第3章

PHP数组与字符串

数组和字符串是PHP中最为重要的两种数据类型，曾有人做过统计，在PHP的项目开发中，至少有30%的代码要处理数组，另有30%以上的代码在操作字符串，两者合计占PHP代码比重高达60%以上，故本章专门讲述这两类数据的操作。

- 掌握PHP数组的定义与操作。
- 掌握PHP字符串的定义与操作。
- 了解正则表达式及其使用。

3.1 数组及处理

数组把若干数据有序地组织在一起。本节介绍如何创建和初始化数组，以及对数组的各种处理。

3.1.1 数组的创建和初始化

1. 使用array()函数创建一维数组

使用array()函数的语法格式如下：

```
数组名 = array([键名 =>]值,...,[键名 =>]值);
```

每个元素包括键名和值两项，键名可以是整数或字符串。如果全部值未指定键名，则键名默认为从0开始的连续整数。如果只有某些值未指定键名，则该值的键名默认为该值前面最大的整数键名加1后的整数。例如：

```
<?php
  $arr1 = array(1,2,9,10);                              //定义不带键名的数组
  $arr2 = array("color" =>"blue","name" =>"pen");       //定义带键名的数组
```

```
    $arr3 = array(1 => 5, 2 => 6, 4 => 1, 9, 10);          //个别元素没有键名
?>
```

说明：数组 $arr1 的键名为整数键名，分别为 0、1、2、3。数组 $arr2 的键名为字符串键名，分别为"color"和"name"。数组 $arr3 的键名分别为 1、2、4、5、6。

对于数组，在调试程序时可以用 print_r 函数来显示数组各元素的键名和值，print_r 函数的语法格式如下：

```
print_r(数组名)
```

例如：

```
<?php
    $arr1 = array("a" => 5,"b" => 10,20);
    print_r( $arr1);                    //输出: Array ( [a] => 5 [b] => 10 [0] => 20 )
    echo "<br>";
    $arr2 = array(2 => 4, "color" => "red", 5, 3 => 7);
    print_r( $arr2);                    //输出: Array ( [2] => 4 [color] => red [3] => 7 )
?>
```

注意：*在数组 $arr1 中，第 3 个值 20 的键名为 0；在数组 $arr2 中，第 3 个值 5 被系统自动设置键名为 3，但是由于后面又有 3=>7 自定义了一个键名 3，因此后面的值 7 覆盖了前面相同键名的值。*

数组创建之后，可以使用“数组名[键名]”的形式来访问一维数组元素，例如：

```
<?php
    $arr1 = array("a" => 5,"b" => 10,20);
    echo $arr1["a"];                    //输出: 5
    echo $arr1["b"];                    //输出: 10
    echo $arr1[0];                      //输出: 20
?>
```

数组创建之后，可以使用 count()和 sizeof()函数获得数组元素的个数，例如：

```
<?php
    $array = array(1,2,3,6 => 7,8,9,5,10);
    echo count( $array);                //输出: 8
    echo sizeof( $array);               //输出: 8
?>
```

2. 使用 array()函数创建二维数组

通过对 array()函数的嵌套使用，可以创建二维数组，语法格式如下：

```
数组名 = array( [键名 1 =>] array(值 1,...,值 n),
                [键名 2 =>] array(值 1,...,值 n)
                );
```

说明：内层的每个 array()函数表示一行，键名表示行号。若省略键名，则默认为从 0 开始的连续整数。

二维数组元素的表示形式如下：

```
数组名[键名 1][键名 2]
```

例如：

```
<?php
   $ arr1 = array("color" = > array("红色","绿色","蓝色"),
                  "number" = > array(1,2,3,4,5)
                );
   echo $ arr1["color"][0], $ arr1["number"][4];          //输出：红色 5
   print_r( $ arr1);
   echo "< br >";
    $ arr2 = array(array("红色","绿色","蓝色"),array(1,2,3,4,5));
   echo $ arr2[0][0], $ arr2[1][4];                       //输出：红色 5
   print_r( $ arr2);
?>
```

程序解释：print_r($ arr1)语句的运行结果为 Array ([color] => Array ([0] => 红色[1] =>绿色[2] =>蓝色) [number] => Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5))。

print_r($ arr2)语句的运行结果为 Array ([0] => Array ([0] =>红色[1] =>绿色[2] =>蓝色) [1] => Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5))。

3. 使用变量名建立数组

通过使用 compact()函数，可以把多个变量，甚至数组紧凑成一个数组，其中，变量名成为数组元素的键名，变量值成为数组元素的值。语法格式如下：

```
数组名 = compact("变量名",...["数组名"])
```

