PHP正则表达式教程

**1、入门简介**

在编写处理字符串的程序或网页时，经常会有查找符合某些复杂规则的字符串的需要。正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。

很可能你使用过Windows/Dos下用于文件查找的通配符(wildcard)，也就是\*和?。如果你想查找某个目录下的所有的Word文档的话，你会搜索\*.doc。在这里，\*会被解释成任意的字符串。和通配符类似，正则表达式也是用来进行文本匹配的工具，只不过比起通配符，它能更精确地描述你的需求——当然，代价就是更复杂——比如你可以编写一个正则表达式，用来查找所有以0开头，后面跟着2-3个数字，然后是一个连字号“-”，最后是7或8位数字的字符串(像010-12345678或0376-7654321)。

1. **基本语法**

正则表达式(regular expression)描述了一种字符串匹配的模式，可以用来检查一个字符串串是否含有某种子串、将匹配的子串做替换或者从某个串中取出符合某个条件的子串等。格式如下：

**/正则表达式/[模式修正符]**

正则表达式包含：

1. 普通字符：包含a-z、A-Z、0-9
2. **元字符：包含特殊符号、转义字符、限定符、定位符**
3. 模式修正符：用固定的字符，表示特定含义，是正则补充说明的
4. **普通字符**

普通字符包括没有显式指定为元字符的所有可打印和不可打印字符。这包括所有大写和小写字母、所有数字、所有标点符号和一些其他符号。

1. **转义字符 \**

将下一个字符标记为一个特殊字符、或一个原义字符、或一个 向后引用、或一个八进制转义符。例如，'n' 匹配字符 "n"。'\n' 匹配一个换行符。序列 '\\' 匹配 "\" 而 "\(" 则匹配 "("。

**常用转义字符：**

\d 匹配一个数字字符。等价于 [0-9]

\D 匹配一个非数字字符。等价于 [^0-9]

\w 匹配包括下划线的任何单词字符。等价于[A-Za-z0-9\_]

\W 匹配任何非单词字符。等价于 [^A-Za-z0-9\_]

\b 匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'

\B 匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'

\cx 匹配由 x 指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符

\f 匹配一个换页符。等价于 \x0c 和 \cL

\n 匹配一个换行符。等价于 \x0a 和 \cJ

\r 匹配一个回车符。等价于 \x0d 和 \cM

\s 匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [ \f\n\r\t\v]

\S 匹配任何非空白字符。等价于 [^ \f\n\r\t\v]

\t 匹配一个制表符。等价于 \x09 和 \cI

\v 匹配一个垂直制表符。等价于 \x0b 和 \cK

1. **特殊字符**

所谓特殊字符，就是一些有特殊含义的字符

**( )**  标记一个子表达式的开始和结束位置，子表达式可以获取供以后使用

**.** 匹配除换行符 \n之外的任何单字符

**[** 标记一个中括号表达式的开始

**'n'**  匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("

**|** 指明两项之间的一个选择

**[^]**  否定符

**注意：**

1. “()”把字符串组合在一起。

“()”符号包含的内容必须同时出现在目标对象中。例如“（abc）”等的字符串匹配，则匹配“abc34”或“12abctt”

1. \\1 提取第一位的属性

例如：/^\d{2} ([\W])\d{2}\\1\d{4}$/ 匹配“12-31-2006”、“09/27/1996”、“86 01 4321”等字符串。

但上述正则表达式不匹配“12/34-5678”的格式，这是因为模式“[\W]”的结果“/”已经被存储，下个位置“\1”引用时，其匹配模式也是字符“/”。

1. 如果我们希望在正则表达式中实现类似编程逻辑中的**“或”运算**，在多个不同的模式中任选一个进行匹配的话，可以使用**管道符 “|”**。例如：/to|too|2/上述正则表达式将会与目标对象中的 “to”, “too”, 或 “2” 相匹配。
2. 与定位符 “^” 不同，否定符“[^]”规定目标对象中不能存在模式中所规定的字符串。

