Regular Expression Examples

Generally, RegularExpressions are defined in terms of whether a string matches the given RegularExpression, although much more complex variants are certainly in widespread use.

Simple examples:

- Any string containing "a" matches the regex /a/.
- A string beginning with "a" matches the regex /^a/
- A string ending with "a" matches the regex /a$/
- A string s matches the regex /r1r2/ iff there exist substrings s1 and s2 such that s = s1.s2 and s1 matches /r1/ and s2 matches /r2/.
- A string s matches the regex /r1|r2/ iff either s matches /r1/ or s matches /r2/ (or both).
- A string s matches the regex /r1*/ iff either s contains the empty string (always true) or s matches /r1(r1*)/ (or of course both).
- A string s matches the regex /[r1]/ iff s is exactly one character and that character appears in the set r1.
- To have a] in the set, place it first.
- To invert a set, precede it with a ^.
- Numeric and alphabetic sets may be abbreviated similar to [0-9], [a-z] and [A-Z].

Here are some more:

- /.*/ matches anything.
- /colo(u|)r/ matches either "color" or "colour". (This can also be written /colou?r/.)
- /(0|1|2|3|4|5|6|7|8|9)*(1|3|5|7|9)/ matches all odd positive integers. (Shorter written as /[0-9]*[13579]/. -- ChristofferHammarstrom)
- /(1|)(0(0*)1)*(0*)/ matches all strings of 0s and 1s without two 1s in a row.
- /0?[1-9]|[12][0-9]|3[01]/ matches all valid days of the months with 31 days.

-- RaphLevien

RegXy

Another example, as in a beer program in RegXy, which is an programming language (EsotericProgrammingLanguage) with only 2 things, RegularExpressions and if/goto commands:

- 0/^.*$//

- a//z9az8az7az6az5az4az3az2az1aza/
- b/z([0-9]?)a/x$19x$18x$17x$16x$15x$14x$13x$12x$11x$10/
- b?/z([0-9]?)a/b
- c/x([0-9][0-9]?)/$1 botles uv beer. $1 botles uv beer on the wall, $1 botles uv beer, Take 1 down and pass it around, /
- c?/x([0-9][0-9]?)/c
- d/^99 botles uv beer. //
- e/, 0 botles uv beer.*$/, No more botles uv beer on the wall./

Syntax: Query: LABEL/expr/JUMP_TARGET_ON_MATCH

Change: LABEL/expr/SUBSTITUTE/

(Do you see how simple this is?)

Example usage of RegularExpressions in Validation

I have had great success using RegularExpressions in validation routines. They're perfectly suited for those ever changing user requirements like "We can't permit commas, periods or ampersands in field X. The field also has to be between 8 and 12 characters, begin with an S and contain another letter at position 6".

Coding this example would give no room for changes, where as a regular expression can be stuck in a database field which can be modified later if necessary. Besides, the code to test the string is much simpler when using a regular expression.

Shhhhhh... you don't want TopMind to see this, do you? :p

A little story from last night's ExtremeProgramming meeting drives home an aspect of RubyLanguage. We had two pairs working, and someone from the other pair called out, "What's the name of the 'string ends with' method?" I called back, "You want regular expression matching. Stick a dollar sign at the end." Well, they pored through the String API docs to discover that there's no built-in method for determining whether a string ends with a given substring. I called out, "Python has such a thing; Ruby does not. Use a regular expression, and stick a dollar sign at the end."

In the meantime, my pair and I needed to test whether a filename ended with a particular extension. If we were working in PythonLanguage we would have written

```
fname.endswith('.txt')
```

This would return True or False. In RubyLanguage we did a RegularExpression match:

```
fname =~ /\.txt$/
```

This returns the index of where the match starts, or nil if there's no match. (In Ruby, nil is false and a number -- even the number 0 -- is true.) We joked that we could write a String#endswith method and sell it to the other pair.

The moral of the story is, RegularExpression handling isn't built in to RubyLanguage just as a convenience; you're actually expected to *use* regexes, even for common things like an "ends with" test.

-- ElizabethWiethoff

Guides to RegularExpressions

- http://zytrax.com/tech/web/regex.htm

CategoryRegularExpressions

View edit of October 1, 2011 or FindPage with title or text search