

多媒體網頁設計及互動程式設計文憑課程

Part II: Interactive Application Development (Lesson 21)

Content:

1. Introduction to PHP and server-side programming / Installation and Configuration of Apache 2 and PHP 5 on Windows platform
2. Basic PHP Syntax : Data Types, Variables, Arrays, Function and Loop
3. Introduction of Database Programming and MySQL / Installation and Configuration of MySQL 5 and related utilities
4. Implementing user authentication and authorization using cookie and session (Building a membership system)
5. SQL & Database Programming in PHP I (Corporate News with Content Management System)
6. SQL & Database Programming in PHP II (Search function, Interactive Enquiry Forms, Online Questionnaire and Hit Counter)
7. Online Product Catalogue with Content Management System
8. Shopping Cart and Payment Gateway Integration
9. Interactive Voting System and Interactive Bidding System
10. Online Forum Setup and Configuration using phpBB



Contents

2.1 HTML 簡介 4

什麼是 HTML.....	4
開始撰寫	4
文件命名	5
什麼是延伸檔名.....	5
利用瀏覽程式開啟 HTML 文件	6
HTML 標籤	7
標籤的格式	7
單身標籤	9

2.2 PHP 基本語法 10

將 PHP 嵌入 HTML 內	10
PHP 標籤的使用	11
PHP 標籤類別.....	12
SHORT STYLE	12
XML STYLE	12
SCRIPT STYLE.....	12
ASP STYLE.....	12
PHP 敘述 (PHP STATEMENT).....	13
PHP 的註解	13
變數.....	14
識別字 (IDENTIFIERS).....	14
變數的型別	15
常數.....	16
變數範圍	16
陣列.....	17
語法.....	17
PHP 超全域變數.....	22
\$GLOBALS	22
\$_SERVER.....	22

\$_GET	22
\$_POST	22
\$_COOKIE	22
\$_FILES	23
\$_ENV	23
\$_REQUEST	23
\$_SESSION	23
運算子	24
複合指派運算子	24
前後遞增與遞減還算子	25
參考	26
關係運算子	26
相等運算子	26
運算子參考	27
算術運算子：	27
設定運算子：	27
邏輯運算子：	28
流程控制	29
IF..ELSE 迴圈	29
DO..WHILE 迴圈	32
FOR 迴圈	34
SWITCH 迴圈	35
其它的流程控制	36
函式 (FUNCTION)	38
2.3 練習	40
2.4 資源	43

2.1 HTML 簡介

什麼是 HTML

HTML 不是一種電腦語言。HTML 是 Hypertext Markup Language 的縮寫。你必須先輸入文字，再加入印表機控制碼(或是 HTML 旗標)，必須等到列印之後(瀏覽時)才能看到放大、縮小或粗斜體等效果。Hyper 的意義是"在.....之上"、"非直線性的、跳躍的"。網際網路(WWW)帶給人類不同於以往的改變，步幅之大也是前人難以想像的。HTML 讓你在任何時間、任何地點都能享用網際網路的好處。Text 就是"文字"，翻譯時可以省略。它也告訴你 HTML 不是天書，只是簡單易懂的文件。Mark up 就是"標記"。它使用一些旗標來控制你的文件。Language 就是"語言"，翻譯為語法。它不是繞口難記的電腦語言，它只是簡單的白話英語。當然它也有文法規定，這才能放諸四海皆準。

開始撰寫

你必須利用文字處理程式來撰寫 HTML，如第一章內我們而安裝好的 Notepad++ 編輯器。當你完成 HTML 文件時，你需要一個瀏覽程式來檢查成果，瀏覽程式翻譯 HTML 文件成為網頁，例如 Microsoft 的 Internet Explorer。現在許多人進入學校或補習班學習 HTML，他們會告訴你去買一些 HTML 撰寫軟體，例如 Microsoft 的 Front Page。這類軟體的確會帶給你一些方便，但是它們卻增加學習的困難度，因為它們拿去了一半工作，很多時只需簡單的 drag & drop 動作就可完成一個 HTML 的編輯。我們建議必須使用純文字處理程式來撰寫 HTML，等你結業之後再作選擇是否要添購這類軟體。其實，許多有經驗的網頁工作者只使用 Notepad 而已。有一件事你必須牢記在心，HTML 必須是純文字文件，而且它的延伸檔名必須是 .html 或 .htm。

文件命名

文件命名是很重要的工作，你必須替你的 **HTML** 文件取一個簡單易記而不重覆的名字，任何名字都可以，不過最好是由英文字母(**A-Z, a-z**)及數字(**0-9**)組成。大部份的網路主機不支援中文檔名，而且將大小寫英文字母視為不同字母，所以千萬要注意。而且每一份 **HTML** 文件都需要延伸檔名。以下是命名的步驟：

1. 取一個名字，任何名字。例如：**test**
2. 加上延伸檔名 **.html** (或 **.htm**)。例如：**test.html** 或 **test.htm**

什麼是延伸檔名

延伸檔名可協助電腦辨別文件格式，**.html** 或 **.htm** 的作用是告訴電腦這是一份 **HTML** 文件。當你在處理圖像、音樂檔案或文書檔案時還會遇上其它延伸檔名，所以記得替你所有的網頁文件加上合適的延伸檔名。

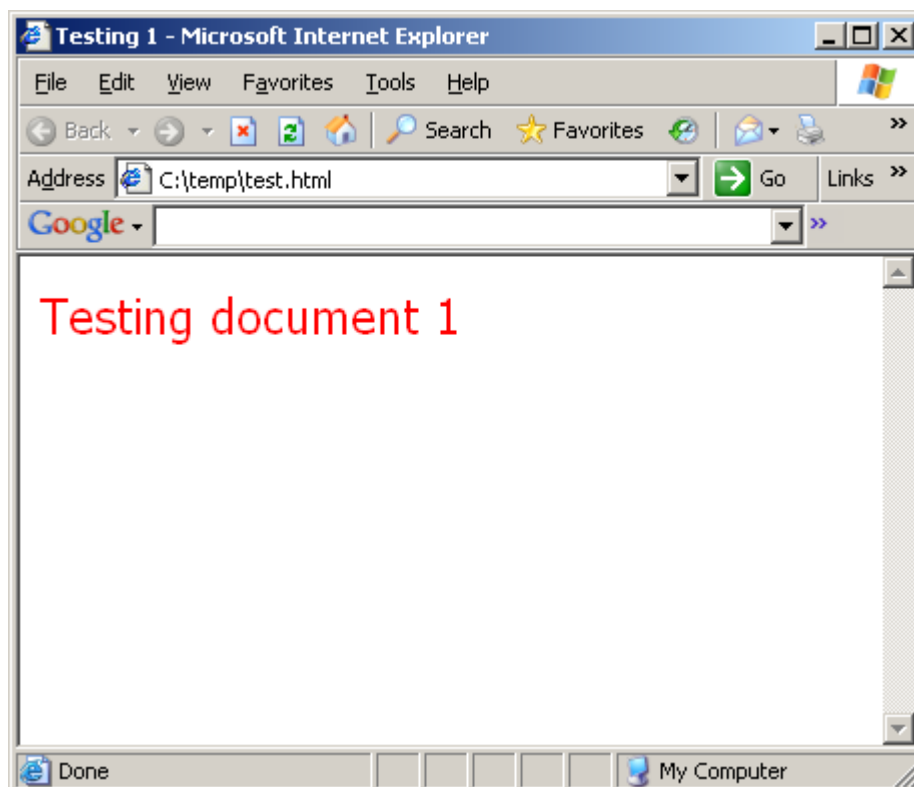
.html 和 **.htm** 有什麼不同，它們和蘋果電腦及視窗 **3.1/95/98/NT** 版本有什麼關係？蘋果電腦的 **MAC OS** 及 **PC** 的視窗 **3.1/95/98/NT/2000/XP** 版本的專業技術名詞都叫"作業系統"，作業系統管理電腦的各項作業。任一版本的 **MAC OS** 及 **Window 95/98/NT/2000/XP** 以上版本都可以支援 **3** 個以上英文字母的延伸檔名，但重前的 **Window 3.1** 版只能支援 **3** 個英文字母的延伸檔名。其實它們處理 **.html** 和 **.htm** 的方式是相同的。

