第8章　PHP主页设计

随着Internet技术的不断普及和发展，越来越多的应用服务开始采用基于　WWW的Browser/Server形式，网络系统开发人员迫切需要一种高效的 Web交互式数据库开发环境，这种开发环境应该具有以下的特点：

1) 强大的数据库支持能力。

2) 跨平台运行能力。

3) 高速运行环境。

4) 强大的字符串和图形处理功能。

5) 与HTML无缝集成。

现有的技术解决方案主要采用服务器端脚本技术，即客户端浏览器产生 HTTP请求，由通过服务器端脚本对HTTP请求处理并产生响应，最终在客户端产生动态的 HTML文档。通常采用的服务器端脚本技术主要分为非嵌入式的服务器端脚本和嵌入式服务器端脚本两种。非嵌入式服务器端脚本即 CGI(Common Gateway Interface)，在应用中常用的 NSAPI、ISAPI和FastCGI等都是对它的改善和扩展。嵌入式服务器端脚本随服务器种类的不同有很多种，目前最常用的是Microsoft的Active Server Pages(ASP)。由于非嵌入式的服务器端脚本和嵌入式服务器端脚本的工作原理不同，其在功能上也各有优劣。非嵌入式的服务器端脚本 (即CGI)由于还要处理HTTP请求/响应的输入、输出部分，因此开发过程比较复杂，运行速度也比较慢，但由于它本身是位于服务器之外、作为系统的一个标准进程运行的，可以不受服务器进程本身功能的限制，处理功能要强于嵌入式服务器端脚本，并且由于它是对整个 HTTP流进行操作的，可以由MIME映射产生图像流、视频流和音频流等多种的服务形式。嵌入式服务器端脚本是内嵌于Web服务器进程之内的，其脚本架构于 HTML文档的基础之上，由Web服务器进程负责查找、处理和替换服务器处理的部分，并在基于本身 HTML的基础上产生输出。这种脚本的功能是由Web服务器的功能所决定的，因此在扩展性能方面开发难度比较大，并且可移植性不好，因为脚本的功能必须依赖于服务器。在实际的网络应用开发过程中，嵌入式服务器端脚本由于开发周期短、编程形式简单、与 HTML结合比较好而受到了很多系统开发人员的青睐，成为Web数据库开发方案的主流，其中应用最多的开发环境即 Microsoft的Active Server Pages(ASP)。但是ASP技术也有其不足之处，主要表现为 ASP服务器脚本不具备跨平台运行能力，并且其通过ODBC连接数据库的效率比较低，其宿主语言 VBScript对文本和图像的处理能力不够强等。下面介绍另外一种嵌入式服务器端脚本语言 PHP，供Linux网站建设者参考。

8.1　内嵌式脚本语言PHP概述

8.1.1　PHP 发展历史

最初的PHP3的雏形诞生于1994年秋季，Rasmus Lerdorf编写了一个用于和他的在线简历的访问者保持联系的CGI，这就是最初的未发布的 PHP。1995年初发布了第一个公开的 PHP，

在当时只是被用作一个个人主页开发工具。1995年，Rasmus Lerdorf重写了整个解析器，并取名为PHP/FI 2，这是第二个公开发行的版本。FI来源于他写的另外一个将HTML数据和数据集成的软件包。此后PHP/FI便以惊人的速度传播开来，人们开始大量在网页设计中使用它。到1997年，Zeev Suraski和Andi Gutmans全面重写了该分析器(Parser)，大量的PHP/FI中的功能被移植，其中很大部分已经完全改写。这构成了我们今天使用的PHP3的核心部分。

到1999年，PHP/FI和PHP 3都已经有了很多的商业版本，比如我们常用的Red Hat Linux中就包含了PHP3的分析器。根据NetCraft的保守估计，世界上已经有至少15万商业站点使用PHP，甚至超过了Netscape公司企业版网络服务器的用户数量。

8.1.2 PHP的主要技术特点

PHP最初只是用C语言开发的一个CGI程序，上文介绍的发展和流传，现在已经几乎成为一门语言。它具有如下的技术特点，如果你希望通过简单的方法使用这些特性的话，PHP是最好的选择：

1) 多种数据库支持，PHP内置了对大多数数据库系统的支持，并且可以根据开发的需要进行扩充。

2) 强大的字符串处理功能和图形支持，PHP内置有功能强大的字符串处理函数，并且可以利用图像函数动态产生Gif图像并输出到服务器，这对于以文本和图形内容为主的Web应用是非常有用的。

3) HTTP验证和数据加密，PHP可以直接操作HTTP流，因此可以通过写HTTP头“WWW-Authenticate”实施客户端HTTP验证，并且可以对数据进行多种形式的加密。

4) 网络通信功能，PHP提供了Socket通信接口，可以方便地实现网络通信与服务功能

5) 多平台代码的一致性，PHP可以运行在多种操作系统平台上，并且在多种平台上可以保证脚本代码的一致性，这为整个应用系统的扩充和移植打下了良好的基础。

另外，PHP可以在Windows 95/98/NT下工作，也可以在Linux/UNIX下工作，由于Windows和Linux/UNIX系统的服务机制不同，PHP在这两种平台上的运行机制也不同，PHP根据不同的平台自动选择执行的方式并返回结果，但对于开发人员和客户端而言，这种机制是完全透明的，无论从编程角度或客户端，完全体会不到这种区别的存在。因此，PHP的程序是与开发平台无关的。

