Berner Fachhochschule - HTI

PHP, Reg Exp

Dr. E. Benoist

Winter Term 2006-2007

PHP Regular Expressions

- Motivation
- Find a pattern
  `preg_match()`
- Find and Replace Patterns
  `preg_replace()`
- Usefull functions
  `preg_split()`
  `preg_quote()`
  `preg_grep()`
- Pattern Syntax

Regular Expressions

What to do?

- Search an expression
- Parse a file
- Scan an input and replace the content

Read a File

Example

```
<?
// get a web page into an array
//$fcontents = file ('http://www.libe.com');

// Read a File from the local disk into an array
$fcontents = file ('testFile.php');

// Print out the array
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b> " .
        htmlspecialchars ($line) . "<br>\n";
}
?>
```

Functionalities of Regular Expressions

Main functions:

- **preg_match()**
- **preg_match_all()** Correspondance Operator
- **preg_replace()** Substitution Operator
- **preg_split()** Split Operator
- **preg_quote()** Quote regular expression characters

preg_match()

Perform a regular expression match

int preg_match (string pattern, string subject
[, array matches])

- Searches subject for a match to the regular expression given in pattern.
- If `matches` is provided, then it is filled with the results of search. `$matches[0]` will contain the text that match the full pattern, `$matches[1]` will have the text that matched the first captured parenthesized subpattern, and so on.
- Returns true if a match for pattern was found in the subject string, or false if not match was found or an error occurred.

preg_match() (Cont.)

Example 1

```
// the "i" after the pattern delimiter indicates
// a case-insensitive search
if (preg_match ("/php/i",
                    "PHP est le meilleur langage pour le web.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

preg_match() (Cont.)

Example 2

```
// the \b in the pattern indicates a word boundary,
// so only the distinct word "web" is matched,
// and not a word partial like "webbing" or "cobweb"
if (preg_match ("/\bweb\b/i",
                "PHP a the web scripting language")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
if (preg_match ("/\bweb\b/i",
                "PHP is a website scripting language")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

preg_match_all()

Perform a global regular expression match

int preg_match_all (string pattern, string subject,
array matches [, int order])

Searches subject for all matches to the regular expression given in pattern and puts them in matches in the order specified by order. After the first match is found, the subsequent searches are continued on from end of the last match.

preg_match_all() (Cont.)

Order can be one of two things

- **PREG_PATTERN_ORDER**
  Orders results so that `$matches[0]` is an array of full pattern matches, `$matches[1]` is an array of strings matched by the first parenthesized subpattern, and so on.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
        "<b>example: </b><div align=left>this is a test</div>",
        $out, PREG_PATTERN_ORDER);
print $out[0][0]." , ".$out[0][1]."\n";
print $out[1][0]." , ".$out[1][1]."\n";

# Output
# <b>example: </b>, <div align=left>this is a test</div>
# example: , this is a test
```

preg_match_all() (Cont.)

- **PREG_SET_ORDER**
  Orders results so that `$matches[0]` is an array of first set of matches, `$matches[1]` is an array of second set of matches, and so on.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
     $out, PREG_SET_ORDER);
print $out[0][0]. ", ". $out[0][1]. "\n";
print $out[1][0]. ", ". $out[1][1]. "\n";

# Output
# <b>example: </b>, example:
# <div align=left>this is a test</div>, this is a test
```

preg_replace()

Perform a regular expression search and replace

mixed preg_replace (mixed pattern, mixed replacement,
mixed subject [, int limit])

Searches subject for matches to pattern and replaces them with replacement . If limit is specified, then only limit matches will be replaced; if limit is omitted or is -1, then all matches are replaced.

Example 1

```
$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
                   "/^\s*{(\w+)}\s*=/");
// Use a backward reference on the regExp \1 = first ()
$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");
print preg_replace ($patterns, $replace,
                    "{startDate} = 1999-5-27");

# Output
# $startDate = 5/27/1999
```

preg_replace() (Cont.)

Example 2 (Using /e modifier)

This would capitalize all HTML tags in the input text.

```
preg_replace ("/(<\/?)(\w+)([^>]*>)/e",
                "'\\1'.strtoupper('\\2').'\\3'",
              $html_body);
```

preg_split()

Split string by a regular expression

array preg_split (string pattern, string subject [, int limit [, int flags]])

Returns an array containing substrings of subject split along boundaries matched by pattern.

