Simple RegEx Tutorial

Regular Expression can be used in Content Filter conditions.

Regular Expressions can be extremely complex but they are very flexible and powerful and can be used to perform comparisons that cannot be done using the other checks available.

There follows some very basic examples of regular expression usage. For a complete description please visit www.regular-expressions.info.

'^' and '$'

First of all, let's take a look at two special symbols: '^' and '$'. These symbols indicate the start and the end of a string, respectively:

"^The"	matches any string that starts with "The".
"of despair$"	matches a string that ends in with "of despair".
"^abc$"	a string that starts and ends with "abc" - effectively an exact match comparison.
"notice"	a string that has the text "notice" in it.

You can see that if you don't use either of these two characters, you're saying that the pattern may occur anywhere inside the string -- you're not "hooking" it to any of the edges.

'*', '+', and '?'

In addition, the symbols '*', '+', and '?', denote the number of times a character or a sequence of characters may occur. What they mean is: "zero or more", "one or more", and "zero or one." Here are some examples:

"ab*"	matches a string that has an a followed by zero or more b's ("ac", "abc", "abbc", etc.)
"ab+"	same, but there's at least one b ("abc", "abbc", etc., but not "ac")
"ab?"	there might be a single b or not ("ac", "abc" but not "abbc").
"a?b+$"	a possible 'a' followed by one or more 'b's at the end of the string: Matches any string ending with "ab", "abb", "abbb" etc. or "b", "bb" etc. but not "aab", "aabb" etc.

Braces { }

You can also use bounds, which appear inside braces and indicate ranges in the number of occurrences:

"ab{2}"	matches a string that has an a followed by exactly two b's ("abb")
"ab{2,}"	there are at least two b's ("abb", "abbbb", etc.)
"ab{3,5}"	from three to five b's ("abbb", "abbbb", or "abbbbb")

Note that you must always specify the first number of a range (i.e., "{0,2}", not "{,2}"). Also, as you might have noticed, the symbols '*', '+', and '?' have the same effect as using the bounds "{0,}", "{1,}", and "{0,1}", respectively.

Now, to quantify a sequence of characters, put them inside parentheses:

"a(bc)*"	matches a string that has an a followed by zero or more copies of the sequence "bc"
"a(bc){1,5}"	one through five copies of "bc."

'|' OR operator

There's also the '|' symbol, which works as an OR operator:

"hi|hello"	matches a string that has either "hi" or "hello" in it
"(b|cd)ef"	a string that has either "bef" or "cdef"
"(a|b)*c"	a string that has a sequence of alternating a's and b's ending in a c

('.')

A period ('.') stands for any single character:

"a.[0-9]"	matches a string that has an a followed by one character and a digit
"^.{3}$"	a string with exactly 3 characters

Bracket expressions

specify which characters are allowed in a single position of a string:

"[ab]"	matches a string that has either an a or a b (that's the same as "a|b")
"[a-d]"	a string that has lowercase letters 'a' through 'd' (that's equal to "a|b|c|d" and even "[abcd]")
"^[a-zA-Z]"	a string that starts with a letter
"[0-9]%"	a string that has a single digit before a percent sign
",[a-zA-Z0- 9]$"	a string that ends in a comma followed by an alphanumeric character

You can also list which characters you DON'T want -- just use a '^' as the first symbol in a bracket expression (i.e., "%[^a- zA-Z]%" matches a string with a character that is not a letter between two percent signs).

In order to be taken literally, you must escape the characters "^.[$()|*+?{\" with a backslash ('\'), as they have special meaning. On top of that, you must escape the backslash character itself in PHP3 strings, so, for instance, the regular expression "(\$|A)[0-9]+" would have the function call: ereg("(\\$|A)[0-9]+", $str) (what string does that validate?)

Just don't forget that bracket expressions are an exception to that rule--inside them, all special characters, including the backslash ('\'), lose their special powers (i.e., "[*\+?{}.]" matches exactly any of the characters inside the brackets). And, as the regex manual pages tell us: "To include a literal ']' in the list, make it the first character (following a possible '^'). To include a literal '-', make it the first or last character, or the second endpoint of a range."

See Also

Shared

Database Settings

Access Mode

Schedule

Domain Admin Rights

Select Accounts

Account Options