第 3 章　动手写第一个 PHP 脚本

所谓 PHP 脚本，其实就是一串指令，告诉 PHP 处理引擎应该完成什么动作。理论上来说，PHP 脚本可以只包含一条命令，也可以包含成千上万条命令，这完全取决于读者的需要。对于一个 PHP 脚本来说，PHP 处理引擎是按照从上到下、从左到右的顺序一条一条处理的，直到引擎指针指向脚本的最后一行命令。

那么，我们可以用 PHP 脚本做些什么事情呢？按照本书 1.2 节中的说法，我们可以编写 PHP 脚本实现在特定的网页显示特定的内容、将用户在表单中填写的内容存入数据库、将某目录中的文件备份到指定的存储设备上。PHP 几乎无所不能，只要读者肯下功夫，学习 PHP 是一件十分简单并且充满乐趣的事。

在本章里，我们将要动手写下第一个 PHP 脚本。

3.1　何谓 PHP 命令

按照表现形式的不同，PHP 命令可以分为简单命令和复杂命令两种。如何判断一条命令是简单还是复杂呢？

3.1.1　简单命令

每条简单的 PHP 命令都在告诉 PHP 处理引擎执行一个动作。最常见的 PHP 命令就是 echo 命令，它的功能是显示和输出信息。在第 1 章里，我们就已经见过这条命令了。

现在再来详细地看一下这个命令。

【例 3.1】 echo 命令。

```
1   echo "Hi";
```

在这条命令中，有三个部分组成。它们分别是命令关键字 echo、命令对象 Hi 和行结束符（;）。当 PHP 处理引擎读到这条命令时，它首先会看到这条命令的关键字，通过关键字了解命令要求完成的动作；然后再读取命令的对象，并按照命令关键字的要求完成对对象的处理；最后引擎会读取行结束符来结束对这条命令的执行。

刚才说到，echo 命令的功能是显示和输出信息，那么当 PHP 处理引擎读到这条命令时，就会输出一个简单的字符串“Hi”。

这个例子十分好懂，也没有什么好讲的。不过有的同学可能对行结束符产生了兴趣。行结束符和平常 Word 文档里的回车符有什么本质上的区别么？为了讲清楚这个问题，再来看几条命令。

【例 3.2】 行结束符。

```
1   echo "Great!
2   I hope I can finally get there!";
3
4   echo "Great!"; echo "Well done!";
```

看到这里，有的同学可能就不淡定了：这到底算是三条命令还是两条命令呢？嗯，这个问题问得好。其实答案也很简单，那就是只有当行结束符出现的时候，一条命令才算结束，无论一条命令被切成了几段写在了几行里。同理，若干条简单命令只有行结束符齐全，也可以挤在一行里抱团取暖，就像例 3.2 中的第 4 行一样。即便如此，还是建议大家一行只写一条简单命令，这样在后期进行错误定位的时候会好过得多。

PHP 引擎其实无从知晓代码的内容，它只知道寻找行结束符。在两个行结束符之间的内容就会被 PHP 引擎当成一条命令加以执行。于是上面这条命令的结果就是另一个简单的字符串“Great! I hope I can finally get there!”。

好奇的同学可能又要问了：如果在一个脚本里一个行结束符都没有的话，是不是所有的代码就会一起执行呢？答案是肯定的，但是你却看不到你想要的结果，看到的只是如下的一条报错信息：

```
Parse error: expecting "," or ";" in file.php on line 6
```

在报错信息中，你会看到出错的文件名和具体的行号，以及可能解决问题的办法。通常情况下，在每一条命令结尾处加上一个分号就能解决这个问题。

对于一个只有几条命令组成的 PHP 脚本文件来说，定位错误是十分容易的一件事情。但是通常情况下，一个 PHP 脚本怎么着也得有个上百条命令。因此选用一款可以显示行号的编辑器就成了一个明智的选择。否则，你就只能从上往下一条一条地数了。

