**PHP学习笔记**

**V1.0.0**

**Author: clearuo**

Date: 2016-04-06

## 一、基础知识

### 1.1理解constant函数可以动态输出不同的常量的含义

定义了常量，那么就要使用常量，那么如何获取常量值呢？

获取常量值的有两种方法取值。第一种是使用常量名直接获取值；例如计算圆周率的面积，如下（“\*”是运算符，相当于乘号，后面部分的小节会介绍）：

|  |
| --- |
| <? php define("PI", 3.14); $r = 1; $area = PI \*$r \*$r; //计算圆的面积 ?> |

第二种是使用constant()函数。它和直接使用常量名输出的效果是一样的，但函数可以动态的输出不同的常量，在使用上要灵活、方便，其语法格式如下：

mixed constant(string constant\_name)

第一个参数constant\_name为要获取常量的名称，也可为存储常量名的变量。如果成功则返回常量的值，失败则提示错误信息常量没有被定义。（注：mixed表示函数返回值类型为多种不同的类型，string表示参数类型为字符串类型）

例如：

|  |
| --- |
| <? php $p = ""; //定义圆周率的两种取值 define("PI1", 3.14); define("PI2", 3.142); //定义值的精度 $height = "中"; //根据精度返回常量名，将常量变成了一个可变的常量 if($height == "中"){     $p = "PI1"; } else if($height == "低"){     $p = "PI2"; } $r = 1; $area = constant($p) \*$r \*$r; echo $area; ?> |

### 1.2 php中的赋值运算符

PHP的赋值运算符有两种，分别是：

(1)“=”：把右边表达式的值赋给左边的运算数。它将右边表达式值复制一份，交给左边的运算数。换而言之，首先给左边的运算数申请了一块内存，然后把复制的值放到这个内存中。

(2)“&”：引用赋值，意味着两个变量都指向同一个数据。它将使两个变量共享一块内存，如果这个内存存储的数据变了，那么两个变量的值都会发生变化。

### 1.3 php中的比较运算符



### 1.4 php的错误控制运算符

PHP中提供了一个错误控制运算符“@”，对于一些可能会在运行过程中出错的表达式时，我们不希望出错的时候给客户显示错误信息，这样对用户不友好。于是，可以将@放置在一个PHP表达式之前，该表达式可能产生的任何错误信息都被忽略掉；

如果激活了track\_error（这个玩意在php.ini中设置）特性，表达式所产生的任何错误信息都被存放在变量$php\_errormsg中，此变量在每次出错时都会被覆盖，所以如果想用它的话必须尽早检查。

需要注意的是：错误控制前缀“@”不会屏蔽解析错误的信息，不能把它放在函数或类的定义之前，也不能用于条件结构例如if和foreach等。

|  |
| --- |
| <?php    $conn = @mysql\_connect("localhost","username","password");  echo "出错了，错误原因是：".$php\_errormsg; ?> |

### 1.5 获取当前时间

|  |
| --- |
| <?php date\_default\_timezone\_set('Asia/ShangHai'); $now = date('Y-m-d H:i:s',time()); echo $now; |

### 1.6 获取随机数

|  |
| --- |
| <?php $num = rand(1,50);//获取1至50的随机数，包含1和50 ?> |

### 1.7 php函数之可变函数

所谓可变函数，即通过变量的值来调用函数，因为变量的值是可变的，所以可以通过改变一个变量的值来实现调用不同的函数。经常会用在回调函数、函数列表，或者根据动态参数来调用不同的函数。可变函数的调用方法为变量名加括号。

|  |
| --- |
| class book {     function getName() {         return 'bookname';     } } $func = 'getName'; $book = new book(); $book->$func(); |

可变函数也可以用在对象的方法调用上。

|  |
| --- |
| class book {     function getName() {         return 'bookname';     } } $func = 'getName'; $book = new book(); $book->$func(); |

### 1.8 session与cookie的异同

cookie将数据存储在客户端，建立起用户与服务器之间的联系，通常可以解决很多问题，但是cookie仍然具有一些局限：

cookie相对不是太安全，容易被盗用导致cookie欺骗

单个cookie的值最大只能存储4k

每次请求都要进行网络传输，占用带宽

session是将用户的会话数据存储在服务端，没有大小限制，通过一个session\_id进行用户识别，PHP默认情况下session id是通过cookie来保存的，因此从某种程度上来说，seesion依赖于cookie。但这不是绝对的，session id也可以通过参数来实现，只要能将session id传递到服务端进行识别的机制都可以使用session。