利用瀏覽程式開啟 **HTML** 文件

當你將 **HTML** 文件儲存在軟碟片或硬碟中以後，你就可以利用瀏覽程式開啟它們。這裡有一個範例：

```
<html>
<head>
<title>Testing 1</title>
</head>
<body>
<font size=5 color=red>Testing document 1</font>
</body>
</html>
```

把檔案儲存成 **test.html** 並用瀏覽器打開：



HTML 標籤

HTML 是非常簡單、邏輯的格式，就好像聽你指揮往前、往後、向上或向下一樣容易。只要記住一點，HTML 是文字格式，它的主要目的也在控制文字編排。當你寫了長長一篇文章之後，你總不會將文章不分段的一口氣讀(列印)完吧！你將文章分段、每個段落之前加上標題，而標題文字使用較大字型、文章重點則加上底線或是改為粗體字型、註解文字使用較小字型。在 HTML 裡，這些功能皆透過一系列的標籤來完成。每一個標籤都代表一個指令，例如你想將文章中某一個字改為較大字型，你在那一字前面加上一個放大標籤，在那個字的後前加上一個回復標籤。一行字或整個段落也是如此控制。粗體、斜體字型都有相對應的標籤，控制方法也相同。

標籤的格式

所有的標籤 (指令) 都有一固定的格式，姑且稱為 HTML 文法。它們一定由一個"小於符號"開始，<，再由一個大於符號結束，>，絕對沒有例外。標籤多半是英文單字的縮寫，學習 HTML 只是要記得這些標籤而已。以下是一個將文字粗體化的例子：

一些主要的 HTML 標籤都是成對出現，它們也都很容易記憶。以下列出三個常用的文字控制標籤：

效果

控制碼

說明

範例

粗體(Bold)

B

粗體(Bold)

粗體(Bold)

斜體(Italic)

<I>斜體(Italic)</I>

斜體(Italic)

鉛字體(Typerwriter)

TT

<TT>鉛字體(Typerwriter)</TT>

鉛字體(Typerwriter)

是否能一次使用兩個或兩個以上控制標籤？是的。但要記得不管使用幾個控制標籤，不要忘了成雙成對的文法規定。例子：

<I>粗體和斜體</I>：粗體和斜體

<TT>粗體和鉛字體</TT>：粗體和鉛字體

當你使用兩個或兩個以上控制標籤時，務必注意啟動和回復的順序。看看接下來的說明：

<I><TT>Testing</TT></I>錯誤， 應在 </I> 之後。

注意！回復標籤的順序和啟動標籤的順序完全相反。

<I><TT>Testing</TT></I>：正確

單身標籤

現在介紹幾個常用的單身標籤，意思是它們不需要回復標籤。

標籤

說明

<HR>

HR 是 Horizontal Reference 的縮寫，它會在網頁中顯示一條水平的分隔線。

BR 是 Break 的縮寫，它的作用為另起一行，(跳行)。

<P>

P 是 Paragraph 的縮寫，它的作用為另起一段，(分段)。它的功能和
 有點兒類似，不過段落與段落之間的距離較大，意即 <P> 多跳一行。

雖然 <P> 是一個單身標籤，不過你仍可以加上回復標籤 </P>，協助你保持成雙成對的良好文法習慣。

每一張網頁都是由 <HTML> 標籤開始。顯而易見的，你必須宣告這是一份 HTML 文件。利用 <TITLE> 和 </TITLE> 標籤給你的網頁一個標題。看看命令列的上方，(如果你的瀏覽程式使用全螢幕，看看螢幕的頂端)，是否出現 <TITLE> 標籤的字樣，這就是利用 <TITLE> 和 </TITLE> 標籤所宣告的本頁標題，在啟動標籤和回復標籤之間的文字就會出現在命令列上方。到網頁結束時，一定是由 </HTML> 標籤結束。

2.2 PHP 基本語法

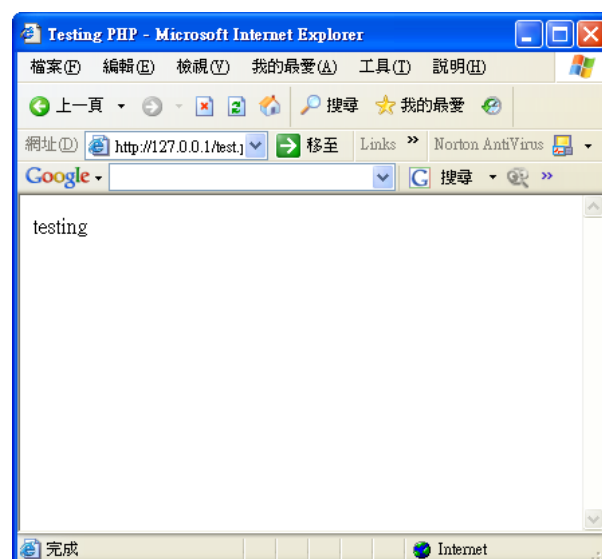
到目前為止，我們所用的都是單一的 HTML 文件，要加入動態內容，我們必須加入 PHP 的語句。以下將會介紹 PHP 的一些基本語法。

將 PHP 嵌入 HTML 內

將以下程式儲存成 test.php 檔在 C:\AppServ\www 文件夾中。：

```
<html>
<head>
<title>Testing PHP</title>
</head>
<body>
<? echo "testing"; ?>
</body>
</html>
```

打開瀏覽器並瀏覽至以下網址， <http://localhost:8080/test.php> ，我們將看到以下結果：



注意我們寫的 **PHP** 程式碼是如何嵌入平常的 **HTML** 檔案中。試著從您的瀏覽器中瀏覽這些原始碼，你會看到下列式碼：

```
<html>
<head>
<title>Testing PHP</title>
</head>
<body>
testing
</body>
</html>
```

