PHP 中正则表达式详解

概述

正则表达式是一种描述字符串结果的语法规则，是一个特定的格式化模式，可以匹配、替换、截取匹配的字符串。常用的语言基本上都有正则表达式，如 JavaScript、java 等。其实，只有了解一种语言的正则使用，其他语言的正则使用起来，就相对简单些。

正则表达式的基本知识汇总

行定位符（^与$）

行定位符是用来描述字符串的边界。“$”表示行结尾“^”表示行开始如"^de"，表示以 de 开头的字符串 "de$",表示以 de 结尾的字符串。

单词定界符

我们在查找的一个单词的时候，如 an 是否在一个字符串”gril and body”中存在，很明显如果匹配的话，an 肯定是可以匹配字符串“gril and body”匹配到，怎样才能让其匹配单词，而不是单词的一部分呢？这时候，我们可以是哟个单词定界符\b。

\ban\b 去匹配”gril and body”的话，就会提示匹配不到。

当然还有一个大写的\B，它的意思，和\b 正好相反，它匹配的字符串不能使一个完整的单词，而是其他单词或字符串中的一部分。如\Ban\B。

选择字符(|)，表示或

选择字符表示或的意思。如 Aa|aA，表示 Aa 或者是 aA 的意思。注意使用”[]”与”|”的区别，在于”[]”只能匹配单个字符，而”|”可以匹配任意长度

的字符串。在使用”[]”的时候，往往配合连接字符”-“一起使用，如[a-d],代表 a 或 b 或 c 或 d。

排除字符，排除操作

正则表达式提供了”^”来表示排除不符合的字符，^一般放在[]中。如[^1-5]，该字符不是 1~5 之间的数字。

限定符(？*+{n，m})

限定符主要是用来限定每个字符串出现的次数。

限定字符	含义
？	零次或一次
*	零次或多次
+	一次或多次
{n}	n 次
{n,}	至少 n 次
{n,m}	n 到 m 次

如(D+)表示一个或多个 D

点号操作符

匹配任意一个字符（不包含换行符）

表达式中的反斜杠(\)

表达式中的反斜杠有多重意义，如转义、指定预定义的字符集、定义断言、显示不打印的字符。

转义字符

转义字符主要是将一些特殊字符转为普通字符。而这些常用特殊字符有”.” ,”?”、”\”等。

指定预定义的字符集

字符	含义
\d	任意一个十进制数字[0-9]
\D	任意一个非十进制数字
\s	任意一个空白字符(空格、换行符、换页符、回车符、字表符)
\S	任意一个非空白字符
\w	任意一个单词字符
\W	任意个非单词字符

###显示不可打印的字符

字符	含义
\a	报警
\b	退格
\f	换页
\n	换行
\r	回车
\t	字表符

括号字符()

在正则表达式中小括号的作用主要有:

1、改变限定符如（|、* 、^)的作用范围

如(my|your)baby ,如果没有"()" ,|将匹配的是要么是 my ,要么是 yourbaby,有了小括号，匹配的就是 mybaby 或 yourbaby。

2、进行分组，便于反向引用

反向引用，就是依靠子表达式的"记忆"功能，匹配连续出现的字串或是字符。如(dqs)(pps)\1\2，表示匹配字符串 dqsppsdqspps。在下面 php 应用中，我将详细展开学习反向引用。

模式修饰符

模式修饰符的作用是设定模式，也就是正则表达式如何解释。php 中主要模式如下表：

修饰符	说明
i	忽略大小写
m	多文本模式
s	单行文本模式
x	忽略空白字符

正则表达式在 php 中应用

php 中字符串匹配

所谓的字符串匹配，言外之意就是判断一个字符串中，是否包含或是等于另一个字符串。如果不使用正则，我们可以使用 php 中提供了很多方法进行这样的判断。

不使用正则匹配

1、strstr 函数

string strstr (string haystack,mixedneedle [, bool $before_needle = false])

注 1：haystack 是当事字符串，needle 是被查找的字符串。该函数区分大小写。

注 2：返回值是从 needle 开始到最后。

注 3：关于$needle，如果不是字符串，被当作整形来作为字符的序号来使用。

注 4：before_needle 若为 true,则返回前东西。

stristr 函数与 strstr 函数相同，只是它不区分大小写

2、strpo 函数

int strpos (string haystack,mixedneedle [, int $offset = 0])

