台北市教育網路中心

PHP & MySQL Introduction

Yung-Zen Lai

yzlai@hotmail.com

2004/Jan

Table of Contents

PHP，即代表"PHP: Hypertext Preprocessor"，為一種廣泛使用的開放原始碼、一般用途之腳本語言，尤其適合於網站開發而且能內嵌於 HTML 中。它的語法接近 C，Java，和 Perl，而且容易學習。此語言的主要目標是讓網站開發者能快速寫出動態生成的網頁，但 PHP 有更多的功能。

MySQL 是一個真正的多用戶、多執行緒 SQL 資料庫伺服軟體。SQL（結構化查詢語言）是世界上最流行的和標準化的資料庫語言。MySQL 是以一個主從架構的實現，它由一個伺服器程式 mysqld 和很多不同的用戶執行緒組成。

MySQL 主要目標是快速、健壯和易用。

PHP 與 MySQL 組合起來，成為一個在教育機構及小型企業除了 Microsoft ASP(.Net) + MS-SQL 架構外，最快，也最方便能實現網頁動態與資料庫化的另一 Solution。

Chapter 1. PHP

1.1 Basic

1.1.1 Escaping from HTML – 從 HTML 中跳脫

範例程式 1-1 index.php

```
<html>
<body>
<h1>這是 HTML 本文</h1>
<?
    echo “<h2>Hello World!  現在時間爲”;
    echo date (‘Y/m/d H:i:s’);
    print “</h2>\n”;
?>
</body>
</html>
```

請留意上述程式碼和其他語言如 Perl 或 C 所編寫的CGI程式之間的分別 – 與其使用大量的指令來輸出 HTML，我們寫了一個內嵌了代碼的 HTML 腳本來做某些事（以上述例子來說，輸出一行 H2 標題文字）。PHP 代碼包含在特殊的開始和結束標記，讓您可以隨時進出「PHP 模式」。

和 Java Script 之類的用戶端語言不同的是，PHP 的代碼是在伺服器端執行。如果您的伺服器中有一個和上述例子中類似的腳本，用戶看到的將只是執行後的輸出，而無法知道該腳本的代碼內容。

當 PHP 解譯器在解析一個檔案時，一般只是將檔案內容讀取並顯示出來。但若遇到了一些特定的標記時，它才會開始進行它的解譯。它目前支援四種從 HTML 本文中跳脫的格式，不過一般常見到的只有兩種，分別爲 <? // PHP Code … ?> 及 <?php # PHP Code … ?> 。（還有一種簡潔的方式 <?= expression; ?> ，只有在很短的敘述中使用）

它也可以支援 ASP 的<% %> 標籤，不過這需要特別的設定才行。

1.1.2 Dealing with Forms – 存取表單變數

範例程式 1-2　　postform.html

```
<form action="action.php" method="POST">
  Your name: <input type="text" name="name" />
  Your age: <input type="text" name="age" />
  <input type="submit">
</form>
```

上面是個很普通的 HTML Form，如何可以在我們的 PHP 程式中，取出使用者輸入的 name 及 age 值呢…

範例程式 1-3　　action.php

```
Hi <?php echo $_POST["name"]; ?>.
You are <?php echo $_POST["age"]; ?> years old.
```

發現了嗎？在 method 爲 POST 時，只要利用 $_POST 這個關聯式陣列即可取出使用者所輸入的值。同樣地，在 method 爲 GET 時，也是利用 $_GET 關聯式陣列來取得參數。

範例程式 1-4　　getform.html

```
<form action="action.php" method="GET">
  Your name: <input type="text" name="name" />
  Your age: <input type="text" name="age" />
  <input type="submit">
</form>
```

在較舊版 PHP 預設定中，其實可以直接使用 $name, $age 系統 Global 變數來取得 form 所傳遞過來的數值。不過此方法若是沒有特別的注意的話，會造成一些安全上的疑慮（例如 form 的變數名稱與我們所使用的變數同名，那麼一些變數的起始值就會爲 form 所傳遞過來的數值。），所以在 PHP 4.1 之後，預設定將這些表單數值定爲非 Global 變數。

（當然也可以修改設定將這些表單值設爲 Global 變數，但強烈建議不要這麼做，除非是爲了相容於舊有且難以維護的系統。）

1.1.3 Types – 資料型態

PHP 支援八種簡單的資料型態，其中含四種 scalar(boolean, integer, float, string)，兩種 compound(array, object)以及兩種 special types(resource, NULL)。

