Regular Expressions

… a powerful tool in a skilled hand

15. 10. 2010

Agenda

- Introduction
- Special Characters
- Sets
- Character Classes
- Simple examples
- Multipliers
- Number Quantifiers
- Subexpressions
- Regex in PHP
- Regexp in real use
- Try Out ...
- Read more...
- Q/A

Introduction

- A pattern that either matches or doesn't match a given string or substring. Result of comparison will either be true or false.

  art :: matches art in 'art', 'article' , 'artifact', 'martial', 'cart', 'mart'

- Use and syntax of regex is the same across many Unix programs (vi, sed, awk etc.) and programming / scripting languages(Perl , Java, PHP etc.)
- Regexp is supported in all major development environments.
- Uses:
  - **Search for the existence of a pattern**
  - **Validate User Input data in web forms**
  - **Bulk Search and replace at ease.**
  - **String manipulation**

Special Characters

- A period (.) - matches any single character
  www.ibm.com matches patterns like “www1ibmacom”, ”wwwaibmscom”
- A pipe (|) - either what comes before or what comes after.
  jpg|png :: **matches 'jpg' or 'png‘**
- A caret (^) at the beginning of a regexp - will only match if it starts at the beginning of the comparison string
  ^art :: **matches 'article' & 'artifact‘ but not ‘mart’**
- A dollar sign ($) at the end of a regexp - will only match if it ends at the end of the comparison string
  art$** :: **matches 'cart' & 'mart‘ but not ‘arts’ ^art$ :: **matches - 'art'**

Special Characters (...)

- All regex are case sensitive unless told not to be so. – with the use of ‘i’

- **"WWW.ibm.com"** does not match "www"
  "WWW.ibm.com" | egrep -i "www“ matches WWW.ibm.com

- A backslash (\) means escape the next character if it is a special one.

  www\.ibm\.com matches exact pattern “www.ibm.com”
  \? matches a question mark
  \/ matches a forward slash
  \\ matches a backslash

Sets

- A character set is a group of characters from which only one is desired.
  [0123456789] – matches any single number
- Sets can use ranges of characters
  [4-9] – matches any digit in the range 4 to 9
- A dash can be represented in a set by placing it first
  [-aeiou] – matches a dash or a vowel
- A caret (^) at the beginning of a set negates.
  [^1-4] – matches any character which isn't 1,2,3 or 4

Character classes

- To represent a bunch of characters as a single item

  alpha :: any letter, same as [A-Za-z].
  upper :: any upper-case letter; same as [A-Z].
  lower :: any lower-case letter; same as [a-z].
  digit :: any digit; same as [0-9].
  alnum :: any alphanumeric character; = [A-Za-z0-9].
  xdigit :: any hexadecimal digit; = [0-9A-Fa-f].

- If the character after the backslash is not a special one, then it may be an escape sequence.

  \l - Lowercase next character
  \n - newline character
  \r - Return character
  \s - white space
  \S - non white space = **^\s**
  \t - Tab character

Simple examples

- \d\d\.\d\d\.\d\d\d\d
  matches patterns like “01.01.2000”
- \w\w\w, \d\d \w\w\w \d\d\d\d
  matches patterns like “Wed, 21 Jul 2000”
- “.. \[[0-9]\]:” matches patterns like SL [9]: , IQ [5]:
- “[a-zA-Z]99” matches patterns like s99, K99, S99
- “([wx])([yz])” matches 'wy','wz','xy' or 'xz‘
- “([A-Z]{3}|[0-9]{4})” matches three UC letters OR 4 numbers
- s!^(.*)(\r?\n\1)+$!\1!g – deletes similar duplicate lines

Multipliers

Any character or character class can be assigned a multiplier - say whether a character must exist, is optional, may exist for a certain minimum or maximum ...

- Plus (+) :: One or more
  A+ - A followed by any no. of additional A's
- Asterisk (*) :: anything
  A* - A followed by anything
- Question Mark (?) :: Zero or more occurrences
  A? - Either A or no As
- Curly Brackets({}) :: A specific range of occurrences
  A{2,4} - 2 As or more but no more than 4.
  [[:digit:]]{1,6} - 1 number (0-9) or more, but no more than 6.