注 1：可选的 offset 参数可以用来指定从 haystack 中的哪一个字符开始查找。返回的数字位置是相对于 haystack 的起始位置而言的。

stripos -查找字符串**首次出现**的位置（**不区分大小**定）

strrpos -计算指定字符串在目标字符串中**最后一次出现**的位置

strripos -计算指定字符串在目标字符串中**最后一次出现**的位置（**不区分大小写**）

使用正则进行匹配

在 php 中，提供了 preg_math()和 preg_match_all 函数进行正则匹配。关于这两个函数原型如下：

int preg_match|preg_match_all (string $pattern , string $subject [, array &$matches [, int $flags = 0 [, int $offset = 0]]])

搜索 subject 与 pattern 给定的正则表达式的一个匹配. pattern:要搜索的模式，字符串类型。

subject :输入字符串。

matches:如果提供了参数 matches，它将被填充为搜索结果。 matches[0]将包含完整模式匹配到的文本，matches[1]将包含第一个捕获子组匹配到的文本，以此类推。

flags:flags 可以被设置为以下标记值：PREG_OFFSET_CAPTURE 如果传递了这个标记，对于每一个出现的匹配返回时会附加字符串偏移量(相对于目标字符串的)。 注意：这会改变填充到 matches 参数的数组，使其每个元素成为一个由 第 0 个元素是匹配到的字符串，第 1 个元素是该匹配字符串 在目标字符串 subject 中的偏移量。

offset:通常，搜索从目标字符串的开始位置开始。可选参数 offset 用于 指定从目标字符串的某个未知开始搜索(单位是字节)。

返回值：preg_match()返回 pattern 的匹配次数。 它的值将是 0 次（不匹配）或 1 次，因为 preg_match()在第一次匹配后 将会停止搜索。preg_match_all()不同于此，它会一直搜索 subject 直到到达结尾。 如果发生错误 preg_match()返回 FALSE。

实例 1

- 判断字符串”http://blog.csdn.net/hsd2012“中是否包含 csdn?

解法一（不适用正则）：

- 如果不适用正则，我们使用 strstr 或者 strpos 中任意一个都可以，在此，我将使用 strstr 函数，代码如下：
- $str='http://blog.csdn.net/hsd2012';function checkStr1($str,$str2)

- {
- return strstr1($str,$str2)?true:false;
- }echo checkStr($str,'csdn');

解法二：使用正则

- 因为我们只需要判断是否存在即可，所以选择 preg_match。
- $str='http://blog.csdn.net/hsd2012';$pattern='/csdn/';function checkStr2($str,$str2)
- {
- return preg_match($str2,$str)?true:false;
- }echo checkStr2($str,$pattern);

实例 2（考察单词定界符）

- 判断字符串”I am a good boy”中是否包含单词 go
- 首先判断是单词，而不是字符串，因此比较的时候，需要比较是否包含’ go ‘，即在字符串 go 前后有一个空格。
- 解析：如果使用非正则比较，只需要调用上面的 checkStr1()函数即可，注意，第二个参数前后要加一个空格,即’ go ‘。如果使用正则，
- 我们可以考虑使用单词定界符\b，那么$pattern=’/\bgo\b/’;然后调用 checkStr2 函数即可.

例 3（考察反向引用)

- 判断字符串”I am a good boy”中是否包含 3 个相同的字母
- 解析：此时，如果我们不使用正则，将会很难判断，因为字母太多了，我们不可能去将所有字母分别与该字符串比较，那样工作量也比较大。这时候涉及到了正在的反向引用。在 php 正则表达式中，通过\n，来表示第 n 次匹配到的结果。如\5 代表第五次匹配到的结果。那么本题的$pattern='/(\w).*\1.*\1/';

- 主要注意的是，在使用反向匹配的时候都需要使用(),反向匹配时，匹配()里面出现的字符或字符串。

php 中字符串替换

不使用正则

php 中当替换字符串的时候，如果不适用正则，我们通常使用 substr、mb_substr、str_replace、substr_replace 关于这几个函数区别如下表。

函数符	功能	描述
str_replace(find,replace,string,count)	使用一个字符串替换字符串中的另一些字符。	find 必需。规定要查找的值。replace 必需。规定替换 find 中的值的值。string 必需。规定被搜索的字符串。count 可选。一个变量，对替换数进行计数。
substr_replace(string,replacement,start,length)	把字符串的一部分替换为另一个字符串。适合用于替换自定位置的字符串。	string 必需。规定要检查的字符串。replacement 必需。规定要插入的字符串。start 必需。规定在字符串的何处开始替换。

