# 9.1.1认识正则表达式

### 正则表达式简介:

正则表达式是用于描述字符排列和匹配模式的一种语法规则。它主要用于字符串的模式分割、匹配、查找及替换操作。到目前为止，我们前面所用过的精确（文本）匹配也是一种正则表达式。

在PHP中，正则表达式一般是由正规字符和一些特殊字符（类似于通配符）联合构成的一个文本模式的程序性描述。

PHP中，正则表达式有三个作用：

匹配，也常常用于从字符串中析取信息。

用新文本代替匹配文本。

将一个字符串拆分为一组更小的信息块。

一个正则表达式中至少包含一个原子。

在PHP中有两套正则表达式函数库，两者功能相似，只是执行效率略有差异：

一套是由PCRE（Perl Compatible Regular Expression）库提供的。使用“preg\_”为前缀命名的函数；

一套由POSIX（Portable Operating System Interface of Unix ）扩展提供的。使用以“ereg\_”为前缀命名的函数；

PCRE来源于Perl语言，而Perl是对字符串操作功能最强大的语言之一，PHP的最初版本就是由Perl开发的产品。

PCRE语法支持更多特性，比POSIX语法更强大。

举例：想一想这两个正则表达式做什么用？

/^-?\d+$|^-?0[xX][\da-fA-F]+$/

[/^[0-9a-zA-Z\_-]+@[0-9a-zA-Z\_-]+(\.[0-9a-zA-Z\_-]+){0,3}$/](mailto:/^[0-9a-zA-Z_-]+@[0-9a-zA-Z_-]+(/.[0-9a-zA-Z_-]+){0,3}$/)

视频中的note

/\*

\* 1. 正则表达式，就是一个匹配的"模式"

\* 2. 正则表达式本身， 就是一个字符串(中有一些语法规则, 特殊字符)

\*

\*

\* 正则表达式这个字符串， 一定要在对应的函数中使用， （分割的函数，替换的函数 ）

\*/

# 9.2.1正则表达式的语法规则

### 定界符

### 原子

### 元字符

### 模式修正符

### Perl兼容正则表达式

正则表达式作为一个匹配的模版，是由原子（普通字符，例如字符a到z）、特殊字符（元字符，例如\*、+和？等）、以及模式修正符三部分组成的文字模式。

一个最简单正则表达式至少包含一个原子。

将下面的正则表达式拆分如下：

'/<a.\*?(?:|\\t|\\r|\\n)?href=[\"]?(.+?)[\"]?(?:(?:|\\t|\\r|\\n)+.\*?)?>(.+?)<\/a.\*?>/sim'

— 定界符：两个斜线”/”。

— 原子用到了< a href = ‘ “ / >等普通字符和\t \r \n等转义字符

— 元字符使用了 [] () | . ? \* + 等具有特殊含义的字符

— 用到了模式修正符是在定界符最后一个斜线之后的三个字符: s i m

### 定界符

在程序语言中，使用与Perl兼容的正则表达式，通常都需要将模式表达式放入定界符之间，如“/”。

作为定界符常使用反斜线“/”，如“/apple/”。用户只要把需要匹配的模式内容放入定界符之间即可。作为定界的字符也不仅仅局限于“/”。除了字母、数字和斜线“\”以外的任何字符都可以作为定界符，像“#”、“|”、“!”等都可以的。

— /<\/\w+>/ --使用反斜线作为定界符合法

— |(\d{3})-\d+|Sm --使用竖线”|”作为定界符合法

— !^(?i)php[34]! --使用竖线”!”作为定界符合法

— {^\s+(\s+)?$} --使用竖线”}”作为定界符合法

— /href=‘(.\*)’ --非法定界符，缺少结束定界符

— 1-\d3-\d3-\d4| --非法定界符，缺少其实定界符

## 视频上的note

<?php

$str = "aa1aaaaa3aaaa5aaaa77777aaaa2aaaa9aaaaa0aaaaaaa<br>";

$reg = "/\d/";

/\* 定界符号： 多种都可以， 常用为 //

\*

\* 原子 : 最少的一个匹配单位 (放在定界符中), 在一个正则表达式中， 至少要有一个原子

\* 元字符: 元字符不能单独使用的，修饰原子， 是用来扩展原子功能的和限定功能 (写在定界符号中)

