第 3 章

数　　组

Chapter 3

数组是一个能在单个变量中存储多个值的特殊变量，PHP 中的数组实际上是一个有序映射，是一种把数据(value)关联到键(key)的类型。其中，键可以是一个整型 int 或字符串 string，值可以是任意类型的值。

PHP 的数组类型在很多方面做了优化，既可以把它当成真正的数组，也可以看作是列表(向量)、散列表(是映射的一种实现)、字典、集合、栈、队列等数据类型。同时数组元素的值也可以是另一个数组，这样就可以构成多维数组。

3.1　数组的定义

PHP 中定义数组，可以用 array()语言结构来新建一个数组，自 PHP 5.4 起可以使用短数组定义语法，用[]替代 array()，推荐使用[]来定义数组。PHP 中用数字作为键名的数组一般称为索引数组；用字符串表示键的数组称为关联数组。

3.1.1　定义索引数组

索引数组是指定义数组时不指定数据的键，由系统自动分配一个唯一整数索引号作为键，索引默认从 0 开始。

语法格式为

```
变量=array(value1,value2,…,valuen);
```

或者

```
变量=[ value1,value2,…,valuen];
```

【例 3.1】 定义索引数组示例，这里用到的 print_r()函数可以打印输出整个数组内容及结构，按照一定格式显示键和元素。

```
<?php
    $a = array('华为','三星','vivo','oppo');
    $b = ['华为','三星','vivo','oppo'];
    print_r($a);
    print_r($b);
?>
```

两种形式的运行结果是一致的，索引数组中的键是从数字 0 开始，依次递增，不需要特别指定，如图 3.1 所示。

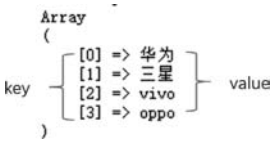

图 3.1　索引数组

3.1.2　定义关联数组

关联数组是指定义数组时指定数据的键，键一般都是自定义的字符串 string。

语法格式为

```
变量=array(key =>value,
      ...
      key=>value,
      )
```

或：

```
变量=[key =>value,
      ...
      key=>value,
      ]
```

【例 3.2】 定义关联数组示例。

```
<?php
    $age1=array(
        "Peter"=>"35",
        "Ben"=>"37",
        "Joe"=>"43",
    );
    $age2=[
        "Peter"=>"35",
        "Ben"=>"37",
        "Joe"=>"43",
    ];
    echo "<pre>";
    var_dump($age1,$age2)
?>
```

两种形式的运行结果是一致的，关联数组中的键不是数字，而是字符串，如图 3.2 所示。

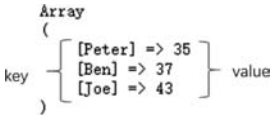

图 3.2　关联数组

3.1.3　直接动态定义数组

直接使用[]动态定义数组，其语法格式为

```
$arr[] = $value
```

或

```
$arr[$key] = $value
```

语法说明如下。

“$arr[] = 10;”表示如果数组 $arr 不存在，则创建一个数组，并将当前元素的下标置 0。

“$arr[] = 20;”表示如果数组 $arr 已存在，则增加一个数组元素，下标为最大下标加 1。

$key 代表元素的下标，可以是字符，也可以是整数。

$value 代表元素的值，可以是任何类型。

3.2 数组的操作

3.2.1 访问数组元素

1. 索引数组

对于索引数组，可以使用数组元素的索引号访问到数组中的元素，其语法格式为

```
数组名[index]
```

【例 3.3】 通过索引号访问索引数组元素变量，这里用到的 unset()函数用来销毁指定的变量。

```
<?php
    $cars=["Volvo","BMW","Toyota"];
    print_r($cars);
    echo "<br/>";
    $cars[3] = "Mazda"; //添加一个数组元素(注意：下标为 3 的数组元素不存在，因此新增该元素)
    print_r($cars);
    echo "<br/>";
    unset($cars[0]);    //删除一个数组元素，注意剩余数组元素位置没有发生变化
    print_r($cars);
    echo "<br/>";
    $cars[1]="Cadillac"; //修改一个数组元素
    print_r($cars);
    echo "<br/>";
?>
```

