PHP 初學者實務手冊

Jollen Chen 著

仕橙3G教室研究中心

PHP 初學者實務手冊

Jollen Chen

This book is for sale at http://leanpub.com/php-beginner-practice

This version was published on 2013-10-07

Contents

1 PHP 基礎教材

1.1 踏出 PHP 的第一步

本章將帶領讀者進入 PHP 程式語言的世界。對不曾接觸過 PHP 的讀者而言，本章是您學習 PHP 的最佳教材。如果您早就熟悉 PHP，也可以瀏覽一下本章的內容，看看是否有學習上的遺漏。

開始學習前的準備工作：安裝 PHP + Apache + MySQL。這部份在網路上有非常多資源，請參考相關文件，在 Windows/Ubuntu/Mac 上安裝開發環境。

推薦的文章：

- Windows 使用者請參考 http://blog.roodo.com/esabear/archives/15069653.ht
- Mac 使用者請參考 http://www.mamp.info/en/index.html
- Ubuntu 使用者請參考 http://linuxg.net/install-lamp-on-ubuntu-13-04-raring-ringtail/

測試時，不會有瀏覽器的相容性問題，任何瀏覽器（Chrome/Firefox/Safari/Opera/IE）皆可使用。

1.2 如何開始撰寫 PHP 程式

PHP 是一種內嵌於 HTML 文件裡的程式語言，因此 PHP 的程式碼必須寫在 HTML 的檔案裡，為了能明辨出 PHP 的程式碼與 HTML，因此我們必須加上特製的標籤 (tag)。

當伺服器送出網頁時，會先直譯執行 PHP 的程式碼，而使用者端 (client) 所能看到的只有 ** PHP 的輸出 **，因此，我們

的輸出必須符合 HTML 語法的規範，那如何在網頁裡寫 PHP 呢？有 4 種格式：

第 1 種是正規的寫法。

```
<?php
   echo "Hello! World!";
?>
```

「」代表的是程式碼的結束，我們就將 PHP 的程式碼寫在這兩個標籤之間。另外，也有人將開始標籤裡的「php」省略，變成底下第 2 種寫法。

第 2 種是精簡的寫法。

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

程式碼的部份以包圍住。

第 3 種，當我們利用 FrontPage 編輯含有 PHP 程式的網頁時，就必須使用這種方式：

```
<script language="php">
   echo "Hello! World!";
</script>
```

這種寫法可以確保我們的 PHP 程式碼不會被 FrontPage 或其它網頁編輯工具重新排版，當我們想要保持原來的程式撰寫風格時，建議採用這種寫法。

第 4 種是 ASP 的寫作方式，利用 FrontPage 編輯含 PHP 程式碼的網頁時，也可以使用這種方式。

```
<%
   echo "Hello! World!";
%>
```

最後將 PHP 的程式檔案存成.php 的副檔名即可。

1.3 撰寫程式時的注意事項

使用第一種方法時，必須修改 php.ini 設定檔，設定 short_open_tag=On 才能使用；如果您以 source code 自行編譯來安裝 PHP，那麼在執行 configure 時，必須加上 [[-enable-short-tags]] 的參數。

php.ini 設定檔到底在那裡?

1. Linux 環境的使用者：php.ini 設定檔位於 php 安裝目錄下的 lib/php.ini，以第 1 章的安裝範例來講，就是 /usr/local/php/lib/php.ini。
2. MS-Windows 環境的使用者：利用 PHP 安裝程式安裝的話，我們可以在 windows\ 目錄下 (for Windows 95/98/Me) 或是 winnt\ (for Windows NT/2000/XP) 目錄下找到 php.ini 設定檔。手動安裝的話也應該將 php.ini 設定檔放到 windows 或是 winnt 目錄下。### 框 (說明)

當我們使用第 4 種方法時，必須修改 php.ini 設定檔, 將 asp_tags 設定成 On 才能使用這種方式。

撰寫 PHP 程式碼的方式又稱為「HTML 跳離」(HTML escape)。例如我們有一份 HTML 的檔案:

```
<html>

<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; cha\
rset=big5">
<title>PHP Example.</title>
</head>
<body BGCOLOR="#FFFFFF">
<?
echo "<p>Hello!</p><br>";
echo "<p>World...</p><br>";
?>
</body>
</html>
```