举例如下：

```
<?php
   $ num = 8;
   $ str = "abc";
   $ arr = array(2,4,6);
   $ newarr = compact("num", "str", "arr");
   print_r( $ newarr);
?>
```

运行结果：

```
Array ( [num] => 8 [str] => abc [arr] => Array ( [0] => 2 [1] => 4 [2] => 6 ) )
```

即数组 $ newarr 包含 5 个元素：$ newarr[" num"]，$ newarr[" str"]，$ newarr["arr"][0]，$ newarr["arr"][1]，$ newarr["arr"][2]。

与 compact()函数对应的是 extract()函数，作用是将一个数组分离成多个变量，语法格式如下：

```
extract(数组名)
```

例如：

```
<?php
```

```
    $a = array("key1" => 1,"key2" => 2,"key3" => 3);
    extract( $a);      //数组 $a 被分离成 $key1、$key2、$key3
    echo " $key1  $key2  $key3";
?>
```

注意：在 extract(数组名)中，数组的键名必须是字母开头的字符串。

4. 建立指定范围的数组

使用 range()函数可以建立一个值在指定范围内的数组，语法格式如下：

```
数组名 = range(初值,终值[,步长值])
```

注意：若初值＜终值，则步长值为正数；若初值＞终值，则步长值为负数。若省略步长值，则默认为 1。例如：

```
<?php
    $array1 = range(1,5);      //输出: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )
    $array2 = range(2,10,2);   //输出: Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 [4] => 10 )
    $array3 = range("a","e"); //输出: Array ( [0] => a [1] => b [2] => c [3] => d [4] => e )
    print_r( $array1);
    print_r( $array2);
    print_r( $array3);
?>
```

5. 自动建立数组

数组可以不事先创建，而是直接赋值，数组会自动创建。例如：

```
<?php
    $arr[0] = "a";
    $arr[1] = "b";
    $arr[2] = "c";
    print_r( $arr);            //输出: Array ( [0] => a [1] => b [2] => c )
?>
```

说明：在第一个语句运行时，如果 $arr 数组不存在，则自动创建一个只有一个元素的 $arr 数组，后续的语句将在这个数组中添加新值。

3.1.2　键名和值的操作

对于数组的键名和值，有不少函数能操作它们。下面介绍一些常用函数。

1. 存在性检查

(1) array_key_exists()函数

格式：

```
array_key_exists(键名,数组名)
```

功能：检查数组中是否存在某个键名，若存在，则返回 true。

(2) in_array()函数

格式：

```
in_array(值,数组名)
```

功能：检查数组中是否存在某个值，若存在，则返回 true。

例如：

```
<?php
  $ array = array(1,2,3,5 = > 4,7 = > 5);
  if (array_key_exists(0, $ array)) echo "数组中存在键名 0";
  if (in_array(5, $ array)) echo "数组中存在值 5";
?>
```

2. 获取和输出

(1) array_keys()函数

格式：

```
数组名 2 = array_keys(数组名 1)
```

功能：将数组 1 中的所有键名存入数组 2 中。

(2) array_values()函数

格式：

```
数组名 2 = array_values(数组名 1)
```

功能：将数组 1 中的所有值存入数组 2 中。

例如：

```
<?php
  $ arr = array( "color" = > "red","name" = > "Sandy","age" = > 20);
  $ keys = array_keys( $ arr);
  $ values = array_values( $ arr);
  print_r( $ keys);          //输出: Array ( [0] = > color [1] = > name [2] = > age )
  print_r( $ values);        //输出: Array ( [0] = > red [1] = > Sandy [2] = > 20 )
?>
```

3. 遍历数组

与数组的遍历有关的函数有：

```
next(数组名)                //把数组指针移向下一个元素
prev(数组名)                //把数组指针移向上一个元素
reset(数组名)               //把数组指针移到第一个元素
end(数组名)                 //把数组指针移到最后一个元素
key(数组名)                 //取数组当前元素的键名
each(数组名)                //取数组当前元素的键名和值,并把指针移向下一个元素
list(var1,var2,...) = arr;  //把数组 arr 各值分别赋给各变量 var1,var2,...
```

例如：

```
<?php
    $ arr = array( "color" = >"red","name" = >"Sandy","age" = > 20);
    for( $ i = 0; $ i < count( $ arr); $ i++){
        echo key( $ arr).",";
        next( $ arr);
    }
```

```
    $arr2 = array("one","two","three");
    list($v1, $v2, $v3) = $arr2;
    echo "$v1, $v2, $v3";
?>
```