8.2 PHP语句

8.2.1 初识PHP

现在我们正式进入PHP3的世界，首先对它来一点感性的认识。和所有HTML文件一样，PHP文件也是标准的ASCII文本，所以可以用任意的文本编辑器编写PHP文件，也可以在网页制作工具(如Microsoft FrontPage)中打开HTML编辑窗口，向其中添加PHP代码。注意，文件的后缀名应该是PHP或者PHP3，这样，当Web服务器收到相应的访问请求以后，才会自动调用PHP3解释器，进行PHP代码的解释工作。其实就是在服务器端运行这个程序：

```
<HTML>
  <HEAD>
```

```
    <TITLE>PHP3 Script-1</TITLE>
  </HEAD>
  <BODY>
    <CENTER>
    <?
      echo "Hello, world! ";
    ?>
    </CENTER>
  </BODY>
</HTML>
```

把它放在任何一个能通过 HTTP访问到的目录，取名 index.php，再用浏览器连接该文件，就可以看见其效果了，本书已经在上一章中介绍了 PHP3的安装配置方法。请注意，只有在服务器端正确安装和配置了 PHP3.0才能看到你希望的结果，而且必须通过浏览器的访问(http://hostname/index.php)才行，这也是一般服务器端脚本语言调试起来比较麻烦的地方。

从这个例子应该很容易看出 PHP3的基本语法：用＜？和？＞括起来的部分就是 PHP的服务器端脚本程序段，解释器对其余部分不作处理，而这一部分代码则在服务器一端执行后以它的输出代替，这里的 echo语句输出一个无格式的字符串。因此，客户端收到的内容是这样的(可以通过浏览器选择“ 源文件 ”看到这些内容)：

```
<HTML>
  <HEAD>
    <TITLE>PHP3 Script-1</TITLE>
  </HEAD>
  <BODY>
    <CENTER>
    Hello, World!
    </CENTER>
  </BODY>
</HTML>
```

也就是说， PHP程序虽然是文本，但客户端只能看见在服务器端解释后的结果，因此对客户端是保密的，这同样是绝大部分服务器端脚本的特点。

8.2.2 PHP语句和HTML分离

上边已经提到， ＜？和？＞的作用是告诉服务器，这两个标记之间是 PHP语句，需要在服务器端事先执行，再把结果送出到客户机上。还有其他两种方法可以表示 PHP语句：

```
<Php
  echo "Hello,World!";
? >
```

这种方式一般用于XML页面。

```
<SCRIPT LANGUAGE="php">
  echo "Hello,World!";
</SCRIPT>;
```

有些主页设计工具不支持＜？...？＞这样的结构，有时会自动将＜？ ...？＞改为它认为“ 正确 ”的东西，比如常用的Microsoft FrontPage，这时候就可以用上面的方法。

在同一个标签内，表示语句结束的分隔符是分号“ ； ”。当然，结束标签“ ？＞ ”也表示

语句的结束，所以最后一个语句末尾不用加分号。

在本书的很多示例中，为了简洁和直接起见，省略了<？和？>符号，但是读者应当明白这些是PHP语句，在实际的脚本文件中应当位于<？和？>之间。

8.3 PHP中的变量

8.3.1 变量名和变量类型

PHP是一种解释性的语言，就像在 BASIC中一样，不必进行变量说明，甚至可以直接调用从来没有定义和赋值的变量，当然得到的结果一般是一个空字符串 (有些时候也可能会是莫名其妙的值)。

很多脚本语言用一个特殊的符号作前缀来表示变量名，在 PHP中，这个前缀是“ $ ”号，例如:

```
<?
   $A=1;
   $B=1.0;
   $C="Hello! ";
?>
```

以上都是合法的PHP变量赋值。这些变量分别是整数、双精度数和字符串类型。 PHP支持的变量类型有：

integer：整数

```
$A=100              //十进制正整数
$A=-100             //十进制负整数
$A=0100             //八进制整数，由0作前缀
$A=0x100            //十六进制整数，由0x作前缀
```

double： 双精度数

```
$A=0.0111           //双精度数
$A=1.11e-2          //用科学计数法表示的双精度数
```

string：字符串

在PHP中，字符串用单引号或双引号表示起始。需要注意的是，这两种引号在使用中有一些细微的差别：对于双引号而言，可以在字符串中使用所谓“ 转义字符 ”，见表8-1。

表8-1 换码符号及其说明

换码符号	说 明
\n	回车：光标到下一行行首
\r	光标到本行行首
\t	制表符，光标到下一个制表位
\\	字符“ \ ”
\$	字符“ $ ”
\ ”	字符“ " ” (双引号)

另外，系统还将用双引号包括的字符串中的变量名用该变量的值替换。例如：

```
<?
  $A="world";
  echo "Hello $A!";
```

```
?>
```

等效于：

```
<?
  echo "Hello world!";
?>
```

array：数组。

object：对象。

pdfdoc：PDF文本(仅仅在起用PDF支持之后生效)。

pdfinfo：PDF信息(仅仅在起用PDF支持之后生效)。

对于一般类型的变量，只要简单地对它赋值就可以实现初始化的工作，第一次为变量赋值的时候，就同时分配了内存空间，同时也决定了变量的初始类型。变量类型通常不由程序员来设置；它在运行时会由 PHP依据上下文来自动设定。当然，如果在程序中有必要将某个变量转换到指定的类型，也还可以对该变量进行强制类型转换，语法和 C语言中的一样。或者使用下一章介绍的函数“ settype() ”。 变量类型强制转化的例子如下：

```
<?
   $A="10 minutes";
   $B=35;
   $A=(int)$a;
   $B=(string)$b;
?>
```

