Введение в регулярные выражения в PHP

Регулярные выражения - это такие команды для сложного поиска и замены. Существует несколько функций PHP для работы с регулярными выражениями. Мы начнем знакомится с ними на примере preg_replace. Эта функция первым параметром принимает что менять, а вторым - на что менять, а третьим параметром - строку, в которой нужно заменять:

```
<?php
	preg_replace(что менять, на что, строка);
?>
```

При этом первым параметром наша функция принимает не просто строку, а *регулярное выражение*, представляющее собой строку с набором команд, расположенных внутри символов решетки #. Эти решетки называются *ограничителями* регулярных выражений.

После ограничителей можно писать *модификаторы* - команды, которые изменяют общие свойства регулярного выражения.

Сами регулярные выражения состоят из двух типов символов: из тех, которые обозначают сами себя и из символов-команд, которые называются *специальные символы*.

Буквы и цифры обозначают сами себя. В следующем примере мы с помощью регулярного выражения заменим букву 'a' на '!':

```
<?php
	preg_replace('#a#', '!', 'bab'); // вернет 'b!b'
?>
```

А вот точка является специальным символом и обозначает *любой символ*. В следующем примере мы найдем строку по такому шаблону: буква 'x', затем любой символ, затем опять буква 'x':

```
<?php
	preg_replace('#x.x#', '!', 'xax eee'); // вернет '! eee'
?>
```

Задача 1

Дана строка:

```
<?php
	$str = 'ahb acb aeb aeeb adcb axeb';
?>
```

Напишите регулярку, которая найдет строки 'ahb', 'acb', 'aeb' по шаблону: *буква 'a', любой символ, буква 'b'*.

Задача 2

Дана строка:

```
<?php
	$str = 'ahb acb aeb aeeb adcb axeb';
?>
```

Напишите регулярку, которая найдет строки 'abba', 'adca', 'abea' по шаблону: *буква 'a', два любых символа, буква 'b'*.

Операторы повторения символов в регулярках

Бывают ситуации, когда мы хотим указать, что символ повторяется заданное количество раз. Если мы знаем точное число повторений, то можно просто написать его несколько раз - #aaaa#. Но что делать, если мы хотим сказать такое: *повторить один или более раз*?

Для этого существуют операторы *(квантификаторы)* повторения: плюс + (один и более раз), звездочка * (ноль или более раз) и вопрос ? (ноль или один раз). Эти операторы действуют на тот символ, который стоит перед ними.

Давайте посмотрим на работу этих операторов на примерах.

Пример

Найдем все подстроки по шаблону *буква 'x', буква 'a' один или более раз, буква 'x'*:

```
<?php
	$str = 'xx xax xaax xaaax xbx';
	$res = preg_replace('#xa+x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'xx ! ! ! xbx'
```

Пример

Найдем все подстроки по шаблону *буква 'x', буква 'a' ноль или более раз, буква 'x'*:

```php
<?php
	$str = 'xx xax xaax xaaax xbx';
	$res = preg_replace('#xa*x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'! ! ! ! xbx'
```

Пример

Найдем все подстроки по шаблону *буква 'x', буква 'a' ноль или один раз, буква 'x'*:

```php
<?php
	$str = 'xx xax xaax xbx';
	$res = preg_replace('#xa?x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'! ! xaax xbx'
```

Практические задачи

Задача 3

Дана строка:

```php
<?php
	$str = 'aa aba abba abbba abca abea';
?>
```

Напишите регулярку, которая найдет строки по шаблону: *буква 'a', буква 'b' один или более раз, буква 'a'*.

Задача 4

Дана строка:

```php
<?php
	$str = 'aa aba abba abbba abca abea';
?>
```

Напишите регулярку, которая найдет строки по шаблону: *буква 'a', буква 'b' ноль или более раз, буква 'a'*.

Задача 5

Дана строка:

```
<?php
	$str = 'aa aba abba abbba abca abea';
?>
```

Напишите регулярку, которая найдет строки по шаблону: *буква 'a', буква 'b' один раз или ниодного, буква 'a'*.

Задача 6

Дана строка:

```
<?php
	$str = 'aa aba abba abbba abca abea';
?>
```

Напишите регулярку, которая найдет строки 'aa', 'aba', 'abba', 'abbba', не захватив 'abca' и 'abea'.

Группирующие скобки в регулярках PHP

В предыдущих примерах операторы повторения действовали только на один символ, который стоял перед ними. Что делать, если мы хотим подействовать им на несколько символов?

Для этого существуют группирующие скобки '(' и ')'. Они работают так: если что-то стоит в группирующих скобках и сразу после ')' стоит оператор повторения - он подействует на все, что стоит внутри скобок.

