && B`	And	TRUE if both A and B are TRUE.
`A || B`	Or	TRUE if either A or B is TRUE.

Example

```
(:if [ name SomePage and group SomeGroup ]:)    equivalent to (:if name SomeGroup.SomePage:)
```

Note:

- Spaces around operators and brackets are required.
- No specific feedback is given for syntax errors or unbalanced brackets.
- Use round brackets (not square) for nested expressions.

Thus, the following is a valid way of building an expression that shows the following contents only when the user is either the administrator, or is logged in and the time is later than the given date:

```
(:if [ auth admin || ( authid && date 2006-06-01 ) ] :)
```

Nesting with square brackets will silently fail to work as expected:

```
(:if [ auth admin || [ authid && date 2006-06-01 ] ] :)
```
NOTE: Doesn't Work!
A common use of these complex tests are for expressions like:

```
(:if expr auth admin || auth attr || auth edit :)
[[Logout -> {$Name}?action=logout]]
(:ifend:)
```

which provides a *logout* link only when the browser has admin, attr, or edit permissions.

admins (advanced)

Creating new conditions

See Cookbook:ConditionalMarkupSamples.

See also special references for the use of {*$Variables}.

Retrieved from https://www-sop.inria.fr/mascotte/EULER/wiki/pmwiki.php/PmWiki/ConditionalMarkup
Page last modified on July 10, 2009, at 08:01 PM