php半小时精通正则表达式

php半小时精通正则表达式

想必很多人都对正则表达式都头疼。今天，我以我的认识，加上网上一些文章，希望用常人都可以理解的表达方式。来和大家分享学习经验。

开篇，还是得说说 ^ 和 $ 他们是分别用来匹配字符串的开始和结束，以下分别举例说明：

"^The"：开头一定要有"The"字符串；
"of despair$"：结尾一定要有"of despair" 的字符串；

那么，
"^abc$"：就是要求以abc开头和以abc结尾的字符串，实际上是只有abc匹配；
"notice"：匹配包含notice的字符串；

你可以看见如果你没有用我们提到的两个字符（最后一个例子），就是说模式（正则表达式）可以出现在被检验字符串的任何地方，你没有把他锁定到两边。

接着，说说 '*' '+' 和 '?'
他们用来表示一个字符可以出现的次数或者顺序，他们分别表示：
"zero or more"相当于{0,}
"one or more"相当于{1,}
"zero or one."相当于{0,1}

这里是一些例子：

"ab*"：和ab{0,}同义，匹配以a开头，后面可以接0个或者N个b组成的字符串("a", "ab", "abbb", 等)；
"ab+"：和ab{1,}同义，同上条一样，但最少要有一个b存在 ("ab" "abbb"等)；
"ab?"：和ab{0,1}同义，可以没有或者只有一个b；
"a?b+$"：匹配以一个或者0个a再加上一个以上的b结尾的字符串。

要点： **'*' '+' 和 '?' 只管它前面那 单个 字符,但是当前面的字符被小括号包起来时,就匹配小括号里面的所有的字符。**

你也可以在**大括号里面限制字符出现的个数**，比如：

"ab{2}": 要求a后面一定要跟两个b（一个也不能少）("abb")；
"ab{2,}": 要求a后面一定要有两个或者两个以上b(如"abb" "abbbb" 等)；
"ab{3,5}": 要求a后面可以有2－5个b("abbb", "abbbb", or "abbbbb")。

现在我们把**几个字符放到小括号里**，比如：

"a(bc)*": 匹配 a 后面跟0个或者一个"bc";
"a(bc){1,5}": 一个到5个 "bc";

还有**一个字符 '|'**，相当于OR操作：

"hi|hello": 匹配含有"hi" 或者 "hello" 的 字符串；
"(b|cd)ef": 匹配含有 "bef" 或者 "cdef"的字符串；
"(a|b)*c": 匹配含有这样多个（包括0个）a或b，后面跟一个c的字符串；

一个点**('.')**可以代表**所有的单一字符，不包括"\n"**

如果，**要匹配包括"\n"在内的所有单个字符** 用 **'[\n.]'** 这种模式。

"a.[0-9]": 一个a加一个字符再加一个0到9的数字；
"^.{3}$": 三个任意字符结尾。

中括号括住的内容只匹配一个单一的字符

"[ab]": 匹配单个的 a 或者 b (和 "a|b" 一样);
"[a-d]": 匹配'a' 到'd'的单个字符 (和"a|b|c|d" 还有 "[abcd]"效果一样);

一般我们都用 **[a-zA-Z]** 来指定字符为**一个大小写英文**：

"^[a-zA-Z]": 匹配以大小写字母开头的字符串；
"[0-9]%": 匹配含有 形如 x% 的字符串；
',[a-zA-Z0-9]$': 匹配以**逗号再加一个数字或字母结尾**的字符串；

你也可以把你不想要得字符列在中括号里，你只需要在总括号里面使用'^' 作为开头 "%[^a-zA-Z]%" 匹配含有两个百分号里面有一个非字母的字符串。

要点：^用在中括号开头的时候，就表示 排除括号里 的字符。

为了PHP能够解释，你必须在这些字符面前后加"，并且将一些字符转义。

不要忘记在中括号里面的字符是这条规则的例外—在中括号里面，所有的特殊字符，包括(")，都将失去他们的特殊性质 "[*\+?{}.]"匹配含有这些字符的字符串：

还有，正如regx的手册告诉我们："如果列表里含有']'，最好把它作为列表里的第一个字符(可能跟在'^'后面)。如果含有'-'，最好把它放在最前面或者最后面， or 或者一个范围的第二个结束点[a-d-0-9]中间的'-'将有效。

看了上面的例子，你对{n,m}应该理解了吧。要注意的是，n和m都不能为负整数，而且n总是小于m。这样，才能 最少匹配n次且最多匹配m次。如"p{1,5}"将匹配 "pvpppppp"中的前五个p