使用正则

如果使用正则替换，php 中提供了 preg_replace _callback 和 preg_replace 函数，preg_replace 原型如下：

mixed preg_replace (mixed pattern,mixedreplacement , mixed subject[,intlimit = -1 [, int &count]])函数功能描述 :在字符串 subject 中，查找 pattern,然后使用 replacement 去替换，如果有 limit 则代表限制替换 limit 次。pregreplacecallback 与 pregreplace 功能相识，不同的是 pregreplaceback 使用一个回调函数 callback 来代替 replacement.–例 1 将字符串” hello,中国” 中的 hello 替换为’你好’;如果不是用正则:str=’ hello,中国’ ;

str=strreplace(’hello’,’你好’,str)

或是使用 str=substrreplace(str,’ 你好’ ,0,5)

使用正则

pattern=’/hello/’;str=preg_replace (pattern,’你好’,str);

例 2

去除字符串” gawwenngeeojjgegop” 中连续相同的字母

$str='gawwenngeeojjgegop';$pattern='/(.)\1/';$str=preg_replace($pattern,'',$str);

解析：当然这样可能会遇到，当第一次去除了重复了字符串后，又出来重复的字符串。如字符串味’ gewwenngeeojjgegop’ ,针对这中问题，当然，这样的话，通过判断，继续替换下去。

例 3

将字符串中” age13gegep3iorji65k65k” ;中出现的连续两个数字改为第二个数字，如字符串中 13 被改为 3

$str='age13gegep3iorji65k65k';$pattern='/(\d)(\d)/';$str=preg_replace($pattern,'$2', $str);

解析：$n 在正则表达式外使用反向引用。n 代表第几次匹配到的结果。

php **中字符串分割**

不使用正则

php 提供了 explode 函数去分割字符串，与其对应的是 implode。关于 explode 原型如下：

array explode (string delimiter,stringstring [, int $limit])

delimiter：边界上的分隔字符。

string：输入的字符串。

limit：如果设置了 limit 参数并且是正数，则返回的数组包含最多 limit 个元素，而最后那个元素将包含 string 的剩余部分。如果 limit 参数是负数，则返回除了最后的 -limit 个元素外的所有元素。如果 limit 是 0，则会被当做 1。

使用正则

关于通过正则表达式进行字符串分割，php 提供了 split、preg_split 函数。preg_split() 函数，通常是比 split() 更快的替代方案。

array preg_split (string pattern,stringsubject [, int limit=－1[,intflags = 0]])

例题

将字符串 'http://blog.csdn.net/hsd2012/article/details/51152810' 按照' /' 进行分割

解法一：

```
$str='http://blog.csdn.net/hsd2012/article/details/51152810';$str=explode('/', $str);
```

解法二：

```
$str='http://blog.csdn.net/hsd2012/article/details/51152810';$pattern='/\//';  /*因为/为特殊字符，需要转移*/$str=preg_split ($pattern, $str);
```

php **中贪婪匹配与惰性匹配**

- 贪婪匹配：就是匹配尽可能多的字符。

比如，正则表达式中 m.*n，它将匹配最长以 m 开始，n 结尾的字符串。如果用它来搜索 manmpndegenc 的话，它将匹配到的字符串是 manmpndegen 而非 man。可以这样想，当匹配到 m 的时候，它将从后面往前匹配字符 n。

- 懒惰匹配：就是匹配尽可能少的字符。

有的时候，我们需要并不是去贪婪匹配，而是尽可能少的去匹配。这时候，就需要将其转为惰性匹配。怎样将一个贪婪匹配转为惰性匹配呢？只需要在其后面添加一个”？”即可。如 m.*?n 将匹配 manmpndegenc，匹配到的字符串是 man。

• 函数符	• 描述
• *?	• 零次或多次，但尽可能少的匹配
• +?	• 一次或多次，但尽可能少的匹配
• ??	• 0 次或 1 次，但尽可能少的匹配
• {n,}?	• 至少 n 次，但尽可能少的匹配
• {n,m}?	• n 到 m 次 ，但尽可能少的匹配

php **正则表达式之回溯与固态分组**

回溯

首先我们需要清楚什么是回溯，回溯就像是在走岔路口，当遇到岔路的时候就先在每个路口做一个标记。如果走了死路，就可以照原路返回，直到遇见之前所做过的标记，标记着还未尝试过的道路。如果那条路也走不能，可以继续返回，

