19

- String Processing with Regular Expressions in PHP

19.7 String Processing with Regular Expressions

- Text manipulation is usually done with regular expressions—a series of characters that serve as *pattern-matching* templates (or search criteria) in strings, text files and databases.
- Function `preg_match` uses regular expressions to search a string for a specified pattern using Perl-compatible regular expressions (PCRE).
- If a pattern is found, `preg_match` returns the length of the matched string—which evaluates to `true` in a boolean context.
- *Anything enclosed in single quotes in a `print` statement is not interpolated, unless the single quotes are nested in a double-quoted string literal.*
- Function `preg_match` takes two arguments—a regular-expression pattern to search for and the string to search.
- Function `preg_match` performs *case-insensitive pattern matches*.

```
<!DOCTYPE html>

<!-- Fig. 19.9: expression.php -->
<!-- Regular expressions. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>Regular expressions</title>
      <style type = "text/css">
         p { margin: 0; }
      </style>
   </head>
   <body>
      <?php
         $search = "Now is the time";
         print( "<p>Test string is: '$search'</p>" );

         // call preg_match to search for pattern 'Now' in variable search
         if ( preg_match( "/Now/", $search ) )
            print( "<p>'Now' was found.</p>" );

         // search for pattern 'Now' in the beginning of the string
         if ( preg_match( "/^Now/", $search ) )
            print( "<p>'Now' found at beginning of the line.</p>" );

```

Fig. 19.9 | Regular expressions. (Part 1 of 3.)

```
26          // search for pattern 'Now' at the end of the string
27          if ( !preg_match( "/Now$/", $search ) )
28             print( "<p>'Now' was not found at the end of the line.</p>" );
29
30          // search for any word ending in 'ow'
31          if ( preg_match( "/\b([a-zA-Z]*ow)\b/i", $search, $match ) )
32             print( "<p>Word found ending in 'ow': " .
33                $match[ 1 ] . "</p>" );
34
35          // search for any words beginning with 't'
36          print( "<p>Words beginning with 't' found: " );
37
38          while ( preg_match( "/\b(t[[:alpha:]]+)\b/", $search, $match ) )
39          {
40             print( $match[ 1 ] . " " );
41
42             // remove the first occurrence of a word beginning
43             // with 't' to find other instances in the string
44             $search = preg_replace("/" . $match[ 1 ] . "/", "", $search);
45          } // end while
46
47          print( "</p>" );
48       ?><!-- end PHP script -->
49    </body>
50 </html>h
```

Fig. 19.9 | Regular expressions. (Part 2 of 3.)

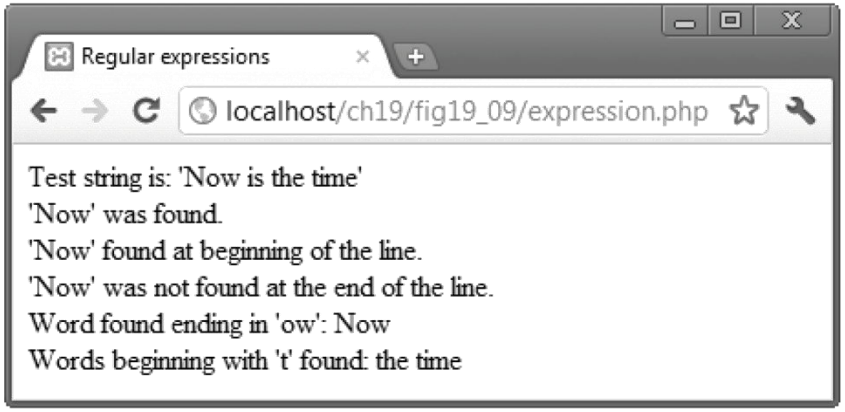

Fig. 19.9 | Regular expressions. (Part 3 of 3.)

19.7.2 Representing Patterns

- Regular expressions can include metacharacters such as ^, $ and . that specify patterns.
- For example, the caret (^) metacharacter matches the beginning of a string, while the dollar sign ($) matches the end of a string.
- The period (.) metacharacter matches any single character.
- Bracket expressions are lists of characters enclosed in square brackets ([]) that match any single character from the list.
- Ranges can be specified by supplying the beginning and the end of the range separated by a dash (-).

19.7.2 Representing Patterns

- The \b before and after the parentheses indicates the beginning and end of a word, respectively—in other words, we're attempting to match whole words.
- Quantifiers are used in regular expressions to denote how often a particular character or set of characters can appear in a match.

Quantifier	Matches
{*n*}	Exactly *n* times
{*m*,*n*}	Between *m* and *n* times, inclusive
{*n*,}	*n* or more times
+	One or more times (same as {1,})
*	Zero or more times (same as {0,})
?	Zero or one time (same as {0,1})

Fig. 19.10 | Some regular expression quantifiers.

19.7.3 Finding Matches

- The optional third argument to function preg _match is an array that stores matches to each parenthetical statement of the regular expression.
- The first element stores the string matched for the entire pattern, and the remaining elements are indexed from left to right.
- To find multiple instances of a given pattern, we must make multiple calls to `preg_match`, and remove matched instances before calling the function again by using a function such as `preg_replace`.

19.7.4 Character Classes

- Character classes are enclosed by the delimiters `[:` and `:]`.
- When this expression is placed in another set of brackets, it is a regular expression matching all of the characters in the class.
- A bracketed expression containing two or more adjacent character classes in the class delimiters represents those character sets combined.

Character class	Description
`alnum`	Alphanumeric characters (i.e., letters [a-zA-Z] or digits [0-9])
`alpha`	Word characters (i.e., letters [a-zA-Z])
`digit`	Digits
`space`	White space
`lower`	Lowercase letters
`upper`	Uppercase letters

Fig. 19.11 | Some regular expression character classes.

19.7.5 Finding Multiple Instances of a Pattern

- Function `preg_replace` takes three arguments—
  - the pattern to match,
  - a string to replace the matched string and
  - the string to search. The modified string is returned.