PHP 所看到的是自 HTML 跳離的程式碼：

```
echo "<p>Hello!</p><br>";
echo "<p>World...</p><br>";
```

接下來 PHP 瀏覽器只能看到最後的 output：

```
<html>

<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; cha\
rset=big5">
<title>PHP Example.</title>
</head>

<body BGCOLOR="#FFFFFF">

<p>Hello!</p><br>
<p>World...</p><br>
```

```
</body>
</html>
```

當我們利用瀏覽器瀏覽時，看到的畫面是：

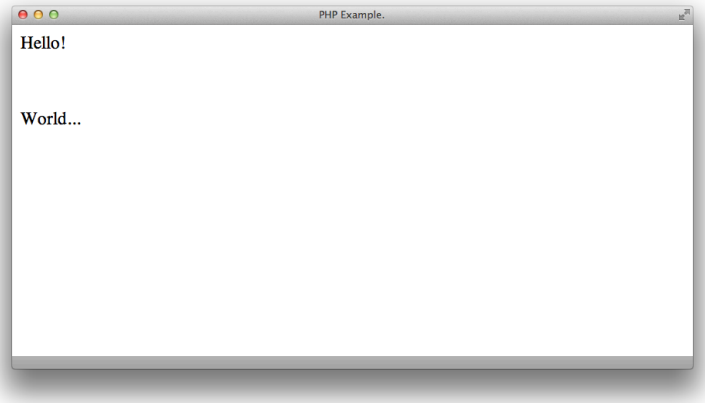

圖 1.1：程式範例 **example-1-1.php**

大多數的開發者，都是採用第 2 種形式；若是在 ASP 裡撰寫 PHP 程式時，請使用第 2 或第 4 種形式。

1.4 PHP 程式的副檔名

對於 PHP 文件而言，其慣用的副檔名為：

1. PHP/FI 2.0 副檔名為.phtml
2. PHP 3 的副檔名為.php3
3. PHP 4 的副檔名為.php
4. PHP 5 的副檔名為.php
5. 副檔名.phps 表示顯示 PHP 程式的原始碼

副檔名雖然可藉由修改 apache 的設定檔而自訂，但建議不要這麼做。因為本書是以 PHP 4/5 為主，所以副檔名一律使用.php，同時請確定 apache 的設定是否正確，否則無法正常處理 PHP 檔案。

1.5 PHP 敘述的結束

echo 是 PHP 的語法，用來做輸出。而整個 echo 語法的撰寫則是一行完整的敘述，敘述結束時必須以分號做結尾，例如底下的範例：

```
<?php
   echo "Trust me!<br>";
   echo "You can make it.<br>"
?>
```

另外，敘述是容許斷行的，請看底下的例子：

```
<?php
   echo "Trust me!<br>
You can make it.<br>"
?>
```

PHP 程式碼是以分號做為一行敘述的結束，但是最後一行敘述可省略分號，例如：

```
<?php
   echo "Trust me!<br>";
   echo "You can make it.<br>" //省略分號
?>
```

1.6 PHP 與 HTML 的換行方式

PHP 程式碼本文利用 Enter 做的換行動作並不等於在出現在瀏覽器畫面的換行，瀏覽器輸出畫面的換行必須使用 HTML 語法中的標籤，例如：

```
<?php
   echo "Trust me!<br>";
   echo "You can make it.<br>"
?>
```

在瀏覽器裡的輸出為：

```
Trust me!
You can make it.
```

如果寫成：

```
<?php
   echo "Trust me!";
   echo "You can make it."
?>
```

則在瀏覽器上看到的畫面就會變成：

```
Trust me!You can make it.
```

要記得，我們的輸出必須符合 HTML 的語法，才能被瀏覽器正常顯示。如果使用 '\n' 控制字元來換行的話，只有 HTML 的內文 (網頁原始碼) 才會被影響，對於 HTML 的輸出則沒有任何影響，例如：

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<?php
   echo "Trust me!<br>";
   echo "You can make it.<br>"
?>
</BODY>
</HTML>
```

輸出的結果雖然是:

```
Trust me!
You can make it.
```

在瀏覽網頁時，如果檢視網頁的原始碼，這份文件的內容其實是:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
Trust me!<br>You can make it.<br></BODY>
</HTML>
```