找到下一个标记，如此重复，直到找到出路，或者直到完成所有没有尝试过的路。

首先我们看例题

$str='aageacwgewcaw';$pattern='/a\w*c/i';$str=preg_match($pattern, $str);

看到上面的程序，可能都清楚是什么意思，就是匹配$str 是否包含这样一个由”a+0 个或多个字母+c”不区分大小写的字符串。但是至于程序怎样去匹配的呢？匹配的过程中，回溯了多少次呢？

匹配过程	接下来操作描述
‘a\w*c’ 中 a 匹配到‘aageacwgewcaw’ 中第一个字符 a	\w 进行下一个字符匹配
因为\w 是贪婪匹配，会一直匹配到‘aageacwgewcaw’ 中最后一个字符 w	c 进行下一个字符匹配时
‘a\w*c’ 中 c 发现没有可以匹配的	于是\w 匹配进行第一次回溯，匹配到倒数第二个字符 a
‘a\w*c’ 中 c 发现还是没有可以匹配的	于是\w 匹配进行第二次回溯，匹配到倒数第三个字符 c
‘a\w*c’ 中 c 匹配成功	匹配结束返回结果

现在，如果我们将 pattern 改为 pattern=’/a\w*?c/i’;又会回溯多少次呢？正确答案是回溯四次。

固态分组

固态分组，目的就是减少回溯次数， 使用(?>…)括号中的匹配时如果产生了备选状态，那么一旦离开括号便会被立即 引擎抛弃掉。举个典型的例子如：

‘\w+:’ 这个表达式在进行匹配时的流程是这样的，会优先去匹配所有的符合\w 的字符，假如字符串的末尾没有’ :’ ，即匹配没有找到冒号，此时触发回溯机制，他会迫使前面的\w+释放字符，并且在交还的字符中重新尝试与’ :’ 作比对。但是问题出现在这里: \w 是不包含冒号的，显然无论如何都不会匹配成功，可是依照回溯机制，引擎还是得硬着头皮往前找，这就是对资源的浪费。所以我们就需要避免这种回溯，对此的方法就是将前面匹配到的内容固化，不令其存储备用状态！，那么引擎就会因为没有备用状态可用而只得结束匹配过程。大大减少回溯的次数。

如下代码，就不会进行回溯：

$str='nihaoaheloo';$pattern='/(?>\w+):/';$rs=preg_match($pattern, $str);

当然有的时候，又需慎用固态分组，如下,我要检查$str 中是否包含以 a 结尾的字符串，很明显是包含的，但是因为使用了固态分组，反而达不到我们想要的效果。

$str='nihaoahelaa';$pattern1='/(?>\w+)a/';$pattern2='/\w+a/';$rs=preg_match($pattern1, $str);//0$rs=preg_match($pattern2, $str);//1

php 中其他常用字符串操作函数

字符串截取

string substr (string string,intstart [, int length])stringmbsubstr(stringstr , int start[,intlength = NULL [, string $encoding = mb_internal_encoding()]])

字符串中大小写转换

strtoupper

strtolower

ucfirst

ucwords

字符串比较

-strcmp、strcasecmp、strnatcmp

字符串翻转

strrev($str);

字符串随机排序

string str_shuffle (string $str)

补充

怎样进行邮箱匹配，url 匹配，手机匹配

使用 preg_match 函数进行匹配，以下内容从 TP 中复制而来。

邮箱验证

pattern='/\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/' ;

url 匹配

pattern='/^http(s?):\/\/(?:[A-za-z0-9-]+\.)+[A-za-z]{2,4}(:\d+)?(?:[\/\?#][\/=\?%\-&~`@[\]\':+!\.#\w]*)?/' ;

手机验证

pattern='/1[3458]\d10/' ;

php **中正则的优缺点**

php 中正则在某些时候，能帮我们解决 php 函数很多困难的匹配或是替换。然后 php 中正则的效率，往往是我们需要考虑的，所以在某些时候，能不用正则还是尽量不去用它，除非，某些场合必须用到，或是我们能够有效减少其回溯次数。