PHP - Regular Expressions

Info319E

PHP - Regular Expressions

- Regular expressions are nothing more than a sequence or pattern of characters itself. They provide the foundation for pattern-matching functionality.
- Using regular expression you can search a particular string inside a another string, you can replace one string by another string and you can split a string into many chunks.
- PHP offers functions specific to two sets of regular expression functions, each corresponding to a certain type of regular expression. You can use any of them based on your comfort.

POSIX Regular Expressions

- The structure of a POSIX regular expression is not dissimilar to that of a typical arithmetic expression: various elements (operators) are combined to form more complex expressions.
- The simplest regular expression is one that matches a single character, such as g, inside strings such as g, haggle, or bag.
- Lets give explanation for few concepts being used in POSIX regular expression. After that we will introduce you with regular expression related functions.

Brackets

- Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Sr.No	Expression & Description
1	**[0-9]** It matches any decimal digit from 0 through 9.
2	**[a-z]** It matches any character from lower-case a through lowercase z.
3	**[A-Z]** It matches any character from uppercase A through uppercase Z.
4	**[a-Z]** It matches any character from lowercase a through uppercase Z.

Brackets

- The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from b through v.

Quantifiers

- The frequency or position of bracketed character sequences and single characters can be denoted by a special character.
- Each special character having a specific connotation. The +, *, ?, {int. range}, and $ flags all follow a character sequence.

Quantifiers

Sr.No	Expression & Description
1	**p+** It matches any string containing at least one p.
2	**p*** It matches any string containing zero or more p's.
3	**p?** It matches any string containing zero or more p's. This is just an alternative way to use p*.
4	**p{N}** It matches any string containing a sequence of **N** p's
5	**p{2,3}** It matches any string containing a sequence of two or three p's.
6	**p{2, }** It matches any string containing a sequence of at least two p's.
7	**p$** It matches any string with p at the end of it.
8	**^p** It matches any string with p at the beginning of it.

Examples

Sr.No	Expression & Description
1	**[^a-zA-Z]** It matches any string not containing any of the characters ranging from a through z and A through Z.
2	**p.p** It matches any string containing p, followed by any character, in turn followed by another p.
3	**^.{2}$** It matches any string containing exactly two characters.
4	**(.*)** It matches any string enclosed within and .
5	**p(hp)*** It matches any string containing a p followed by zero or more instances of the sequence php.

Predefined Character Ranges

Sr.No	Expression & Description
1	**[[:alpha:]]** It matches any string containing alphabetic characters aA through zZ.
2	**[[:digit:]]** It matches any string containing numerical digits 0 through 9.
3	**[[:alnum:]]** It matches any string containing alphanumeric characters aA through zZ and 0 through 9.
4	**[[:space:]]** It matches any string containing a space.

PHP's Regexp PERL Compatible Functions

Sr.No	Function & Description
1	**preg_match()**The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.
2	**preg_match_all()**The preg_match_all() function matches all occurrences of pattern in string.
3	**preg_replace()**The preg_replace() function operates just like ereg_replace(), except that regular expressions can be used in the pattern and replacement input parameters.
4	**preg_split()**The preg_split() function operates exactly like split(), except that regular expressions are accepted as input parameters for pattern.
5	**preg_grep()**The preg_grep() function searches all elements of input_array, returning all elements matching the regexp pattern.
6	**preg_quote()**Quote regular expression characters

PHP - Function preg_match()

- The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.

```
<?php
   $line = "Vi is the greatest word processor ever created!";
   // perform a case-Insensitive search for the word "Vi"

   if (preg_match("/\bVi\b/i", $line, $match)) :
      print "Match found!";
      endif;
?>
```

PHP - Function preg_match_all()

- int preg_match_all (string pattern, string string, array pattern_array [, int order]);
- The preg_match_all() function matches all occurrences of pattern in string. $pattern_array[0] will contain elements matched by the first parenthesized regexp, $pattern_array[1] will contain elements matched by the second parenthesized regexp, and so on.

```
<?php
   $userinfo = "Name: <b>John Poul</b> <br> Title: <b>PHP Guru</b>";
   preg_match_all ("/<b>(.*)<\/b>/U", $userinfo, $pat_array);

   print $pat_array[0][0]." <br> ".$pat_array[0][1]."\n";
?>
```

Function preg_replace()

- The preg_replace() function operates just like POSIX function ereg_replace(), except that regular expressions can be used in the pattern and replacement input parameters.

```php
<?php
   $copy_date = "Copyright 1999";
   $copy_date = preg_replace("([0-9]+)", "2000", $copy_date);

   print $copy_date;
?>
```

Function preg_split()

- The preg_split() function operates exactly like split(), except that regular expressions are accepted as input parameters for pattern.
- If the optional input parameter limit is specified, then only limit number of substrings are returned.
- array preg_split (string pattern, string string [, int limit [, int flags]]);

```php
<?php
   $ip = "123.456.789.000"; // some IP address
   $iparr = split ("/\./", $ip);

   print "$iparr[0] <br />";
   print "$iparr[1] <br />" ;
   print "$iparr[2] <br />"  ;
   print "$iparr[3] <br />"  ;
?>
```

Character	Description
.	a single character
\s	a whitespace character (space, tab, newline)
\S	non-whitespace character
\d	a digit (0-9)
\D	a non-digit
\w	a word character (a-z, A-Z, 0-9, _)
\W	a non-word character
[aeiou]	matches a single character in the given set
[^aeiou]	matches a single character outside the given set
(foo|bar|baz)	matches any of the alternatives specified

Modifier	Description
i	Makes the match case insensitive
m	Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
o	Evaluates the expression only once
s	Allows use of . to match a newline character
x	Allows you to use white space in the expression for clarity
g	Globally finds all matches
cg	Allows a search to continue even after a global match fails

Simple regex

Regex quick reference
[abc] A single character: a, b or c
[^abc] Any single character but a, b, or c
[a-z] Any single character in the range a-z
[a-zA-Z] Any single character in the range a-z or A-Z
^ Start of line
$ End of line
\A Start of string
\z End of string
. Any single character
\s Any whitespace character
\S Any non-whitespace character
\d Any digit
\D Any non-digit
\w Any word character (letter, number, underscore)
\W Any non-word character
\b Any word boundary character
(...) Capture everything enclosed
(a|b) a or b
a? Zero or one of a
a* Zero or more of a
a+ One or more of a
a{3} Exactly 3 of a
a{3,} 3 or more of a
a{3,6} Between 3 and 6 of a