3.1.2　复杂命令

把若干条简单命令放到一对花括号里，这些命令就组成了一个复杂命令。一个复杂命令通常包含若干条简单命令，甚至还会嵌套一些复杂命令。最常见的复杂命令块就是条件命令，只有满足特定条件时，花括号中的简单命令才会被执行，如例 3.3 所示。

【例 3.3】 If 条件命令。

```
1   if (time is in the morning)
2   {
3     get up;
4     brush my teeth;
5     wash my face;
6     put on my jacket;
7     go to work;
8   }
```

在上面这个例子里只出现了一对花括号。这对花括号中包含了 5 条简单命令。这个例子可以做如下的解读：

```
早晨，我起床后会先刷牙，再洗脸，然后穿上我的夹克出门上班。
```

在这句简单的描述中，我们可以发现作为条件的时间是早晨。也就是说只有早晨，我

才会做如下的动作：起床、刷牙、洗脸、穿衣和出门上班。于是，需要把条件写在花括号外的 if 子句中，然后把当条件满足时需要完成的动作依次写在花括号内，从而完成一个复杂的条件命令。

对于一条复杂命令来说，PHP 会一次性读取这条复杂命令所有的内容。值得注意的是，花括号后面是不需要加行结束符（;）的。

另外，大家还要注意务必让花括号内的每条语句都缩进若干字符并使它们保持左对齐。这个要求并不是必须的。但是，如果你和其他的同事都在编辑同一个 PHP 脚本，为了他人阅读的方便还是建议大家照做。

3.2　如何写代码

第 1 章里提到动态网页这个概念。为了使网页“动”起来，就得在 HTML 代码中插入 PHP 脚本，然后将这些内嵌 PHP 脚本的 HTML 文件保存为扩展名为.php 的文件。如此一来，PHP 处理引擎才会处理文件中的 PHP 脚本。

本节将讨论一个 PHP 脚本应该包括的元素。

3.2.1　PHP 标记对

只有当 PHP 脚本被插入扩展名为.php 的 HTML 文件中时，PHP 引擎才会处理这些脚本。那么应该把这些脚本插入到 HTML 文件中的什么地方呢？先来看例 3.4 中的代码。

【例 3.4】 PHP 标记对。

```
1   <?php
2   …
3   PHP statements
4   …
5   ?>
```

所有的 PHP 脚本都应该被包含在如例 3.4 所示的标记对中。读者也可以使用“<?”和“?>”来标记一个 PHP 脚本的起止。前提是修改了 php.ini 文件中关于启用短标记对的相关内容。

一般来说，使用短标记并不是一个好主意。如果把使用短标记对的 HTML 文件转移到一台没有启用短标记对的服务器上，那么所有的 PHP 脚本都会失效。尤其是对于租用服务器的开发者来说，这样做的后果是致命的，因为大多数供应商并不允许修改 php.ini 文件。这样一来，编码时少敲几个字母的代价也忒大了些。所以还是建议大家尽量使用完整的 PHP 标记对。

第 1 章的结尾曾经提到：如果用户通过浏览器发出访问请求，PHP 标记对间的所有 PHP 脚本都会被送到 PHP 处理引擎进行处理。然后服务器将经过处理的页面下发到发出请求的浏览器。该页面中所有的 PHP 脚本都已经被替换成了相应的处理结果。在浏览器里通过查看源代码的方式是无法看到任何 PHP 脚本的。

例如，读者可以在 HTML 代码中加入如例 3.5 所示的 PHP 脚本，然后将 HTML 代码

保存为一个 PHP 文件。

【例 3.5】 PHP 脚本。

```
1   <?php
2       echo "This line is brought to you by PHP.";
3   ?>
```