我們看不到原始的 **PHP** 碼，這是因為 **PHP** 解讀器讀取程式，並由程式運作的結果取代原先的程式碼。這意味著我們可以利用 **PHP** 來製作任何瀏覽器都可以讀取的 **HTML**，換句話說，使用者的瀏覽器不需要懂 **PHP** 語言。這簡單說明了伺服器端程式語言的基本概念。不同於 **Javascript** 及其他客戶端程式語言，可以在使用者電腦的網路瀏覽器中解譯和執行；**PHP** 是在網路伺服器端解譯和執行的。

PHP 標籤的使用

前面範例中的 **PHP** 程式碼的開始符號是 `<?` 結束符號是 `?>`。這與所有的 **HTML** 標籤相類似，因為 **HTML** 的開始符號是小於 (`<`)，結束符號是大於 (`>`)。這些符號叫做 **PHP 標籤 (PHP Tags)**，是用來告訴網絡伺服器 **PHP** 程式從哪裡開始、到哪裡結束。標籤內的所有內容則會被當作 **PHP** 來作闡釋，任何標籤外的內容則會被當作一般的 **HTML**。**PHP** 標籤讓我們可以脫離 **HTML**。任何標籤形式都是可以接受的，我們這種用的是簡短形式 (**short style**)。如果你無法有效執行這段程式，這可能是因為你在安裝 **PHP** 時設定不接受短標籤。

PHP 標籤類別

事實上我們可以使用的 PHP 標籤有四種，下面每一組程式片段的意義都是相同的。

Short style

```
<? echo "testing"; ?>
```

這是 PHP 程式設計師用來寫 PHP 程式臨時的預設標籤形式。

XML style

```
<?php echo "testing";?>
```

這種形式的標籤可以用於 XML(Extensible Markup Language) 文件。如果你計畫要用 XML 撰寫你的網站，就必須使用這種標籤。

SCRIPT style

```
<SCRIPT LANGUAGE="php">echo "testing";</SCRIPT>
```

這種標籤是最長標籤，如果你使用 JavaScript 或 VBScript，你會對它非常熟悉。當你使用的是 HTML 編輯器而對其他形式的標籤都束手無策時，SCRIPT style 會很有用。

ASP style

```
<% echo "testing"; %>
```

這種標籤形式與 Active Server Pages (ASP) 用的相同，如果你有安裝 asp-tages 設定亦可使用。如果你使用的編輯器是 ASP，你可能會想用此種標籤。

PHP 敘述 (PHP statement)

我們利用開始及結束這兩個標籤之內的 **PHP 敘述 (PHP statement)** 來告訴 **PHP** 編輯器要做些什麼。在這個範例中我們只有用一種形式的敘述：

```
echo "testing";
```

你一定已經猜到了，用 **echo** 結構會產生很簡單的結果，它將字串呈現在瀏覽器上。你會看到在 **echo** 敘述結尾時會出現一個分號，這是用來分開 **PHP** 敘述的。如果你曾用過 **C** 或 **Java**，你會很熟悉分號的這種用法。結尾時遺漏分號是常見的語法錯誤，但同樣的也很容易發現並改正。

PHP 的註解

程式中的註解可以做為我們閱章程式時的筆記。註解可以用來解釋程式的目的、誰撰寫此程式、為何這支程式要這樣寫、最後修改的日期等等。你幾乎可以在所有的 **PHP** 程式中找到註解，除了那些很簡單的 **PHP** 程式。**PHP** 解譯器會忽略註解中的內容，基本上 **PHP** 會把註解視為空白。

PHP 支援 **C**, **C++**, **Shellscript** 形式的註解。這個 **C** 型別、多行的註解就有可能在我們 **PHP** 程式的開端出現：

```
/* Author: FevaWorks
```

```
Last modified: 1 June 2006 */
```

多行註解必須開始於 **/*** 並結束於 ***/**，多行註解不能是巢狀的。

你也可以在 **C++** 型別中使用單行註解：

```
echo "testing"; // commend
```

在這種型別中，所有註解符號 (**//**) 後的文字都是註解，一直到此行結束或是 **PHP** 標籤的結束符號出現為止，就看這兩項哪一個先出現而決定。

變數

我們在 `echo` 敘述中所連結起來的變數和字串是兩種不同的東西。變數是資料的代表符號，字串則是資料的本身。當我們需要運用原始資料的一部分於程式上時，我們會把它們稱為文字 (**literal**) 來與變數做區別。 `$UserName` 是一個變數，就是一個用來代替使用者輸入資料的代表符號。換句話說，`"UserName"`，就是一個文字，它會以表面的字元呈現。

識別字 (Identifiers)

識別字就是變數的名字。以下是識別字的一些簡單規則：

- 識別字的長度不限且可以由文字、數字、底線和金錢符號所組成。
- 識別字的第一個字不可以是一個數字。
- PHP 裡的識別字大小寫是有差異的，`$UserName` 與 `$username` 是不相同的，這是常見的文法錯誤，但 PHP 的內建函數則不在此限。
- 變數的識別字可以與內建函數取相同名稱，但是容易造成混淆，所以最好還是避免使用同一名稱。同樣的你也不能建立一個與內建函數相同名稱的函數。

變數的型別

變數的型別是依據儲存在變數中的資料種類而定。

PHP 支援下列幾種資料型別：

- 整數 (integer) - 使用於資料是整數時
- 浮點數 (double) - 使用於資料是實數時
- 字串 (string) - 使用於多字元的字串
- 陣列 (array) - 用來儲存相同類別的多種資料

例如：

```
$integer_var = 0;  
$double_var = 0.00;  
$string_var = "Testing";
```

我們可以利用型別轉換來假裝一個變數或值是不同型別。這項作業的方法與在 C 語言裡相同，只要在你想轉換的變數前面加一個小括弧，裡面放置你想要的暫時性型別即可。舉例來說，我們可以使用轉換 (cast) 來宣告前面提到的變數：

```
$integer_var = 0;  
$double_var = (double)$integer_var;
```

第二行的意思是「把儲存於 `$integer_var` 的值解釋為浮點數，並將它儲存至 `$double_var`」。變數 `$double_var` 會變成浮點數型別，被轉換的變數並不會改變型別，因此 `$integer_var` 的型別還是整數。

常數

如同我們前面所看到的，我們可以改變儲存於變數中的值。我們也可以宣告常數，常數可以儲存一個值，但是它的值一經設定後在此程式中就無法更改了。在我們的範例中，我們可以假設將每個販售的產品的價格設成常數，你可以利用 `define()` 函數來定義這些常數：

```
define("APPLEPRICE", 4);  
define("ORANGEPRICE", 3);  
define("BANANAPRICE", 7);
```

將這幾行加入你的程式中。你可以看到常數的名稱都用大寫字母，這是沿襲 C 語言的做法，讓我們可以一眼就可分辨出變數及常數。這個協定並不是必要的，但可以使你的程式更容易閱讀及率佳言畫。常數和變數最重要的不同點在於：提及常數時不需要在常數前面加上金錢符號。如果你要使用常數的數值，就直接使用它的名稱。舉例來說，如果我們要用先前創造的常數時，我們可以輸入：

```
echo ORANGEPRICE;
```

除了你所定義的常數外，PHP 設定了許多自己的常數。想要總覽這些常數有一個簡單的方法，就是執行 `phpinfo()` 指令。

```
<?phpinfo();?>
```