為了美化 HTML 的內文，我們在 PHP 裡加上 '\n' 控制字元，來做文件內文的換行:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<?php
   echo "Trust me!<br>\n";
   echo "You can make it.<br>\n"
?>
</BODY>
</HTML>
```

再檢視網頁原始碼就可以看到較為組織化的內容了：

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
Trust me!<br>
You can make it.<br>
</BODY>
</HTML>
```

1.7 資料型態與變數

PHP 所支援的資料型態 (data types) 有 8 種：

1. boolean
2. integer
3. floating (double)
4. string
5. array

6. object
7. resource
8. NULL

第 1~4 種型別稱為 scalar types ()，第 5~6 種稱為 compound types (複合型別)，最後第 7 與第 8 種則是特殊型別。

PHP 的變數屬鬆散資料型別，雖然支援 8 種不同的型態，但在初始化變數時不必宣告型態，而是在計算時動態 (dynamic) 決定，而非由撰寫程式的人決定。如果要强制設定變數的資料型別的話，可以利用 settype() 函數，或利用 C 語言的强迫轉型方式 (type casting)。

PHP 的鬆散資料型別，即我們給定什麼值，該變數即為什麼型別，或是如何使用該變數，該變數即為適當的型別，例如：

```
<?php
$foo = "0";  // $foo 為 string (ASCII 48)
$foo++;      // $foo 變成 string "1" (ASCII 49)
$foo += 1;   // $foo 變成 integer (2)
$foo = $foo + 1.3;  // $foo 變成 double (3.3)
$foo = 5 + "10 Little Piggies"; // $foo 為 integer (15)
$foo = 5 + "10 Small Pigs";     // $foo 為 integer (15)
?>
```

另外還有一點，PHP 的變數都是以 $ (dollar sign) 開頭，並且變數名稱有大小寫之分，例如：

```
$name
$Name
```

是兩個不同的變數。

當我們指定值 (value) 給變數時，事實上我們是指定 expression 最後的值給變數，例如：

```
$a = 5+5*2;
```

$a 的值為 5+5*2 最後的結果，即 15。

在 PHP 裡，除了給定值給變數外，還可以給定 reference 給變數。也就是，該變數為另外一個變數的 reference，reference 的意義很像是 “becomes an alias for” 或是 “points to”，給定 reference 的方法為，在原來的變數前加上 & (ampersand) 符號，再指定給另外一個變數。例如：

```
<?php
$foo = 5;               // $foo 為 integer 5
$num = &$foo;           // $num 為 $foo 的 reference
?>
```

上例中，&$foo 表示將 $foo 的 reference 指定給 $num 變數，當我們改變 $num 的值，等於改變 $foo 的值，也就是：

```
$num = 10;
echo $foo;                  // 輸出 10
```

使用參考時，來源必須是一個變數名稱，例如：

```
<?php
$foo = 10;
$bar = &$foo;                   // 正確的寫法
$bar = &(1+2*2);          // 錯誤的寫法!!!

function test() {
   return 0;
}

$bar = &test();     // 錯誤的寫法!!!
?>
```

1.8 型別轉換

PHP 裡要做强迫轉換變數型態的方法有 2 種：

1. 使用 C 語言的 type casting 語法，例如：

```
$x = 5;
$y = (double) $x;                    //括弧裡可以有 tab 或空白 (space)
```

可使用的 cast 型別有：

- (int)，(integer) – 轉換成 integer 型別
- (bool)，(boolean) – 轉換成 boolean 型別
- (float)，(double)，(real) – 轉換成 float 型別
- (string) - 轉換成 string 型別
- (array) - 轉換成 array 型別
- (object) - 轉換成 object 型別

type casting 型別轉換範例。

範例 1：

```
<?php
$x = "5";
$number = (int)$x;
echo $number; //輸出 5
?>
```

範例 2：

```php
<?php
$x = “foo” ;
$arr = (array)$x;
echo $arr[0]; //輸出 “foo”
?>
```

範例 3：

```php
<?php
$x = “foo” ;
$obj = (object)$x;
echo $obj->scalar; //輸出 “foo”
?>
```

1. 使用 settype 函數：

```
int settype(string var, string type);
```

將 var 變數轉換為 type 型別，可指定的型別參數有：

- ”boolean”(PHP 4.2.0 與之後的版本也可以簡寫成“bool”)
- “integer” (PHP 4.2.0 與之後的版本也可以簡寫成“int”)
- ”float”(PHP 4.2.0 與之後的版本才支援)
- “double”
- “string”
- “array”
- “object”
- ”null”(PHP 4.0.8 以後的版本才支援)