运行结果为

```
Array ( [0] => Volvo [1] => BMW [2] => Toyota )
Array ( [0] => Volvo [1] => BMW [2] => Toyota [3] => Mazda )
Array ( [1] => BMW [2] => Toyota [3] => Mazda )
Array ( [1] => Cadillac [2] => Toyota [3] => Mazda )
```

2. 关联数组

对于关联数组，通过数组元素的键名可以方便地访问到数组中的元素。其语法格式为

数组名[key]

【例 3.4】 通过键名访问关联数组元素变量。

```
<?php
    $age=[
        "Peter"=>"35",
        "Ben"=>"37",
    ];
    print_r($age);
    echo "<br/>";
    $age["Tom"] = "22"; //添加一个数组元素(注意：key 为 Tom 的数组元素不存在,因此新增该
                          元素)
    print_r($age);
    echo "<br/>";
    unset($age["Ben"]);//删除一个数组元素
    print_r($age);
    echo "<br/>";
    $age["Tom"]=30;     //修改一个数组元素
    print_r($age);
    echo "<br/>";
?>
```

运行结果为

```
Array ( [Peter] => 35 [Ben] => 37 )
Array ( [Peter] => 35 [Ben] => 37 [Tom] => 22 )
Array ( [Peter] => 35 [Tom] => 22 )
Array ( [Peter] => 35 [Tom] => 30 )
```

3.2.2 foreach 遍历数组

对数组的遍历可以使用 for 循环或使用 foreach 语句,对于索引数组,因为有下标值,配合 count()函数可以计算出数组的长度,所以可以方便地使用 for 循环遍历整个数组；但是对于关联数组,只能使用 foreach 语句,获取 key 和 value 的值。

foreach 语句提供了遍历数组的简单方式,foreach 仅能够应用于数组和对象,如果尝试应用于其他数据类型的变量或者未初始化的变量,将发出错误信息。foreach 有两种语法格式如下。

```
foreach (array_expression as $value)
     statement
```

说明：遍历给定的 array_expression 数组。每次循环中,当前数组元素的值被赋给 $value 并且数组内部的指针向前移一步(因此下一次循环中将会得到下一个数组元素)。

```
foreach (array_expression as $key => $value)
     statement
```

说明：遍历给定的 array_expression 数组。每次循环中,当前数组元素的值被赋给 $value,同时当前数组元素的键名被赋给变量 $key。

运行原理如图 3.3 所示。

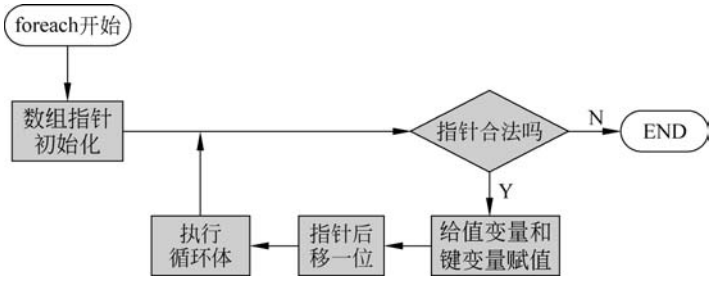

图 3.3 foreach 原理

【例 3.5】 对索引数组和关联数组分别遍历。

```
<?php
    $cars=["Volvo","BMW","Toyota"];
    $arrlength=count($cars); //获取数组的元素的个数,也就是数组长度
    for($x=0;$x<$arrlength;$x++){
        echo $cars[$x]." ";
    }
    echo "<br/>";
    foreach($cars as $c){
        echo $c." ";
    }
    echo "<br/>";
    $age=["Peter"=>"35","Ben"=>"37","Joe"=>"43"];
    foreach($age as $x_key=>$x_value){
        echo "Key=" . $x_key . ", Value=" . $x_value;
        echo "<br>";
    }
?>
```