这样，我们把变量$A强制转化为了整型，它的值是整数 10；变量$B被强制转化为字符串型，其值是“ 35 ”。

8.3.2 深入了解变量类型转化

上一节中已经说过， PHP中不需要进行显式的类型声明，一个变量的类型取决于它被赋予的当前值的类型。也就是说，如果你给变量赋一个字符串值，它就成为字符串变量。如果接着又赋给它一个整型值，它立刻变成了一个整型变量。一个 PHP自动类型转换的例子是加法操作符“ + ”。如果任何一个操作数是double型，则所有的操作数转化为double型来计算，结果当然也是double型。否则，所有的操作数按integer类型来计算，结果也是integer型的。注意：操作数本身的类型没有发生变化。例如：

```
$MyVar="0";   //声明一个字符串变量，ASCII值为48。
$MyVar++;     //还是字符串变量，ASCII值为49，表示字符串"1"。
$MyVar+=1;    //现在成为一个整型变量，当前值为2。
$Temp=$MyVar+1.3;   // $Temp是双精度型，当前值3.3。$MyVar还是整型。
$MyVar=$MyVar+1.3;   //现在$MyVar被转化成为双精度型，当前值3.3。
```

在PHP中，强制类型转换和在C语言中类似：在变量前添加前缀，在括号中写出目标类型名。例如：

```
$MyVar= 10;    // $MyVar被初始化为整型变量
$Temp=(double) $MyVar;    // $Temp是双精度型变量，初值10.0
```

在PHP3中允许的前缀有：

```
(int), (integer)转化为整型
(real), (double), (float)转化为双精度型
(string)转化为字符串型
(array)转化为数组
```

(object)转化为对象

8.3.3 数组类型

对于PHP中的数组类型，一般有两种方法赋值：一是使用 array()函数；更简单的是像普通的赋值语句那样，把值直接赋给某一个数组变量，注意数组元素会自动扩展，所以不必事先设定上限，这也是一个十分方便的特性。对确定元素赋值的例子如下：

```
$A[0]=2;
$A[1]=3;
```

甚至可以这样用：

```
$A["a"]=1;
$A["b"]="Hello!"
```

如果只是要简单地将元素连续地添加到数组尾部，只需简单地将值赋给不带下标的数组变量：

```
$A[]=2;
$A[]=3;
```

这段语句的运行结果和第一段例子程序是一样的。

8.3.4 动态变量

动态变量指变量名可以被动态地赋值和使用，这就意味着变量名本身也是一个变量，例如：

```
$a = "hello";        变量$a的值为字符串hello
$$a = "world";      现在，$$a意味着$hello
Echo "$a ${$a}";
```

这时，我们定义了两个PHP变量，分别是$a和$hello。上面最后一条语句等价于：

```
echo "$a $hello";
```

它们都将输出：hello world

如果在数组中使用动态变量，就必须解决二义性的问题。例如对于语句 $$a[1]，解析器需要知道，你是要引用以$a[1]的值为变量名的变量呢，还是名为 $$a的数组的第1个元素，也就是下标[1]和那一个变量结合的问题。一般说来，第一种情况最好写成 ${$a[1]}，第二种情况则写成${$a}[1]。

8.3.5 类和对象

PHP中类的定义和C++相似：

```
Class clMyClass
{
  Function ShowTip()
  {
    Echo "Now running function in MyClass";
  }
}
```

对象不能像普通变量一样用赋值来初始化，而需用 new语句建立该类型的新对象实例，这

时系统为它分配储存空间：

```
$A = new clMyClass;
$A ->ShowTip ();
```

上面这个例子中，第一个语句新建一个 clMyClass类的对象实例，第二个语句调用这个对象实例的函数ShowTip()。

8.3.6 变量作用域

最后需要说明的就是变量的作用域，由于 PHP在这方面的和C的差异，常常导致实际编程中的错误调用。在PHP中，凡是没有声明作用域的变量，一律认为是局部的，先看一个例子：

```
$A=1;
Function TestA ()
{
  Echo $a;
}
Test ();
```

这段程序不会输出任何的东西，因为 echo语句要输出局部变量$A，但是在函数内的$A从未被赋过值，于是被认作一个空字符串。注意，在 C语言中全局变量可以直接在函数内引用，除非它被一个局部变量所覆盖，但是在 PHP中要在函数内部使用全局变量必须显式地说明：

```
$A=1;
$B=2;
Function Sum ()
{
  Global $A, $B;
  $B = $A + $B;
}
Sum ();
Echo $B;
```

上面程序将输出"3"。通过在函数内部的global语句将$A和$B声明为全局，从而使用这两个变量。不过，基于结构化程序设计思想，一般不主张使用全局变量，而建议使用参数传递的办法

有关函数作用域的另外一个重点是静态 (static)变量。static变量存在于局部函数中，但当程序离开这个函数时系统将保留它的值，下次重新进入这个函数的时候，原来的值仍然有效，当使用函数的递归调用的时候，往往要用到静态变量。

先看不使用静态变量的例子：

```
Function CallCounter ()
{
  $Count=0;
  $Count++;
  Echo $Count;
}
```

程序员希望在每次调用这个函数的时候显示输出它被调用的次数，并用变量 $Count作为调用次数的计数。但是这样实现不了预计的功能，因为每次调用时它都先把 $Count初始化为0，再自增，然后显示出“ 1 ”，最后，当函数调用结束时，变量 $Count就被释放了，因此没有

“记忆”任何以前的数值。要使记数程序有效计数而不丢掉当前的记数结果， $Count要声名为static变量：

```
Function CallCounter ()
{
  Static $Count=0;
  $Count++;
  Echo $Count;
}
```

现在，每次调用函数它都会显示并且增加 $CallCount的值，而且把这个值保留下来。

8.4 外界变量和交互式网页

我们在网上看到通常的主页都向访问者提供信息，包括文字、图像、声音甚至多媒体动画，这样的网页可以做得十分漂亮，初级的网页仅仅能够向客户端送出信息。

8.4.1 PHP从客户端获取信息的方法：外界变量

在PHP中，通过外界变量，就可以十分方便地从客户端获取信息。顾名思义，外界变量就是从当前的PHP脚本文件以外定义和赋值的变量。对于程序员而言，外界变量的值是“自动”获得的，只需要在脚本程序中引用变量名就可以了，至于具体的外界变量产生机制，完全可以不去理会。