範例程式 1-5　　types.php

```
<?
$bool = TRUE;   // a Boolean
$str  = "foo";   // a string
$int  = 12;     // an integer

echo gettype($bool); // prints out "boolean"
echo gettype($str);  // prints out "string"

// If this is an integer, increment it by four
if (is_int($int)) {
   $int += 4;
}

// If $bool is a string, print it out
// (does not print out anything)
if (is_string($bool)) {
   echo "String: $bool";
}
?>
```

由於 PHP 是一種弱型別(loosely-typed)語言，所以我們可以隨時的將變數的型別轉換(cast)。不過對於程式寫作算是兩面刃，一方面很方便，另一方面就得在字串及數值的處理上小心一點。我們也可以使用 settype(), gettype() 來設定或取得變數的型態。

範例程式 1-6　　type_juggling.php

```
<?
$foo = "0";  // $foo is string (ASCII 48)
$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
?>
```

1.1.4 Variables – 變數

PHP 的特色之一就是在使用變數前並不需要宣告，當第一次指派值給這個變數時變數會自然產生。而 PHP 裡變數的取名方式跟一般程式語言都相似，就不在這部份贅述。這邊要注意的就是 PHP 裡有許多系統預先設定好的變數以及變數範圍（scope）。

系統預設定好的變數大都是存在關聯式陣列中，例如 $_SERVER, $_ENV, … 等。而跟使用者輸入有關的變數陣列則有 $_REQUEST, $_POST, $_GET, $_COOKIE, $_SESSION, $_FILES。

$_SERVER 儲存的是一些關於 Web Server, headers, paths, and scripts 的資訊

$_ENV 內是關於此 PHP parser 所執行環境的環境變數。

$_REQUEST 則是使用者這個 HTTP 要求的一些相關資訊。

$_POST 則是儲存以 POST 方法所傳遞過來的表單數值。

$_GET 儲存著以 GET 方法所傳遞過來的表單數值。

$_COOKIE 則是有著 HTTP cookies 的值。

$_SESSION 關於此次 HTTP session。

$_FILES 則與使用者上傳檔案時的一些資訊有關。

例如 $_SERVER ['REMOTE_ADDR'] 即爲伺服器這邊所看到之遠端使用者連線的 IP Address。

PHP 中有四種基本的變數生存範圍：

1. 內建的全域變數，在整個程式中都可以隨時讀取。
2. 程式開頭所宣告的整體變數，適用於整個程式，不過並非內建函式。
3. 僅應用於部份副程式、函數之內的區域變數。
4. 函數內宣告成廣域變數的變數，指的是參考到同名的廣域變數。

上面所提到的系統預設變數皆屬於第 1 種內建的全域（Global）變數，即在整個程式中都可以隨時讀取。

PHP 裡還有一個非常強大的變數，它叫做「變數的變數」（Variable variables）。變數的變數可以讓我們動態地改變變數的名稱。要做到改變變數的名稱的方法很簡單，就是利用一個變數的值來當另一個變數的名稱。

範例程式 1-7　　variable_variables.php

```
<?
$var_name = “students”;
$$var_name = 35;                // ${$var_name} = 35; 亦可。
// 以上結果相同於 $students = 35;
?>
```

1.1.5 Constants – 常數

一般程式語言，除了可以宣告變數之外，也可以使用一些不能在執行時被更動的常數值來使用，PHP 當然也不例外。我們可以利用 define() 函數來設定常數。

範例程式 1-8 constants.php

```
<?
define (“PI”, 3.1415926);
define (“LOGIN_ATTEMPT”, 3);
?>
```

上面即爲兩個常數的宣告，一般程式寫做習慣常數都會以全大寫來與變數做爲區別。而常數之範圍皆爲全域（global）。

1.1.6 Expressions – 敘述式

敘述式是一個程式語言中最重要的組成部份，所有的程式執行，都是由一個個的敘述式所組合起來的。

類似 C, Perl, 以及 Java，PHP 也是使用「;」來分別一個個敘述式。就像前面例子中的 echo date(‘Y/m/d H:i:s’); 或 $bool = TRUE; 或 define(“PI”, 3.1415926)。這些每個都是一個敘述式。

1.1.7 Operators – 運算子

運算子是一種可以用來將值與變數加以運算的符號，它又分成許多類，例如數學運算子、字串運算子、關係運算子、邏輯運算子…等。以下就略爲介紹一些常用的運算子。

數學運算子：

運算子	名稱	範例
+	加	$a + $b
-	減	$a - $b
*	乘	$a * $b
/	除	$a / $b
%	取餘數	$a % $b

字串運算子：

```
$a = “This is string A”;
$b = “This is string B”;
$result = $a . $b;
```