运行结果为

```
Volvo BMW Toyota
Volvo BMW Toyota
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

3.3 多维数组

PHP 支持二维数组和多维数组,它们在实际编程中也经常用到。

3.3.1 二维数组

二维数组是指特殊的一维数组,这个一维数组的元素是一个一维数组,即将两个一维数组组合起来就可以构成一个二维数组,使用二维数组可以保存较为复杂的数据,在一些场合经常用到。

1. 索引二维数组

【例 3.6】 索引二维数组示例。

```
<?php
    $cars = [
       ["Volvo",22,18],
       ["BMW",15,13],
       ["Land Rover",17,15]
    ];
    echo $cars[0][0].": 库存: ".$cars[0][1].", 销量: ".$cars[0][2].".<br/>";
    echo $cars[1][0].": 库存: ".$cars[1][1].", 销量: ".$cars[1][2].".<br/>";
    echo $cars[2][0].": 库存: ".$cars[2][1].", 销量: ".$cars[2][2].".<br/>";
    //在 for 循环中使用另一个 for 循环,来获得 $cars 组中的元素(仍需使用两个索引)
    for ($row = 0; $row < count($cars); $row++) {
       $len = count($cars[$row]);
       echo "第 $row 行<br/>";
       for ($col = 0; $col < $len; $col++) {
          echo $cars[$row][$col]." ";
       }
       echo "<br/>";
    }
?>
```

例 3.6 中二维数组包含了四个数组,并且它有两个索引(下标):行和列。如需访问 $cars 数组中的元素,必须使用两个索引(行和列)。运行结果为

```
Volvo: 库存: 22, 销量: 18.
BMW: 库存: 15, 销量: 13.
Land Rover: 库存: 17, 销量: 15.
第 0 行
Volvo 22 18
第 1 行
BMW 15 13
第 2 行
Land Rover 17 15
```

2. 关联二维数组

【例 3.7】 关联二维数组示例。

```
<?php
    $person = [
         'Lily' => ['age'=>'20','hobby'=>'sleep'],
         'Tom' => ['age'=>'12','hobby'=>'eat']
    ];
    echo "<pre>";
    print_r($person);
    //遍历二维数组
    foreach ($person as $name => $msg) {
         echo "$name:";
         foreach ($msg as $key => $value) {
```

```
            echo "$key=>$value ";
        }
        echo "<br/>";
    }
?>
```

例 3.7 中，Lily、Tom 对应的值分别是个一维数组，这 2 个一维数组组成了一个二维数组。运行结果为

```
Array(
    [Lily] => Array(
            [age] => 20
            [hobby] => sleep
    )
    [Tom] => Array(
            [age] => 12
            [hobby] => eat
    )
)
Lily:age=>20 hobby=>sleep
Tom:age=>12 hobby=>eat
```

3.3.2 多维数组

参考二维数组，可以很容易地创建三维数组、四维数组或者其他更高维数的数组，PHP 中对多维数组没有上限的固定限制，但是随着维数的增加，数组会越来越复杂，对于阅读调试和维护都会稍微困难些。

【例 3.8】 定义三维数组示例。

```
<?php
    $arr = [
        '安徽' => [
            '阜阳'=>['阜南县','临泉县'],
            '合肥'=>['蜀山区','长丰县']
        ],
        '河南' => [
            '洛阳'=>['西工区','老城区'],
            '郑州市'=>['中原区','金水区']
        ]
    ];
    echo "<pre>";
    //var_dump($arr);
    print_r($arr);
    echo $arr['安徽']['合肥'][1];      //长丰县
?>
```

例 3.8 定义一个三维数组，其中“安徽”对应的是一个二维数组，“阜阳”“合肥”分别对应一个一维数组，“河南”也对应一个二维数组。“安徽”和“河南”分别对应一个二维数组，它俩组合起来形成一个三维数组。

运行结果为