当用户请求该页面时，服务器会先查看文件的扩展名。当服务器发现该文件是一个 PHP 文件时，安装在服务器上的 PHP 处理引擎就会检查该文件里的 PHP 标记对、执行标记对中的脚本、并输出相应的结果。在本例中，服务器上的 PHP 处理引擎会执行 PHP 标记对中的 echo 命令，并输出处理结果，也就是“This line is brought to you by PHP”这句话。

当执行完文件中应该执行的所有脚本后，服务器会用脚本的执行结果替换相应的脚本，然后将处理后的 HTML 文件下发到用户的浏览器中。用户就能看到上面那句话了。

3.2.2 注释脚本

看到这一节的标题，有的同学会问：为什么要注释脚本呢？

注释对于脚本来说十分重要。通常情况下，我们会使用注释来描述代码，告诉阅读脚本的人某一段代码可以实现的功能以及该功能是如何实现的。当脚本十分复杂，让人无法很快读懂时，注释就显得尤为重要了。但是如果代码只有自己一个人在维护，那么是不是就不用注释了？自己写的代码难道自己还看不懂吗？对于这个问题，我只能用一句俗语来回答：“好记性不如烂笔头。”既然我们可以很方便地在脚本旁边注明一下某段脚本的功能，为什么不呢。更何况，脚本会变得越来越复杂，总有一天会需要很多的人来一起维护。写上注释就可以避免出现代码无法维护的情况，提高代码的利用效率。

所谓注释，其实就是写在脚本旁边用于说明代码的一段文字。PHP 处理引擎在碰到注释时会直接忽略。也就是说，注释一定是给人看的，那么写注释的时候言简意赅就显得十分必要了。那么 PHP 处理引擎如何区别脚本和注释呢？还是通过一个例子来说明一下。

【例 3.6】 注释示例。

```
1   /*   在这儿写注释
2   在这儿写更多的注释     */
```

在例 3.6 中，我们看到了如下的两个符号：“/*”和“*/”，这样的注释标记称为长注释标记。PHP 处理引擎在看到这一对符号时，就会直接忽略它们之间的所有内容。大家可以在开始写脚本之前，在开头的地方注释一段，写一写脚本的名字、描述、作者信息和写作时间等信息，以后查找起来也会非常方便。例 3.7 就是一段脚本说明。

【例 3.7】 脚本说明。

```
1   /*  name: hello.php
2       description: Displays "Hello World!" on a web page.
3       written by: Joe Programmer
4       created on: Feb 1st, 2012
5       modified on: Mar 15th, 2012
6   */
```

值得注意的是，长注释标记不支持嵌套。也就是说，如果出现了如例 3.8 这样的注释

标记，PHP 会报错。

【例 3.8】 错误的注释嵌套。

```
/* 这是一条注释
  /* 这是另一条注释 */
*/
```

有人可以会问了，这不是挺工整的吗，为什么会报错呢？我们来分析一下：按照之前的说法，PHP 处理引擎在见到“/*”符号时，就会忽略之后的所有内容，直到它遇到了“*/”符号。这样看来，在例 3.8 中，PHP 处理引擎会把第一行的“/*”和第二行的“*/”当成是注释的开始和结尾，而把第二行开头的“/*”当成了注释的一部分。那么第三行的“*/”就形只影单无人顾了。PHP 处理引擎也会因为无法处理这个形只影单的标识符而报错。

对于注释内容如例 3.7 这样比较多的情况来说，这个注释标记还显得不是很累赘。如果注释只有一行，还要陪上 4 个字符的注释标记对，效率实在是太低了。其实 PHP 还提供了两种短注释，标识符是井号（#）或双斜杠（//），如例 3.9 所示。

【例 3.9】 短注释。

```
# This is a comment.
//This is another comment.
```

那么这两种标识符有什么区别呢？井号（#）只能用在一行的开始，而双斜杠（//）可以用在一行的中间。当需要在某一行命令后进行注释时，可以使用双斜杠（//），如例 3.10 所示。

【例 3.10】 双斜杠注释符可以用在一行的末尾。

```
<?php
    echo "Hello World!";    //打印“Hello World!”
?>
```

在本书中，这三种注释标识都会用到。为了统一风格，也为了帮助大家养成注释脚本的习惯。本书做出如下的规定：

- 脚本的开始使用长注释书写脚本说明。
- 若一个脚本中包含有若干个模块，在每个模块开始前用短注释标识（#）说明模块的功能。
- 在重要的命令行后用短注释标识（//）说明该命令行的作用。

3.3　实战练习：向世界说 Hello!