由三种方法可以产生外界变量，并通过它从客户端获取信息，有些信息是由用户 (网站访问者)输入或者选取的；还有些信息，例如访问者的 IP地址等，是网络自动传送的。下面将分别介绍这些用法。

8.4.2 使用HTTP Form获取信息

在HTML表单提交的时候，表单内各域的内容是包含在HTTP头里传送到服务器的，PHP解释程序将在接受、处理表单的PHP程序中自动转化成同名的标准PHP变量，看下面这个例子：

```
<HTML>
  <HEAD>
    <TITLE>PHP3 Test Page</TITLE>
  </HEAD>
  <BODY bgcolor="#ffffaa">
    <CENTER>
    <?
      echo "<h2><font color='#ff0000'>欢迎光临PHP3测试主页<br>";
      if($uname) echo 原来你是 $uname 呀，很高兴认识你!";
    ?>
    <FORM action="test.php" method="post">
       你的大名是：<input type="text" name="uname"><br>
            <input type="submit" value="点击这里，告诉我”>
    </FORM>
    </CENTER>
  </BODY>
</HTML>
```

将这个文件命名test.php，放到可以访问的目录，用浏览器访问它，注意一定不要在本地直接打开文件。先看第一次浏览的结果，窗口内显示了表单内容 (见图8-1)。

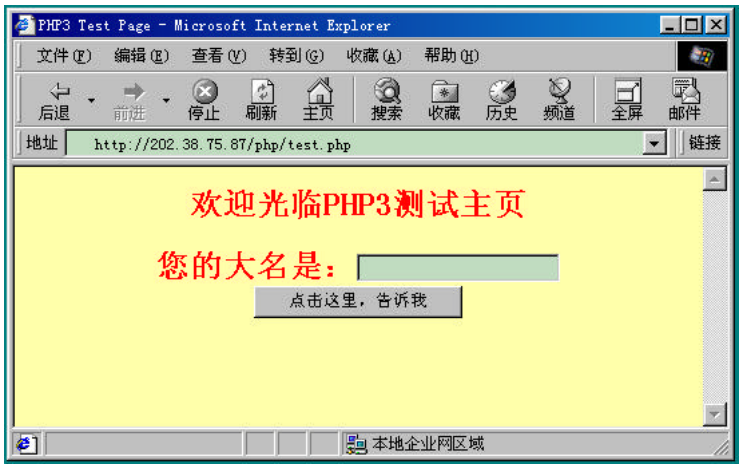

图 8-1

这里设定表单的处理程序就是它本身 (form action=“ test.php ”)，为了区别第一次访问是显示的内容和处理提交信息后的内容，在 PHP代码段中使用判断语句，如果变量 $uname相当于布尔型的“ 假 ”(空字符串或者 0)，也就是说没有收到表单提交的内容，就不显示“ 原来你是… ”的内容。

现在在对话框内填入文字信息，然后点击提交按钮，如上文所述，服务器收到表单以后，用test.php来处理它，这时，自动生成了一个名为 $uname的变量(其实是客户端送来的HTTP标头里包含的)，这里的变量名由表单中文字输入框的 name属性决定。在echo语句中，由于字符串被双引号包括， PHP自动将变量名用它的值替换。这时在浏览器上看到下面的样子，注意我们仍然保留了表单部分，用户可以继续输入新的内容并且提交表单，在浏览器上将看到新的输入(见图8-2)。

图 8-2

8.4.3 HTTP Cookies与客户端信息

HTTP cookies 最初由Netscape's Spec定义，利用HTTP头，将一些数据信息保存在客户浏览器端，对于Microsoft Windows 98用户，在Windows目录底下有一个cookie目录，在里边可以看到一些文件名具有 user@website形式的文件，这就是该 website保存在你计算机里的cookies。一般而言，网站利用cookies和用户保持联络或验证用户身份。

PHP对HTTP cookies提供支持，可以很方便地使用 setcookie() 函数来设置cookie的值。Cookies包含在HTTP请求的头部的一部分，所以必须在任何输出数据返回给用户浏览器前调用setcookie()函数，这一点类似Header() 函数的限制。任何用户端返回的cookies都将被自动转换为标准的PHP变量，就像GET和POST方法提交的表单数据。设置cookie时注意两点：

首先，如果要在一个 cookie中设置多个值，要给这个 cookie的名字后边加上方括号 ([])，就像定义一个数组。例如：

```
setcookie("MyCookie[]","Testing", time()+3600);
```

其次，同名的cookie第二次赋值将覆盖掉你的浏览器中原有的那一个 cookie值，除非它们具有不同的路径或域。

8.4.4 使用环境变量获取客户端信息

随HTTP请求发到服务器的还包括“ 环境变量 ”的值， PHP自动将环境变量转换为普通的变量。也可以可使用getenv()函数。例如，下边两个语句完成相同的功能：

```
echo $REMOTE_ADDR;
echo getenv("REMOTE_ADDR");
```

两者都返回客户端的 IP地址，但是后一种方法比较好，可以避免因为同名的表单变量或cookie引起的混淆。

8.5 基本控制流程：分支和循环

8.5.1 条件语句和分支结构

1. IF和IF...ELSE分支语句

PHP中也包含分支和循环语句，语法和 C语言相似，有经验的程序员可以很轻易地上手。PHP的条件语句一般有下面几种形式：

```
If  (Expression)
  Statement;
```

