

BSc.(PS)II 26/3/2020

Regular expressions continued...

Reading Records with a Pattern Separator:

Problem:

You want to read in records from a file, in which each record is separated by a pattern you can match with a regular expression.

Solution:

Read the entire file into a string and then split on the regular expression:

```
$contents = file_get_contents('/path/to/your/file.txt');  
$records = preg_split('/[0-9]+ \) /', $contents);
```

Discussion

This breaks apart a numbered list and places the individual list items into array elements.

So if you have a list like this:

- 1) Godel
- 2) Escher
- 3) Bach

you end up with a four-element array, with an empty opening element. That's because

`preg_split()` assumes the delimiters are between items, but in this case, the numbers are before items:

```
array(4) {  
  [0]=>  
  string(0) ""
```

```
[1]=>
string(7) "Gödel
"
```

```
[2]=>
string(7) "Escher
"
```

```
[3]=>
string(5) "Bach
"
}
```

From one point of view, this can be a feature, not a bug, because the n th element holds the n th item. But, to compact the array, you can eliminate the first element:

```
$records = preg_split('/[0-9]+ \) /', $contents);
array_shift($records);
```

Note:

PHP array_shift() is an inbuilt function that removes the first element from an **array** and returns the value of a removed item.

Another modification you might want is to strip newlines from the elements and substitute the empty string instead:

```
$records = preg_split('/[0-9]+ \) /', str_replace("\n", "", $contents));
array_shift($records);
```

PHP doesn't allow you to change the input record separator to anything other than a newline, so this technique is also useful for breaking apart records divided by strings.

For preg_split : <https://www.youtube.com/watch?v=rMOUDoB1oq4>

Using a PHP Function in a Regular Expression

Problem

You want to process matched text with a PHP function. For example, you want to decode all HTML entities in captured subpatterns.

Solution

Use `preg_replace_callback()`.

Instead of a replacement pattern, give it a callback function.

This callback function is passed an array of matched subpatterns and should return an appropriate replacement string.

Example 23-15 decodes entities between

`<code> </code>` tags.

Example 23-15. Generating replacement strings with a callback function

```
$h = 'The &lt;b&gt; tag makes text bold:
```

```
<code> &lt;b&gt;bold&lt;/b&gt;</code>';
```

```
print preg_replace_callback('@ <code>(.*?)</code> @','decode', $h);
```

```
// $matches[0] is the entire matched string
```

```
// $matches[1] is the first captured subpattern
```

```
function decode($matches) {
```

```
return html_entity_decode($matches[1]);
```

```
}
```

Example 23-15 prints:

The tag makes text bold: bold

Discussion

The second argument to `preg_replace_callback()` specifies the function that is to be called to calculate replacement strings. Like everywhere the PHP “callable” pseudotype is used, this argument can be a string or an array.

Use a string to specify a function name.

To use an object instance method as a callback, pass an array whose first element is the object and whose second element is a string containing the method name.

To use a static class method as a callback, pass an array of two strings: the class name and the method name.

In PHP 5.4.0 and later, you can pass a variable containing an anonymous function, or define the function inline with the call to `preg_replace_callback()`.

The callback function is passed one argument: an array of matches. Element 0 of this array is always the text that matched the entire pattern. If the pattern given to `preg_replace_callback()` has any parenthesized subpatterns, these are present in subsequent elements of the `$matches` array.

The keys of the `$matches` array are numeric, even if there are named subpatterns in the pattern.

If you are providing an anonymous function as a callback, it can be memory intensive if the function creation is inline with the call to `preg_replace_callback()` and inside a loop.

If you want to use an anonymous function with `preg_replace_callback()`, store the anonymous function callback in a variable. Then, provide the variable to `preg_replace_callback()` as the callback function.

Example 23-16 uses an anonymous function to apply the transformation in Example 23-15 to every line in a file.

Example 23-16. Generating replacement strings with an anonymous function

```

$callbackFunction = function($matches) {
return html_entity_decode($matches[1]);
};
$fp = fopen(__DIR__ . '/html-to-decode.html','r');
while (! feof($fp)) {
$line = fgets($fp);
print
preg_replace_callback('@ <code>(.*?)</code> @',$callbackFunction,
$line);
}
fclose($fp);

```

Example 23-16 uses the anonymous function declaration syntax introduced in PHP 5.3.0.

If you're using an older version of PHP, you can use `create_function()` to build your callback, as follows:

```

$callbackFunction = create_function('$matches','return
html_entity_decode($matches[1]);');
$fp = fopen(__DIR__ . '/html-to-decode.html','r');
while (! feof($fp)) {
$line = fgets($fp);
print
preg_replace_callback('@ <code>(.*?)</code> @',$callbackFunction,$line);
}
fclose($fp);

```

So you can avoid using it, you should be aware of the e pattern modifier. This causes the replacement string to be evaluated as PHP code. This pattern modifier is deprecated as of PHP 5.5.0. Using the e modifier opens up remote code execution security vulnerabilities when user input is part of the text that `preg_replace()` operates on. If you encounter code using `preg_replace()` with the e modifier, convert it to use `preg_replace_callback()` instead.

Refer this link for more help: <https://www.phptutorial.info/?preg-replace-callback>

For any doubt contact 9873961590