複合指派運算子：

```
$a++;
$a--;
$a += 5;
$a -= 3;
$a *= 2;
$a /= 10;
```

關係運算子：

運算子	名稱	用法
==	相等	$a == $b
===	全等	$a === $b
!=	不等	$a != $b
<>	不等	$a <> $b
<	小於	$a < $b
>	大於	$a > $b
<=	小於等於	$a <= $b
>=	大於等於	$a >= $b

邏輯運算子：

運算子	名稱	用法	結果
!	NOT	!$b	若$b 爲 false 或 0 則傳回 true
&&	AND	$a && $b	$a 與 $b 皆爲 true 則傳回 true
||	OR	$a || $b	$a 或 $b 爲 true 則傳回 true
and	AND	$a and $b	與&&同
or	OR	$a or $b	與||同

位元運算子：

&	位元 AND	$a & $b	取$a 與$b 做 AND 運算的結果
|	位元 OR	$a | $b	取$a 與$b 做 OR 運算的結果
~	位元 NOT	~$b	取$a 做 NOT 運算後的結果
^	位元 XOR	$a ^ $b	取$a 與$b 做 XOR 運算的結果
<<	左移	$a << $b	將$a 左移$b 個位元
>>	右移	$a >> $b	將$a 右移$b 個位元

執行運算子：

`command`	執行 command 指令

1.1.8 Control Structures – 控制架構

控制架構是一種允許我們控制程式或命令執行流程的語言中的架構，我可以可以將它們群集爲條件（或分支）架構，還有重複架構或迴圈。接下來就來看看一些常見的控制架構。

如果我們希望依一些變數的值來決定下一步該執行哪些程式碼時，這類型就稱之爲 conditionals。常見的如 if, else, elseif, switch 等。

if, else, elseif 條件式敘述

```
if ( expr1 )
    statement1
elseif ( expr2 )
    statement2
else
    statement3
```

範例程式 1-9 control_structure_if.php

```
<?
if ( $a > $b )
    print “a is bigger than b”;

if ( $a > $b ) {
    // $a 大於 $b
    print “a is bigger than b”;
} else
    print “a is NOT bigger than b”;

if ( $a > $b ) {
    // $a 大於 $b
    print “a is bigger than b”;
} elseif ( $a == $b ) {
    // $a 等於 $b
    print “a is equal to b”;
} else {
    // $a 小於 $b
    print “a is smaller than b”;
}
?>
```

switch 條件式敘述

範例程式 1-10　　control_structure_switch.php

```
<?
switch ( $char )
{
case ‘a’:
      echo “Char is a”;
      break;
case ‘b’:
      echo “Char is b”;
      break;
default:
      echo “Char is unknown”;
      break;
}
?>
```

電腦最拿手的事情之一就是重複性動作的自動化。如果有件事是需要以同樣方式執行好幾次，就可以使用迴圈來完成這部份的工作。常見的有 for, while 迴圈等。

for 迴圈

```
for (expression1; condition; expression2 )
      expression3
```

範例程式 1-11　　control_structure_for.php

```
<?
$score_sum = 0;
for ( $i = 0; $i < $students; $i++ )
      $score_sum += $score[$i];
?>
```

while 迴圈

```
while ( condition )
      expression;
```

範例程式 1-12　　control_structure_while.php

```
<?
$i = 0;
$score_sum = 0;
While ( $i < $students ) {
    $score_sum += $score[$i];
    $i++;
}
?>
```

跳脫或者跳向下一次的迴圈控制結構與 C 語言一樣，都是使用 break 與 continue 兩個指令。

1.1.9 Functions – 函數

在大多數的程式語言中都存有函數，函數是用來將單獨且定義明確的工作之程式碼分離出來。這使得程式變得易於閱讀與維護，且在我們能在這行相同的工作時，可重複地使用這段程式碼。

函數是一個自我獨立的程式模組。它規定了其呼叫界面、完成一些工作且選擇性的傳回結果。在此之前，我們已經見過一些函數了。前幾小節我們呼叫過 date(), settype(), gettype(), …等簡單的函數，只是忽略了其細節。在這小節，我們將會更詳盡地介紹如何呼叫函數和撰寫函數。