当表达式的值大于 1的时候(也就是逻辑表达式结果为真)，执行程序段；否则跳过程序段执行后边的语句。

```
If  (Expression)
  Statement1;
Else
  Statement2;
```

当表达式的值为“ 真 ”的时候，执行后续的程序段 1；否则转向执行 else子句以后的程序段2。

2. ELSEIF多分支语句

使用ELSEIF语句，当条件语句的判断结果为“非”的时候，还可以进入第二层条件判断，语法如下：

```
If  (Expression1)
  Statement1;
Elseif (Expression2)
  Statement2;
......
Else
  Statement3;
```

当第一个表达式的值为“真”，执行第一个程序段；否则进入第二个表达式的判断，为“真”则执行程序段2，否则进入下一次判断.....如果最后有else字句的话，则在所有的条件表达式都不满足的时候执行else字句以后的程序段。

特别要注意的是，在同一组if...elseif...elseif.....else...的结构中，如果在某种情况下使得一个以上的条件被满足，PHP3解释器仅执行第一个被满足的if或elseif后的程序段。请看下面这段某Web BBS的欢迎程序：

```
......
Echo "你已经光临小站" . $nLoginCount .次" ";
If($nLoginCount>=100)
  Echo "现在是本站的一般站友！";
elseif ($nLoginCount>=500)
  Echo "是本站的老朋友了！";
else
  Echo "还是本站的新手呢！";
......
```

上面的程序希望实现这样的功能：根据用户登录次数多少，分别显示不同的欢迎信息。但是，仔细分析可以发现，这段程序其实达不到预计效果：对于一个登录者的登录次数$nLoginCount(这是从用户数据库中取得的数据，具体的内容请参看本书的相关部分)，当数值超过500的时候，第一个判断条件“If ($nLoginCount>=100)”已经被满足，系统将显示“现在是本站的一般站友！”，然后就跳出这段多分支条件结构，而后面的elseif语句就被跳过。总而言之，对于elseif条件后的程序段，仅仅当本模块前面的条件判断都为“假”，且本模块条件为“真”的时候，才能够被执行。修正后的程序段是这样的：

```
......
Echo "你已经光临小站" . $nLoginCount .次" ";
If($nLoginCount>=500)
  Echo "是本站的老朋友了！";
elseif ($nLoginCount>=100)
  Echo "现在是本站的一般站友！";
else
  Echo "还是本站的新手呢！";
......
```

最后，我们再看一个多分支语句的例子：

```
<?
  If ($a > $b)
    {
```

```
    print "a is greater than b";     大于$b
    }
  Elseif ($a == $b)
    {
    print "a is equal to b";         等于$b
    }
  Else
    {
    print "a is smaller than b";     小于$b
}
? >
```

3. SWITCH语句实现多分支结构

在实际问题中常常需要用到多分支的选择，比较简单的情况可以像上面的例子那样，使用多重嵌套的IF语句来处理。但对于分支较多的情况，嵌套的 IF语句层数多，使得程序冗长而且可读性降低。对于这种情况， PHP提供了Switch语句来直接处理多分支选择。 Switch语句的语法结构如下：

```
Switch(Expression)
{
  Case Expression1:
    Statement1;
  Case Expression2:
    Statement2;
  ......
  Default:
    Statement;
}
```

Switch后面的表达式(Expression)通常是变量或者包含变量的算式，而 Case后面的表达式(Expression1、Expression2...)通常是常量。

Switch执行的过程是这样的：一条语句一条语句地扫描源程序段，并且首先计算 Switch表达式的值；当没有Case语句表达式的值与之匹配(相等)的时候，不执行任何代码；当出现一个Case表达式与Switch表达式的值相等的时候，从这个分支开始执行语句，直到 Switch模块结束；如果没有任何分支被匹配，默认 (Default)分支将被执行。要注意的是，在每一个分支的程序段末尾，我们必须加上一条 Break语句，否则后续的分支将继续被执行，直到整个 Switch模块结束。

通过下面这个实例，可以更好地理解 Switch多分支语句：

```
......
Switch($nLoginCount)
{
  Case 100:
    Echo "恭喜！你从今天起成为小站的一般站友！";
    Break;
  Case 500:
    Echo "恭喜！从今天起，你成为小站的老朋友了！";
    Break;
  Case 1000:
    Echo "恭喜恭喜！到今天，你已经光临小站一千次了！";
```

```
    Break;
  Default:
    Echo "这是你第" . $nLoginCount .次光临小站";
}
......
```

4. 条件语句与HTML的交互文法

上文所述的条件分支程序段应该位于同一标签内。对于某些情况下的需要， PHP也提供了在不同的标签内使用分支结构的方法。这时，在 If语句后用冒号“ :”代替大括号，然后就可以在PHP语句中加入HTML，在分支结束的时候，使用 Endif语句。语法结构如下：

```
<? If (Expression): ?>
HTML Statement
<? Endif ?>
```

实际的例子如下：

```
<? If ($A==5): ?>
A = 5
<?Endif; ?>
```

这里，在两个PHP标签中插入了一段HTML代码，只有在if条件满足的时候(这里就是$a等于5)，才会执行这段代码 (显示“ A=5 ”的信息)。同样可以用这种格式改写上面第 2小节的一段程序：

```
<?
  If ($a > $b):
    print "a is greater than b";     大于$b
  Elseif ($a == $b):
    print "a is equal to b";         等于$b
  Else:
    print "a is smaller than b";     小于$b
  Endif;
? >
```

8.5.2 循环语句

基本程序控制流程的另一类型是循环， PHP支持多种循环格式，一般情况下任何一种格式都可以实现相同的功能，但是，熟练掌握这几种格式，可以使程序编制更容易，程序段更加简洁、高效。