运行结果：

```
color,name,age,one, two, three
```

3.1.3　数组的排序

PHP 提供了许多数组排序函数，使一维数组的排序变得非常简单。

1. 升序排序

(1) sort()函数

格式：

```
sort(数组名)
```

功能：对数组的值进行升序排序，并将数组的键名修改为从 0 开始的整数键名。

(2) asort()函数

格式：

```
asort(数组名)
```

功能：对数组的值进行升序排序，但保持数组的键名和值之间的关联。

例如：

```
<?php
  $arr1 = array("a" => 5,"x" => 3,5 => 7,"c" => 1);
  $arr2 = array(2 =>"c",4 =>"a",1 =>"b");
  sort($arr1);
  asort($arr2);
  print_r($arr1);          //输出: Array ( [0] => 1 [1] => 3 [2] => 5 [3] => 7 )
  print_r($arr2);          //输出: Array ( [4] => a [1] => b [2] => c )
?>
```

2. 降序排序

rsort()、arsort()分别对应于上面的 sort()、asort()函数，但是排序是降序的。

3. 对多个数组同时排序

array_multisort()函数可以一次对多个一维数组排序。语法格式如下：

```
array_multisort(数组名,...,数组名)
```

功能：首先对第一个数组的值升序排列，其他数组中值的顺序按照第一个数组的对应顺序排列。数组列表中所有数组的长度必须相等。例如：

```
<?php
  $ar1 = array(3,5,2,4);
  $ar2 = array(8,6,9,7);
  array_multisort($ar1, $ar2);
```

```
  print_r( $ ar1);            //输出: Array ( [0] => 2 [1] => 3 [2] => 4 [3] => 5 )
  echo "< br />";
  print_r( $ ar2);            //输出: Array ( [0] => 9 [1] => 8 [2] => 7 [3] => 6 )
?>
```

说明：第一个数组中值的原先顺序是 3,5,2,4,对应的第二个数组中值的顺序是 8,6,9,7,排序后第一个数组中的值为 2,3,4,5,第一个元素中的值为 2,对应于第二个数组中的值为 9,因此 9 成为第二个数组排序后的第一个元素,以此类推。

4. 打乱数组的顺序

格式：

```
shuffle(数组名);
```

功能：打乱数组的顺序,并将数组的键名修改为从 0 开始的整数键名。

【例 3-1】 产生 10 个[1,100]范围内的互不重复的随机整数。

```
<?php
  $ arr = range(1,100);
  shuffle( $ arr);
  for( $ i = 0; $ i < 10; $ i++)
  {
      echo  $ arr[ $ i]. ",";
  }
?>
```

5. 按相反顺序排序

格式：

```
数组名 2 = array_reverse(数组名 1 [,key]);
```

功能：将数组 1 按相反顺序排序,生成数组 2。若 key 取 true,则数组 2 保持原来的键名；若 key 取 false,则数组 2 的键名修改为从 0 开始的整数键名。key 省略时为 false。

例如：

```
<?php
  $ array = array("a" => 1,2,3,4);
  $ ar1 = array_reverse( $array);        //$ ar1 的键名修改为从 0 开始的整数键名
  $ ar2 = array_reverse( $array,true);   //$ ar2 保持原来的键名
  print_r( $ar1);                        //输出: Array([0] => 4 [1] => 3 [2] => 2 [a] => 1)
  print_r( $ar2);                        //输出: Array([2] => 4 [1] => 3 [0] => 2 [a] => 1)
?>
```

3.2 字符串操作

字符串是很常用的数据类型,特别是网页源代码本身就是字符串。因此,字符串有很多操作函数。由于 PHP 是弱语言类型,所以当使用字符串操作函数时,其他类型的数据也会被当作字符串来处理。

3.2.1　常用的字符串函数

1. 计算字符串的长度

格式 1：

```
strlen(字符串)
```

功能：charset＝ GB2312，每个汉字为 2 个字符；charset＝UTF-8，每个汉字为 3 个字符。

格式 2：

```
mb_strlen(字符串,编码方式)
```

功能：编码方式为 GB2312，每个汉字为 2 个字符；编码方式为 UTF-8，每个汉字为 3 个字符。例如，mb_strlen(字符串,"GB2312")。

2. 改变字母大小写

```
strtolower(字符串)                    //将字符串转化为小写字母
strtoupper(字符串)                    //将字符串转化为大写字母
```

3. 删除字符串的首尾空格

```
ltrim(字符串)                         //删除字符串首部空格
rtrim(字符串)                         //删除字符串尾部空格
trim(字符串)                          //删除字符串首、尾空格
```