下列是一個最簡單的函數呼叫：

```
function_name ();
```

上式呼叫了一個 funcion_name 的函數，且不需要任何的參數，此外，它忽略了任何函數可能會傳回的值。

有些函數就是用以上的方式來呼叫，例如 phpinfo()。但大部份的函數確實需要一個或更多的參數，而參數就是呼叫函數時，傳遞給函數並且會影響其執行結果的資訊。

```
function_name(“parameter”);
```

在上式中，我們所用的參數字串只有 parameter。而依程式的不同，以下的函數呼叫也是正確的：

```
function_name (35);
function_name ( $students);
function_name ( $students, $classes);
```

最後兩列的$students, $classes 可以是包含陣列在內的任何一種 PHP 變數。

參數可以是任何型態的變數，可是特定的函數通常需要特定的資料型態。使用函數的原型（prototype），我們可以知道函數需要多少個參數，每個參數所代表的意義及其所需的資料型態。以下爲 fopen()函數的原型：

```
int fopen ( string filename, string mode, [int use_include_path]);
```

原型會告訴我們一些事，其中重要的是它可以讓我們知道如何正確地使用。

PHP 內建函數讓我們可以使用檔案及資料庫，或是建立圖形程連結其他的伺服器。然而，在我們的工作上可能常常會需要做一些其他程式語言設計師所沒有想到，或是不需要做的事，這時候就可能會需要「自訂函數」來協助我們更快、更容易的完成所需要的程式。

範例程式 1-13　　my_func1.php

```
<?
function    my_function ()
{
    echo “my function was called”;
}
my_function();
?>
```

上式就是一個最簡單的自訂函數，它沒有傳遞任何參數，也只是印出一行文字在我們的視窗上，並沒有做什麼特別的事情。

爲了完成一些工作，大部份的函數會需要一個或者是更多的參數，以下是一個需要參數的函數：

範例程式 1-14　　my_func2.php

```
<?
function larger ( $a, $b)
{
    if ( $a > $b )
        return $a;
    return $b;
}
echo larger ( 5, 15);
?>
```

上式是一個傳回 $a 與 $b 值中較大值的函數，並將較大值指派給 $c 變數。

有些時候，函數中運算所需要的運算值也包含著一些可能在主程式中所用到的變數，這時候就又跟變數的生存範圍（scope）扯上關係了。

在前幾節說過如 $_POST, $_GET, …等系統預設變數爲全域變數，這部份沒有生存範圍的問題。但其他使用者的自訂變數，如果只是依一般的參數傳遞方式傳給函數去使用，那麼此變數在函數裡若有更動的話，對原來程式裡此變數的值是**沒有** 影響的。因爲一般的傳遞方式只是傳遞變數的值（pass by value）到函數裡另一個同名的變數罷了，也就是程式執行到函數裡時，會有兩個同爲 $var_name 的變數，一個爲原來程式碼中的，另一個爲函數中的。而函數中所使用 $var_name 之祇始值爲原先原來程式碼中所傳遞進來的。所有在函數中所使用到的，都是函數中的 $var_name，更動它的值，並不會影響到原來程式中的$var_name 之值。

若我們一定要將原來程式中的$var_name 值更動的話，其中一個解決方式是在函數中宣告$var_name 爲全域變數。但一般並不建議這麼使用，因爲會破壞程式的獨立性。另一種方法即是使用傳遞參照（pass by reference）來傳遞參數值。此方法會將原來變數的參照（記憶體位址）傳遞給函數裡的同名變數，使兩個變數都同樣的指向同一塊記憶體位址。一般只要在變數前加上「&」即可，如下所示：

```
function function_pass_by_reference( &$var_name );
```

1.1.10 Classes and Objects – 類別與物件

現代的程式語言通常支援甚至還需要以物件導向的方法來協助軟體開發，PHP 當然也少不了這一塊。

範例程式 1-15　　class_and_object.php

```
<?
class classname
{
    var $attribute1;
    var $attribute2;

    function classname ( $param1, $param2)
    {
        $this->attribute1 = $param1;
        $this->attribute2 = $param2;
    }

    function sum ()
    {
        return $this->attribute1 + $this->attribute2;
    }
```

```
    function larger ()
    {
        if ( $this->attribute1 > $this->attribute2 ) {
            return $this->attribute1;
        }
        return $this->attribute2;
    }
}

$a = new classname (3, 5);
print $a->sum() . "\n";
?>
```