這樣會列出 PHP 先前定義的函數和常數，以及其他有用的資訊。

變數範圍

範圍 (scope) 是指一個特定變數在程式文本中可以看得到的範圍。PHP 中有三種基本的範圍類型如下：

- 程式中所宣告的廣場變數在整個程式文本中均可以看到，但是不包括函數在內。
- 函數中所使用的變數只歸屬於此函數。
- 函數內宣告成廣場域變數的變數指的是參考到同名的廣場變數。

陣列

PHP 中的陣列實際上是一個有序圖。圖是一種把 **values** 映射到 **keys** 的型別。此型別在很多方面做了優化，因此你可以把它當成真正的陣列來使用，或列表（矢量），散列表（是圖的一種實現），字典，集合，棧，隊列以及更多可能性。因為可以用另一個 PHP 陣列作為值，也可以很容易地模擬樹。

解釋這些結構超出了本手冊的範圍，但對於每種結構你至少會發現一個例子。要得到這些結構的更多訊息，我們建議你參考有關此廣闊主題的外部著作。

語法

定義 `array()`

可以用 `array()` 語言結構來新建一個 **array**。它接受一定數量用逗號分隔的 **key => value** 參數對。 **key** 可以是 **integer** 或者 **string**。如果鍵名是一個 **integer** 的標準表達方法，則被解釋為整數（例如 "8" 將被解釋為 8，而 "08" 將被解釋為 "08"）。如果省略鍵名，則取當前最大的整數索引，而新的鍵名將是該值加一。整數可以為負，所以對於負的索引也是這樣。例如最大的索引是 -6 將導致新的鍵名是 -5。如果還不存在整數索引，則鍵名將為 0（零）。如果你指定的鍵名已經有了值，則該值會被覆蓋。使用 **true** 作為鍵名將使 **integer 1** 成為鍵名。使用 **false** 作為鍵名將使 **integer 0** 成為鍵名。使用 **NULL** 作為鍵名將等同於使用空字串。使用空字串作為鍵名將新建（或覆蓋）一個用空字串作為鍵名的值，這和用空的方括號不一樣。不能用陣列和對象作為鍵名。這樣做會導致一個警告：
Illegal offset type。

```
array( [key =>] value
    , ...
    )
// key is either string or nonnegative integer
// value can be anything
```

可以通過明示地設定值來改變一個現有的陣列。這是通過在方括號內指定鍵名來給陣列設值實現的。也可以省略鍵名，在這種情況下給變數名加上一對空的方括號（“[]”）。 `$arr[key] = value;`

```
$arr[] = value;
// key is either string or nonnegative integer
// value can be anything
```

如果 `$arr` 還不存在，將會新建一個。這也是一種定義陣列的替換方法。要改變一個值，只要給它賦一個新值。如果要刪除一個鍵名／值對，要對它用 `unset()`。

註：`unset()` 函式允許取消一個陣列中的鍵名。要注意陣列將不會重建索引。

```
$a = array( 1 => 'one', 2 => 'two', 3 => 'three' );
unset( $a[2] );
/* 將產生一個陣列，定義為
   $a = array( 1=>'one', 3=>'three');
   而不是
   $a = array( 1 => 'one', 2 => 'three');
*/
```

`foreach` 控制結構是專門用於陣列的。它提供了一個簡單的方法來遍歷陣列。

例子 2.2.1 :

PHP 中的陣列型別有非常多的用途，因此這裡有一些例子展示陣列的完整威力。

```
// this
$a = array( 'color' => 'red'
           , 'taste' => 'sweet'
           , 'shape' => 'round'
           , 'name'  => 'apple'
           ,          4          // key will be 0
           );

// is completely equivalent with
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name'] = 'apple';
$a[]       = 4;          // key will be 0

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// will result in the array array( 0 => 'a' , 1 => 'b' , 2 => 'c' ),
// or simply array('a', 'b', 'c')
```

例子 2.2.2 使用 array() :

```
// Array as (property-)map
$map = array( 'version'    => 4
              , 'OS'       => 'Linux'
              , 'lang'     => 'english'
              , 'short_tags' => true
            );

// strictly numerical keys
$array = array( 7
               , 8
               , 0
               , 156
               , -10
            );

// this is the same as array( 0 => 7, 1 => 8, ...)

$switching = array(          10 // key = 0
                   , 5    => 6
                   , 3    => 7
                   , 'a'  => 4
                   ,          11 // key = 6 (maximum of integer-indices was 5)
                   , '8'  => 2 // key = 8 (integer!)
                   , '02' => 77 // key = '02'
                   , 0    => 12 // the value 10 will be overwritten by 12
            );

// empty array
$empty = array();
```

例子 2.2.3 集合：

```
$colors = array('red','blue','green','yellow');

foreach ( $colors as $color ) {
    echo "Do you like $color?\n";
}

/* output:
Do you like red?
Do you like blue?
Do you like green?
Do you like yellow?
*/
```

本例產生一個基於一的陣列。 **例子 2.2.4 基於一的陣列：**

```
$firstquarter = array(1 => 'January', 'February', 'March');
print_r($firstquarter);

/* output:
Array
(
    [1] => 'January'
    [2] => 'February'
    [3] => 'March'
)
*/
```

PHP 超全域變數

\$GLOBALS

包含一個引用指向每個當前腳本的全域範圍內有效的變數。該陣列的鍵標為全域變數的 名稱。從 PHP 3 開始存在 \$GLOBALS 陣列。

\$ _SERVER

變數由 Web 伺服器設定或者直接與當前腳本的執行環境相關聯。類似於舊陣列 \$HTTP_SERVER_VARS 陣列（依然有效，但反對使用）。

\$ _GET

經由 HTTP GET 方法送出至腳本的變數。類似於舊陣列 \$HTTP_GET_VARS 陣列（依然有效，但反對使用）。

\$ _POST

經由 HTTP POST 方法送出至腳本的變數。類似於舊陣列 \$HTTP_POST_VARS 陣列（依然有效，但反對使用）。

\$ _COOKIE

經由 HTTP Cookies 方法送出至腳本的變數。類似於舊陣列 \$HTTP_COOKIE_VARS 陣列（依然有效，但反對使用）。

\$ FILES

經由 HTTP POST 文件上傳而送出至腳本的變數。類似於舊陣列 `$HTTP_POST_FILES` 陣列（依然有效，但反對使用）。詳細訊息請參閱 POST 方法上傳。

\$ ENV

執行環境送出至腳本的變數。類似於舊陣列 `$HTTP_ENV_VARS` 陣列（依然有效，但反對使用）。

\$ REQUEST

經由任何用戶輸入機制送出至腳本的變數，因此該陣列並不值得信任。所有包含在該陣列中的變數的存在與否 以及變數的順序均按照 `php.ini` 中的 `variables_order` 配置指示來定義。該陣列沒有直接模擬 PHP 4.1.0 的早期版本。參照 `import_request_variables()`。

註：當執行於 命令行 模式時，這個陣列將 不會 包含 `argv` 和 `argc` 入口；它們已經存在於陣列 `$_SERVER` 中。

\$ SESSION

當前註冊給腳本會話的變數。類似於舊陣列 `$HTTP_SESSION_VARS` 陣列（依然有效，但反對使用）。詳細訊息，請參照 Session 處理函式 章節。