\*

\* 模式修正符号: 修正， 对模式（正则）修正, （写在定界符号外面， 写在右边）

\*

\*/

echo $str."<br>";

echo preg\_replace($reg, "#", $str)."<br>";

print\_r(preg\_split($reg, $str));

echo '<br>';

if(preg\_match($reg, $str, $arr) ) {

echo "正则表达式 <b>{$reg}</b> 和字符串 <b>{$str}</b> 匹配成功!<br>";

print\_r($arr);

} else {

echo "没有匹配上<br>";

}

# 9.2.4正则表达式中的原子

原子是正则表达式的最基本的组成单元，而且在每个模式中最少要少包含一个原子。原子是由所有那些未显示指定为元字符的打印和非打印字符组成，具体分为5类。

1. 普通字符作为原子： 如 a~z、A~Z、0~9 等

2. 一些特殊字符和转义后元字符作为原子：

所有标点符号，但语句特殊意义的符号需要转义后才可作为原子，如：\” \’ \\* \+ \? \. 等

3. 一些非打印字符作为原子： 如：\f \n \r \t \v \cx

4. 使用“通用字符类型”作为原子：如：\d \D \w \W \s \S。

5. 自定义原子表([])作为原子：如：’/[apj]sp/’ ’/[^apj]sp/’

([原子表]) 在原字表中不需要加转意符号默认视为一个原子

\*\*\*\*

?: 可以让括号里的作用只用到第一、第二个功能

\*\*\*\*

### 正则表达式中常用的非打印字符

|  |  |
| --- | --- |
| **原子字符** | **含义描述** |
| **\cx** | 匹配由x指明的控制字符。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。 |
| **\f** | 匹配一个换页符。等价于 \x0c或\cL |
| **\n** | 匹配一个换行符。等价于 \x0a或\cJ (不知道) |
| **\r** | 匹配一个回车符。等价于 \x0d或\cM |
| **\t** | 匹配一个制表符。等价于 \x09或\cI (不知道) |
| **\v** | 匹配一个垂直制表符。等价于\x0b或\cK (不知道) |

### 正则表达式中常用的“通用字符类型”

|  |  |
| --- | --- |
| **原子字符** | **含义描述** |
| **\d** | 匹配任意一个十进制数字，等价于[0-9] |
| **\D** | 匹配任意一个除十进制数字以外的字符，等价于[^0-9] |
| **\s** | 匹配任意一个空白符，等价于[\f\n\r\t\v] |
| **\S** | 匹配除空白符以外任何字符，等价于[^\f\n\r\t\v] |
| **\w** | 匹配任意一个数字、字母或下画线，等价于[0-9a-zA-Z\_] |
| **\W** | 匹配一个除数字、字母或下画线以外的任意一个字符，等价于[^0-9a-zA-Z\_] |

## 视频note

<?php

/\*

\* 原子：

\*

\* 1. 打印字符(a-z A-Z 0-9 !@#$%^&\*()\_+) 和非打印字符

\*

\*

\*

\* \ 转义字符

\*

\* 一、 可以将有意义的字符转成没有意义的原子字符 (将有意义变成原子)

\* 二、 可以将没有意义的字符转成有意义的原子

\* 三、 a-z A-Z 0-9 \_ 所有没有意义的字符， 加上转义也没有意义的， \@ 都是可加可不加的

\*

\*

\* 2. 所有的数字， 所有的字， 所有的空白, 所有字母， 特殊符号

\*

\*

\* \d 代表任意一个数字 [0-9]

\* \D 代表任意一个非数字 [^0-9]

\*

\*

\* \w 代表任意一个字 a-z A-Z 0-9 \_ [a-zA-Z\_]

\* \W 代表任意一个非字 除了 a-z A-Z 0-9 \_之外的所有字符 [^a-zA-Z0-9\_]

\*

\*

\* \s 代表空白 [\t\n\f\v]

\* \S 代表非空白 [^\t\n\f\v]

\*

\*

\* 3. 自己定义原子表

\*

\* [13579]

\* [^a-zA-Z] -从哪到哪区间范围 ^除列表中的

\*

\*

\* 4 点(.)能代表所有的

\*

\*/

$str = "fdsaf3ds2af4dsaf8dsafdsa";

$reg = '/[^a-zA-Z]/';

if(preg\_match($reg, $str, $arr) ) {

echo "正则表达式 <b>{$reg}</b> 和字符串 <b>{$str}</b> 匹配成功!<br>";

print\_r($arr);