例如：/[^A-C]/ 目标对象中除A，B，和C之外的任何字符相匹配。

1. **限定符**

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配

**\*** 匹配前面的子表达式零次或多次。例如，zo\* 能匹配 "z" 以及 "zoo"。\* 等价于{0,}

**+** 匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。

**?** 匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。

**{n}**  n是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o

**{n,}** n是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o\*'

**{n,m}** m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

1. **定位符**

定位符使您能够将正则表达式固定到行首或行尾。它们还使您能够创建这样的正则表达式，这些正则表达式出现在一个单词内、在一个单词的开头或者一个单词的结尾。

**^** 匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与 \n 或 \r 之后的位置匹配

**$** 匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，$ 还会与 \n 或 \r 之前的位置匹配

**\b** 匹配一个字边界，即字与空格间的位置

**\B** 非字边界匹配。

1. **模式修正符（Pattern Modifiers）**

模式修正符在忽略大小写、匹配多行中使用特别多，掌握了这一个修正符，往往能解决我们遇到的很多问题。

**i** 匹配时忽略大小写，可同时匹配大小写字母

**m** 将字符串视为多行，当设定了此修正符，行起始(^)和行结束($)除了匹配整个字符串开头和结束外，还分别匹配其中的换行符(\n)的之后和之前

**s** 将字符串视为单行，换行符做普通字符看待，使“.”匹配任何字符

**x** 模式中的空白忽略不计

**U** 匹配到最近的字符串, 禁止贪婪匹配，只跟踪到最近的一个匹配符并结束，常用在采集程序上的正则表达式；

**U** 　模式字符串被当成UTF-8

**e** 将替换的字符串作为表达使用，只有preg\_replace() 使用此修正符，其它 PCRE 函数将忽略之

**A** 如果设定了此修正符，模式被强制为“anchored”，即强制仅从目标字符串的开头开始匹配

**D** 如果设定了此修正符，模式中的行结束($)仅匹配目标字符串的结尾。没有此选项时，如果最后一个字符是换行符的话，也会被匹配在里面，如果设定了 m 修正符则忽略此选项

1. **运算符优先级**

正则表达式从左到右进行计算，并遵循优先级顺序，这与算术表达式非常类似。

下表从**高到低**优先级顺序：

* \ 转义符
* (), (?:), (?=), [] 圆括号和方括号
* \*, +, ?, {n}, {n,}, {n,m} 限定符
* ^, $, \任何元字符、任何字符 定位点和序列（即：位置和顺序）
* | 替换，"或"操作字符具有高于替换运算符的优先级，使得"m|food"匹配"m"或"food"。若要匹配"mood"或"food"，请使用括号创建子表达式，从而产生"(m|f)ood"。

1. **正则函数**

函数的具体使用，我们可以通过PHP手册来找到。

preg\_grep()

preg\_match\_all()

preg\_match()

preg\_quote()

preg\_split()

preg\_replace()

1. **具体实例**
2. /[A-Z]/

上述正则表达式将会与从A到Z范围内任何一个大写字母相匹配。

1. /[a-z]/

上述正则表达式将会与从a到z范围内任何一个小写字母相匹配。

1. /[0-9]/

上述正则表达式将会与从0到9范围内任何一个数字相匹配。

1. /([a-z][A-Z][0-9])+/

上述正则表达式将会与任何由字母和数字组成的字符串，如 “aB0” 等相匹配。

1. 当用户需要在正则表达式的模式中加入元字符，并查找其匹配对象时，可以使用转义字符“\”。

例如：/Th\\*/

上述正则表达式将会与目标对象中的“Th\*”而非“The”等相匹配。

1. 当不需要存储匹配结果时使用非存储模式单元“（？：）”

例如 /(?:a|b|c)(D|E|F)\\1g/　将匹配“aEEg”。在一些正则表达式中，使用非存储模式单元是必要的，否则需要改变其后引用的顺序。上例还可以写成/（a|b|c）(C|E|F)\2g/。

1. 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。

1. 说说 ‘\*’ ‘+’ 和 ‘?’