轉換成功傳回 true，否則傳回 false。當我們不知道某個變數的值是什麼型別時，也可以利用 gettype() 函數來取得。

在 PHP 裡和 C 語言一樣，非零即為 true，例如 if ($x) 等於 if ($x != 0)。

1.9 字串型態轉數值型態

當我們給變數的值是利用雙引號括住數值或字串時，就是指定一個字串給變數，例如：

```
$a = "Hello!";
```

$a 變數的值就是字串。請看底下的範例：

```
<?php
   $a = "hello!";
   echo $a;
?>
```

輸出結果為：

```
hello!
```

PHP 有一項特性，就是 PHP 的變數是在執行時才決定型態的，因此字串也可以用來做計算。PHP 將字串拿來做運算時，會依據底下 2 個原則設法將字串轉成可以計算的型態：

1. 字串中包括 “.” 、”e” 或 “E” 時轉換成 double 型別，否則轉換為 integer
2. 無法轉換時則為 0

之前曾見過這樣的寫法：

```
$foo = 5 + "10 Big Pigs";
```

PHP 會將字串 “10 Small Pigs” 先轉換成 integer 10，再做加法。字串轉數值的範例：

```php
<?php

$foo = 1 + "10.5";      // $foo 為 double (11.5)
$foo = 1 + "-1.3e3";    // $foo 為 double (-1299)
$foo = 1 + "bob-1.3e3"; // $foo 為 integer (1)
$foo = 1 + "bob3";      // $foo 為 integer (1)
$foo = 1 + "10 Small Pigs";     // $foo 為 integer (11)
$foo = 1 + "10 Little Piggies"; // $foo 為 integer (11);
$foo = "10.0 pigs " + 1;        // $foo 為 int (11)
$foo = "10.0 pigs " + 1.0;      // $foo 為 double (11)

?>
```

1.10 初始化變數

PHP 的變數都是以 $ 做為開頭，初始化時直接指定初值即可：

```php
<?php
$name = "Jollen"
$mail = "jollen@o3.net"
$age = 19
?>
```

1.11 初始化陣列

初始化變數時直接指定陣列與元素即可：

```php
<?php
$names[0] = "Jollen"
$names[1] = "Jordan"
$names[2] = "Kitty"
?>
```

PHP 還有另外一個自動設定元素的特異功能:

```php
<?php
$names[] = "Jollen"
$names[] = "Jordan"
$names[] = "Kitty"
?>
```

這個特異功能等於上面的初始化方法，陣列自動由第 0 個元素開始做配置。要注意的是，PHP 和 Perl/C 一樣，陣列的元素都是由 0 開始。其中 [] (中括弧) 不能省略，省略的話 PHP 會以為這個變數是一個 string。

1.12 Type Juggling - 型態間的戰爭

因為 PHP 並沒有精確的精型機制，而是當我們指定什麼樣型別的值給變數，該變數就是什麼樣的型別，例如:

```php
<?php
$a = 5;                 //$a 為 integer
$a = "5";         //$a 為 string
?>
```

變數在做運算時，例如使用 “+”，當 expression 包含各種不同的型態時，就會有 Type Juggling 的動作發生，例如:

```
<?php
$foo = "0";  // $foo 為 string "0" (ASCII 48)
$foo++;      // $foo 為 string "1" (ASCII 49)
$foo += 1;   // $foo 變成 integer (2)
$foo = $foo + 1.1;  // $foo 變成 double (3.1)
$foo = 5 + "15 Persons"; // $foo 的運算結果為 integer (20)
$foo = 5 + "10 Big Pigs";     // $foo 的運算結果為 integer (1\
5)
?>
```

又如：

```
$a = 5;                          // $a 的型別為 integer
$a[0] = "Hi!";         // $a 的型別變成 array
```

這種型別的改變即稱為 “Type Juggling”。

但是，在 PHP 裡，這種 type juggling 的機制還有一個值得討論的問題，就是當 type juggling 發生在字串轉換成陣列時，會發生底下這個問題：

```
$a = "Go";                 // $a 為 string
$a[0] = "N";         // $a 是否被轉型別 array?
```

聰明的讀者應該可以馬上看出，$a[0] 並不一定被轉型成 array，因為 $a[0] 也可能代表 $a 字串的第一個字元，也就是：

```
$a[0] = "G";
$a[1] = "o";
```