} else {

echo "没有匹配上<br>";

}

# 9.2.5正则表达式中的元字符

|  |  |
| --- | --- |
| **元字符** | **含义描述** |
| **\*** | 匹配0次、1次或多次其前的原子 (任意次数) |
| **+** | 匹配1次或多次其前的原子 （至少一次） |
| **？** | 匹配0次或1次其前的原子（最多一次） |
| **|** | 匹配两个或多个分支选择 |
| **{n}** | 表示其前面的原子恰好出现n次 |
| **{n,}** | 表示其前面的原子出现不小于n次 |
| **{n,m}** | 表示其前面的原子至少出现n次，最多出现m次 |
| **^或\A** | 匹配输入字符串的开始位置（或在多行模式下行的开头，即紧随一个换行符之后） |
| **$或\Z** | 匹配输入字符串的结束位置（或在多行模式下行的结尾，即紧随一个换行符之前） |
| **\b** | 匹配单词的边界 (\bis\b) |
| **\B** | 匹配除单词边界以外的部分 (\Bis\b) |
| **( )** | 匹配其整体为一个原子，即模式单元。可以理解为由多个单个原子组成的大原子 |

## 视频note

<?php

/\* 元字符： 不能在正则表达式中单独使用， 用来修饰原子的。

\*

\* \* 用来修饰其前面的原子可以出现0个1个或多个 (任意次) {0,}

\*

\* + 用来修饰其前面的原子可以出现1次或多次， 不能没有至少要出现在次 {1,}

\*

\* ? 用来修饰其前面的原子可以出现0次或1次 {0,1}

\*

\* {m} m = 3 用来修饰其前面原子只能出现 m次， 多一次和少一次都不可以

\* {n,m} {2,5} 用来修饰其前面的原子出现个数在n到m之间，包括n和m

\* {n,} {2,} 用业修饰其前面的原子可以出现至少次

\* {,m} {,5} 不行， 可以以0开始

\*

\* | 是或的关系， 表示它两边的原子，只要有一个出现就可以, 但是， ｜优先级是最低的， 比如出现oricle|mysql 的话先匹配这两的前面出现的那个词。

\*

\* ^ 或 \A 表示必须以什么开始, 这个必须写正则表达式的最前面

\* $ 或 \Z 表示必须以什么结束， 这个必须写正则表达式的最后面

\*

\*

\* \b 单词边界

\* \B 不是单词边界

\*

\*

\* .

\* []

([原子表]) 在原字表中不需要加转意符号默认视为一个原子

\*

\*/

$str = "this is island";

$reg = "/\Bis\b/";

if(preg\_match($reg, $str, $arr)) {

echo "正则 <b>{$reg}</b>, 和字符串<b>{$str}</b>匹配成功<br>";

print\_r($arr);

} else {

echo "匹配失败!<br>";

}

# 9.2.7

<?php

/\* 元字符： 不能在正则表达式中单独使用， 用来修饰原子的。

\*

\* \* 用来修饰其前面的原子可以出现0个1个或多个 (任意次) {0,}

\*

\* + 用来修饰其前面的原子可以出现1次或多次， 不能没有至少要出现在次 {1,}

\*

\* ? 用来修饰其前面的原子可以出现0次或1次 {0,1}

\*

\* {m} m = 3 用来修饰其前面原子只能出现 m次， 多一次和少一次都不可以

\* {n,m} {2,5} 用来修饰其前面的原子出现个数在n到m之间，包括n和m

\* {n,} {2,} 用业修饰其前面的原子可以出现至少次

\* {,m} {,5} 不行， 可以以0开始

\*

\* | 是或的关系， 表示它两边的原子，只要有一个出现就可以, 但是， ｜优先级是最低的，

\*

\* ^ 或 \A 表示必须以什么开始, 这个必须写正则表达式的最前面

\* $ 或 \Z 表示必须以什么结束， 这个必须写正则表达式的最后面

\*

\*

\* \b 单词边界

\* \B 不是单词边界

\*

\*

\* .