“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 ‘?’ 只管它前面那个字符。

1. 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”)。

1. 现在我们把一定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”；

1. 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；

1. 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，怎么办？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。

1. 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；

1. “[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；

1. 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。

1. **常用正则表达式**

1、非负整数：^\d+$

2、正整数：^[0-9]\*[1-9][0-9]\*$

3、非正整数：^((-\d+)|(0+))$

4、负整数：^-[0-9]\*[1-9][0-9]\*$

5、整数：^-?\d+$

6、非负浮点数：^\d+(\.\d+)?$

7、正浮点数：

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

8、非正浮点数：^((-\d+\.\d+)?)|(0+(\.0+)?))$

9、负浮点数：^(-((正浮点数正则式)))$

10、英文字符串：^[A-Za-z]+$

11、英文大写串：^[A-Z]+$

12、英文小写串：^[a-z]+$

13、英文字符数字串：^[A-Za-z0-9]+$

14、英数字加下划线串：^\w+$

15、E-mail地址：^[\w-]+(\.[\w-]+)\*@[\w-]+(\.[\w-]+)+$

16、URL：^[a-zA-Z]+://(\w+(-\w+)\*)(\.(\w+(-\w+)\*))\*(\?\s\*)?$

或：^http:\/\/[A-Za-z0-9]+\.[A-Za-z0-9]+[\/=\?%\-&\_~`@[\]\':+!]\*([^<>\"\"])\*$

17、邮政编码：^[1-9]\d{5}$

18、中文：^[\u0391-\uFFE5]+$

19、电话号码：

^((\(\d{2,3}\))|(\d{3}\-))?(\(0\d{2,3}\)|0\d{2,3}-)?[1-9]\d{6,7}(\-\d{1,4})?$

20、手机号码：^((\(\d{2,3}\))|(\d{3}\-))?13\d{9}$

21、双字节字符(包括汉字在内)：^\x00-\xff

22、匹配首尾空格：(^\s\*)|(\s\*$)（像vbscript那样的trim函数）

23、匹配HTML标记：<(.\*)>.\*<\/\1>|<(.\*) \/>

24、匹配空行：\n[\s| ]\*\r

25、提取信息中的网络链接：(h|H)(r|R)(e|E)(f|F) \*= \*('|")?(\w|\\|\/|\.)+('|"| \*|>)?

26、提取信息中的邮件地址：\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*

27、提取信息中的图片链接：(s|S)(r|R)(c|C) \*= \*('|")?(\w|\\|\/|\.)+('|"| \*|>)?

28、提取信息中的IP地址：(\d+)\.(\d+)\.(\d+)\.(\d+)

29、提取信息中的中国手机号码：(86)\*0\*13\d{9}

30、提取信息中的中国固定电话号码：(\(\d{3,4}\)|\d{3,4}-|\s)?\d{8}

31、提取信息中的中国电话号码（包括移动和固定电话）：(\(\d{3,4}\)|\d{3,4}-|\s)?\d{7,14}

32、提取信息中的中国邮政编码：[1-9]{1}(\d+){5}

33、提取信息中的浮点数（即小数）：(-?\d\*)\.?\d+

34、提取信息中的任何数字 ：(-?\d\*)(\.\d+)?

35、IP：(\d+)\.(\d+)\.(\d+)\.(\d+)

36、电话区号：/^0\d{2,3}$/

37、腾讯QQ号：^[1-9]\*[1-9][0-9]\*$

38、帐号(字母开头，允许5-16字节，允许字母数字下划线)：^[a-zA-Z][a-zA-Z0-9\_]{4,15}$

39、中文、英文、数字及下划线：^[\u4e00-\u9fa5\_a-zA-Z0-9]+$