```
Array(
    [安徽] => Array(
            [阜阳] => Array(
                    [0] => 阜南县
                    [1] => 临泉县
            )
            [合肥] => Array(
                    [0] => 蜀山区
                    [1] => 长丰县
            )
    )
    [河南] => Array(
            [洛阳] => Array(
                    [0] => 西工区
                    [1] => 老城区
            )
            [郑州市] => Array(
                    [0] => 中原区
                    [1] => 金水区
            )
    )
)
长丰县
```

【例 3.9】 遍历三维数组示例。

```
<?php
    $arr = [
          '安徽省' => [
              '阜阳市'=>['阜南县','临泉县'],
              '合肥市'=>['蜀山区','长丰县']
          ],
          '河南省' => [
              '洛阳市'=>['西工区','老城区'],
              '郑州市'=>['中原区','金水区']
          ]
    ];
    //遍历这个三维数组
    foreach ($arr as $princsname=>$princs) {
    //从三维数组中取出的每个 key 对应的 value 都是一个二维数组
         echo $princsname."<br/>";
         foreach ($princs as $cityname=>$citys) {
         //从二维数组中取出的每个 key 对应的 value 都是一个一维数组
              echo "  --".$cityname."<br/>";
              foreach ($citys as $city) {
                    echo "  ----".$city."<br/>";
              }
         }
    }
?>
```

运行结果为

```
安徽省
  --阜阳市
  ----阜南县
  ----临泉县
  --合肥市
  ----蜀山区
  ----长丰县
河南省
  --洛阳市
  ----西工区
  ----老城区
  --郑州市
  ----中原区
  ----金水区
```

实训：输出杨辉三角前5行

要求：输出如下形式的杨辉三角形。

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

参考代码如下。

```
<?php
    for ($i=0; $i <5 ; $i++) {
         $arr[$i][0] = 1;
         $arr[$i][$i] = 1;
    }
    print_r($arr);
    for ($i=2; $i < 5; $i++) {
         for ($j=1; $j <$i ; $j++) {
              $arr[$i][$j] = $arr[$i-1][$j-1]+$arr[$i-1][$j];
         }
    }
    print_r($arr);
    for ($i=0; $i < 5; $i++) {
         for ($j=0; $j < $i; $j++) {
              echo $arr[$i][$j]." ";
         }
         echo "<br/>";
    }
?>
```

第 4 章 Web 前端和后台数据交互

Chapter 4

在动态网页开发中，后台程序需要接收客户端的数据进行业务处理，再把处理的结果返回到客户端，例如用户的注册、登录，条件查询等操作。用户在客户端输入的数据是如何发送到服务器端的呢？这里，介绍两种常见的前端和后台数据通信的方式：表单和 Ajax 技术。

4.1 表单与服务器的交互

表单是网页设计中前端学习的很重要的一部分，表单的提交过程也是向后台发送数据的过程。表单提交数据的方式分为 POST 和 GET 两种，其实，POST 和 GET 是 HTTP 请求的两种方式，都可实现将数据从浏览器向服务器发送带参数的请求，本质上并没有区别。

4.1.1 GET 表单提交

在 PHP 中，有个 $_GET 变量，其作用就是获取通过前台以 GET 方式发送的数据。

$_GET 变量是一个数组结构的变量，是 PHP 提供的大量的预定义变量中的一个。在 PHP 中，预定义的 $_GET 变量用于收集来自 method＝"get"表单中的值，它通过 URL 参数传递给 GET 数组。

【例 4.1】 定义 form.html 文件，使用 get 方式提交数据到服务器端。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>$_GET 示例</title>
    </head>
    <body>
        <form method="get" action="welcome.php" >
            用户名：<input type="text" name="fname"><!—注意这里的元素变量会作为 GET 数
                    组的 key-->
            密码：<input type="password" name="fpass"><!—注意这里的元素变量会作为 GET
                  数组的 key-->
            <input type="submit" value="登录">
        </form>
    </body>
</html>
```

这里 action 属性代表的是想要接收表单提交的数据的网页地址，如果 action 属性值为空，则表示数据提交到当前页面，method 属性代表表单数据提交的方式。运行效果如图 4.1 所示。

当用户填写此表单并单击登录按钮后，表单数据会发送到名为 welcome.php 的 PHP 文