上式就是一個簡單的類別定義與實體化。這部份的程式寫作概念就先略過不提，待有興趣的人自行發掘。

1.1.11 Code Reuse –重複使用程式碼

重複使用程式碼可以使得程式變得更一致、更可靠且更容易維護。前面提到的函數其實就是重複使用程式碼的一種，現在我們來看看另一種的重複使用程式碼。

PHP 所提供兩種既簡單又有用的敘述 — require() 與 include()，能使我們重複使用任何型式的程式碼。使用這兩個內建函數可以載入檔案到 PHP 程式中。而這個檔可以包含任何的資料，通常會是 PHP 敘述、文字、HTML 標籤、PHP 函數或是 PHP 類別。這些敘述類似於 C 語言中所提供的 #include 敘述，現在就來看看它們的用法。

範例程式 1-16 require_exam.php

```
<?
require ('class_and_object.php');
print $a->larger();
?>
```

以上就是一個將我們之前在物件部份的程式碼所包含進來的小程式。當我們執行 require('class_and_object.php'); 時，PHP parser 將會自動載入所要求的 class_and_object.php 檔案。並且執行該檔案中的程式。也就是說，當我們載入 require_exam.php 時，它會執行像是以下的程式碼：

```
class classname
{
    var $attribute1;
    var $attribute2;

    function classname ( $param1, $param2)
    {
        $this->attribute1 = $param1;
        $this->attribute2 = $param2;
    }
…
…
}

$a = new classname (3, 5);
print $a->sum() . “\n”;
print $a->sum() . “\n”;
```

當使用到 require() 時，必須注意到不同的延伸檔名和 PHP 標籤的之處理方法。由於一般非 .php 的延伸檔名儲存在網站的目錄下時，使用者便可以直接載入並讀取程式碼，所以一般建議延伸檔名取名爲 .inc.php 或是放置於網站目錄外。

include()與 require()敘述是非常相似的，但在運作上仍有些顯著的不同。include()敘述是每當其敘述被執行時才會載入，如果不是在執行狀態便不會去載入，然而 require()敘述則是第一次被解析時便會被執行，不管是否含有程式區塊。

範例程式 1-17 include_exam.php

```
<?
$a = $_GET[‘a’];
if ($a > 10) {
    include (“a_is_greater_than_ten.php”);
} else {
    include (“a_is_less_or_equal_to_ten.php”);
}
?>
```

在上式中，a_is_greater_than_ten.php 與 a_is_less_or_equal_to_ten.php 只會有一個被載入和使用。若是用 require() 的話，則是兩個檔案皆會被載入，但只有其中一個的程式碼被使用。

除非伺服器非常的忙碌，否則以上兩種方法間的差異非常些微。可是這並不表示在條件敘述句中使用 require() 敘述是比較沒有效率的。

如果分不清楚 include() 與 require() 的差異的話，就一直使用 require() 吧! 它會比 include() 快上一些些，不用再動態的載入所需要的檔案。

1.2 Arrays – 陣列

1.2.1 What is an array? – 何謂陣列

簡單的說，陣列就是一群相同資料型態變數的群集。

一般變數通常爲一個統量變數，純量變數是一個被命名的位置，這個位置用來儲存一個數值；同樣地，陣列也是一個被命名的位置，而儲存的是一組相同資料型態的數值。因此陣列可以讓我們將普通純量群組起來。

將一群變數轉換成一組陣列後，我們就可以方便地利用它來作很多事。運用上一章的迴圈概念，我們可以利用對陣列裡的每一個數值做同樣的動作來節省工夫。如此一來，只須少少數行程式碼，便可以把一個班級的所有學生各科分數平均、標準差算出來，也可以利用 PHP 裡陣列的特殊排序函數把排名給輕易解決。

1.2.2 Indexed Array – 數值索引陣列

大部份的程式語言都支援數值索引陣列，PHP 跟 C 語言一樣，是由 0 啓始，但我們也可以改變它。例如我們使用 $score_computer 陣列來儲存每個學生的電腦科成績…

範例程式 1-18 indexed_array1.php

```
<?
$score_computer = array ( 80, 91, 75);
?>
```

利用上面的程式碼可以建造出一個擁有三個元素的陣列。值得注意的是 array() 是一個程式結構而不是一個函數。

由於數值索引陣列的索引值是一連串的數字，因此我們只要指定它的索引數值即可很方便的存取所需要的陣列元素。

而是否需要初始化陣列取決於您所需要的內容。由於 PHP 的弱型別特性，我們也可以不要先以 array() 敘述建造出一個陣列，而以直接指派的方法來建造它。

範例程式 1-19 indexed_array2.php

```
<?
$score_computer[0] = 80;
$score_computer[1] = 91;
$score_computer[2] = 75;
?>
```