Number Quantifiers

- Specify number of occurrences, how many times previous character should occur.
- G* - 0 or more G
- G+ - 1 or more occurrence of G
- G? - 0 or 1 occurrence of G
- V{5} - Exactly 5 times
- S{3,} - 3 or more ; at least 3
- V{2,3} - from 2 to 3 times

SubExpressions

- A way of grouping characters together.
- Used to reference the entire group at once.
- To group characters, place them within '()'.

(Name) = name ;; (Name)+ = name, namename

A pipe within a subExpression means either the first group of text or the second (or more).

(Na|me) = Na or me ;; (Name|Date) = Name or date

Back referencing ; reference one or more groups directly. (\) followed by a no. that specifies which subexpression we want.

(name)\1 = namename

(name|date)\1 = namename or datedate

Regexp in PHP

preg_replace – search and replace

```
<? php
$string = 'Jul 12, 2000';
$pattern = '/(\w+) (\d+), (\d+)/i';
$replacement = '$1y 21, $3';
echo preg_replace($pattern, $replacement, $string);
?>
```

Try !!! Swapping '12' to '21' using regexp instead of literal substitution of 12 by 21

Regexp in PHP

preg_match() – match a pattern – returns 1 for match else 0.

```
<?php

if (preg_match("/\bweb\b/i", "PHP is a web scripting language.")) {
    echo "A match was found.";

} else { echo "A match was not found."; }

if (preg_match("/\bweb\b/i", "PHP is the best website scripting
    language.")) {echo "A match was found.";

} else { echo "A match was not found.";}

?>
```

Regexp in PHP

split – split a string based on regexp

```
<? // Delimiters may be slash, dot, or hyphen
$date = "01/05/1970";
list($day, $Month, $year) = split('[/.-]', $date);
echo "Month: $month; Day: $day; Year: $year <br/>\n";
?>
```

... explore more regex in PHP

Try out

What is $1, $2? What is the end result? Dissect the regexp and analyse first and predict the result. Then try the code.

```
<?php
$s = '<a href="http://www.php.net”> PHP web site </a> ';
$s .= '<a href="http://www.gmail.com"> Gmail </a> ';
$s .= '<a href="http://www.ibm.com"> IBM </a>';
$s =
   preg_replace('/<a[^>]*?href=[\'"](.*?)[\'"][^>]*?>(.*?)<\/a>/si','<a
   href="$1" target="_blank">$2</a>',$s);
echo $s;
?>
```

Regexp in real use

- Practical use to check password strength
  - <? $password = "Fyfjk34sdfjfsjq7";
  - if (preg_match("/^.*(?=.{8,})(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).*$/", $password)) { echo "Your passwords is strong."; }
  - else {echo "Your password is weak."; } ?>
  - (?=.*{8,}) - checks if there are at least 8 characters in the string.
  - (?=.*[0-9]) – checks for "zero or more alphanumeric characters, then any digit". Checks for at least one number.
  - (?=.*[a-z]) and (?=.*[A-Z]) looks for LC and UC letter anywhere.

Regexp in real use

Only allow plain text and URLs – no other HTML tags or scripts allowed in a textbox area as input

```
if ( preg_match('#(<script)([^\s]*)#', $caption) ||
     preg_match('#(</script>)([^\s]*)#', $caption) ||
     preg_match('#(<\?)([^\s]*)#', $caption)||
     preg_match('#(\?>)([^\s]*)#', $caption)||
     preg_match('#(<\%)([\s]*)#', $caption)||
     preg_match('#(\%>)[^\s]*)#', $caption)(
{
   DisplayErrorMessage("0", "Invalid Caption <br> Caption can
   have only plain text and reference URLs  <br>  No other HTML
   tags allowed " , "javascript:history.go(-1)");
}
```

Read more...

- Books
- Mastering Regular expressions by Jeffrey E. F. Friedl (O'Rielly)
- Sams Teach Yourself Regular Expressions in 10 Minutes by Ben Forta
- Regular Expressions Cookbook by Jan Goyvaerts (O'Rielly)

- Web references
- http://www.regular-expressions.info/
- http://www.phpf1.com/tutorial/php-regular-expression.html
- http://weblogtoolscollection.com/regex/regex.php
- .....lot many web references

…The best use of regexp ensures that "only" the desired input gets into the system, thereby ensuring better security of the system.

… Excellent tool for Sysadmins for log analysis, passwd file search, file manipulation etc...

ksri@tifr.res.in

(Q/A) / Discussion