Examples

```
// Get the parts of a search string.
// split the phrase by any number of commas or space characters,
// which include " ", \r, \t, \n and \f
$keywords = preg_split ("/[\s,]+/",
                        "hypertext language, programming");
// Splitting a string into component characters.
$str = 'string';
$chars = preg_split('//', $str, 0, PREG_SPLIT_NO_EMPTY);
print_r($chars);
```

preg_quote()

Quote regular expression characters

string preg_quote (string str [, string delimiter])

preg_quote() takes str and puts a backslash in front of every character that is part of the regular expression syntax.

```
// In this example, preg_quote($word) is used to keep the
// asterisks from having special meaning to the regular
// expression.
$textbody = "This book is *very* difficult to find.";
$word = "*very*";
$textbody = preg_replace ("/".preg_quote($word)."/",
                          "<i>".$word."</i>",
                          $textbody);
```

preg_grep()

Return array entries that match the pattern

array preg_grep (string pattern, array input)

preg_grep() returns the array consisting of the elements of the input array that match the given pattern.

```
// return all array elements
// containing floating point numbers
$fl_array = preg_grep ("/^(\d+)?\.\d+$/", $array);
```

Subpatterns

Subpatterns are delimited by parentheses (round brackets), which can be nested.
Marking part of a pattern as a subpattern does two things:

- It localizes a set of alternatives.

  // This will match any of:
  // "cataract", "caterpillar" or "cat"
  /cat(aract|erpillar|)/

- It sets up the subpattern as a capturing subpattern

  /the ((red|white) (king|queen)) (wins)/

  If the string "the red king wins" is pased to this RegExp, the captured substring are "red king", "red", "king", "wins" and are numbered 1, 2, 3 and 4.

Patern Modifiers

Modificator	Meaning
i	letters in the pattern match both upper and lower case letters
m	Treat the string like multiple lines.
s	Treat the string like a single line.
x	ignores spaces and comments
e	(only used by *preg_replace()*)
...	...

Characters and Quantificators

Specials Characters

Character	Meaning
^	search the begining of a string
$	seach the end of the string
\b	search the extremity of a word
\n	line feed
\r	carriage return
\t	tabulation
\f	newpage
\s	space = [\t\n\r\f]
\S	any character not a Space
\e	escape
\d	digit, equal to :[0-9]
\D	non numeric
\w	alpha-numerical character (for word) = [0-9a-zA-Z]
\W	character not in a word

Characters and Quantificators

Specials Characters

The POSIX norm defines some named classes of chararcters.

Class	Matches
`[[:alnum:]]`	Alphanumeric characters
`[[:alpha:]]`	Alphabetic characters
`[[:lower:]]`	Lower case
`[[:upper:]]`	Uppercase
`[[:digit:]]`	Decimal digit
`[[:xdigit:]]`	Hexadecimal digit
`[[:punct:]]`	Punctuation
`[[:blank:]]`	Tabs and spaces
`[[:space:]]`	Whitespace character
`[[:cntrl:]]`	Control characters
`[[:print:]]`	All printables characters
`[[:graph:]]`	All printable characters except for space

Characters and Quantificators (Cont.)

Quantificators

Expression	example	Meaning
[]	[0-9a-z]	interval
$*$	$\backslash w*$	Any repetition (≥ 0)
$+$	$\backslash w+$	Any repetition (≥ 1)
$\{n, m\}$	$\backslash w\{10, 15\}$	between n and m times
$\{n,\}$	$\backslash w\{10,\}$	at least n times
$\{n\}$	$\backslash w\{10\}$	exactly n times
?	$\backslash .?$	zero or one time

Examples

Usefull regular Expressions

```
# match any empty line
/^$/
# match any email address
/\w+\.?\w*\@(\w+|\.){1,3}\w{2,3}/
# match any line with at least 80 characters
/.{80,}/
```

Corresponding variables

```
// get an access log file into an array and scan it
$fcontents = file ('/home/bie/logs/access_log');
$directoriesGET = array();
while (list ($line_num, $line) = each ($fcontents)) {
    // Print out the line
    echo "<b>Line $line_num:</b> ". htmlspecialchars ($line)."<br>\n";
    // First test, we are looking for the IP adress of Altair.
    if (preg_match ("/147.87.65.34/",$line)){
        echo "comes from me<br>\n";
    }
    // Get element matched by a RegExp with $matches
    // Any letter or digit : \w
    // Any number of repetition : *
    if (preg_match ("/GET \/((\w|~)*)/",$line,$matches)){
        print "GET directory = ".$matches[1]."<br>\n";
        $directoriesGET[$matches[1]]++;
    }
}
```