虽然之前就已经提到过 echo 命令的功能和用法，但是并没有形成一个系统的概念。为了更好地使用这条十分常用的命令，很有必要在正式地用 PHP 脚本向世界说你好之前系统地讲解一下这条命令。

3.3.1　echo 命令初识

这条命令可以说是编写 PHP 脚本必用的命令之一。没有哪个脚本在被执行之后不输出信息的。一旦需要输出信息，就一定会用到这条命令。例如，编写了一段在操作系统里查

找某个特定文件的脚本。查找的过程当然是不可见的，但是如果连查找的结果也不输出的话，如何知道这个脚本是不是起作用了，要查找的文件是不是已经找到了呢？对于这样的脚本，通常需要输出的信息包括，查找结果和需要查找文件的文件名等。如果在操作系统中查找到了该文件、则还需要输出该文件的存储路径、存储时间、最后修改时间和摘要信息。这样一来，我们才能知道编写的脚本是不是起了作用、是否找到了需要的文件以及文件的基本情况。

所以说，echo 命令是十分重要也是十分必要的一条命令。按照之前的示例中书写的样式，我们可以总结出 echo 命令的基本样式。

【例 3.11】 echo 命令的基本样式。

```
1    echo output1, output2, output3, …
```

在使用 echo 命令时，一定要注意以下几点：

- 输出的对象，也就是例 3.11 中的 output 参数一定是一个数字或者字符串。所谓数字就是像 1 或者 234 这样的数字（如 echo 93;），而字符串则是一串包含在引号内的字符（如 echo "Hello World!";）。关于引号的使用，将在第 7 章讲字符串型变量的时候会进行详细地讲解。
- 理论上来讲，一条 echo 命令可以输出的对象是无限的，但是当输出对象多于两个时，需要在相邻的输出对象之间加上一个逗号。千万注意，不要画蛇添足的在逗号后面加个空格，否则脚本会报错。
- 空格也是一种字符，可以通过在空格前后添加引号来输出（如 echo "　　";）。

表 3-1 所示列出了使用一些 echo 命令和它们的输出结果。

表 3-1　echo命令的输出结果

echo 命令	输 出 结 果
echo 159;	159
echo "Hello World!";	Hello World!
echo "Hello","World!";	HelloWorld!
echo "Hello"," ","World!";	Hello World!

在使用 echo 命令时，还要注意一些诸如“\n”、“\t”和“\”这样的特殊字符和转义字符。具体的内容会在第 7 章讲解字符串类型的变量时进行详细地讲解。

到这里，使用 echo 命令需要注意的相关内容都呈现给大家了。是不是很简单呢？有没有学会靠事实来说话。下面就让我们进入到实战练习吧。

3.3.2　实战练习——向世界说 Hello!