\* []原字符号 里面不用带转意本身就已经转意了

\*

\*

\* () 第一个作用： 必变优先级别

\* 第二个用用： 将小原子，变成大原子

\* 第三个作用： 子模式, 整个表达式是一个大的模式， 小括号中是每个独立的子模式

\* 第四个作用： 反向引用

\*\*\*\*

?: 可以让括号里的作用只用到第一、第二个功能

\*\*\*\*

\*/

$str = "this http://www.lampbrother.net 2014-01-24 oraclemysqlmysqlmysqlmysqllllllllllll is island";

$reg = '/(?:\d{4})(-|\/)\d{2}\1\d{2}/';

if(preg\_match($reg, $str, $arr)) {

echo "正则 <b>{$reg}</b>, 和字符串<b>{$str}</b>匹配成功<br>";

print\_r($arr);

} else {

echo "匹配失败!<br>";

}

# 9.2.9 模式修正符（单个字符）2

**视频note**

<?php

/\*

\* 模式修正符号(单个字符)

\*

\* 1. 模式修正符号要写在定界符号外， 放在右边

\*

\* "/go\*gle/i"

\* 2. 模式修正符号， 一个字符就是一个功能， 可以组合使用

\*

\* "/go\*gle/ieU"

\*

\* 作用：

\* 模式修正符号可以修正正则表代式的解释, 或扩充了正则表达式的功能

\*

\*

\* i : 修正正则表达式不区分大小写（默认是区分大小写的）

\*

\* m : 修正符号修正正则表达式可以视为多行， 在使用 ^ 或 $ 这两个符号时， 每一行满足都可以。 （默认视为一行处理）

\*

\* s: 修正正则表达式中的 . 可以匹配换行符号 (默认 . 不能匹配回车符号)

\*

\* x: 修正正则表达式，可以省略空白

\*

\* U: (.\*, .+) 正则表达式比较贪婪, .\*? .+?

\*

\*/

$str = "this <b>is</b> a test <b>Web</b> Serve";

$reg = '/\<b\>(.\*?)\<\/b\>/is';

if(preg\_match\_all($reg, $str, $arr)) {

echo "正则 <b>{$reg}</b>, 和字符串<b>{$str}</b>匹配成功<br>";

print\_r($arr);

} else {

echo "匹配失败!<br>";

}

### 9.2.10之后的都是练习！！！(url)

<?php

/\*

\* 正则表达式的编写

\* 1. 就是一种语言, 开发思想放进去

\* 2. 列需求

\*

\*

\*/

$str = "

这是http://www.lampbrother.net网站,

这是http://www.lampbrother.net/php/网站,

这是http://www.lampbrother.net/php/demo.php网站,

这是http://www.lampbrother.net/php/demo.inc.php?username=admin&p=123456网站,

这是http://www.baidu.com网站

这是http://www.aaa.baidu.com网站

这是https://www.lampbrother.net网站

这是ftp://mail.lampbrother.net网站

这是ftps://www.lampbrother.net网站

";

$reg= '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/';

if(preg\_match\_all($reg, $str, $arr)) {

echo '<pre>';

echo "正则 <b>{$reg}</b>, 和字符串<b>{$str}</b>匹配成功<br>";

print\_r($arr);

echo '</pre>';

} else {

echo "匹配失败!<br>";

}

%是干啥用的(注意)

一个中文会带两个百分号加数字的形式来显示

%234253%

# 9.2.11 email

<?php

/\*

\* 正则表达式的编写

\* 1. 就是一种语言, 开发思想放进去

\* 2. 列需求, 一条一条的满足

\*

\*

\*/

$str = "

这是meizi@lampbr-other.net网站,

这是mei-zi@lampbr.other.net网站,

这是mei+zi+zi+zi+zi@lampbrother.net网站,

这是mei.zi@lampbrother.net网站,

";

$reg = '/\w+([+-.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*/i';

if(preg\_match\_all($reg, $str, $arr)) {

echo '<pre>';

echo "正则 <b>{$reg}</b>, 和字符串<b>{$str}</b>匹配成功<br>";

print\_r($arr);

echo '</pre>';

} else {

echo "匹配失败!<br>";

}

# 9.3.1 字符串的匹配与查找之(strstr、strpos、substr)

### 视频note

<?php

/\*

\* 分割、匹配、查找、替换

\*

\* 1. 字符串处理函数 (处理快， 但有一些做不到)

\*

\* 2. 正则表达式函数 (功能强大，但效率要低)