當 $a[0] = “N” 發生時，可能變成：

```
$a = "No";                    // $a 仍為 string
```

或是:

```
$a[0] = "N";          // $a 轉型成 array
```

過去舊版的 PHP（PHP 3）支援字串的字元索引的方法和陣列元素的索引方式一樣，這是在舊版 PHP 版本裡才需要特別提出的一個問題。舊版 PHP 的使用者如果要將 string 型別轉換成 array，可利用 type casting 的方式來解這樣的問題:

```
<?php
$a = "Go";
$a = (array)$a;
echo $a[0];           // 輸出 'Go'
$a[0] = "N";
echo $a[0];                   // 輸出 'N'
?>
```

type juggling 還有一個常被忽略的重點就是，++ 與 – 運算子所做的運算是 ASCII 與算術運算，而 + 運算子只做算術運算，看個例子就可以了解是什麼意思了:

```
<html>
<head>
<title>test</title>
</head>

<body>
<?php
   $foo = "01";
   echo "<p>\$foo  => $foo</p>";
   $foo++;
```

```
    echo "<p>\$foo++  => $foo</p>";

    $foo = $foo+1;
    echo "<p>\$foo = \$foo+1  => $foo</p>";
?>
</body>
</html>
```

其結果為：

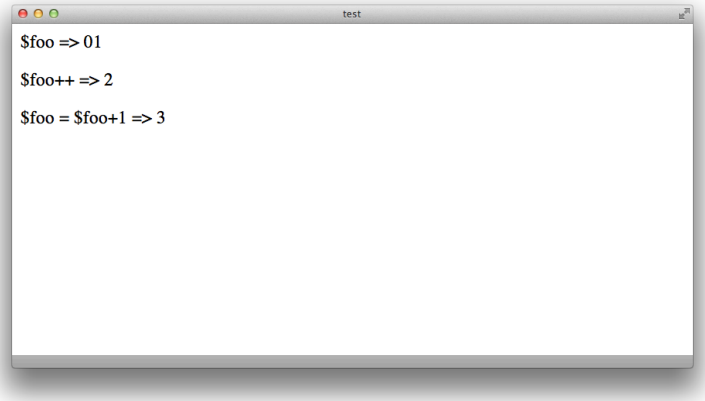

圖 1.2：程式範例 **example-1-2.php**

注意 $foo = $foo+1 的計算結果為 3，而不是 “03”，這就是筆者所要特別强調的地方，+ 運算子只做算術運算，這點在其它語言如 Perl 也有類似情況。

1.13 物件的生成

在 PHP 裡要生成物件和 C++ 一樣，首先必須先定義好 class，然後再利用關鍵字 new 來建立物件：

```
<?php
class person {
   function who() {
      echo "I'm Jollen.";
   }
}

$man = new person;
$man->who();                    // 輸出 I'm Jollen.

?>
```

PHP 5 新增了許多物件方面的功能。

1.14 區域變數

在 function 裡初始化的變數即區域變數。為什麼叫區域變數呢? 因為區域變數只有在 function 裡可以被「看見」，請看底下的範例:

```
<?php
function sum() {
   $a = 1;
   $b = 2;
   echo $a+$b;
}

sum();
?>
```

執行結果:

```
3
```

第 2 個輸出的結果為 0，這是因為 function 裡的 $a 只有在 function 裡才能被看到，在 function 外區域變數就不能被看見。

1.15 全域變數

在區域變數範圍之外所宣告的變數即全域變數，例如：

```
<?php
$a = 1;

function sum() {
   echo $a;
}

sum();
?>
```

這段程式碼執行後不會有任何輸出，因為 $a 變數是屬於區域變數。請注意 PHP 鬆散資料型別的變數使用前是不需要經過任何宣告的，包括 printf()、給定初值等。

因為區域變數的可見度會蓋掉全域變數，所以 sum() 所 echo 出的 $a 變數是一個區域變數，那該如何告訴 function 變數是一個全域變數呢？利用 global 關鍵字即可：

```
<?php
$a = 1;

function sum() {
   global $a;

   $a = $a*100;
}

sum();
echo $a;
?>
```