## 正则表达式

### 2.0 正则表达式教程

<http://deerchao.net/tutorials/regex/regex.htm>

### 2.1 什么是正则表达式

正则表达式是对字符串进行操作的一种逻辑公式，就是用一些特定的字符组合成一个规则字符串，称之为正则匹配模式。

|  |
| --- |
| $p = '/apple/'; $str = "apple banna"; if (preg\_match($p, $str)) {     echo 'matched'; } |

其中字符串'/apple/'就是一个正则表达式，他用来匹配源字符串中是否存在apple字符串。

PHP中使用PCRE库函数进行正则匹配，比如上例中的preg\_match用于执行一个正则匹配，常用来判断一类字符模式是否存在。

### 2.2 正则表达式的基本语法

PCRE库函数中，正则匹配模式使用分隔符与元字符组成，分隔符可以是非数字、非反斜线、非空格的任意字符。经常使用的分隔符是正斜线(/)、hash符号(#) 以及取反符号(~)，例如：

/foo bar/

#^[^0-9]$#

~php~

如果模式中包含分隔符，则分隔符需要使用反斜杠（\）进行转义。

/http:\/\//

如果模式中包含较多的分割字符，建议更换其他的字符作为分隔符，也可以采用preg\_quote进行转义。

$p = 'http://';

$p = '/'.preg\_quote($p, '/').'/';

echo $p;

分隔符后面可以使用模式修饰符，模式修饰符包括：i, m, s, x等，例如使用i修饰符可以忽略大小写匹配：

$str = "Http://www.imooc.com/";

if (preg\_match('/http/i', $str)) {

echo '匹配成功';

}

### 2.3 元字符与转义

正则表达式中具有特殊含义的字符称之为元字符，常用的元字符有：

\ 一般用于转义字符

^ 断言目标的开始位置(或在多行模式下是行首)

$ 断言目标的结束位置(或在多行模式下是行尾)

. 匹配除换行符外的任何字符(默认)

[ 开始字符类定义

] 结束字符类定义

| 开始一个可选分支

( 子组的开始标记

) 子组的结束标记

? 作为量词，表示 0 次或 1 次匹配。位于量词后面用于改变量词的贪婪特性。 (查阅量词)

\* 量词，0 次或多次匹配

+ 量词，1 次或多次匹配

{ 自定义量词开始标记

} 自定义量词结束标记

//下面的\s匹配任意的空白符，包括空格，制表符，换行符。[^\s]代表非空白符。[^\s]+表示一次或多次匹配非空白符。

$p = '/^我[^\s]+(苹果|香蕉)$/';

$str = "我喜欢吃苹果";

if (preg\_match($p, $str)) {

echo '匹配成功';

}

元字符具有两种使用场景，一种是可以在任何地方都能使用，另一种是只能在方括号内使用，在方括号内使用的有：

\ 转义字符

^ 仅在作为第一个字符(方括号内)时，表明字符类取反

- 标记字符范围

其中^在反括号外面，表示断言目标的开始位置，但在方括号内部则代表字符类取反，方括号内的减号-可以标记字符范围，例如0-9表示0到9之间的所有数字。

//下面的\w匹配字母或数字或下划线。

$p = '/[\w\.\-]+@[a-z0-9\-]+\.(com|cn)/';

$str = "我的邮箱是Spark.eric@imooc.com";

preg\_match($p, $str, $match);

echo $match[0];

### 2.4 贪婪模式与懒惰模式

正则表达式中每个元字符匹配一个字符，当使用+之后将会变的贪婪，它将匹配尽可能多的字符，但使用问号?字符时，它将尽可能少的匹配字符，既是懒惰模式。

贪婪模式：在可匹配与可不匹配的时候，优先匹配

//下面的\d表示匹配数字

$p = '/\d+\-\d+/';

$str = "我的电话是010-12345678";

preg\_match($p, $str, $match);

echo $match[0]; //结果为：010-12345678

懒惰模式：在可匹配与可不匹配的时候，优先不匹配

$p = '/\d?\-\d?/';

$str = "我的电话是010-12345678";

preg\_match($p, $str, $match);

echo $match[0]; //结果为：0-1

当我们确切的知道所匹配的字符长度的时候，可以使用{}指定匹配字符数

$p = '/\d{3}\-\d{8}/';

$str = "我的电话是010-12345678";

preg\_match($p, $str, $match);

echo $match[0]; //结果为：010-12345678