在本小节里，我们将使用 PHP 基础知识来写一段脚本，用不同的语言向世界说 Hello！

在示例脚本中，会出现一些之前没有见过的内容，比如说数组、再比如说循环等等。这些都是 PHP 的基础知识，大家会在后续的章节中学到。另外，脚本中涉及到关于 HTML 和 JavaScript 的相关知识，书中会附带着给予相应的解释，应该不会给大家的阅读带来影响。不过，这里也有一个小建议，那就是尽可能多的掌握 HTML 和 JavaScript，特别是在 HTML 5 大行其道的年代，这一点非常的重要。

好了，言归正传。首先，先来新建一个名为 01_hello_world.php 的文件，然后在文件中写下一些 HTML 代码：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Hello World!</title>
        <style>
            #msg {
                font-size:500%;
                margin-left:60px;
                width:900px;
                height:300px;
                line-height:300px;
                text-align:center;
                overflow:hidden;
            }

            #lang {
                font-size:150%;
                margin-left:120px;
                height:80px;
                line-height:80px;
                text-align:right;
            }
        </style>
    </head>
    <body>
        <div id="msg"></div>              //输出问候的位置
        <div id="lang"></div>             //输出该问候对应语言的位置
    </body>                               //在该标签前插入 PHP 脚本
</html>
```

在上面这份 HTML 文件中，定义了两个 div 标签，分别用来存放问候语和该问候语对应的语言。

接下来，需要在“**</body>**”标签前插入如下的 PHP 脚本：

```
<?php
    $msgArray = array('Hello World!',
                '世界，您好！',
                '世界，您好！',
                'こんにちは，世界！',
                '안녕하세요!',
                'Bonjour le monde!',
                'Hallo Welt!',
                'สะบายดีโลก',
                'saluton mondo');  //定义了一个数组，用于存放问候语

    $langArray = array('English',
                       '简体中文',
                       '繁体中文',
                       '日本語',
                       '한국어',
                       'française',
                       'Deutsch',
```

```
                        'ພາສາລາວ',
                        'Esperanto'); //定义了另一个数组，用于存放问候语对应的语言
    $idPrev = 0;

    /*因为在下面的循环里使用了 rand()产生的随机数，为了避免重复出现两句相同的问候
    这里定义了一个比较变量*/

    for ($i=0;$i<=10;$i++){          //开启一个循环，用于随机输出问候语
        $id = rand(0,8);             //获取一个 8 以下的随机数
        if($idPrev === $id) {        //判断当前循环中产生的随机数与上一次循环中产
                                       生的循环次数是否相同
            $id = $idPrev + 1;
            if($id > 8)
            {
                $id = 8;
            }
        }                            //<=中产生的随机数是否相同
        $msg = $msgArray[$id];       //获取产生的随机数对应的问候语
        $lang = $langArray[$id];     //获取产生的随机数对应的问候语使用的语言
        $idPrev = $id;               //为下一次循环比较随机数做准备
        echo '<script
language="javascript">document.getElementById("msg").innerHTML="'.$msg.
'"</script>';
        echo '<script
language="javascript">document.getElementById("lang").innerHTML="'.$lan
g.'"</script>';

        /*上面使用学习到的 echo 命令输出了两个字符串。在这两个字符串中我们使用了
            JavaScript 脚本。这个脚本的向 HTML 文件中的 ID 为 msg 的 DIV 标签输出
        变量$msg 的内容，同时向 ID 为 lang 的 DIV 标签输出变量$lang 的内容*/