甚至 PHP 還有一個特色，在數值索引陣列裡，若是不指定索引值而指派一個值給某一陣列的話，它會自動地從索引值的最後一個接下去新增陣列元素。

範例程式 1-20　　indexed_array3.php

```
<?
if ( isset ($score_computer) )
    unset ($score_computer);
$score_computer[] = 80;
$score_computer[] = 91;
$score_computer[] = 75;
?>
```

例如上式與前二個建造數值陣列所產生的結果是完全一樣的。(加上前面的 if …程式碼是爲了怕之前已有變數使用了此名稱，而造成不可預期的影響。)

由於數值陣列的索引是一連串連續的數字，是故我們可以用 for 迴圈更容易的對它做存取。

範例程式 1-21　　indexed_array4.php

```
<?
for ( $i = 0; $i < 3; $i++ )
    echo "$score_computer[$i] ";
?>
```

這段迴圈會顯示出前面所建造出的 $score_computer 中前三個元素的值，但是比起利用人工鍵入一行行程式碼來顯示多個學生的電腦科成績省事多了。也不用爲了 35 個學生使用 35 個不同的變數。

1.2.3 Associative Array – 關聯式陣列

關聯式陣列不太容易利用迴圈來存取，但卻允許索引值更具意義。它允許我們可以利用任意的 key 或索引值（index）來和每個值產生關聯。

如我們之前所見的系統預設變數 $_POST, $_GET, $_SERVER, …等，其實都是關聯式陣列。它們的索引值都不像上一小節所提到的數值，而是對我們人類更有解讀意義的字串。

範例程式 1-22　　asso_array1.php

```
<?
$score_computer2 = array ('apply' => 80, 'orange' => 75, 'banana' => 91);
?>
```

範例程式 1-23　　asso_array2.php

```
<?
```

```
$score_computer2[‘apple’] = 80;
$score_computer2[‘orange’] = 75;
$score_computer2[‘banana’] = 91;
?>
```

上兩例中，同樣的建造了兩個關聯式陣列，其中索引值爲 apple 者，數值爲 80；索引值爲 orange 者，數值爲 75；索引值爲 banana 者，數值爲 91。由第二個例子中我們可以很清楚的看到，對於關聯式陣列的存取，跟一般數值索引陣列一樣，只要直接指定它的索引值即可。

但是關聯式陣列的索引值並不像數值索引陣列的索引值有連續的關係存在。那我們該如何存取它的所有元素呢，就先賣個小關子，後兩節的陣列的操作就會提到了。

1.2.4 Multi-Dimensional Array – 多維陣列

陣列可以不必只是單純的列出索引值與值，陣列中每個位址都可以儲存另一個陣列，如此便可以建立一個二維的陣列，我們可以試著將二維陣列想像成一個帶有長、寬、列與行的矩陣。

例如我們希望可以用一個二維陣列便儲存整個班級中所有學生的所有科目成績，而不是各科各用一個一維陣列來儲存。

範例程式 1-24　　multi_dim_array1.php

```
<?
$score = array ( array (‘chinese’ => 80, ‘english’ => 82, ‘computer’ => 85),
                 array (‘chinese’ => 70, ‘english’ => 95, ‘computer’ => 90));
?>
```

範例程式 1-25　　multi_dim_array2.php

```
<?
$score[0][‘chinese’] = 80;
$score[0][‘english’] = 82;
$score[0][‘computer’] = 85;
$score[1][‘chinese’] = 70;
$score[1][‘english’] = 95;
$score[1][‘computer’] = 90;
?>
```

上兩例產出的 $score 陣列是相同的，這就是一個二維陣列的小應用，藉由這個陣列中各元素也可以是另一個陣列的做法，同樣地也可以發展出多維陣列。

1.2.5 Operations of Arrays – 陣列的操作

我們要知道一個陣列中有多少元素呢？可以使用 count() 函數：

範例程式 1-26　　count_sample.php

```
<?
require (‘indexed_array3.php’);
echo count ($score_computer)
?>
```

