正则表达式

> 作者：王猛
>
> Email：bietushiwo@gmail.com
>
> 微博：@王猛
>
> QQ：672725440

> 说明： 本文档用于上课教案和学员复习，可传播可分享，如有错误，请联系老王，感谢矫正与探讨。

第1章 认识正则

所谓正则，其实就是字符串规则表达式，比如说大家熟悉的"*"代表的是所有字符.其实不应该叫正则表达式，你叫它规则表达式更好，因为它的主要作用就是，通过规则，找出你想要找的东西。

1.描述你要找的字符串的规律
2.调用函数，执行该正则表达式

1.php

```php
//把字符串的'hi'找出来
//规律:'hi'
$str = 'hi，this is his history';
$patt = '/hi/';
preg_match_all($patt，$str，$res);
print_r($res);
```

程序员都会用到，但是平常用的不多，所以容易忘;

入手:找谁？怎么找？找几个？

- 具体字符(字面值) --> 比如说就找a，b，hi
- **字符边界(下面加粗)** --> 从哪开始到哪结束
- 字符集合[ace]，[0123456789] --> 里面任意条件符合找出来
- 字符补集[^ qxz]:不在qxz范围内 --> 里面任意条件符合的不要
- 字符范围[a-z0-9] --> 注意:必须是连续的，你不能写a-Z
- 字符簇(系统定义好的常用集合，在第二章) --> 系统定义好的常用集合

字符边界

- `^` 匹配字符串的开始
- `$` 匹配字符串的结尾
- `\b` 匹配单词的开始和结尾(边界)
- `\B` 匹配单词的非边界

第2章 常用字符簇

簇	代表
.(点)	任意字符，不含换行
\w	[a-zA-Z0-9 _]
\W	\W的补集
\s	空白符，包括\n\r\t\v等
\S	非空白符
\d	[0-9]
\D	非数字

第3章 单词匹配

```php
// 把字符串的hi单词找出来
// 规律，单词开始处=>hi=>单词结束处 \b
$str = 'hi  ,  this is some history book';
$patt = '/\bhi\b/';
preg_match_all($patt , $str , $res);
print_r($res);

//把包括在单词内部的hi找出来
$patt = '/\Bhi\B/';
$str = ''
```

第4章 集合与补集示例

```php
/*
给定一组手机号，必须由[0123456789]组成的，才选出来从哪找?从字符串的开始找，找到字符串的结束 ^ $
找谁[01235689]
找几个?11个
*/
$arr = array('13800138000','13487656887','434456','45454353434543');
//$patt = '/^[^47]{11}$/'; //补集方法
$patt = '/^[01235689]{11}$/';//集合方式
foreach($arr as $v){
    preg_match_all($patt,$v,$res);
    print_r($res);
}
```

第5章 字符范围

```php
//试着找纯字母组成的单词
$str = 'o2o， b2b，hello，wordl， that';
//$patt = '/\[a-zA-Z]{1，}\b'; //{1，}最少1个字母
$patt = '/\b[a-zA-Z]+\b';
preg_match_all($patt,$str,$res);
print_r($res);
```

第6章 字符簇

就是系统规定好的标识方法

```php
$str = 'tommorw is another day，o2o ，you dont bird me i dont bird you';
$patt = '/\W{1，}';// \w \w[a-zA-Z0-9_]的补集
//preg_split 通过正则的表达式，分割字符串
print_r(preg_split($patt,$str));

//把多个空格或者制表符换成一个空格
$str = 'a     b     hello         world';//'a b hello world';
$patt = '/\s{1，}/'; //\s空白符，包括 \n\r\t\v 等
//preg_replace - 执行一个正则表达式的搜索和替换
echo preg_replace($patt,' ',$str);
```

第7章 找几个

- *匹配前面的子表达式零次或多次。
- +匹配前面的子表达式一次或多次。
- \? 匹配前面的子表达式零次或一次。

- {n} n是一个非负整数。匹配确定的 n 次。
  {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。。
- {n，} n 是一个非负整数。至少匹配n 次。

```php
$str = 'longren lao wang meng ge bi ';
// 5个字母组成的单词
//$patt = '/\b[a-zA-Z]{5}\b/';

// 3-5个字母组成的单词
//$patt = '/\b[a-zA-Z]{3，5}\b/';

// 5个以上字母组成的单词
//$patt = '/\b[a-zA-Z]{5，}\b/';

preg_match_all($patt， $str， $res);
print_r($res);
/*
某编辑部，键盘坏了，0键弹不出来，经常打出多个0
于是god打成good，gooood，请把这些单词替换成god
*/
$s = 'goooood，goood，gooooooooood';
$p = '/go+d/';
print_r(preg_replace($p，'god'，$s));
```