1. WHILE循环

While循环就是结构化程序设计中所谓“ 当 ”型循环，它的语法结构如下：

```
While (Expression)
  Statement;
```

While语句的的含义非常明显：当 Expression条件表达式为真的情况下循环地执行嵌套的语句(Statement)，否则退出循环。在每一次循环开始的时候检查表达式的值，因此，如果这个值在循环体内部被修改，执行过程将不会被终止，直到该次循环体结束，再次检查表达式的值。如果While表达式的值在开始就是假，则循环体不会被执行。图 8-3可以帮助你更好地理解While循环的实质。

和在C语言中一样，如果循环体超过一条语句，应该用大括号将它们括起来。另外，和上

面介绍的条件分支语句类似， While语句也支持HTML交互文法，这时用Endwhile表示循环体结束：

```
<? While(Expression): ?>
  HTML Statement
 <? Endwhile; ?>
```

2. DO...WHILE循环

DO…WHILE循环就是“直到”型循环，语法结构如下：

```
Do
  Statement;
While(Expression);
```

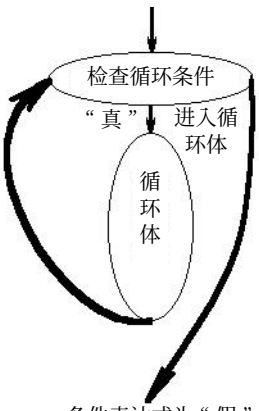

图 8-3

DO...WHILE循环和WHILE循环十分相似，但是它的循环条件检查是在每个循环过程最后进行。这造成的最主要的差别在于 DO...WHILE循环的第一个循环过程总是会被执行，第一次循环条件的检验在循环体执行一次以后进行。而 WHILE循环中的第一个语句不一定被执行，因为第一次的循环条件检验在循环体执行之前。

3. FOR循环

在PHP中FOR循环是最复杂的循环。它的特性和它在 C语言中的表现类似。 FOR循环的语法如下：

```
FOR (Expression1; Expression2; Expression3)
  Statement;
```

循环开始时先计算(执行)第一个表达式Expression1。每一个循环开始前，进行循环条件检验：第二个表达式 Expression2将被计算，如果它的值为真，则该继续执行下一次循环体；如果它的值为假，则循环结束。在每次循环的最后 (循环体执行结束以后)第三个表达式将被执行。

任何一个表达式都可以为空，当表达式二为空的时候， PHP和C一样默认它为真，这是除非遇到break或者return之类的强制调转语句，否则循环将一直进行下去。

下面这个例子很有意思，它的功能是显示各种不同的 HTML标题：

```
<HTML>
  <HEAD>
    <TITLE>PHP3 Test Page</TITLE>
  </HEAD>
  <BODY bgcolor="#ffffaa">
    <CENTER>
    <?
      FOR($I=1;$I<=6;$I++)
        echo "<H".$I.">这是标题".$I."\n";
    ?>
    </CENTER>
  </BODY>
</HTML>
```

看看源文件，可以对PHP的“服务器端解释”机制有更深入的理解：

```
<HTML>
  <HEAD>
```

```
    <TITLE>PHP3 Test Page</TITLE>
  </HEAD>
  <BODY bgcolor="#ffffaa">
    <CENTER>
    <H1>这是标题1
    <H2>这是标题2
    <H3>这是标题3
    <H4>这是标题4
    <H5>这是标题5
    <H6>这是标题6
    </CENTER>
  </BODY>
</HTML>
```

4. BREAK语句

BREAK无条件跳出当前的循环体，提前结束循环。

5. CONTINUE语句

CONTINUE结束当前循环体的执行，跳到循环的开始处，进行下一次循环条件的检验。

8.6 表达式基础

上面的例子中已经用到很多 PHP运算符，对于稍有经验的程序员来说，简单的运算符是一目了然的事，我们只进行最简单的说明。

8.6.1 算术运算符

PHP 的算术运算符和C语言中的定义完全一样，见表 8-2。

表8-2 算术运算符及其说明

算 符	名 称	说 明
+	加法运算符	$A+$B返回两个变量的和
-	减法运算符	$A - $B返回两个变量的差
*	乘法运算符	$A*$B返回两个变量的乘积
/	除法运算符	$A/$B返回$A除以$B的商，如果两个操作数都是整型的话，返回整除的结果，小数部分被舍去。如果任一个操作是浮点型，返回浮点型值
%	取模运算符	$A%$B返回$A除以$B的余数

8.6.2 字符串运算符

PHP中只有一个字符串运算符：“ . ”，表示字符串连接。用“ $A . $B ”返回将串$B连接在串$A尾部得到的合并字符串。

8.6.3 赋值运算符

赋值运算符的真正含义是将运算符右部表达式的值赋予算符左边的变量名。 PHP 和C语言一样，除了最简单的赋值算符“ = ”以外，还提供和 C语言中类似的组合赋值操作符，例如：

```
$A+=5;     //等价于：$A=$A+5
$B.= "end";      //等价于：$B=$B . "end"
```

8.6.4 位运算符

PHP中提供了位运算符，对操作数按照二进制位进行运算，见表 8-3。

表8-3 位运算符及其说明

算 符	名 称	说 明
&	按位与	$A & $B 变量$A和$B按位进行“与”运算
|	按位或	$A | $B 变量$A和$B按位进行“或”运算
~	按位非	~ $A 对变量$A按位取非

8.6.5 逻辑运算符

见表8-4。

表8-4 逻辑运算符及其说明

算 符	名 称	说 明
and	与	$A and $B 变量$A和$B同为“真”时结果为“真”
or	或	$A or $B 变量$A和$B有一个为“真”时结果为“真”
xor	异或	$A xor $B 变量$A和$B有且只有一个为“真”时结果为“真”
!	非	!$A 变量$A为“假”时结果为“真”
&&	与	$A && $B 变量$A和$B同为“真”时结果为“真”，同and
||	或	$A || $B 变量$A和$B有一个为“真”时结果为“真”，同or

