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:)
```

where "cond" names a condition to be tested, and "param" is a parameter or other argument to the condition.

Note that `(:if:)` without parameters and `(:ifend:)` are identical. Also note that `(:if cond:)` automatically closes a previous conditional. For nested multiple levels, see Nested conditionals.

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. To use this the wiki must include recipe Auth User.
`(:if true:)`	-	always include text
`(:if false:)`	-	always exclude text (same as a comment)
`(:if attachments:)`	-	current page has one or more attachments If used in a sidebar, header, or footer the condition applies to the main page.

`(: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 if VALUE is omitted, as in the following examples.) (Note that VALUE can be a recognizable date via strtotime() whereas DATE [or DATE 1? and DATE 2? below] have a more fixed format which explicitly must exclude spaces. Any spaces in DATE 1? or DATE 2? cause unpredictable results.)
`(:if date DATE.. VALUE:)`	-	true if VALUE (or current date if omitted) is DATE or later (unlimited)
`(:if date ..DATE VALUE:)`	-	true if VALUE (or current date if omitted) is DATE or earlier (unlimited)
`(:if date DATE1..DATE2 VALUE:)`	-	true if VALUE (or current date if omitted) is in range DATE1 to DATE2 (inclusive) *dates are in standard format yyyy-mm-dd or yyyymmdd or yyyymmddThhmm (note the "T" between the date and the hour, and also 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 ontrail WikiTrailPage ThisPage:)`	-	true if This Page? is in a list used as a trail on Wiki Trail Page?

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.

Note: Although there is no built-in conditional markup to test ?action=, you can use `(:if equal {$Action} ACTION:)`

to test what the current action being requested is.

Negated Conditions

Negated forms of conditions also work:

`(:if !attachments:)`	-	this page has no attachments
`(:if ! name PAGENAME:)`		current page is NOT named "PAGENAME"
`(:if name -PAGENAME :)`		

Note that `(:if cond:)` automatically closes a previous conditional. Thus, the following two examples have identical meaning:

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

Conditions can be nested from 2.2.beta 66. 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 ignored (: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 Pm Cal? and begins with 2005
`(:if group PmWiki* :)`	-	current page is in group Pm Wiki or a group beginning with Pm Wiki
`(: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 (PT Vs?), page variables (P Vs?) and markup expressions can be used in conditional

markup. They will be assigned/evaluated before the condition(s).

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:)
```

Important Notes:

- Spaces are *required* around operators and brackets.
- 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}.