運算子

運算子可以用來處理數字、字串及其它需要比較運算的條件。
PHP 的運算子和 **C** 語言的運算子與很類似，對於有經驗的程式設計人員，應可以很順利的掌握 **PHP** 的運算子。

不同的運算子，其實還是有優先順序，就像小時候在學數學時，老師會教：先乘除、後加減。在 **PHP** 的運算優先順序可以參考下面的表格，在混合式的情形下，愈往下表示優先權愈高。至於符號代表的意義，散見於以下的章節中。

就像先哲說的：物有本末、事有終始，知所先後，則近道矣，在運算時只要照著運算優先順序寫出來的程式，應該不會發生結果和預期不同的情形。在寫作時多注意細節，可以減少除錯的痛苦。

複合指派運算子

除了簡單的指派之外，還有組複合指派運算子。每個運算子都用以對變數進行運算，再將結果指派回變數本身。例如：

```
$a += 5;
```

與下列寫法是樣的

```
$a = $a + 5;
```


前後遞增與遞減還算子

前後遞增 (++) 與遞減 (--) 運算子類似 += 與 -= 運算子，但是有些許不同。所有的遞增運算子都有兩種功能 - 遞增與指派。細想下列範例：

```
$a = 4;  
echo ++$a;
```

第二行用到前遞增運算子，之所以如此稱呼是因為 ++ 在 \$a 之前。所造成的效果是：第一，將 \$a 加 1；第二，傳回加 1 後的值。本例中 \$a 會遞增為 5 並傳回 5 且將 5 顯示出來。整個運算式的值為 5。（值得注意的是儲存於 \$a 的值已然改變，我們不是只傳回 \$a + 1 而已。）

如果 ++ 位於 \$a 之後，則所用的就是後遞增運算子，功能也不一樣，下述範例：

```
$a = 4;  
echo $a++;
```

這個例子裡的效果剛好相反，也就是首先傳回 \$a 的值並顯示出來，接著再將 \$a 遞增。整個運算式的值為 4。也就是會顯示出來的值。但是在執行該敘述之後的 \$a 值則變成 5。也許你會聯想到，-- 運算子的行為與之類似。其差別事實上只在 -- 會遞減而非遞增。

參考

在 PHP4 新增了一個參考運算子，**&(ampersend)**，用以與指派結合。通常將變數指派給另一變數時，會建立第一個變數的副本並將之儲存於記憶體中的某處。例如：

```
$a = 5;  
$b = $a;
```

這兩行程式會建立一個變數 **\$a** 的值的副本並將之儲存於變數 **\$b**。如果接下來我們改變 **\$a** 的值，**\$b** 並不會跟著改變。

關係運算子

關係運算子用以比較二值。根據比較的結果，使用這類運算子的運算式所傳回的結果不是邏輯上的 **true** (真) 就是 **false** (假)。

相等運算子

相等關係運算子，**==** (兩個等於符號) 讓你測試二值是否相等。例如我們可能用到這樣的運算式：

```
$a == $b;
```

來測試儲存於 **\$a** 與 **\$b** 的值是否相等。若相等會傳回 **true** 否則傳回 **false**。相等運算子很容易與指派運算子 **=** 混淆。雖然這樣不會造成執行上的錯誤，但通常不會傳回你想要的結果。

運算子參考

以下是一些常用運算子的參考。

算術運算子：

符號 意義	
+	加法運算
-	減法運算
*	乘法運算
/	除法運算
%	取餘數
++	累加
--	遞減

設定運算子：

符號 意義	
=	將右邊的值連到左邊
+=	將右邊的值加到左邊
-=	將右邊的值減到左邊
*=	將左邊的值乘以右邊
/=	將左邊的值除以右邊
%=	將左邊的值對右邊取餘數
.=	將右邊的字串加到左邊

邏輯運算子：

符號 意義

< 小於

> 大於

<= 小於或等於

>= 大於或等於

== 等於

!= 不等於

&& 而且 (And)

and 而且 (And)

|| 或者 (Or)

or 或者 (Or)

xor 互斥 (Xor)

! 不 (Not)

除了上述的運算符號之外，還有一些運算符號難以歸類：

符號 意義

\$ 變數

& 變數的指標 (加在變數前)

=> 陣列的元素值

? : 三元運算子

流程控制

由於 PHP 的大部份語法都是承襲了 C 語言的語法，因此，在流程控制方面，也是有著和 C 語言極類似的迴圈。PHP 的迴圈不像 ASP 般可以使用 goto 的 BASIC 語法，PHP 是結構化的程式語言，流程的設計上有一定的規定，而不能用 BASIC 的觀念來亂跳到別的區段中。

PHP 的語法沒有像 C 語言的 `main(){}` 區段，其實整個 PHP 主頁面（就是瀏覽器輸入的 URL）就是 `main(){}` 區段，這點和其它的解譯程式，如 Perl、Python、Shell Script 倒是很像。

在流程的區段分隔符號上，都是使用 `{` 當作區段的開頭，用 `}` 當作結尾，和 C 語言相同。不過 C 可以定義 `begin` 當開頭、`end` 當結尾（像 Pascal），而 PHP 中不能做這種特殊的定義。

而 PHP 語法中在每道指令結束時都要加上分號 `;`，但是在區段結尾符號 `}` 後面不用加上分號結束。

if..else 迴圈

if..else 迴圈有三種構造

第一種是只有用到 if 條件，當作單純的判斷。解釋成 "若發生了某事則怎樣處理"。語法如下：

```
if (expr) {  
    statement  
}
```

其中的 `expr` 為判斷的條件，通常都是用邏輯運算符號 (logical operators) 當判斷的條件。而 `statement` 為符合條件的執行區段程式，若程式只有一行，可以省略大括號 `{}`。

範例：本例省略大括號。

```
<?php
if (date("D") == "Sat") echo "周末了，狂歡去";
?>
```

範例：本例的執行區段有三行，不可省略大括號。

```
<?php
if (file_exists("/usr/local/lib/php3.ini")) {
    echo "以下是 PHP3 的設定檔<p><pre>\n";
    readfile("/usr/local/lib/php3.ini");
    echo "</pre>\n";
}
?>
```

第二種是除了 **if** 之外，加上了 **else** 的條件，可解釋成 "若發生了某事則怎樣處理，否則該如何解決"。語法如下

```
if (expr) {
    statement1
} else {
    statement2
}
```

範例：上面的例子來修改成更完整的處理。其中的 **else** 由於只有一行執行的指令，因此不用加上大括號。

```
<?php
$f="/usr/local/lib/php3.ini";
if (file_exists($f)) {
    echo "以下是 PHP3 的設定檔<p><pre>\n";
    readfile($f);
    echo "</pre>\n";
}
```

```
} else echo "很抱歉，找不到 $f";  
?>
```

第三種就是巢狀的 **if..else** 迴圈，通常用在多種決策判斷時。它將數個 **if..else** 拿來合併運用處理。直接看下面的例子

```
<?php  
if ($a > $b) {  
    echo "a 比 b 大";  
} elseif ($a == $b) {  
    echo "a 等於 b";  
} else {  
    echo "a 比 b 小";  
}  
?>
```

