Systems Programming & Scripting

Lecture 18: Regular Expressions in PHP

Regular Expressions

- As with most scripting languages, PHP has good support for regular expressions
- For the notation of basic regular expressions see Lecture 12
- Here we discuss aspects specific to PHP
- Regular expressions are often used when iterating over the contents of a file, etc
- Regular expression are not only used to match strings, but also to replace

PHP & Regular Expressions

- PHP includes three sets of functions that support regular expressions.
- POSIX regular expressions: regex function set
- Extended regular expressions: (mb_)ereg function set
- Perl-style regular expressions: preg function set

The ereg Function Set

- Regular expressions are passed as strings.
- Supported in PHP 3, 4, 5.
- Part of core PHP.

```
int ereg (string pattern, string subject [, array groups])
```

- Checks if *pattern* matches *subject.*
- Returns TRUE (same as 1) on a successful match or FALSE (same as 0) if there is no match.
- If the 3rd parameter is specified, the function will store the substrings matched in the elements of the array.

Examples

- `ereg('[0-9]+', 'the number is 432');`
- Matches one or more digits
- `ereg('pet.*(cat|dog)$', 'my pet is a cat');`
- Matches pet, followed by any sequence of characters, and cat or dog at the end
- `ereg('pet.*(cat|dog)$', 'my pet is a cat', $matched);`
- As above, storing the matched sub-expressions in the array matched
- Result: $matched = array ('my pet is a cat', 'cat')

The ereg Function Set (cont'd)

```
string ereg_replace(string pattern, string
  replacement, string subject)
```

- Replace all matches of *pattern* with *replacement.*

```
array split(string pattern, string subject [,
  int limit)
```

- Splits *subject* into an array of strings using the regular expression *pattern*

Ref: http://www.php.net/manual/en/function.ereg.php

Examples

- `ereg_replace('USER', 'hwloidl', 'insert the user name here: USER');`
- Replaces USER by hwloidl in the string
- `$matched = split(',','one,two,three');`
- Splits at each ',' in the string
- Result: $matched is array('one','two','three')
- `$terms = split('[/+*-]', '3*5+i/6-12')`
- Result: $term is array('3', '5', 'i', '6', '12')

The mb-ereg Function Set

- Very similar to the ereg function set with one difference:
  - ereg functions treat the regular expression and the subject strings as *8-bit characters*.
  - mb_ereg supports *multi-byte characters* with various encodings.

Ref: http://uk2.php.net/manual/en/function.mb_ereg.php

The preg Function Set

- Regular expressions specified as a string using Perl syntax.
- Functions include:
  - `preg_match`
  - `preg_match_all`
  - `preg_grep`
  - `preg_replace`
  - `preg_replace_callback`

Ref: http://www.php.net/manual/en/function.preg-match.php

Perl Regular Expressions

- Patterns are delimited by / eg `/pattern/`
- Additional character classes are provided:
  - \s whitespace; \S non-whitespace
  - \w word ; \W non-word
  - \d digit ; \D non-digit
- Trailing options can modify the behaviour:
  - /regexp/i … match case insensitive
  - /regexp/x … remove whitespace and comments
  - /regexp/e …if string is PHP code, evaluate it

Examples

- `preg_match('/here\s*(\w+)/', 'match word after here test', $matches);`
- $matches[1] is 'test'
- `preg_match('/I like (?i:PHP)/', 'I like pHp');`
- ?i makes match against PHP case-insensitive
- `preg_replace('/(\w)\w*\s*(\w+)/', '\2, \1.', array('Will Smith', 'John Doe'));`
- Result: array ('Smith, W.', 'Doe, J.')
- `preg_replace('/<.*?>/', '!', 'beware of <b>this</b>')`
- Result: 'beware of !this!'

Examples

- `preg_split('{[+*/-]}', '3+5*9/2');`
- Result: array ('3', '5', '9', '2');
- `preg_grep('/\.txt$/', $filenames);`
- Result: array of filenames ending with .txt
- `preg_quote('/usr/local/etc/resolv/conf', '/');`
- Creates a regular expression, matching exactly this string, escaping every '/'
- Many, many more features...

Example: Credit Card Validation

```
<?php
// Credit Card Validator from
// "Programming PHP", Rasmus Lerdorf et al, Chapter 4

function isValidCreditCard ($inCardNumber, $inCardType) {
  // assume it is ok
  $isValid = true;

  // strip all non-numbers
  $inCardNumber = ereg_replace('[^0-9]', '',
                               $inCardNumber);
  // $inCardNumber = ereg_replace('[^[:digit:]]', '',
                                  $inCardNumber);
```


Example: Credit Card Validation

```
// make sure card number and type match
switch ($inCardType) {
case 'mastercard':
  $isValid = ereg('^5[1-5].{14}$', $inCardNumber);
  break;
case 'visa':
  $isValid = ereg('^4.{15}|^4.{12}$', $inCardNumber);
  break;
case 'amex':
  $isValid = ereg('^3[47].{13}$', $inCardNumber);
  break;
case 'discover':
  $isValid = ereg('^6011.{12}$', $inCardNumber);
  break;
case 'diners':
  $isValid = ereg('^30[0-5].{11}|^3[68].{12}$',
                  $inCardNumber);
  break;
}
```


Example: Credit Card Validation

```
if ($isValid) {
  // reverse the string
  $inCardNumber = strrev($inCardNumber);

  // total the digits in the number, doubling in odd positions
  $theTotal = 0;
  for ($i = 0; $i <strlen($inCardNumber); $i++) {
    $theAdder = (int) $inCardNumber[$i];

    // double if odd numbered position
    if ($i % 2) {
       $theAdder = $theAdder << 1;
       if ($theAdder > 9) { $theAdder -= 9; }
    }
    $theTotal += $theAdder;
  }
  // Valid cards will divide evenly by 10
  $isValid = (($theTotal % 10) == 0);
}
return $isValid;
} ?>
```


Wrapper: Credit Card Validation

```
<html> ...
<body>
<h1>Credit card validation</h1>

<?php
include 'cc_valid.php';

$debug = 1;

$cc_number = $_GET['number'];
$cc_type   = $_GET['type'];
$self = basename($_SERVER['PHP_SELF']);

if ($debug){
  print "<font size=-1 color=grey>\n";
  print "<p>Received: number=$cc_number;type=$cc_type<br>\n";
  print "</font>\n";
}
```

Wrapper: Credit Card Validation

```
// did we receive a number to validate
if ($cc_number == "" || $cc_type == "") {
  // No: show the form to enter one
?>

<h2>Enter Number and Type of your Card:</h2>
<p>
<form action="<?php echo $self; ?>" method="get">
<p>Credit Card Number: <input type="text" name="number" /></
p>
<p>Credit Card Type: <input type="text" name="type" /></p>
<p><input type="submit" value="validate" /></p> </form>
```


Wrapper: Credit Card Validation

```
 <?php
} else { // Yes: check the number
?>
<h2>The result of validation is:</h2>

<?php
  if (isValidCreditCard ($cc_number, $cc_type)) {
    echo "The credit card $cc_number of type $cc_type is
valid. Congrats!";
  } else {
    echo "I am sorry, but the credit card $cc_number of type
$cc_type is NOT valid.";
  }
}
?>
</body>
</html>
```


Exercise

- Check the web page at
  http://www.macs.hw.ac.uk/~hwloidl/SPS/te
  work through the regular expression examples in the first section, and then do the regular expression exercise described in the second section. There is an PHP template that you have to complete for this exercise.