8.6.6 比较运算符

PHP比较运算符合C语言中的定义完全一样，见表 8-5。

表8-5 比较运算符

算 符	名 称	算 符	名 称	算 符	名 称
==	等于	<	小于	<=	小于或等于(不大于)
!=	不等于	>	大于	>=	大于或等于(不小于)

8.7 PHP常用函数和MySQL数据库访问函数

8.7.1 PHP内置数学函数

下面简要列出PHP支持的常用数学函数，参数含义一目了然的，就不多加说明了。

Abs()：返回参数绝对值。

Acos：返回参数的反余弦。

Asin：返回参数的反正弦。

Atan：返回参数的反正切。

BinDec：将参数由二进制数转化为十进制数，参数应该以字符串形式保存。

Cos：返回参数(弧度)的余弦值。

DecBin：将参数由十进制数转化为二进制形式，注意返回值是一个字符串。

DecHex：将参数由十进制数转化为十六进制数的形式，注意返回值是一个字符串。

DecOct：将参数由十进制数转化为八进制数的形式，注意返回值是一个字符串。

Exp(arg)：返回e^{arg}的值。

Floor：返回大于或等于参数值的最小整数。

Getrandmax：返回rand()函数能够返回的最大随机数值。

HexDec：将参数由十六进制数转化为十进制数，参数是字符串形式。

Log：返回参数以自然对数(e)为底的对数值。

Log10：返回参数以10为底的对数值。

Max：返回多个参数的最大值。

Min：返回多个参数的最小值。

OctDec：将参数由八进制数转化为十进制数，参数是字符串形式。

Pi：返回圆周率π

Pow(float base, float exp)：返回base的exp次方，即$base^{exp}$。

Rand：返回当前伪随机序列的下一个数。随机数下界为 0，上界可由gettandmax()函数返回。如果想要求常数a到b之间的随机数(b>a)，可以用rand() % (b-a)+a。注意，取随机数前一定要用srand()函数初始化伪随机序列。

Round：返回参数经四舍五入取整的值，但返回值是一个双精度数，小数部分为 0。

Sin：返回参数(弧度)的正弦值。

Sqrt：返回参数的平方根。

Srand：初始化伪随机序列。

Tan：返回参数(弧度)的正切值。

8.7.2 日期时间函数

下面介绍PHP支持的时间、日期函数。

1. mktime函数

int mktime(int hour, int minute, int second, int month, int day, int year,int [is_dst]);

返回一个标准的UNIX/Linux timestamp时间、日期值，以提供下面的函数作为参数使用。注意参数依次是小时、分、秒、月、日、年，与通常的 UNIX/Linux系统不同。

2. date函数

string date(string format, int timestamp);

第一个参数是格式字符串， PHP将相应的格式替换转化为时间、日期值以后作为返回值。第二个参数是时间、日期值(timestamp，可用mktime函数获得)，如果缺省的话，用当前时间。替换规则如下。

a：上下午，用“ am ”、“ pm ”表示。

A：上下午，用“ AM ”、“ PM ”表示。

d：该月日数，用两位表示，不足的在前边添零，如一号为“ 01 ”。

D：星期几的英文简称。

F：月份的英文全称，如一月用“ January ”表示。
h：用12小时制两位表示的小时数，如下午1点为“ 01 ”。
H：用24小时制两位表示的小时数，如上午1点为“ 01 ”，下午1点为“ 13 ”。
g：用12小时制按实际位数表示的小时数，如下午1点为“ 1 ”。
G：用24小时制按实际位数表示的小时数，如上午1点为“ 1 ”，下午1点为“ 13 ”。
i：用两位表示的分钟数。
j：该月日数，用实际位数表示，如1号为“ 1 ”。
l：星期几的英文全称。
L：是否是闰年，是闰年则替换为1，否则替换为0。
m：两位表示的月份数。
n：按实际位数表示的月份数。
M：月份的英文简称。
s：两位表示的秒数。
t：该月份的总天数。
U：从1970年1月1日零点起的秒数。
w：用数字表示的星期几。
Y：四位表示的年份数。
y：两位表示的年份数。
z：从当年1月1日算起的天数。

3. getdate函数

array getdate(int timestamp);

将一个标准的UNIX/Linux timestamp时间、日期值转化为一个以字符串为下标的数组，可从中提取时间、日期各分量值。各下标的意义如下：

seconds：秒数。
minutes：分钟数。
hours：小时数。
mday：日数。
mon：月份数。
year：年号。
yday：1月1日以来的天数。
weekday：星期几的英文全名。
month：月份的英文全名。

4. gmdate函数

string gmdate(string format, int timestamp);

gmdate函数与date函数类似，但先根据服务器所处的时区将时间转换成格林威治标准时以后再替换输出。

5. time函数

```
int time(void);
```

返回1970年1月1日零点以来的秒数。

8.7.3 PHP的数据库功能及对MySQL数据库访问

前面已经说过，PHP具有强大的数据库功能，可以支持各种平台下的多种数据库的访问，并且，这些数据库的支持都是内置在PHP3系统之中，不必程序员自己编写接口。

PHP3.0支持以下数据库：

Oracle	Adabas D	Sybase	FilePro	dBase
Velocis	Solid	Informix	mSQL	MySQL
Unix dbm	Postgre SQL	Empress	InterBase	ODBC

