Regular Expression

AENDA

What is Regular Expression?
How we can use it in Php?
User Cases
Rules for RE

Regular Expression

A way to do pattern matching

For Example :

Match all files with extension .jpeg or .jpg or .png

Match all files with start from some letter or ending with some letter (doc1,doc2,doc3,doc_1)

Validating Numbers or Decimal

RE Applications in Real World

- Search pattern in DNA
- Check syntax of a programming language
- Searching files in File System
- Input Validation i.e. Username, Contact No, Email-Address, Domain Name
- Text Parsing and many more ...

What we need ?

1) Target : It may be Text/String/File
2) Regular Expression : A formula for matching the pattern
3) Method : where we can pass Target and RE to get results

Note : RE almost same for any language

Syntax of RE

It is a String start with / and End with / and between / / , need a Regular Expression Rule

i.e.

$regex = “/ Rule /”;

Simple Example : User's Feedback

Match the users feedback about animals.
User1 : I like cats
User2: I like dogs
User3: I like cats and dogs
User 4: I like cats and birds

We have to match , which users have like cats.

Approach: Parse all the lines and check which user have written cats.

Solution

```
<?php
$regex = “/cats/”;
echo preg_match($regex,"I like cats but not dogs");
?>
```

Output : 1

Continued..

Match either cats or dogs

```
$regex = "/cats|dogs/";

echo preg_match($regex,"I like cats but not dogs");
echo preg_match($regex,"I like cats");
echo preg_match($regex,"I like dogs");
echo preg_match($regex,"I like birds");
```

Output : will true except last statement

Rules 1 : Period (.)

1) period (.) : means “**match any character**,” so the pattern .at matches bat, cat, and even rat.

```
$regex = "/.at/";
echo preg_match($regex,"cat");
echo preg_match($regex,"bat");
echo preg_match($regex,"rat");
echo preg_match($regex,"dog");
```

Output : 1110

Rules 1 : Period (.)

```
$regex = "/..at/"; # we have change . with ..

echo preg_match($regex,"cat");

echo preg_match($regex,"bat");

echo preg_match($regex,"rat");

echo preg_match($regex,"dog");
```

What will be the output?

0000

Rules 1 : Period (.)

```
$regex = "/c.at/";

echo preg_match($regex,"coat");

echo preg_match($regex,"ciat");

echo preg_match($regex,"cat");

echo preg_match($regex,"cooat");
```

Output ?

1100

Rule 2 : asterisk (*)

asterisk means “match 0 or more of the preceding object”

```
$regex = "/c*at/"; #  it matches string at, cat , ccat , cccat, at123 , abcat , abat
echo preg_match($regex,"cat");
echo preg_match($regex,"coat");
echo preg_match($regex,"ciat");
echo preg_match($regex,"cast");
echo preg_match($regex,"a_t");
echo preg_match($regex,"at123");
```

Output : 111001

Rule 2 : asterisk (*)

$regex = "/abc*xy/"; # abcxy, abccxy , abcccxy , kmnabcxymn

```
echo preg_match($regex,"abcxy");
echo preg_match($regex,"abccxy");
echo preg_match($regex,"abcccxy");
echo preg_match($regex,"kmnabcxymn");
```

Output : 1111

Rule 3 : plus sign (+)

$regex = "/ab+at/"; #start with ‘a’ then 1 or more occurence of ‘b’ and end with ‘at’
matches all strings abat , abbat , abbbat , abbbbbbat , xyzabbat , xyzabbatd

```
echo preg_match($regex,"abat");
echo preg_match($regex,"abbat");
echo preg_match($regex,"abbbat");
echo preg_match($regex,"abbbbbbat");
echo preg_match($regex,"xyzabbat");
echo preg_match($regex,"xyzabbatd");
```

Rule 3 : plus sign (+)

```
$regex = "/c+at/"; # 1 or more char ‘c’  before at
echo preg_match($regex,"cat");
echo preg_match($regex,"coat");
echo preg_match($regex,"ciat");
echo preg_match($regex,"cat");
echo preg_match($regex,"cooat");
echo preg_match($regex,"oat");
echo preg_match($regex,"at");
echo preg_match($regex,"abcefgh_at");
Output : 10010000
```

Rule 4 : question mark (?).

question mark means “the preceding object is optional , it matches 0 or 1 of the object that precedes it
colou?r matches both color and colour

```
$regex = "/abc?xy/"; # abcxy, abccxy , abcccxy , kmnabcxymn
echo preg_match($regex,"abxy");# zero c
echo preg_match($regex,"abcxy");# one c
echo preg_match($regex,"abccxy");# two c
echo preg_match($regex,"abcccxy");# three c
echo preg_match($regex,"kmnabcxymn");# one c
```

Functions used for RE matching

preg_match()

preg_match_all()

preg_replace()

Rule 5 : Match specific character set

The character class [aeiou] matches any one of the characters a, e, i, o, and u.