上例只用二層的 **if..else** 迴圈，用來比較 **a** 和 **b** 二個變數。實際要使用這種巢狀 **if..else** 迴圈時，請小心使用，因為太多層的迴圈容易使設計的邏輯出問題，或者少打了大括號等，都會造成程式出現莫名其妙的問題。

do..while 迴圈

do..while 是重複敘述的迴圈，可以分成二種模式。

最單純的就是只有 **while** 的迴圈。用來在指定的條件內，不斷地重複指定的動作。語法如下

```
while (expr) {  
    statement  
}
```

其中的 **expr** 為判斷的條件，通常都是用邏輯運算符號 (logical operators) 當判斷的條件。而 **statement** 為符合條件的執行區段程式，若程式只有一行，可以省略大括號 {}。

下例很有趣，要電腦的瀏覽器出現十次 "以後不敢了" 的字串，前面並加上數字，表示說了第幾次不敢了。(感覺好像是 Web Server 做錯事被處罰)

```
<?php  
$i = 1;  
while ($i <= 10) {  
    print $i++;  
    echo ". 以後不敢了<br>\n";  
}  
?>
```


while 可以不用大括號來包住執行區段，而使用冒號加上 **endwhile**。見下例：

```
<?php
$i = 1;
while ($i <= 10):
    print $i++;
    echo ". 以後不敢了<br>\n";
endwhile;
?>
```

另外一種 **do..while** 迴圈則先執行，再判斷是否要繼續執行，也就是說迴圈至少執行一次，有點像是先斬後奏的方法。這種的迴圈，和單用 **while** 是不同的（單用 **while** 是先判斷再處理）。若讀者熟悉 **Pascal** 語言的話，會發現 **do..while** 迴圈像是 **Pascal** 的 **repeat..until** 迴圈。**do..while** 的語法如下：

```
do {
    statement
} while (expr);
```

for 迴圈

for 迴圈就單純只有一種，沒有變化，它的語法如下

```
for (expr1; expr2; expr3) {  
    statement  
}
```

其中的 **expr1** 為條件的初始值。**expr2** 為判斷的條件，通常都是用邏輯運算符號 (**logical operators**) 當判斷的條件。**expr3** 為執行 **statement** 後要執行的部份，用來改變條件，供下次的迴圈判斷，如加一..等等。而 **statement** 為符合條件的執行區段程式，若程式只有一行，可以省略大括號 {}。

下例是用 **for** 迴圈寫的 "以後不敢了" 的例子，可以拿來和用 **while** 迴圈的比較。

```
<?php  
for ($i=1; $i<=10; $i++) {  
    echo "$i. 以後不敢了<br>\n";  
}  
?>
```

從上例中，可以很明顯的看到，用 **for** 和用 **while** 的不同。實際應用上，若迴圈有初始值，且都要累加(或累減)，則使用 **for** 迴圈比用 **while** 迴圈好。例如將資料從資料庫取出，可能用 **for** 迴圈會比用 **while** 迴圈適合？

switch 迴圈

switch 迴圈，通常處理複合式的條件判斷，每個子條件，都是 **case** 指令區段。在實作上若使用許多類似的 **if** 指令，可以將它綜合成 **switch** 迴圈。語法如下：

```
switch (expr) {  
    case expr1:  
        statement1;  
        break;  
    case expr2:  
        statement2;  
        break;  
    :  
    :  
    default:  
        statementN;  
        break;  
}
```

其中的 **expr** 條件，通常為變數名稱。而 **case** 後的 **exprN**，通常表示變數值。冒號後則為符合該條件要執行的區段。注意要用 **break** 跳離迴圈。

```
<?php  
switch (date("D")) {  
    case "Mon":  
        echo "今天星期一，猴子穿新衣";  
        break;  
    case "Tue":  
        echo "今天星期二，猴子肚子餓";
```

```
        break;
    case "Wed":
        echo "今天星期三，猴子去爬山";
        break;
    case "Thu":
        echo "今天星期四，猴子看電視";
        break;
    case "Fri":
        echo "今天星期五，猴子去跳舞";
        break;
    default:
        echo "今天放假，不管猴子了";
        break;
}
?>
```

很明顯的，上述的例子用 `if` 迴圈就很麻煩了。當然在設計時，要將出現機率最大的條件放在最前面，最少出現的條件放在最後面，可以增加程式的執行效率。上例由於每天出現的機率相同，所以不用注意條件的順序。

其它的流程控制

除了上面的流程控制指令之外，尚有 `break` 及 `continue` 二個流程控制指令。

`break` 用來跳出目前執行的迴圈，如下例：

```
<?php
$i = 0;
while ($i < 10) {
```

```
if ($arr[$i] == "stop") {  
    break;  
}  
$i++;  
}  
?>
```

continue 即刻停止目前執行迴圈，並回到迴圈的條件判斷處，見下例：

```
<?php  
while (list($key,$value) = each($arr)) {  
    if ($key % 2) { // 略過偶數  
        continue;  
    }  
    do_something_odd ($value);  
}  
?>
```

而 BASIC 常用的 **goto** 在 C 及 Borland Pascal 中或許可以使用。但在 PHP 中，由於它的 Web Server Script 特性以及結構化的組成，並不能在 PHP 中使用 **goto** 迴圈指令。

函式 (Function)

在 PHP 中，允許程式設計者將常用的流程或者變數等元件，組織成一個固定的格式。也就是說使用者可以自行組合函式或者是物件。PHP 中的函式 (function) 和 C 語言一樣，包括有傳回值的及無傳回值的，不像 Pascal 分成函式 (function) 和程序 (procedure) 那麼複雜。

在函式的名稱上，PHP 對於大小寫的管制很鬆散。可以在定義函式時寫成大寫的名字，而在使用時使用小寫的名字。總之，對函式而言，不用管大小寫，只要注意名稱沒有重複就好了。

以下就是函式的使用語法：

```
function myfunc($arg_1, $arg_2, ..., $arg_n) {  
    // 執行一些動作  
    return $retval;  
}
```

在使用時，在自定的函式名稱前要加入 **function** 的保留字，表示這是定義使用者自定函式。之後的 **myfunc** 可以是任何的英文字母開頭的字串，字串除了開頭不能是數字或是底線，在第一個字母後可以是阿拉伯數字或者是底線，當然其它的符號或是中文字不能當函式名。

\$arg_1 到 **\$arg_n** 為函式使用的參數，參數之間使用逗號隔開。在參數後的大括號 **{}**，即為整個函式的區段。函式若有傳回值，使用 **return** 可將值傳回。而參數可以事先定義初始值或內定值。有定義內定值的參數在使用函式時可以省略，但一定要放在沒有設定內定值參數的後面，否則 PHP 在解析函式時，會出現錯誤。