        echo str_repeat(' ',1024*64);   //重复输出空格以填满缓存
        flush();                        //刷新缓存
        sleep(1);                       //停顿一秒钟
    }
?>
```

当运行上面这段脚本后，会发现浏览器中出现了在数组中定义的问候语及其使用的语言，如图 3-1 所示。

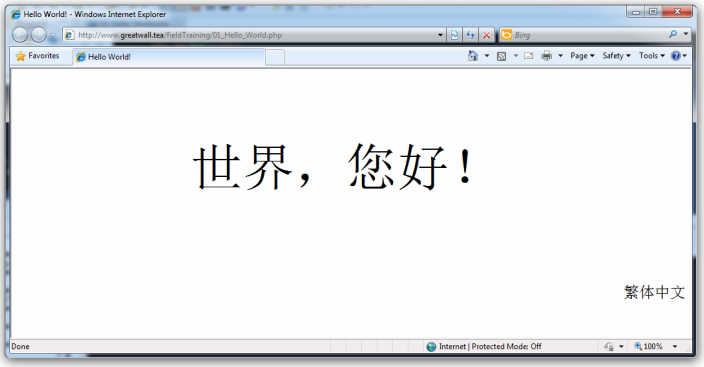

图 3-1　Hello World（繁体中文）

这是怎么做到的呢？一起来看看代码：

在上面这段脚本中，首先定义了两个数组，一个用于存放问候语（$msgArray），另一个则用于存放语言（$langArray）。注意，这两个数组中的问候语和语言是一一对应的，以便于后续从数组中取值时得到的结果也是一一对应的。

由于这些问候语不是按照它们在数组中的顺序在浏览器中出现的。为了防止同一问候语连续多次出现的情况，我们定义了一个变量，用于存放当前问候语对应的数组索引。该变量会在随机产生下一句问候语的数组索引时使用。即$idPrev。这个变量用来存储上一次显示的问候语在数组中的索引。如果当前问候语在数组中的索引与该变量的值相同，则该变量的值加 1，并将加 1 后的结果赋值给存储当前索引的变量。

接下来，我们定义了一个 for 循环用来输出问候语到浏览器。这个 for 循环由四步构成，分别是：

（1）随机生成索引并比较与上一次显示的问候语在数组中的索引。

（2）根据生成的索引在两个数组中查找相应的问候语及其使用的语言。

（3）将查找到的问候语及其使用的语言输出到浏览器。

（4）让系统在刷新缓存后停顿 1 秒钟，再次执行 for 循环。

第一步，定义一个变量$id，用于存储使用 rand()函数生成的一个 8 以内的随机数。若变量$id 的值与$idPrev 相同，则将变量$idPrev 的值加 1 并将计算结果赋值给变量$id，若此时变量$id 的值又大于 8，则将变量$id 的值定为 8。这样一来，就不会出现查找不到问候语的情况了。

第二步，根据生成的索引在两个数组中查找相应的问候语及其使用的语言。具体的做法是，先定义两个变量$msg 和$lang，然后使用在第一步中生成的变量$id 的值做为索引，在数组$msgArray 和$langArray 中查找相应的问候语和使用的语言，并将查找到的内容赋值给变量$msg 和$lang。

第三步，使用 echo 语句输出两条 JavaScript 语句，用来向 HTML 代码中的两个 DIV 标签添加相应的内容。这时，浏览器中就出现了一条问候语及该问候语使用的语言，如图 3-1 所示。

第四步，使用 flush()和 sleep()两个函数用来刷新系统缓存和让服务器暂时休息。关于这两个函数的具体内容，大家可以查看 PHP 官网上的介绍。

至此，一次循环就完成了。浏览器中的信息会每隔一秒钟刷新一次。在刷新 10 次后，浏览器中的信息就不再变化了。读者可以尝试着向数组中添加更多的内容，同时修改循环语句中的变量$i 的最大值，从而展现更加丰富的信息。

在这段脚本中，有很多的概念，比如数组、循环和条件判断等，都是大家第一次遇到。如果不懂，也没有关系，后续的章节中对这些内容进行详细地讲解。现在大家只用看看热闹就好。

3.4　习　　题

（1）请使用 echo 命令在浏览器中输入下面这一段文字：

```
"PHP 是一个免费开源的项目，无论是在 Web 服务器上部署 PHP 引擎，还是使用 PHP 代码编写网站，
你都不需要花费一分钱。天下到底还是有免费的午餐。"
```

（2）请在刚才编写的脚本中使用"//"添加一段注释，注释内容如下：

```
"这是一条注释。"
```

（3）请将如下的内容以注释的形式添加到脚本中：

```
项目名称：向世界说你好
负责人：张小二
开始时间：2013 年 11 月 24 日星期日
结束时间：2013 年 11 月 25 日星期一
```