The class [a-z] matches all lowercase English letters.

The class [a-zA-Z0-9] matches digits and English letters

The class [a-zA-Z0-9_] matches digits, English letters, and the underscore.

How we can write RE for text containing only English characters and Digits ?

Let’s Check..

RE = [a-z0-9]+

This will match “ One of my friend is Sachin”

Also match with “h20” , “cp450” , “50 mg”

But also matches “*hello!!!” , “*/**a” , “_#@4” , as these are contain one or more of a digit or lowercase English letter

Rule 6 : caret (^) ; pattern at the beginning

^[a-z0-9]+ means “begins with one or more of a digit or lowercase English letter,”

Matching all these “a_var” , “0abc”, “9h”, “400” , “ro34” “3**/” ,
But not “*78” , “$3” , “/exe/”

```
$regex = "/^[a-z0-9]/";
echo preg_match($regex,"xy6");
echo preg_match($regex,"@a");
echo preg_match($regex,"*78");
echo preg_match($regex,"kmnabcxymn");
```

Output: 1001

Rule 7 : dollar sign ($) pattern at the end of the string

[a-z0-9]+$ means “ends with one or more of a digit or lowercase English letter”

```
$regex = "/[a-z0-9]$/";
echo preg_match($regex,"xy6");
echo preg_match($regex,"a@");
echo preg_match($regex,"*78");
echo preg_match($regex,"7*");
```

Output : 1010

^[a-z0-9]+$

“contains only one or more of a digit or lowercase English letter.”

```
$regex = "/^[a-z0-9]+$/";
echo preg_match($regex,"xy6");
echo preg_match($regex,"a@");
echo preg_match($regex,"*78");
echo preg_match($regex,"7*");
echo preg_match($regex,"7");
echo preg_match($regex,"50mg");
echo preg_match($regex,"user123")
```

Output : 1000111

Problem : Match Name

Problem : We have to write a RE for “Name”.

A name would have only a-z or A-Z characters, i.e. Robin, Alex, Rahul, Isha

String = “any name” ; // we have to match this pattern

Solution

```
$regex = "/^[a-zA-Z]+$/";

echo preg_match($regex,"Ram");
echo preg_match($regex,"user@123");
echo preg_match($regex,"user123");
echo preg_match($regex,"Jackson");
echo preg_match($regex,"seema");
echo preg_match($regex,"ragga1");
Output : 100110
```

Problem : Match Username(including _ and - but no digits)

Matches “kite” , “good-and-bad”, “robin_k”

But not “kite2” , “$50” , “3-d”

Solutions

```
$regex = "/^[a-zA-Z-_]+$/";
echo preg_match($regex,"Ram");
echo preg_match($regex,"_user");
echo preg_match($regex,"a_user");
echo preg_match($regex,"first-name");
echo preg_match($regex,"user12");
```

Output : 11110

Problem : Match username including (digits, - and _)

Solution

```
$regex = "/^[a-zA-Z-_0-9]+$/";
echo preg_match($regex,"Ram");
echo preg_match($regex,"_user");
echo preg_match($regex,"a_user");
echo preg_match($regex,"first-name");
echo preg_match($regex,"3-d");
echo preg_match($regex,"user12")
echo preg_match($regex,"user12#");
```

Output : 1111110

Rule 8 : ^ a caret inside a character class

[^aeiou] matches everything but not lowercase English vowels

opposite of [aeiou] isn't [bcdfghjklmnpqrstvwxyz]

[^aeiou] also matches uppercase vowels such as AEIOU, numbers such as 123, URLs

Rule 9 : The vertical bar (|) or Pipe

To find various possibilities , i.e. find image files (.jpg, .jpeg , .gif , .png)

```
$text = "The files are computer.gif, mouse.png,brush.jpg and books.jpeg";
if (preg_match_all('/[a-zA-Z0-9]+\.(gif|jpe?g|png)/',$text,$matches)) {
   print "The image files are: " . implode(',',$matches[0]);
}
```

Output : The image files are: computer.gif,mouse.png,brush.jpg,books.jpeg

Problem : Match a tag

```
$tag = "<a href=\"www.mozilla.org\">";#match a tag

$regex_tag  = "/<[^>]+>$/";

echo preg_match($regex_tag, $tag); # matched
```

Problem : Username version 2

```
$regex_username = "/[^a-zA-Z0-9_-]/";#here ^ symbol means negate everything inside brackets

$username = "aman";

echo preg_match($regex_username, $username);# if it return 0; then its fine.
```

Problem : Google, Gooogle, Goooooogle

Match spelling of Google ..