上面的程式碼應該會印出 $score_computer 的陣列元素數目。（以之前例子來看，應該是 3。☺）

那關聯式陣列要怎麼一個個列出來呢？可以搭配使用 reset() 及 each() 函數：

範例程式 1-27　　reset_and_each_sample.php

```
<?
require (‘asso_array2.php’);
reset ($score_computer2);
while ( list ($key, $value) = each ( $score_computer2 ) )
    echo “$key = $value\n”;
?>
```

reset 是將一個陣列的內部指標重設至第一個元素，而 each 則是將陣列的索引值及元素值傳回，並將指標指向下一個陣列元素。

上例應該會產生如下的結果：

```
apple = 80
orange = 75
banana = 91
```

在陣列中對相關的資料進行排序常常會很有用，而對一維陣列排序則非常的簡單。

由於數值索引陣列的索引值是連續的數字，所以對它的索引值做排序並沒有任何意義，我們僅能對此種陣列的元素值做排序。

```
sort ( $indexed_array );      // 對陣列 $indexed_array 中的元素值做升冪排序。
rsort ( $indexed_array);      // 則陣列 $indexed_array 中的元素值做降冪排序。
```

而對於關聯式陣列，由於它的索引值也是有意義的，所以可以有對索引值及對元素值的多種排序。

```
asort ( $asso_array );        // 對陣列 $asso_array 中的元素值做升冪排序。
arsort ( $asso_array );       // 對陣列 $asso_array 中的元素值做降冪排序。
ksort ( $asso_array );        // 對陣列 $asso_array 中的索引值做升冪排序。
krsort ( $asso_array );       // 對陣列 $asso_array 中的索引值做降冪排序。
```

咦，奇怪了，對關聯式陣列做排序有什麼用呢？我們就拿之前的範例程式 1-23 asso_array2.php 來看看不同的結果。

範例程式 1-28　　asort_exam.php

```
<?
require (‘asso_array2.php’);
asort ($score_computer2);
reset ($score_computer2);
while ( list ($key, $value) = each ( $score_computer2 ) )
    echo “$key = $value\n”;
?>
```

上例應該會產生如下的結果：

```
orange =75
apple = 80
banana = 91
```

reset_and_each_sample.php 與 asort_exam.php 兩個程式只相差了一行 asort 敘述，就產生了不同的結果。如果我們再將 asort_exam.php 中的 asort 敘述換成 ksort 的話，又會產生如下的結果：

```
apple = 80
banana = 91
orange = 75
```

除了排序外，還有許多陣列專有的函數，以下就節選一些可能會用的到的，其他的就待各位自行自 PHP 手冊中發掘了。☺

```
mixed array_pop ( array array)
    // Pop the element off the end of array
int array_push ( array array, mixed var [, mixed ...])
// Push one or more elements onto the end of array
mixed array_sum ( array array)
    // Calculate the sum of values in an array
int extract ( array var_array [, int extract_type [, string prefix]])
    // Import variables into the current symbol table from an array
bool in_array ( mixed needle, array haystack [, bool strict])
    // Checks if a value exists in an array
void list ( mixed …)
    // Assign variables as if they were an array
void natsort ( array array)
    // Sort an array using a "natural order" algorithm
mixed next (array array)
```

```
    // Advance the internal array pointer of an array
array range ( int low, int high [, int step])
    // Create an array containing a range of elements
void shuffle ( array array)
    // Shuffle an array
bool usort ( array array, callback cmp_function)
    // Sort an array by values using a user-defined comparison function
bool uasort ( array array, callback cmp_function)
    // Sort an array with a user-defined comparison function and maintain index asso.
bool uksort ( array array, callback cmp_function)
    // Sort an array by keys using a user-defined comparison function
```

1.3 Strings – 字串

在 HTML 的表單之下，使用者輸入的大多都是字串，而之前我們只看到如何將字串從系統預設變數中取出來，以及將兩個字串組合起來，與 Perl 類似，PHP 對字串的處理能力也是非常強大滴，現在我們就來看看 PHP 的字串操作。

1.3.1 Formatting Strings – 字串格式化

通常使用者所輸入的字串都需要進一步的整理，而整理的第一步就是去除多餘不必要的字元，如空白、換行字元等。這個動作不是必要，但是當要將字串儲存進檔案、資料庫，或是利用字串來進行比對時，是非常有用的。

```
string chop ();
    // Alias of rtrim();
string trim ( string str [, string charlist]);
string rtrim ( string str [, string charlist]);
    // Strip whitespace from the end of a string
string ltrim ( string str [, string charlist]);
    // Strip whitespace from the beginning of a string
```