另外就是參數的形態，只要參數是 **PHP** 支援的變數型態都可以使用，無論是陣列、字串、或是整數....等等。傳回值也是一樣。下面即為使用內定值及不用內定值的例子：

```
<?php
function myfunc1($arg_1, $arg_2, $arg_3="我是內定字串") {
    echo $arg_1+$arg_2;
    echo $arg_3."<p>\n";
}

myfunc(3, 4);                // 參數 $arg_3 省略。
myfunc(6, 6, "不用內定值") // 輸入參數 $arg_3。
?>
```

參數的值，通常使用傳值的方式輸入，有時在特別的需求時，可以使用傳址的方式，傳入參數的指標。方法就是在參數的前面加上 **&** 符號即可。如下例：

```
<?php
function myfunc2(&$argstr) {
    $argstr=ereg_replace("/", "-", $argstr);
}

$today="2000/01/01";
myfunc2($today);
echo $today;    // 2000-01-01;
?>
```

2.3 練習

使用 **HTTP Post** 方法遞交數據

1. 把以下程式儲存成 `sample.html` 在 `C:\AppServ\www` 文件夾中。

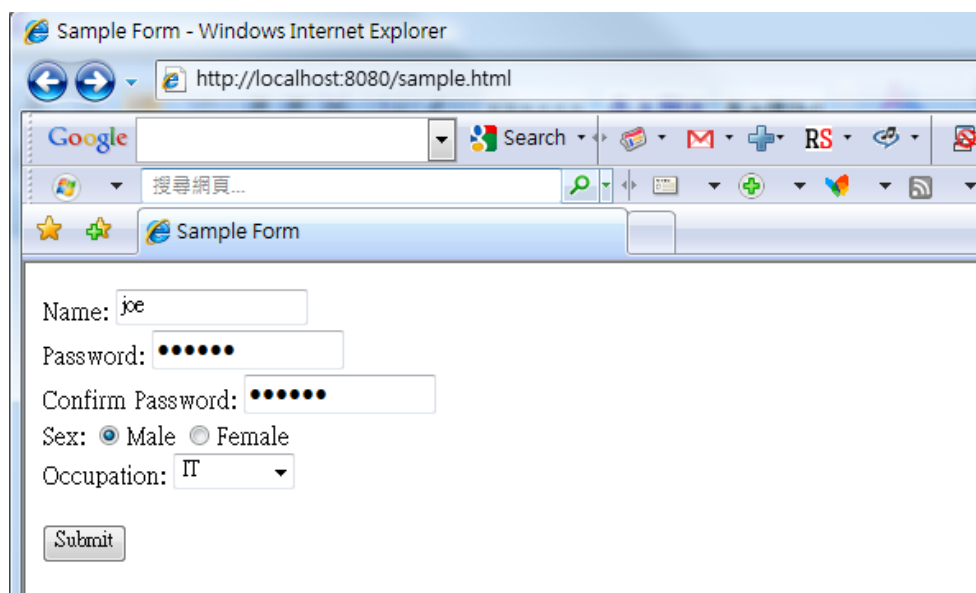
```
<html>
<head>
<title>Sample Form</title>
</head>
<body>
<form action="sample.php" method=post>
Name: <input type="text" name="name" size=15><br>
Password: <input type="password" name="ppassword" size=15><br>
Confirm Password: <input type="password" name="cpassword" size=15><br>
Sex: <input type="radio" value="m" checked name="sex">Male
<input type="radio" name="sex" value="f">Female<br>
Occupation: <select size="1" name="occupation">
<option>IT</option>
<option>Education</option>
</select><br>
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```


2. 把以下程式儲存成 `sample.php` 在 `C:\AppServ\www` 文件夾中。

```
<?php
$name = $_POST["name"];
$password = $_POST["password"];
$cpassword = $_POST["cpassword"];
$sex = $_POST["sex"];
$occupation = $_POST["occupation"];