Solution

```
$regex_tag  = "/^Goo+gle$/";

echo preg_match($regex_tag, $tag);#1
```

Problem : alphabet.alphabet

File name with alphabets and extension with alphabets only.

```
$regex_file = "/[a-z]\.[a-z]/";

$file = "c.a";# characters.characters

echo "\n";

echo preg_match($regex_file, $file);
```

Problem : Title resolver

```
$regex = '/[Mr|Dr|Miss]\.\s[a-zA-Z]+/';
echo preg_match($regex,'e');
echo preg_match($regex,'Mr. A');
echo preg_match($regex,'Mr B');
echo preg_match($regex,'Mr. Rakesh');
echo preg_match($regex,'Miss. A');
```

Output : 01011

Problem : DNA pattern matching

```
$regex = '/GAATTC/';

$dna = "ATCGCGAATTCAC";

echo preg_match($regex,$dna); #1
```

```
# which can have two different sequences:
GGACC and GGTCC.

$regex = '/GGACC|GGTCC/';

$dna = "ATCGGACCATTCAC";

echo preg_match($regex,$dna);#1

$dna = "ATCGGTCCATTCAC";

echo preg_match($regex,$dna);#1
```

DNA Matching cont..

```
$regex = '/GG[A|T]CC/';

$dna = "ATCGGACCATTCAC";

echo preg_match($regex,$dna);#1

$dna = "ATCGGTCCATTCAC";

echo preg_match($regex,$dna);#1
```

OR

```
$regex = '/GG[AT]CC/';

$dna = "ATCGGACCATTCAC";

echo preg_match($regex,$dna);#1
```

DNA Matching cont..

```
$regex = '/GAT?C/';# the T is optional, and the pattern will match either GATC or GAC.

$dna = "ATCGGATCAA";

echo preg_match($regex,$dna);#1

$dna = "ATCGGACAA";

echo preg_match($regex,$dna);#1
```

Question (?) mark to more than one character

Group characters in parentheses

In the pattern `GGG(AAA)?TTT` the group of three As is optional, so the pattern will match either GGGAAATTT or GGGTTT

The pattern `GGGA+TTT` will match three Gs, followed by one or more As, followed by three Ts. So it will match GGGATTT, GGGAATT, GGGAAATT, etc. but not GGGTTT

The pattern `GGGA*TTT` will match three Gs, followed by zero or more As, followed by three Ts. So it will match GGGTTT, GGGATTT, GGGAATTT, etc

Rule 10 : Fixing the repetition by using {lower,upper }

The pattern GA{5}T will match GAAAAAT but not GAAAAT or GAAAAAAT

The pattern GA{2,4}T means G, followed by between 2 and 4 As, followed by T. So it will match GAAT, GAAAT and GAAAAT but not GAT or GAAAAAT

A{3,} will match three or more As, and G{,7} will match up to seven Gs

Problem : Match words not start with vowels

```
<?php
$regex = '/^[^aeiou]/';
$name = array("man","aman","illa", "Lilla","ullu", "kullu","owl",'roul');

foreach ($name as $value) {
   echo preg_match($regex,$value);
}
?>
```

Output : 10010101

preg_match_all() : locate all matches

```
$n = preg_match_all("/cats/i", "Cats are strange. I like cats.", $match);
echo "$n Matches: ";
for ($j=0 ; $j < $n ; ++$j)
   echo $match[0][$j]." ";
```

Output : 2 Matches: Cats cats

preg_replace()

```
$sentence =  "Cats are furry. I like cats.";

echo preg_replace("/cats/i", "dogs" , $sentence);

# dogs are furry. I like dogs
```

/i : not case sensitive

Problem : You want to pull out all words from a string.

```
$text = "Knock, knock. Who's there? r2d2!";

$words = preg_match_all('/\w+/', $text, $matches);

var_dump($matches[0]);
```

\w = word character

```
array(6) {
  [0]=>
  string(5) "Knock"
  [1]=>
  string(5) "knock"
  [2]=>
  string(3) "Who"
  [3]=>
  string(1) "s"
  [4]=>
  string(5) "there"
  [5]=>
  string(4) "r2d2"
}
```

Problem : Nth Match

```
$todo = "1. Wake up  2. Bathing  3. Breakfast 4. Office ";
preg_match_all("/\d\. ([^\d]+)/", $todo, $matches);
print "The second item on the list is: ";
// $matches[1] is an array of each substring captured by
([^\d]+)
print $matches[1][1] . "\n";
print "The entire list is: ";
foreach($matches[1] as $match) {
print "$match\n";
}
```

```
The second item on the list
is: Bathing
The entire list is: Wake up
Bathing
Breakfast
Office
```

Problem : RE for Email Id

Advantage : Extremely Flexible

Disadvantage : Slower than tring or other methods