第8章 或者的用法

```php
//查询纯数字或者纯字母的词
$str = 'hello o2o 2b9 250';
$patt = '/\b[a-zA-Z]+\b|\b[0-9]+\b/';//最少一个
preg_match_all($patt，$str，$res);
print_r($res);

//查询苹果系统的产品
$str = 'ipad，iphone，imac，ipod，iamsorry';
$patt = '/\bi(pad|phone|mac|pod)\b/';
preg_match_all($patt，$str，$res);
print_r($res);
```

第9章 贪婪与非贪婪

```php
$str = 'ksda good goooood good kl s ja dfs dk ';
//把g(任意多的内容)d 这样的字符串，换成god
$patt = '/g.+d/'; //默认贪婪模式(会尽量多匹配)
preg_match_all($patt , $str , $res);
print_r($res); //god is not good

$patt = '/g.+?d/'; //在数量(+ * {n , })限定符后，加?，非贪婪模式
preg_match_all($patt , $str , $res);
print_r($res); //god , good
```

第10章 采集手机号

```php
$str = '王先森，要卖肾，联系手机号:18610886812，备用电话:15615614187，QQ:672725440，email:laowang@qq.com，诚心急卖，身份证号:370125199002220034';
//采集电话号码
$patt = '/\b1[358]\d{9}\b/';
preg_match_all($patt , $str , $res);
print_r($res);
```

第11章 后向引用

找收尾字母相同的单词

```php
$str = 'txt hello , high , bom , mum';
//简化，先找到首尾字母都是t的
$patt = '/\bt\w+t\b/';
preg_match_all($patt , $str , $res);
print_r($res);

此方法重复26次，也能找到

//第n个小括号内的子表达式，命中的内容，后面就用\n来引用
//后向引用
$patt = '/\b([a-z])\w+\1\b/';
//1.单词开始和结束 \b\b
//2.开始的[a-z]都可以.\b[a-z]\b
//3.后面跟什么都行，不管.并且字数不限 \b[a-z]\w+\b
//4.最后一个应该和第一个相同.\b([a-z])\w+\b 子表达式，放在下面另一个数组里面，最后一个引用子表达式匹配出来的结果\b([a-z])\w+\1\b
preg_match_all($patt , $str , $res);
print_r($res);
```

把手机号中间的4位替换为*

```php
$str = '13800138000 , 13426060134 ';
//前3位和后4位放子表达式中，中间4位随便，保留子表达式.替换中间的4位
$patt = '/(\d{3})\d{4}(\d{4})/';
//preg_match_all($patt , $str , $res);
//print_r($res);
echo preg_replace($patt , '\1****\2' , $str);
```

第12章 模式

模式修饰符，可以一定程度上影响正则的解析行为
比如i， 就代表正则不区分大小写， /[a-z A-Z]+/ --->/[a-z]+/i
比如s， 单行模式， 就代表把整个文件看成一个"单行"，忽略回车

```php
$str = 'hello WORLD  ChINa';
//$patt = '/\b[a-z]+\b/'; //hello
$patt = '/\b[a-z]+\b/i'; // 忽略大小写
preg_match_all($patt , $str , $matches);
print_r($matches);


$str = "abc haha
abc dgh";
$patt = '/.+/s'; # single 单行模式，将所有内容看成一整行
preg_match_all($patt , $str , $matches);
print_r($matches);
```

```php
//U 模式，把传入的参数看成unicode字符集的编码，可以判断中文
// http://blog.sina.com.cn/s/blog_640937d101017pca.html
// PHP下正则匹配中文，u模式，\x{4e00}-\x{9fa5}

$str = 'bob李';
$patt = '/^[\x{4e00}-\x{9fa5}]+$/u';
echo preg_match($patt , $str)?'国货':'杂货';
```