\*

\*

\* 注意：如果可以直接使用字符串处理函数处理的字符串，就不要使用正则处理

\*

\*

\*

\* 匹配查找:

\*

\* strstr strpos substr

\*

\*

\* 正则匹配查找

\*

\* preg\_match() preg\_match\_all() preg\_grep();

\*

\*/

### 例子1

strstr ()

$str = "this is a test!";

if(stristr($str, "Test")) {

echo "存在";

}else {

echo "不存在";

}

### 例子2

function getFileName($url) {

$loc = strrpos($url, "/")+1;

return substr($url, $loc);

}

echo getFileName("http://www.baidu.com/aaa/demo.php");

echo getFileName("../images/logo.gif");

# 9.3.2 字符串的匹配与查找之(preg\_match)

### 例子1

<?php

/\* $reg = '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/';

$reg = '/\w+([+-.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\* /i';

\* 分割、匹配、查找、替换

\*

\* 1. 字符串处理函数 (处理快， 但有一些做不到)

\*

\* 2. 正则表达式函数 (功能强大，但效率要低)

\*

\*

\* 注意：如果可以直接使用字符串处理函数处理的字符串，就不要使用正则处理

\*

\*

\*

\* 匹配查找:

\*

\* strstr strpos substr

\*

\*

\* 正则匹配查找

\*

\* preg\_match() preg\_match\_all() preg\_grep();

\*

\*/

header("Content-Type:text/html;charset=utf-8");

?>

<?php

if(isset($\_POST['dosubmit'])) {

if(!preg\_match("/^\S+$/",$\_POST['username'])) {

echo "用户名为空，不成!<br>";

}

if(!preg\_match("/\w+([+-.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*/i", $\_POST['email'])) {

echo "不是正确的email格式";

}

if(!preg\_match( '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/', $\_POST['url'])) {

echo "不是正确的url格式";

}

}

?>

<form action="" method="post">

username: <input type="text" name="username" value="" /><br>

email: <input type="text" name="email" value="" /><br>

url: <input type="text" name="url" value="" /> <br>

<input type="submit" name="dosubmit" value="submit"> <br>

</form>

### 例子2

<?php

/\* $reg = '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/';

$reg = '/\w+([+-.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\* /i';

header("Content-Type:text/html;charset=utf-8");

?>

<?php

if(preg\_match( '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/', $\_POST['url'], $arr)) {

echo '<pre>';

print\_r($arr);

echo '</pre>';

echo "完整的URL是 {$arr[0]} <br>";

echo "协议是 {$arr[1]} <br>";

echo "主机是 {$arr[2]} <br>";

echo "域名是： {$arr[3]} <br>";

echo "顶层域： {$arr[4]} <br>";

echo "资源参数： {$arr[5]} <br>";

}

### 例子3

<?php

if(preg\_match( '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/', $\_POST['url'], $arr)) {

echo '<pre>';

print\_r($arr);

echo '</pre>';

echo "完整的URL是 {$arr[0]} <br>";

echo "协议是 {$arr[1]} <br>";

echo "主机是 {$arr[2]} <br>";

echo "域名是： {$arr[3]} <br>";

echo "顶层域： {$arr[4]} <br>";

echo "资源参数： {$arr[5]} <br>";

}

# 9.3.3 字符串的匹配与查找之(preg\_match\_all与preg\_grep)匹配

### 例子1 preg\_match\_all();

int **preg\_match\_all** ( string $pattern , string $subject [, array &$matches [, int $flags = PREG\_PATTERN\_ORDER [, int $offset = 0 ]]] )

第四个参数 打印的出数组的模式

PREG\_PATTERN\_ORDER和 PREG\_SET\_ORDER

<?php

if(preg\_match\_all( '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/', $\_POST['url'], $arr)) {

echo '<pre>';

print\_r($arr);

echo '</pre>';

echo "完整的URL是:{$arr[0]} <br>";

echo "协议是:{$arr[1]} <br>";

echo "主机是:{$arr[2]} <br>";

echo "域名是:{$arr[3]} <br>";

echo "顶层域:{$arr[4]} <br>";

echo "资源参数:{$arr[5]} <br>";

}

### 例子2 preg\_grep('/\s/', $arr);

**匹配数组中字符串的模式再返回出一个新的数组**

<?php

$arr = array("abcd1", "he llo2", "world", "ni hao");