我们重点介绍PHP对MySQL数据库的访问。

MySQL是小型关系数据库mSQL的一个变种，性能上有较大的提高，主要表现在增加了用户访问控制，扩充了数据类型和加强了对SQL语句的支持。MySQL是通过SQL进行访问的，它支持除嵌套select语句(select...where select ...)以外的全部SQL92规范。同时，在数据量比较大的情况下，MySQL查询速度比mSQL、Red Hat Linux自带的PostgreSQL甚至大型关系数据库Oracle都要快。而且，MySQL和mSQL一样，可以免费得到和安装，但是如果将它作为商业用途，则需要支付一定费用。

下面介绍PHP3内置的MySQL函数，注意，对数据库和表单进行访问和操作的函数都需要一定的权限才能运行，因此编写PHP3代码的时候一定要注意容错性，也就是当访问失败时的处理。

1. 建立和关闭连接

```
int mysql_connect( s[thoisntgn a m[e:]p o r t ] , s t r[iunsge r n a m e ] , s t r i n g
[password] );
int mysql_close(int [link_identifier] );
```

mysql_connect函数建立一个到MySQL数据库服务器的连接，并返回连接号；如果在该服务器端口已经建立起一个连接的话，返回该连接的识别号，不建立新的连接；连接失败返回“0”。第一个参数MySQL数据库服务器主机名(hostname)可以是IP地址或域名地址，缺省状态为本机，还可以指定一个端口号(port)用冒号与主机名分隔；用户名(username)缺省状态下是当前用户的登录名，密码(password)缺省为空字符串(表示没有密码)。这个连接建立以后，会在脚本运行结束时自动终止，也可以调用函数mysql_close强行终止连接。mysql_close函数只有一个参数，即要终止的连接的识别号。下面是一组最简单的应用：

```
<?
...
if ((($nConnect=mysql_connect($host,$user,$password)!=0);
...
mysql_close($nConnect);
...
?>
```

2. 数据库新建、连接和删除

```
int mysql_create_db(string database_name,int[link_identifier] );
int mysql_select_db(string database_name, int [link_identifier] );
int mysql_drop_db(string database_name,int[link_identifier] );
```

mysql_create_db已连接的MySQL服务器上创建一个数据库；mysql_drop_db删除一个数据

库；mysql_select_db选择一个数据库作为当前活动数据库，所有的 mysql_query操作都将在当前活动数据库上进行。第一个字符串参数是数据库名称，第二个变量是连接号，一般由mysql_connect()函数在建立连接时返回。

3. SQL查询和查询结果释放

```
int mysql_query(string query,int[link_identifier] );
int mysql_db_query(string database, string query, int [link_identifier] );
int mysql_free_result(int result);
```

mysql_query()和mysql_db_query()都提供数据库的SQL语言查询功能，两者的差别在于前者使用mysql_select_db()函数选定的当前活动数据库，而后者先选定参数 database指定的数据库为当前活动数据库，再对它进行查询操作。 mysql_free_result()函数用来释放select语句的查询结果，参数是需要释放的查询结果 (query)的ID号。

对于update、insert、delete这样的SQL语句，函数返回 0表示操作失败，其余的数表示操作成功。对于select语句，成功时将返回一个结果识别号 (result identifier)，这个结果ID将惟一地标识当前查询结果 (即当前query)，以后对查询结果进行处理时，就通过这个 ID来确定操作的对象。

4. 查询结果统计

```
int mysql_num_rows(int result);
int mysql_num_fields(int result);
```

这两个函数用来返回一个查询结果 (query)中的字段数目和记录数目。参数是 query的ID，由mysql_query()或mysql_db_query()函数返回。

5. 查询结果的取得

```
int mysql_data_seek(int result, int row_number);
array mysql_fetch_row(int result);
array mysql_fetch_array(int result,int[result_type]);
object mysql_fetch_object(int result, int [result_type]);
int mysql_result(int result, int row, mixed field);
object mysql_fetch_field(int result, int [field_offset] );
```

用select语句对网络数据库查询的结果是一张本地的数据表单 (query)，对它进行操作，就可以取得相应的数据。上面列出的函数都是基于 query的操作，它们的第一个参数都是操作的query的识别(ID)号。

mysql_data_seek()函数将query指针定位于指定的记录，第二个参数是目标记录的行数。mysql_fetch_row()返回一个数组，包含了query当前行(当前记录)的全部数据，下标0的变量值是当前记录第一个字段的内容，下标 1的变量值是第二个字段的内容，依此类推；再次调用此函数，将返回下一条记录的内容，超过 query的末记录时，返回 0。mysql_fetch_array()函数和mysql_fetch_row()相似，但在功能上有了扩充，返回的数组除了用数字下标变量返回字段内容外，还可以用字段名字符串作为下标返回同样的内容。 mysql_fetch_object()也是用来取得query中的一条记录，但它的返回值是一个对象，对象的各个成员变量名就是 query的字段名。mysql_result()直接取得query中指定记录指定字段的数据，后两个参数分别是记录序号和字段名(或者字段序号)。最后请看下边的例子：

```
<?php
mysql_connect($host,$user,$password);
```

```
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_array($result))
{
  echo $row["user_id"];
  echo $row["fullname"];
}
mysql_free_result($result);
?>
```

其中query数据输出的也可以用另外两个函数， 程序可以这样写：

```
while($row = mysql_fetch_row($result))
{
  echo $row[0];
  echo $row[1];
}
```

或者：

```
while($row = mysql_fetch_object($result))
{
echo $row->user_id;
echo $row->fullname;
}
```

6. 字段属性

```
object mysql_fetch_field(int result, int [field_offset] );
```

从query中取得字段的属性，第一个参数为 query的ID，第二个参数为字段序号，缺省情况下取得下一个未取得属性的字段值。返回一个对象，成员变量有：

name、table、max_length、not_null、primary_key、unique_key、multiple_key、numeric、blob、type、unsigned、zerofill。