下面说说以\开头的

\b 书上说他是用来匹配一个单词边界，就是...**比如've\b'，可以匹配love里的ve而不匹配very里有ve**

ve\b: 匹配单词以ve结尾的,

\B 正好和上面的\b相反。例子我就不举了

ve\B: 匹配ve开始的单词

.....突然想起来....可以到 http://www.phpv.net/article.php/251 看看其它用\开头的语法

好，我们来做个应用：如何构建一个模式来匹配货币数量的输入。

构建一个匹配模式去检查输入的信息是否为一个表示money的数字。我们认为一个表示money的数量有四种方式："10000.00" 和 "10,000.00"，或者没有小数部分，"10000" and "10,000"。现在让我们开始构建这个匹配模式：

^[1-9][0-9]*$

这是所变量必须以非0的数字开头。但这也意味着单一的"0"也不能通过测试。以下是解决的方法：

^(0|[1-9][0-9]*)$

"只有0和不以0开头的数字与之匹配"，我们也可以允许一个负号在数字之前：

^(0|-?[1-9][0-9]*)$

这就是：0或者一个以0开头且可能有一个负号在前面的数字。好了，现在让我们别那么严谨，允许以0开头。现在让我们放弃负号，因为我们在表示钱币的时候并不需要用到。我们现在指定模式用来匹配小数部分：

^[0-9]+(\.[0-9]+)?$

这暗示匹配的字符串必须最少以一个阿拉伯数字开头。但是注意，在上面模式中 "10." 是不匹配的， 只有 "10" 和 "10.2" 才可以，你知道为什么吗？

^[0-9]+(\.[0-9]{2})?$

我们上面指定小数点后面必须有两位小数。如果你认为这样太苛刻，你可以改成：

^[0-9]+(\.[0-9]{1,2})?$

这将允许小数点后面有一到两个字符。现在我们加上用来增加可读性的逗号（每隔三位），我们可以这样表示：

^[0-9]{1,3}(,[0-9]{3})*(\.[0-9]{1,2})?$

不要忘记'+'可以被'*'替代如果你想允许空白字符串被输入话，也不要忘记反斜杆'\'在php字符串中可能会出现错误 (很普遍的错误)：

现在，我们已经可以确认字符串了，我们现在把所有逗号都去掉str_replace(",", "", $money)然后在把类型看成double然后我们就可以通过他做数学计算了。

再来一个：

构造检查email的正则表达式

在一个完整的email地址中有三个部分：

1. 用户名 (在 '@' 左边的一切)
2.'@'
3. 服务器名(就是剩下那部分)

用户名可以含有大小写字母阿拉伯数字，句号('.')减号('-')and下划线'_')。服务器名字也是符合这个规则，当然下划线除外。

现在，用户名的开始和结束都不能是句点，服务器也是这样。还有你不能有两个连续的句点他们之间至少存在一个字符，好现在我们来看一下怎么为用户名写一个匹配模式：

^[_a-zA-Z0-9-]+$

现在还不能允许句号的存在。我们把它加上：

^[_a-zA-Z0-9-]+(\.[_a-zA-Z0-9-]+)*$

上面的意思就是说：以至少一个规范字符（除了.）开头，后面跟着0个或者多个以点开始的字符串。

简单化一点， 我们可以用eregi()取代ereg()、eregi()对大小写不敏感， 我们就不需要指定两个范围 "a-z" 和 "A-Z"只需要指定一个就可以了：

^[_a-z0-9-]+(\.[_a-z0-9-]+)*$

后面的服务器名字也是一样，但要去掉下划线：

^[a-z0-9-]+(\.[a-z0-9-]+)*$

好。现在只需要用”@”把两部分连接：

^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*$

这就是完整的email认证匹配模式了，只需要调用：

eregi("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*$",$eamil)

就可以得到是否为email了

正则表达式的其他用法

提取字符串

ereg() and eregi() 有一个特性是允许用户通过正则表达式去提取字符串的一部分(具体用法你可以阅读手册)。比如说，我们想从 path/URL 提取文件名，下面的代码就是你需要：

```
ereg("([^\\/]*)$", $pathOrUrl, $regs);
echo $regs[1];
```

高级的代换

ereg_replace() 和 eregi_replace()也是非常有用的，假如我们想把所有的间隔负号都替换成逗号：

ereg_replace("[\n\r\t]+", ",", trim($str));

最后，我把另一串检查EMAIL的正则表达式让看文章的你来分析一下：

"^[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+'.'@'.'[-!#$%&\'*+\\/0-9=?A-Z^_`a-z{|}~]+\.'.'[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+$"