$content=preg\_grep('/\s/', $arr);

print\_r($content);

?>

# 9.3.4 字符串的分割与连接(explode、implode join、preg\_split)

### 字符串分割：

\* explode() implode() -- join()

\*

\*

### \*正则表达式分割

preg\_split()

**例子1：**

<?php  
//使用逗号或空格(包含" ", \r, \t, \n, \f)分隔短语  
$keywords = preg\_split("/[\s,]+/", "hypertext language, programming");  
print\_r($keywords);  
?>

以上例程会输出：

Array(

[0] => hypertext

[1] => language

[2] => programming

)

**例子2**

<?php  
$str = 'hypertext language programming';  
$chars = preg\_split('/ /', $str, -1, PREG\_SPLIT\_OFFSET\_CAPTURE);  
print\_r($chars);  
?>

以上例程会输出：

Array

(

[0] => Array

(

[0] => hypertext

[1] => 0

)

[1] => Array

(

[0] => language

[1] => 10

)

[2] => Array

(

[0] => programming

[1] => 19

)

)

# 9.3.5 字符串的替换(str\_replace)

### 例子1

<?php

/\*

$num = 0;

$str = "http://www.phpbrother.net/php/demo.php";

$newstr = str\_replace("php", "java", $str, $num);

echo $str."<br>";

echo $newstr."<br>";

echo "替换的次数为：{$num}<br>";

\*/

$num = 0;

//正常 峰哥 妹子

$str="这是一句正常的峰哥句子，妹子但里面有一些不能显示的文字";

$newstr = str\_replace(array("正常", "妹子", "峰哥"), "\*\*", $str, $num);

echo $str."<br>";

echo $newstr."<br>";

echo "替换的次数为：{$num}<br>";

### 例子2

<?php

$num = 0;

//正常 峰哥 妹子

$str="这是一句正常的峰哥句子，妹子但里面有一些不能显示的文字";

$newstr = str\_replace(array("正常", "妹子", "峰哥"), array("很正常", "漂亮", "帅哥"), $str, $num);

echo $str."<br>";

echo $newstr."<br>";

echo "替换的次数为：{$num}<br>";

# 9.3.6

# preg\_replace

(PHP 4, PHP 5)

preg\_replace — 执行一个正则表达式的搜索和替换

### 说明

[mixed](mk:@MSITStore:D:\\兄弟连\\兄弟连新版PHP视频教程(共346讲)\\php_enhanced_zh.chm::/res/language.pseudo-types.html" \l "language.types.mixed) **preg\_replace** ( [mixed](mk:@MSITStore:D:\\兄弟连\\兄弟连新版PHP视频教程(共346讲)\\php_enhanced_zh.chm::/res/language.pseudo-types.html" \l "language.types.mixed) $pattern , [mixed](mk:@MSITStore:D:\\兄弟连\\兄弟连新版PHP视频教程(共346讲)\\php_enhanced_zh.chm::/res/language.pseudo-types.html" \l "language.types.mixed) $replacement , [mixed](mk:@MSITStore:D:\\兄弟连\\兄弟连新版PHP视频教程(共346讲)\\php_enhanced_zh.chm::/res/language.pseudo-types.html" \l "language.types.mixed) $subject [, int $limit = -1 [, int &$count ]] )

搜索subject中匹配pattern的部分， 以replacement进行替换。

# 9.3.7字符串的替换(preg\_replace)1

<?php

header("Content-Type:text/html;charset=utf-8");

/\* $reg = '/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/';

$reg = '/\w+([+-.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\* /i';

\* 分割、匹配、查找、替换

\*

\* 1. 字符串处理函数 (处理快， 但有一些做不到)

\*

\* 2. 正则表达式函数 (功能强大，但效率要低)

\*

\*

\* 注意：如果可以直接使用字符串处理函数处理的字符串，就不要使用正则处理

\*

\*

\*

\* 匹配查找:

\*

\* strstr strpos substr

\*

\*

\* 正则匹配查找

\*

\* preg\_match() preg\_match\_all() preg\_grep();

\*

\*

\* 分割：

\* explode() implode() -- join()

\*

\*

\*正则表达式

preg\_split()

\*

\*

\*

\* 替换

\* 字符串处理 str\_replace()

\*

\*

\* 正则表达式 preg\_replace()