Sets With Negation

```
$fcontents = file ('/home/bie/logs/access_log');

$directories = array();

foreach ($fcontents as $line_num => $line) {
      // RegExp with:
     // - OR : |
     // Set with negation [^ ... ]
     if (preg_match ("/(GET|POST) \/([^\/\s]*)/",$line, $matches)){
          print "GET or POST directory = ".$matches[2]."<br>\n";
          $directories[$matches[2]]++;
     }
}
foreach($directories as $dir => $number) {
    $percentOfGET = (int)(($directories[$dir]/$number)*100);
    print "Directory $dir was seen $number time(s)".
              " ($percentOfGET % of GET)<br>\n";
}
```

Example (Cont)

Pattern repetition

```
// Regexp with repetitions
if (preg_match("/^((\d{1,3}\.){3}(\d{1,3}))/",
                    $line, $matches)){
    print "IP adress = $matches[1]<br>\n";
}
```

Pattern repetition

```
foreach($fcontents as $line_num => $line) {
    if(preg_match(
          "/\"(GET|POST) ([^\?]*)\??(.*)? (HTTP\/\d\.\d)\"/i",
          $line, $matches)){
        print "URL = $matches[2]; Param = $matches[3]; ".
              "Protocol = $matches[4]<br>\n";

        if(preg_match("/[\w_-~\/\.]*(html|jsp|htm|php)/i",
                      $matches[2],$urlContent)){
            print "Suffix = $urlContent[1]<br>\n";
            $suffixes[$urlContent[1]]++;
        }
        // We want to split the URL and get its suffix
        $url_content = preg_split("/[\.\/]/",$matches[2]);
        $suffix = $url_content[count($url_content)-1];
        print "Suffix = $suffix<br>\n";
        $suffixesBis[$suffix]++;
    }
}
```

Pattern repetition (Cont.)

```
while (list ($suf, $number) = each ($suffixes)) {
    print "Suffix $suf was seen $number time(s)<br>\n";
}
while (list ($suf, $number) = each ($suffixesBis)) {
    print "$suf -> $number time(s)<br>\n";
}
```

Example

```
$ok_html = "I <b>love</b> shrimps dumplings.";
$bad_html = "I <b>love</i> shrimps dumplings.";
if (preg_match('@<([bi])>.*?</\1>@', $ok_html)) {
    print ("Good for you! (OK; Backreferences)\n");
}
if (preg_match('@<([bi])>.*?</\1>@', $bad_html)) {
    print ("Good for you! (BAD; Backreferences)\n");
}
if (preg_match('@<[bi]>.*?</[bi]@', $ok_html)) {
    print ("Good for you! (OK; No Backreferences)\n");
}
if (preg_match('@<[bi]>.*?</[bi]@', $bad_html)) {
    print ("Good for you! (BAD; No Backreferences)\n");
}
```

Example

This example prints:

Good for you! (OK; Backreferences)
Good for you! (OK; No Backreferences)
Good for you! (BAD; No Backreferences)

Example

```
$members=<<<TEXT

Name E-Mail Address<br>
--------------------------------------------
Inky T. Ghost inky@pacman.example.com<br>
Donkey K. Gorilla kong@banana.example.com<br>
Mario A. Plumber mario@franchise.example.org<br>
TEXT;

print preg_replace('/([^@\s]+)@(([-a-z0-9]+\.)+[a-z]{2,})/',
                   '\\1 at \\2', $members);
```

This examples prints

```
Name E-Mail Address
--------------------------------------------
Inky T. Ghost inky at pacman.example.com
Donkey K. Gorilla kong at banana.example.com
Mario A. Plumber mario at franchise.example.org
```

Split

Send the content of a String into an Array

```
$line = "142.34.123.12";
@address= split /\./ , $line;
foreach $l(@address){
   print $l." ";
}
print "\n";
# Output:
# 142 34 123 12

$line = 'emmanuel.benoist@bfh.ch';
@address= split /[\.|\@]/ , $line;
foreach $l(@address){
   print $l." ";
}
print "\n";
# Output:
# emmanuel benoist bfh ch
```