4. 字符串查找

用于字符串查找的函数非常多，仅介绍如下两个。

```
strstr(串 1, 串 2 [,是否串 2 之前])
stristr(串 1, 串 2 [,是否串 2 之前])
```

功能：在串 1 中查找串 2，如果查找成功，且省略[是否串 2 之前]，则返回串 1 中从第一次出现串 2 开始直到字符串结尾的字符串，若[是否串 2 之前]取 true 时，则在串 1 中截取串 2 之前的那部分子串。如果查找不成功，则返回 false。[是否串 2 之前]省略时，默认为 false。

stristr 函数与 strstr 作用类似，只是不区分大小写。例如：

```
<?php
  $str = "hello world";
  $find = "llo";
  $res1 = strstr( $str, $find);
  $res2 = strstr( $str, $find, true);
  echo " $res1, $res2";                //输出：llo world,he
?>
```

5. 截取子串

格式：

```
substr(字符串,n,len)
```

功能：对字符串从第 n 个字符开始，截取 len 个字符，形成子串。

说明：1 个字母、数字为一个字符，1 个汉字为 2 个(charset＝GB2312)或 3 个(charset＝UTF-8)字符。

例如，当 charset＝GB2312 时，substr("汕头职院",2,4)＝"头职"；当 charset＝UTF-8，substr("汕头职院",3,6) ＝"头职"。

6. 字符串与 ASCII 码

格式：

```
ord(字符串)                    //返回字符串中第一个字符的 ASCII 码
chr(n)                         //返回 ASCII 码 n 对应的字符
```

例如：

```
<?php
  echo ord("a");               //输出 97
  echo chr(97);                //输出 a
?>
```

7. 字符串的比较

(1) 使用关系运算符比较

数值与字符串比较，或两个数字字符串比较，先统一为数值，再比较。例如，表达式"123.5"＞"9.5"返回 true，即 123.5 大于 9.5。

(2) strcmp(串 1,串 2)：不管字符串如何，都不会转换成数值。

若串 1＞串 2，则 strcmp()返回 1；若串 1＝串 2，则 strcmp()返回 0；若串 1＜串 2，则 strcmp()返回－1。例如，strcmp("123.5","9.5")返回－1，即"123.5"小于"9.5"。

8. 字符串替换

格式：

```
新串 = str_replace(子串 1,子串 2,字符串)
```

功能：将字符串中的子串 1 替换成子串 2，形成新字符串。

例如：

```
<?php
  $ str = "I love you";
  $ replace = "lucy";
  $ end = str_replace("you", $ replace, $ str);
  echo  $ end;                        //输出：I love lucy
?>
```

3.2.2 字符串与 HTML

1. 将特殊字符转换为 HTML 代码

大多数字符转换为 HTML 代码时仍保持不变，但一些特殊字符(例如“＜”和“＞”)转换为 HTML 代码时发生了较大的变化，如表 3-1 所示。

表 3-1　特殊字符和对应的 HTML 代码

特殊字符	字符名称	转换后的 HTML 代码
&	and 符号	&amp;
"	双引号	&quot;
'	单引号	&#039;
<	小于号	&lt;
>	大于号	&gt;

在 PHP 脚本中，htmlspecialchars()函数也可将特殊字符转换为 HTML 代码，其语法格式为：htmlspecialchars(字符串)，功能是：将含有 HTML 标记的字符串编码(如“<”编为“<”，“>”编为“>”)，使浏览器能显示 HTML 标记本身。例如：

```
<?php
  $new = '<a href="test">test</a>';
  //编码成: &lt;a href=&quot;test&quot;&gt;test&lt;/a&gt;
  $str = htmlspecialchars($new);
  echo $str;      //输出: <a href="test">test</a>
?>
```

2. 将 HTML 代码转换为特殊字符

在 PHP 脚本中，htmlspecialchars_decode()函数可将字符串中的 HTML 代码转换为特殊字符，其语法格式为：htmlspecialchars_decode(字符串)。例如：

```
<?php
  $str = "&lt;a href=&quot;test&quot;&gt;test&lt;/a&gt;";
  //输出: <a href="test">test</a>
  echo htmlspecialchars_decode($str);
?>
```

3.2.3　其他字符串函数

1. 字符串转化为数组

格式：

```
数组名 = explode(分隔符,字符串)
```

功能：使用分隔符，将字符串分为若干个子串，并存入数组中。

例如：

```
<?php
  $str = "can you help me";
  $arr = explode(" ", $str);
  print_r($arr);
?>
```

2. 数组转化为字符串

格式：

```
字符串变量 = implode(连接符,数组名)
```