\*

\*

\* strtoupper();把字符串变成

模式修正符e的用法：可以执行字符串里面的函数

\*/

### 例子：

$str = array(

"如果没有一些http://www.baidu.com特殊的<b>替换</b>需5求（<u>比如正则表达式</u>），你应http://www.lampbrother.net该使用该http://bbs.brophp.org函数替9换 <font color='red'>ereg\_replace()</font> 和9 preg\_replace()。 ",

"如果没有一些http://www.baidu.com特殊的<b>替换</b>需5求（<u>比如正则表达式</u>），你应http://www.lampbrother.net该使用该http://bbs.brophp.org函数替9换 <font color='red'>ereg\_replace()</font> 和9 preg\_replace()。 ",

"如果没有一些http://www.baidu.com特殊的<b>替换</b>需5求（<u>比如正则表达式</u>），你应http://www.lampbrother.net该使用该http://bbs.brophp.org函数替9换 <font color='red'>ereg\_replace()</font> 和9 preg\_replace()。 ",

"如果没有一些http://www.baidu.com特殊的<b>替换</b>需5求（<u>比如正则表达式</u>），你应http://www.lampbrother.net该使用该http://bbs.brophp.org函数替9换 <font color='red'>ereg\_replace()</font> 和9 preg\_replace()。 "

);

$reg = array(

'/\<[\/\!]\*?[^\<\>]+?\>/is',

'/(https?|ftps?):\/\/(www|mail|bbs|ftp)\.(.\*?)\.(net|com|org|cn)([\w-\.\/\=\?\&\%]\*)?/e',

'/\d/'

);

$rep = array(

'',

'"<a href=\'$1://$2.$3.$4\'>".strtoupper("$1://$2.$3.$4")."</a>"',

'@'

);

$newstr = preg\_replace($reg, $rep, $str);

echo '<pre>';

print\_r($str)."<br>";

print\_r($newstr)."<br>";

echo '</pre>';

# 9.4字符串比较函数

### 按字节顺序进行字符串比较

PHP中，有多种方法可以对字符串进行比较，除了可以直接使用条件运算符（<, >, ==）加以比较外。

按字节进行字符串的比较

strcmp();（整个比较是否相同）

strncmp();（指定比较长度的字符串比较）

strcasecmp(); （不区分大小写的全串比较）

**例子**

<?php

$userName = "Admin"; //声明一个字符串作为用户名

//if(strcasecmp($userName, “admin”) == 0) { //不区分大小写地比较,相等返回0

if( strcasecmp(strtolower($userName), strtolower(“admin”)) == 0 ) { //采用统一小写后比较

echo "用户名存在";

}

switch(strcmp($ userName, “admin")) { //区分字符串中字母的大小写比较

case 0: echo "两个字符串相等<br>"; break; //两个字符串相等则返回0

case 1: echo "第一字串大于第二字串<br>"; break; //第一个字符串大时则返回1

case -1: echo "第一字串小于第二字串<br>"; break; //第一个字符串小时则返回-1

}

**strnatcmp — 使用自然排序算法比较字符串**

格式：int strnatcmp ( string $str1 , string $str2 )

该函数实现了以人类习惯对数字型字符串进行排序的比较算法，这就是“自然顺序”。注意该比较区分大小写。

与其他字符串比较函数类似，如果 str1 小于 str2，返回负数；如果 str1 大于 str2，返回正数；二者相等则返回 0。

**例子**

<?php

$arr1 = $arr2 = array("img12.png", "img10.png", "img2.png", "img1.png");

echo “使用strcmp排顺序后：\n";

usort($arr1, "strcmp");

print\_r($arr1);

//[0] => img1.png [1] => img10.png [2] => img12.png [3] => img2.png

echo “\n 使用strnatcmp排序后：\n";

usort($arr2, "strnatcmp");

print\_r($arr2);

//[0] => img1.png [1] => img2.png [2] => img10.png [3] => img12.png

?>

正则：自己的理解。

例子：身份证

$reg = "^[1-9]\d{5}[1-9]\d{3}((0\d)|(1[0-2]))(([0|1|2]\d)|3[0-1])\d{3}([0-9]|X)$/";

//身份证的15和18位数的差距。 生日多了年份的两位和最后以为数字。十八位数的身份证1.开头没有0，2.接着任意数字5位，3.日期：年，月，日 4.任意三位。5最后的是任意一个数字或者X.