Давайте посмотрим на примерах.

Пример

В следующем примере шаблон поиска выглядит так: *буква 'x', далее строка 'ab' один или более раз, потом буква 'x'*:

```
<?php
	$str = 'xabx xababx xaabbx';
	$res = preg_replace('#x(ab)+x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'! ! xaabbx'
```

Практические задачи

Задача 7

Дана строка:

```
<?php
	$str = 'ab abab abab abababab abea';
?>
```

Напишите регулярку, которая найдет строки по шаблону: *строка 'ab' повторяется 1 или более раз*.

Экранировка спецсимволов в регулярках PHP

Предположим, что мы хотим сделать так, чтобы спецсимвол обозначал сам себя. Для этого его нужно экранировать с помощью обратного слеша. Давайте посмотрим на примерах.

Пример

В следующем примере автор регулярки хотел, чтобы шаблон поиска выглядел так: *буква 'a', затем плюс '+', затем буква 'x'*. Однако, автор кода не заэкранировал символ '+' и поэтому шаблон поиска самом деле он выглядит так: *буква 'a' один или более раз, потом буква 'x'*:

```
<?php
	$str = 'a+x ax aax aaax';
	$res = preg_replace('#a+x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'a+x ! ! !'
```

Пример

А сейчас автор заэкранировал плюс обратным слешем. Теперь шаблон поиска выглядит так, как надо: *буква 'a', затем плюс '+', затем буква 'x'*.

```php
<?php
	$str = 'a+x ax aax aaax';
	$res = preg_replace('#a\+x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'! ax aax aaax'
```

Пример

В данном примере шаблон выглядит так: *буква 'a', затем точка '.', затем буква 'x'*:

```php
<?php
	$str = 'a.x abx azx';
	$res = preg_replace('#a\.x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'! abx azx'
```

Пример

А следующем примере автор забыл заэкранировать слеш и под регулярку попали все подстроки, так как незаэкранированная точка обозначает любой символ:

```php
<?php
	$str = 'a.x abx azx';
	$res = preg_replace('#a.x#', '!', $str);
?>
```

В результате в переменную запишется следующее:

```
'! ! !'
```

Замечание

Обратите внимание на то, что если вы забудете обратный слеш для точки (когда она должна обозначать сама себя) - этого можно даже не заметить:

```php
<?php
	preg_replace('#a.x#', '!', 'a.x'); // вернет '!', как мы и хотели
?>
```

Визуально работает правильно (так как точка обозначает любой символ, в том числе и обычную точку '.'). Но если поменять строку, в которой происходят замены - мы увидим нашу ошибку:

```
<?php
	preg_replace('#a.x#', '!', 'a.x abx azx'); // вернет '! ! !', а ожидалось '! abx azx'
?>
```

Список специальных символов и обычных

Если экранировать обычный символ - ничего страшного не случится - он все равно будет обозначать сам себя. Исключение - цифры, их нельзя экранировать.

Часто возникает сомнение, является ли данный символ специальным. Некоторые доходят до того, что экранируют все подозрительные символы подряд. Однако, это плохая практика (захламляет регулярку обратными слешами).

Являются спецсимволами: $ ^ . * + ? \ / {} [] () |

Не являются спецсимволами: @ : , ' " ; - _ = < > % # ~ ` & !

Практические задачи

Задача 8

Дана строка:

```
<?php
	$str = 'a.a aba aea';
?>
```

Напишите регулярку, которая найдет строку 'a.a', не захватив остальные.

Задача 9

Дана строка:

```
<?php
	$str = '2+3 223 2223';
?>
```

Напишите регулярку, которая найдет строку '2+3', не захватив остальные.

Задача 10

Дана строка:

```
<?php
	$str = '23 2+3 2++3 2+++3 345 567';
?>
```

Напишите регулярку, которая найдет строки '2+3', '2++3', '2+++3', не захватив остальные (+ может быть любое количество).

Задача 11

Дана строка:

```
<?php
	$str = '23 2+3 2++3 2+++3 445 677';
?>
```

Напишите регулярку, которая найдет строки '23', '2+3', '2++3', '2+++3', не захватив остальные.

Задача 12

Дана строка:

```
<?php
	$str = '*+ *q+ *qq+ *qqq+ *qqq qqq+';
?>
```

Напишите регулярку, которая найдет строки '*q+', '*qq+', '*qqq+', не захватив остальные.

Задача 13

Дана строка:

```
<?php
	$str = '[abc] {abc} abc (abc) [abc]';
?>
```

Напишите регулярку, которая найдет строки в квадратных скобках и заменят их на '!'.