執行結果:

```
100
```

第一段程式碼其實存在了二個變數，一個是全域變數 $a，另一個則是區域變數 $a。在第二段程式碼裡，則只有一個全域變數 $a。

對於全域變數另外一個重點就是，倒底全域變數的範圍為何? 在 PHP 裡，全域變數也稱為 page-scoped 變數，亦即在同一個檔案裡的 PHP 程式都能看到這個全域變數。

1.16 $GLOBALS - 存放全域變數的陣列

PHP 會把全域變數存放在 $GLOBALS 的格式化陣列裡，格式化陣列不同於一般陣列是利用數字索引元素，格式化陣列則是利用字串來索引元素，例如:

```php
<?php
$name = "Jollen";
$mail = "jollen@jollen.org";

function man() {
   echo $GLOBALS["name"] . "<br>";
   echo $GLOBALS["mail"] . "<br>";
}

man();
?>
```

輸出:

```
Jollen
jollen@jollen.org
```

全域變數被存放至 $GLOBALS 陣列中，並利用變數的名字來索引。

1.17 靜態變數

有些程式語言 (例如 C) 具有一種稱做靜態變數 (static variables) 的型別，PHP 也支援靜態變數的寫法。

在 PHP 裡，只有區域變數才能、也才需要宣告成靜態變數，正常的區域變數生命期是在函數執行期間，隨函數的執行結束而結束，而靜態變數的生命期是隨整個 PHP 程式結束而結束，但可見度只有該函數。

我們可以利用關鍵字 static 來宣告靜態變數:

```php
<?php
function sum() {
   static $a = 1;

   if ($a < 10) {
      echo $a;
      $a++;
      sum();
   }
}

sum();
?>
```

輸出結果：

```
123456789
```

區域或全域變數都不是靜態變數，因為函數執行結束後，變數的值並不會被保留。而所謂的靜態變數意思就是說，當函數執行結束後，該變數的值仍然會被保留，因此第 2 次呼叫該函數時，靜態變數之前的值仍然存在。

1.18 以變數為名的變數

所謂「以變數為名的變數」(variable variables) 指的其實就是「動態變數名稱」(dynamic variable names)，直接來看一個例子讀者就可以了解什麼是 variable variables 了：

```php
<?php
$a = "Jollen";
$$a = "Pig!!!";
echo "$Jollen";                  // 輸出為 Pig!!!
echo "${$a}";                   // 輸出為 Pig!!!
echo "$a";                    // 輸出為 Jollen
?>
```

說的嚴謹一點，就是利用變數的值來做為其它變數的名稱，像上面這個例子 $$a 指的就是利用 $a 變數的值做為變數的名稱，所以說穿了上面這個宣告就等於是：

```php
$a = "Jollen";
$Jollen = "Pig!!!";
```

另外，在使用 variable varialbe 時要特別小心，例如在 echo 時：

```php
echo "$a ${$a}";
echo "$a $Jollen";
```

輸出結果一樣都是：

```
Jollen Pig!!!
```

兩者有異曲同工之妙。

1.19 變數的變數陣列

變數變數陣列 (variable variables with arrys) 跟 variable variables 其實是相同的東西，但使用變數變數陣列時，有些小地方必須要特別小心。例如底下是一個錯誤的寫法：

```
$$a[0] = "Kitty!";
```

這個語法錯誤的地方在於 PHP 不曉得 $a 和 $a[0] 那一個才是變數，也就是：

```
$a = "Good";
$a[0] = "Bad";

$$a[0] = "Kitty!";
```

兩者所代表的意思是很模糊的，PHP 分不清楚到底是：

```
$Good[0] = "Kitty!";
```

還是：

```
$Bad = "Kitty!";
```

才是程式設計師想要的。因此，正確的寫法應該使用一對大括弧來告訴 PHP 我們要的是那一種做法：

```
${$a[0]) 等於 $Bad = "Kitty!";
${$a}[1] 等於 $Good[1] = "Kitty!";
```

1.20 讀取 form 的資料

1. 全域變數讀取法

寫過 CGI 的朋友都知道，利用網頁的 form 可以將 client 端的資料傳送至伺服器端。利用 PHP 讀取 form 的資料是非常容易的，而且不像其它程式語言一樣還要判斷並分析這些資料，由 form 傳送過來的資料對 PHP 來講就是全域變數，例如：

```
<form action="reg.php" method="post">
   E-Mail: <input type="text" name="email"><br>
   <input type="submit">
</form>
```

