MARS buildRules

Sébastien Villaume
Manuel Fuentes, Baudouin Raoult, Tiago Quintino
Second Workshop for MARS administrators
7-8 March 2016

ECMWF

purpose

- Construct the MARS tree, i.e. describe how the data is arranged internally.
- Has a massive impact in MARS:
  - On the performance of the whole system
  - Any mistakes can be devastating
  - Test your changes on a scratch server before putting it in production!!!

- Don't start from scratch, look at the ECMWF buildRules (or SMHI ones published in Confluence)

Basic constructs

`#`	comments
`include`	Include a snippet from an external file
`$keyword$`	keyword as a string
`%keyword%`	keyword as an integer
`"someString" 'some string'`	a string
`fail "message"`	fail with error message "message", sent back to the client. Typically used in an "otherwise" clause
`cluster <keyword|string>`	Insert keyword or string into the layout "name"
`push <keyword|string>`	Used in function to return something (a keyword, a string, an object)
`functionName()`	function call
`node <nodeType>(args)`	Insert a node of type "nodeType", can be a tree node or a leaf node

Advanced constructs

`function <name>`
`  Block function`
`end function`
	Define a function

`select <keyword>`
`  when <keyword_value>`
`  Block when`
`  end when`
`  …`
`  otherwise`
`  Block otherwise`
`  end otherwise`
`end select`
	Works like a “case” construct, otherwise to catch everything else

`if <condition> then`
`  Block if`
`else`
`  Block else`
`end if`
	Classic if else construct

operators

==,!=, <,>,<=,>=	Classic comparators
&& ||	Classic boolean operator
+	Concatenate (keyword) strings

Live session