Mastering Regular Expressions in PHP

What are Regular Expressions?

A regular expression is a pattern that can match various text strings. Using regular expressions you can find (and replace) certain text patterns, for example "all the words that begin with the letter A" or "find only telephone numbers". Regular expressions are often used in validation classes, because they are a really powerful tool to verify e-mail addresses, telephone numbers, street addresses, zip codes, and more.

In this tutorial I will show you how regular expressions work in PHP, and give you a short introduction on writing your own regular expressions. I will also give you several example regular expressions that are often used.

Regular Expressions in PHP

Using regex (regular expressions) is really easy in PHP, and there are several functions that exist to do regex finding and replacing. Let's start with a simple regex find.

Have a look at the documentation of the preg match function (http://php.net/preg match). As you can see from the documentation, preg match is used to perform a regular expression. In this case no replacing is done, only a simple find. Copy the code below to give it a try.

```
<?php

// Example string
 str = "Let's find the stuff <bla>in between</bla> these two previous brackets";

// Let's perform the regex
 do = preg match("/<bla>(. )< /bla>/",  str,  matches);

// Check if regex was successful
if ( do = true) {
 // Matched something, show the matched string
 echo htmlentities( matches '0' );

 // Also how the text in between the tags
 echo '<br />' .  matches '1' ;
} else {
 // No Match
 echo "Couldn't find a match";
}

?>
```

After having run the code, it's probably a good idea if I do a quick run through the code. Basically, the whole core of the above code is the line that contains the preg match. The first argument is your regex pattern. This is probably the most important. Later on in this tutorial, I will explain some basic regular expressions, but if you really want to learn regular expression then it's best if you look on Google for specific regular expression examples.

The second argument is the subject string. I assume that needs no explaining. Finally, the third argument can be optional, but if you want to get the matched text, or the text in between something, it's a good idea to use it (just like I used it in the example).

The preg match function stops after it has found the first match. If you want to find ALL matches in a string, you need to use the preg match all function (http://www.php.net/preg match all). That works pretty much the same, so

there is no need to separately explain it.

Now that we've had finding, let's do a find-and-replace, with the preg replace function (http://www.php.net/preg replace). The preg replace function works pretty similar to the preg match function, but instead there is another argument for the replacement string. Copy the code below, and run it.

```
<?php

// Example string
 str = "Let's replace the <bla>stuff between</bla> the bla brackets";

// Do the preg replace
 result = preg replace ("/<bla>(. )< /bla>/", "<bla>new stuff</bla>",  str);

echo htmlentities( result);
?>
```

The result would then be the same string, except it would now say 'new stuff' between the bla tags. This is of course just a simple example, and more advanced replacements can be done.

You can also use keys in the replacement string. Say you still want the text between the brackets, and just add something? You use the 1, 2, etc keys for those. For example:

```
<?php

// Example string
 str = "Let's replace the <bla>stuff between</bla> the bla brackets";

// Do the preg replace
 result = preg replace ("/<bla>(. )< /bla>/", "<bla>new stuff (the old:  1)</bla>",  str);

echo htmlentities( result);
?>
```

This would then print "Let's replace the new stuff (the old: stuff between) the bla brackets". 2 is for the second "catch-all", 3 for the third, etc.

That's about it for regular expressions. It seems very difficult, but once you grasp it is extremely easy yet one of the most powerful tools when programming in PHP. I can't count the number of times regex has saved me from hours of coding difficult text functions.

An Example

What would a good tutorial be without some real examples? Let's first have a look at a simple e-mail validation function. An e-mail address must start with letters or numbers, then have a @, then a domain, ending with an extension. The regex for that would be something like this: a-zA-Z0-9 . - +@ a-zA-Z0-9 - + . a-zA-Z0-9 - . +

Let me quickly explain that regex. Basically, the first part says that it must all be letters or numbers. Then we get the @, and after that there should be letters and/or numbers again (the domain). Finally we check for a period, and then for an extension. The code to use this regex looks like this:

```
<?php

// Good e-mail
 good = "john@example.com";
```

```
// Bad e-mail
 bad = "blabla@blabla";

// Let's check the good e-mail
if (preg match("/  a-zA-Z0-9  . - +@ a-zA-Z0-9 - + . a-zA-Z0-9 - . + /",  good)) {
 echo "Valid e-mail";
} else {
 echo "Invalid e-mail";
}

echo '<br />';

// And check the bad e-mail
if (preg match("/  a-zA-Z0-9  . - +@ a-zA-Z0-9 - + . a-zA-Z0-9 - . + /",  bad)) {
 echo "Valid e-mail";
} else {
 echo "Invalid e-mail";
}

?>
```

The result of this would be "Valid E-mail. Invalid E-mail", of course. We have just checked if an e-mail address is valid. If you wrap the above code in a function, you've got yourself a e-mail validation function. Keep in mind though that the regex isn't perfect: after all, it doesn't check whether the extension is too long, does it? Because I want to keep this tutorial short, I won't give the full fledged regex, but you can find it easily via Google.

Another Example

Another great example would be a telephone number. Say you want to verify telephone numbers and make sure they were in the correct format. Let's assume you want the numbers to be in the format of xxx-xxxxxxx. The code would look something like this:

```
<?php

// Good number
 good = "123-4567890";

// Bad number
 bad = "45-3423423";

// Let's check the good number
if (preg match("/ d{3}- d{7}/",  good)) {
 echo "Valid number";
} else {
 echo "Invalid number";
}

echo '<br />';

// And check the bad number
if (preg match("/ d{3}- d{7}/",  bad)) {
 echo "Valid number";
} else {
 echo "Invalid number";
}

?>
```

The regex is fairly simple, because we use d. This basically means "match any digit" with the length behind it. In this example it first looks for 3 digits, then a '-' (hyphen) and finally 7 digits. Works perfectly, and does exactly what we want.

What exactly is possible with Regular Expressions?

Regular expressions are actually one of the most powerful tools in PHP, or any other language for that matter (you can use it in your mod rewrite rules as well!). There is so much you can do with regex, and we've only scratched the surface in this tutorial with some very basic examples.

If you really want to dig into regex I suggest you search on Google for more tutorials, and try to learn the regex syntax. It isn't easy, and there's quite a steep learning curve (in my opinion), but the best way to learn is to go through a lot of examples, and try to translate them in plain English. It really helps you learn the syntax.

In the future I will dedicate a complete article to strictly examples, including more advanced ones, without any explanation. But for now, I can only give you links to other tutorials:

The 30 Minute Regex Tutorial (http://www.codeproject.com/dotnet/RegexTutorial.asp)

Regular-Expressions.info (http://www.regular-expressions.info/)

Short note about the author

Dennis Pallett is a young tech writer, with much experience in ASP, PHP and other web technologies. He enjoys writing, and has written several articles and tutorials. To find more of his work, look at his websites at http://www.phpit.net, http://www.aspit.net and http://www.webdev-articles.com

dennis@nocertainty.com