PHP 會將 form 傳進來的資料，存放至由 HTML 標籤的 name 欄位所指定的變數裡。以上面的 form 為例，假設 form 傳進 $email=jollen@o3.net，在 PHP 裡就等於：

```
$email = "jollen@jollen.org";
```

這是一個全域變數，而且可以直接取用。傳送進來的資料也可以存成陣列，只要在 form 的 name 欄位做一點手腳即可，利用陣列可以更方便的處理同類型的資料：

```
<form action="interest.php" method="post">
   Name: <input type="text" name="person[name]"><br>
   Email: <input type="text" name="person[email]"><br>
   Your interest: <select multiple name="interest[]">
   <option value="Singing">Singing
   <option value="Sleeping">Sleeping
   <option value="Sport">Sport
   <option value="Reading">Reading
   <option value="other">other
   </select>

   <input type="submit">
</form>
```

在 PHP 裡就可以這樣讀取這些資料：

```
$person["name"];
$person["email"];
$interest[0];
$interest[1];
$interest[2];
$interest[3];
$interest[4];
```

如果您是使用 FrontPage 設計 form，輸入這類的名稱時 (含有 []) 會出現錯誤視窗，不過不用擔心，因為這樣的名稱是瀏覽器可以接受的。

請注意，利用這種方式讀取表單資料是較早期的做法，PHP 會自動將外來的變數註冊成全域變數，但從 PHP 4.2.0 開始，為了安全性與效能考量，預設將不再支援這種做法。

因此，以往的程式若要能在 PHP 4.2.0 與之後的版本也能正常執行，就必須修改 php.ini 設定檔，將底下的設定：

```
register_globals = off
```

改成：

```
register_globals = on
```

如此一來，才能使用全域變數讀取表單資料的使法。

1. track_vars 讀取法

PHP 的 track_vars 功能，經由 POST 到 Server 的表單資料會存放在 $HTTP_POST_VARS 陣列裡，經由 GET 的資料會存放在 $HTTP_GET_VARS 陣列裡。

使用 track_vars 前必須將此功能打開，方法有 2 種：

- 將 php.ini 的 track_vars 參數打開：

```
track_vars = On
```

- 在網頁裡加上

PHP 4.0.3 與以後的版本不再提供 track_vars 的設定項目 (php.ini)，以後 PHP 4 將永遠打開 track_vars 的功能。

1. $_POST 與 $_GET 讀取法

PHP 4.1.0 開始提供經由 $*POST* 與 *$_GET* 陣列存取表單資料的做法，而前面所介紹的 *$HTTP*_VARS 讀取方式是以往的寫法，但仍然可以使用。

例如名稱為 username 的表單資料，經由 POST 方式傳遞時，存取該表單資料時應寫成:

```
$_POST[“username”];    // 等於 $HTTP_POST_VARS[“username”\
];
```

改用 GET 方式傳遞時，則要改成:

```
$_GET[“username”];    // 等於 $HTTP_GET_VARS[“username”];
```

PHP 4.2.0 與之後的版本讀取系統的環境變數時，預設將不會自動將環境變數註冊為全域變數，因此請由 $HTTP_ POST (GET)*VARS* 或 *$* POST(GET) 陣列來讀取或是設定 php.ini 的 register_globals 項目 (On)。

PHP 5 只能經由 $ POST(GET) 陣列來讀取環境變數，以往設定 register_globals 項目與使用 $HTTP_POST(GET)_VARS 的做法將不再支援。

1.21 圖形的超鏈結

在圖形上設定鏈結時，瀏覽器會把滑鼠點選的座標位置傳給伺服器，並由 PHP 做轉換，存放至 var_x 與 var_y 兩個全域變數裡。例如底下的 HTML 語法:

```
<input type="image" src="image.gif" name="sub">
```

此時 (x, y) 的點選座標將會被存放到 (sub_x, sub_y) 裡。

1.22 HTTP Cookies

Cookies 是一種可在瀏覽器存放資料的機制，利用 setcookie() 函數設定 Cookies，因為 Cookies 屬於 HTML 的檔頭，所以 setcookie() 必須在有任何輸出之前呼叫。

Cookies 的內容會在網頁第一次被瀏覽時，由瀏覽器存放於 client 端中的一個檔案，當瀏覽器再次 request 該網頁時，再將 Cookies 傳回給伺服器。