if ($name == "") {
    echo "Please input your name!";
}
else if ($password == "") {
    echo "Please input password!";
}
else if ($password != $cpassword) {
    echo "Confirm password is not equal password!";
}
else {
    if ($sex == "m")
        echo "Welcome, Mr. ".$name;
    else
        echo "Welcome, Miss. ".$name;
    echo "<br>Your occupation is ".$occupation;
}
?>
```

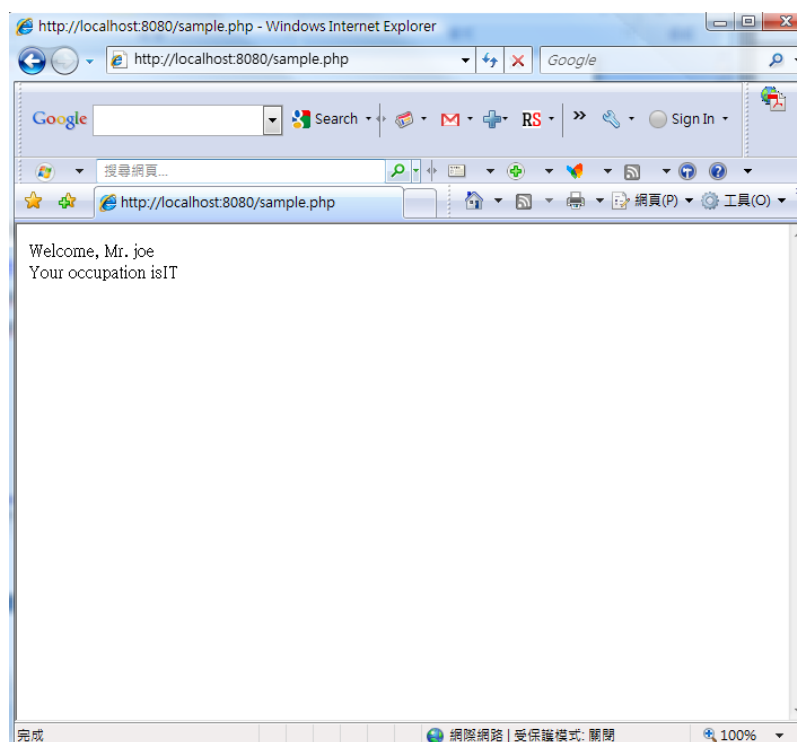
3. 打開瀏覽器並瀏覽至 <http://localhost:8080/sample.html>



The screenshot shows a Windows Internet Explorer browser window titled "Sample Form - Windows Internet Explorer". The address bar displays <http://localhost:8080/sample.html>. The page content includes a form with the following fields and controls:

- Name:
- Password:
- Confirm Password:
- Sex: ☒ Male ☐ Female
- Occupation:
- Submit:

4. 輸入 *Name*、*Password*、*Confirm Password*、*Sex*、*Occupation* 欄位並按 "Submit" 按鈕



The screenshot shows the same Windows Internet Explorer browser window, but the address bar now displays <http://localhost:8080/sample.php>. The page content displays the following text:

Welcome, Mr. joe
Your occupation isIT

2.4 資源

1. 打開瀏覽器並瀏覽至

<http://www1.fevaworks.com/course/cefwebeng/>

Username: cefwebeng

Password: joelam

The screenshot shows a Windows Internet Explorer browser window. The address bar displays the URL <http://www1.fevaworks.com/course/cefwebeng/>. The page content includes the FEVAWORKS logo, the title "CEF Web Engineering Course Material", and a login form. The login form has fields for "Username:" and "Password:", and buttons for "Login Now" and "Reset". Below the form, a text line reads "Please call 3106 8211 if you don't know the username & password". At the bottom, there is a banner for "THOMSON PROMETRIC now at Fevaworks".

CEF Web Engineering Course Material - Windows Internet Explorer

<http://www1.fevaworks.com/course/cefwebeng/>

Google

Google

Search

CEC Web Engineering Course Material

FEVAWORKS

CEF Web Engineering Course Material

Login

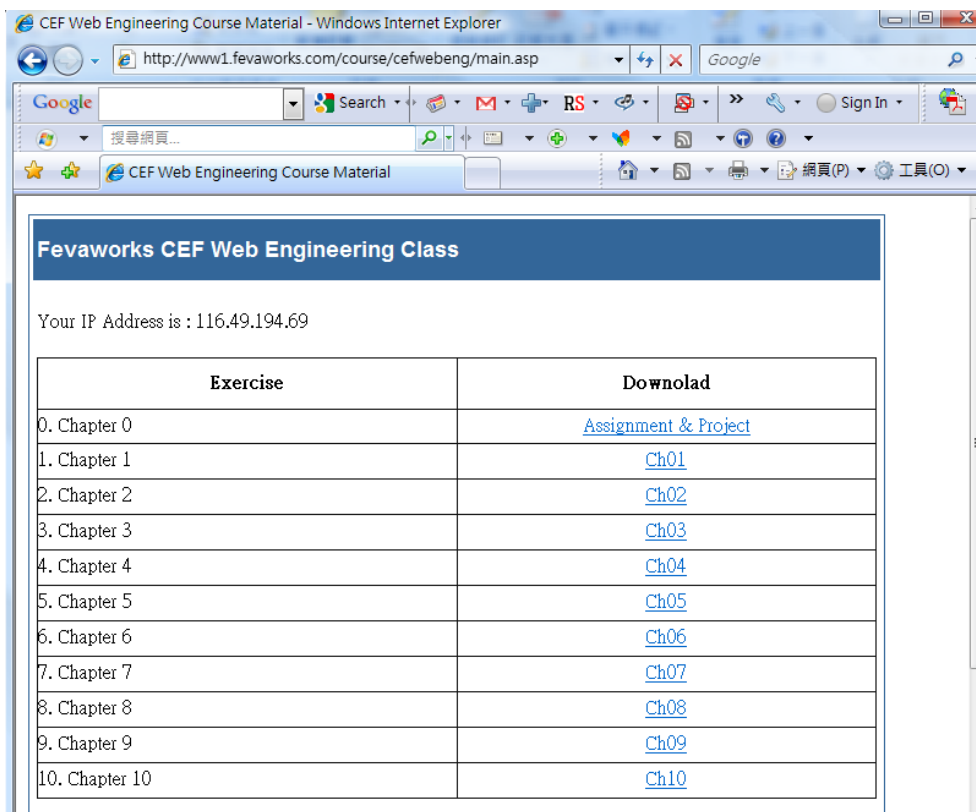
Username:

Password:

Please call 3106 8211 if you don't know the username & password

THOMSON PROMETRIC now at Fevaworks

2. 下載 Chapter 2 “Ch02”



CEF Web Engineering Course Material - Windows Internet Explorer

http://www1.fevaworks.com/course/cefwebeng/main.asp

Google

Google

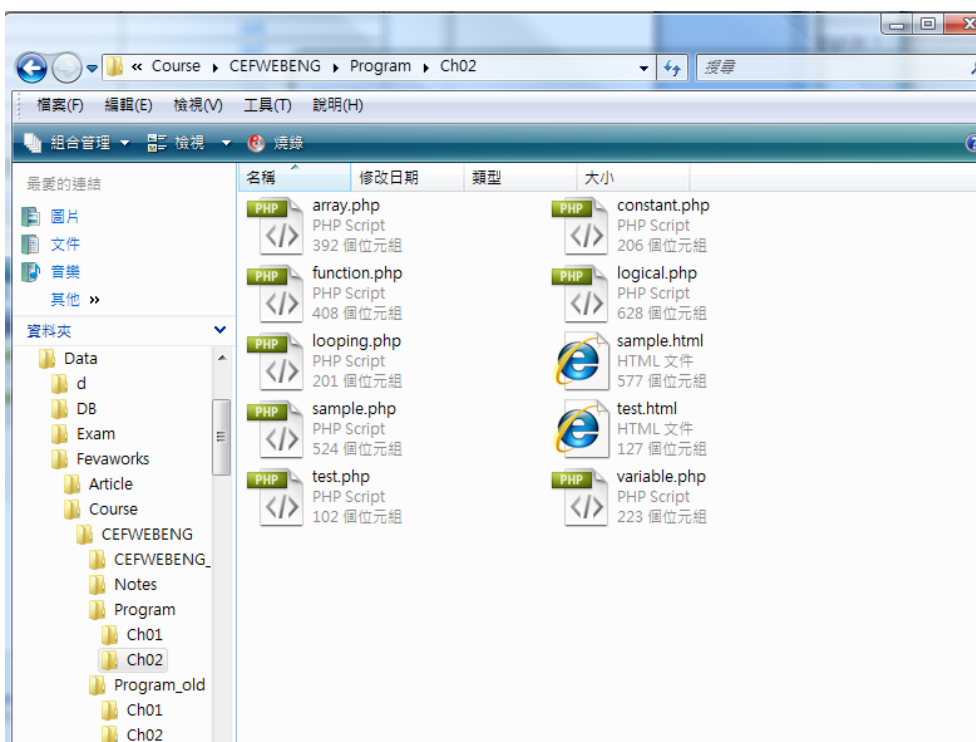
CEF Web Engineering Course Material

Fevaworks CEF Web Engineering Class

Your IP Address is : 116.49.194.69

Exercise	Download
0. Chapter 0	Assignment & Project
1. Chapter 1	Ch01
2. Chapter 2	Ch02
3. Chapter 3	Ch03
4. Chapter 4	Ch04
5. Chapter 5	Ch05
6. Chapter 6	Ch06
7. Chapter 7	Ch07
8. Chapter 8	Ch08
9. Chapter 9	Ch09
10. Chapter 10	Ch10

3. 解壓縮并且打開文件夾



Course > CEPWEBENG > Program > Ch02

檔案(F) 編輯(E) 檢視(V) 工具(T) 說明(H)

組合管理 檢視 燒錄

名稱	修改日期	類型	大小
array.php		PHP Script	392 個位元組
function.php		PHP Script	408 個位元組
looping.php		PHP Script	201 個位元組
sample.php		PHP Script	524 個位元組
test.php		PHP Script	102 個位元組
constant.php		PHP Script	206 個位元組
logical.php		PHP Script	628 個位元組
sample.html		HTML 文件	577 個位元組
test.html		HTML 文件	127 個位元組
variable.php		PHP Script	223 個位元組

資料夾

- Data
- d
- DB
- Exam
- Fevaworks
- Article
- Course
- CEPWEBENG
- CEPWEBENG_
- Notes
- Program
- Ch01
- Ch02
- Program_old
- Ch01
- Ch02