關於版本間的差異，特別整理如下：

- PHP 4.1.0 與之後的版本改由 $_COOKIE 陣列讀取系統環境變數，但 $HTTP_COOKIE_VARS 的做法仍然可以使用。
- PHP 4.2.0 與之後的版本讀取系統的環境變數時，預設將不會自動將環境變數註冊為全域變數，因此請由 $HTTP_COOKIE_VARS 或 $_COOKIE 陣列來讀取或是設定 php.ini 的 register_globals 項目 (On)。
- PHP 5 只能經由 $COOKIE 陣列來讀取環境變數，以往設定 register_globals 項目與使用 $HTTP_COOKIE_VARS 的做法將不再支援。

在進階的教材裡，會再針對 HTTP Cookies 做進一步的說明。

1.23 讀取系統的環境變數

PHP 可以將系統的環境變數自動存成 PHP 的變數，例如：

```
echo $HOME
```

表示顯示系統中的 HOME 環境變數。有時 PHP 的變數會和系統中的環境變數重覆，為了確保我們讀取的是正確的系統環境變數，可以利用 getenv 來讀取環境變數，利用 putenv 存放環境變數。

在 UNIX 系統底下可利用 env 命令來查詢系統的環境變數：

```
linux# env
BASH=/bin/bash
.
.
.
linux# echo $HOME
/root
```

顯示系統的 HOME 環境變數，在 PHP 裡也是利用 $HOME 變數來存取環境變數的 HOME：

```
echo "HOME: ". $HOME . "<br>";
```

輸出結果：

```
HOME: /root
```

只要在環境變數名稱前加上 $ 符號即可。

另外一種讀取系統環境變數的做法則是透過 $HTTP_ENV_VARS 陣列，因此前面的寫法可以改成：

```
echo $HTTP_ENV_VARS[ “HOME” ];
```

關於版本間的差異，特別整理如下：

- PHP 4.1.0 與之後的版本改由 $_ENV 陣列讀取系統環境變數，但 $HTTP_ENV_VARS 的做法仍然可以使用。
- PHP 4.2.0 與之後的版本讀取系統的環境變數時，預設將不會自動將環境變數註冊為全域變數，因此請由 $HTTP_ENV_VARS 或 $_ENV 陣列來讀取或是設定 php.ini 的 register_globals 項目 (On)。
- PHP 5 只能經由 $ENV 陣列來讀取環境變數，以往設定 register_globals 項目與使用 $HTTP_ENV_VARS 的做法將不再支援。

1.24 常數的定義

PHP 有 2 個特別的常數：**FILE** 與 **LINE**，分別代表目前正在被直譯執行的檔案名稱與執行的行數，例如：

```
<?php

function report_error($file, $line, $message) {
    echo "An error occured in $file on line $line: $mes\
sage.";
}

report_error(__FILE__, __LINE__, "Something went wrong!\
");

?>
```

使用者自定常數可使用 define() 函數，這些常數定義後，就不能再被重新定義。例如我們要定義 PI 常數的值為 3.14159：

```php
<?php
define("PI", 3.14159);
echo PI;                          // 輸出為 3.14159
?>
```

要注意的是，常數也有大小寫之分。除了使用者自定的常數外，在 PHP 4 之後的版本裡，也有一些事先定義好的常數：

- **FILE**

目前正在被執行的檔案名稱。假如有一個檔案 (例如 test.php) 被 include 到另外一個檔案執行 (例如 index.php)，則回報回來的檔名為 test.php，即被 include 的檔案。當我們在為 include 許多檔案的 PHP 4 程式除錯時，**FILE** 常數就顯得特別好用。

- **LINE**

回報正被執行的行位置，如果是被 include 的檔案，則和 **FILE** 一樣，回報的是該被 include 的檔案正被執行的行位置。

- PHP_VERSION

PHP 的版本常數。

- PHP_OS

作業系統的名稱常數。

- TRUE

布林值的 true。

- FALSE

布林值的 false。

- E_ERROR

程式因錯誤而中斷執行。這裡的錯誤指的是無法忽略而且影響 PHP 程式繼續執行的錯誤，並非 PHP 語法上的錯誤。

- E_WARNING

PHP 執行時的小錯誤，這些錯誤屬警告訊息，並不影響程式的執行。

- E_PARSE

PHP 的語法有錯。

- E_NOTICE

發生不合法的狀況，但不影響程式的執行，例如計算的變數不存在時。