PHP 提供了四個函數來做這部份的處理（實際上只有三個），trim() 函數自字串的起始及結尾處去除不必要的字元，並傳回處理過的字串。該函數去除的有換行字元「\n」、歸位字元「\r」、平行與垂直跳格字元「\t, \v」、字串結尾字元「\0」以及空白。當然我們也可以自訂需要去除的字元作爲此函數的第二個參數，利用現有的函數快速地達到我們需要的目的。而 rtrim() 及 ltrim() 即是 trim() 的開始及結尾版。

而在字串格式化部份，PHP 也提供了相當多的函數供我們使用。例如：

```
string nl2br (string str);
```

可以將字串中的換行(newline, \n)符號轉換成 HTML 的
標籤。當以瀏覽器顯示字串時，這個函數的功能很有用。

而在這之前，我們對於顯示字串在瀏覽器上只看過 echo() 與 print() 兩個函數。printf() 及 sprintf() 可以讓我們套用一些複雜的格式化功能。基本上 printf() 與 sprintf() 兩個函數幾乎相同，所差別的只是 printf() 會將結果顯示在瀏覽器上，而 sprintf() 則會傳回一個字串。

如果有寫過 C 語言的經驗的話，相信可以很快的就對這兩個函數上手，它們的原型分別是：

```
int printf (string format [, mixed args…]);
string sprintf (string format [, mixed args…]);
```

傳進這兩個函數的第一個參數是字串格式，用以格式字串，而其他參數則是

要代換進格式化字串中的變數。

例如我們想要印出一個學生的各科總分 $total，使用 echo() 的話，我們可以寫成：

```
echo “總分爲: $total.”;
printf (“總分爲 %s.”, $total);
```

上式中，printf 裡面有個%s 的代替字串，它稱之爲轉換規格（conversion specification），這裡是指「以字串代替」的意思。本例中，%s 會被 $total 以字串取代掉。（如果我的 $total 是數字該怎辦？沒關係，PHP 會自然將其做型別轉換）

使用 printf() 的好處是可以利用更有用的轉換規格來格式化字串。例如 $average 若是一個浮點數的話，我們就可以 %.2f 來顯示它的精確度到小數點兩位。

```
printf (“平均分數爲 %.2f.”, $average);
```

我們可以在格式字串裡使用多個轉換規格。假設有 n 個轉換規格，那麼在格式字串後就會有 n 個參數，也就是 printf() 總共會有 n+1 個參數。

所有的轉換規格都以 % 符號起始，如果想要印出 % 符號，就要使用%%。

代碼	意義
b	將變數解釋爲整數並以二進位制顯示。
c	將變數解釋爲整數並以字元顯示。
d	將變數解釋爲整數並以十進位制顯示。
f	將變數解釋爲浮點數並以浮點數顯示。
o	將變數解釋爲整數並以八進位制顯示。
s	將變數解釋爲字串並以字串顯示。
x	將變數解釋爲整數並以十六進位制，小寫英文字母 a-f 顯示。
X	將變數解釋爲整數並以十六進位制，大寫英文字母 A-F 顯示。

字串大小寫同樣也是可以被格式化的，我們可以使用函數來將字串中所以英文字母換成大寫或是小寫，或者是將字串中第一個字或每個字的第一個字母轉換成大寫。

```
string strtoupper ( string str);      // Make a string uppercase
string strtolower ( string str);      // Make a string lowercase
string ucfirst (string str);          // Make a string's first character uppercase
string ucwords (string str);   // Uppercase the first character of each word in a string
```

由於有些字元在 C 語言、HTML 語言或者是資料庫 SQL 指令中有特別的意義，所以一般字串並不能直接印出或者是儲存入資料庫，這時 PHP 也有相對的函數來幫我們做這部份的處理：

```
string addslashes ( string str);          // Quote string with slashes
string stripslashes ( string str);        // Un-Quote string with addslashes()
```

```
string addcslashes ( string str);          // Quote string with slashes in a C style
string stripcslashes ( string str);   // Un- Quote string with addcslashes()
```

最常使用到的就是使用者輸入的資料要存進資料庫時，在 SQL 指令最好加上 addslashes()，以免發生不測。

1.3.2 Merge & Split Strings – 字串的合併與分離

我們常常只需要字串的某一部份，例如我們也許只需要某句裡的某幾個字，或者是將一個 email address 分成好幾個部份，PHP 提供了一些好用的字串函數及正規表示式函數來供我們使用，先來看看簡單的字串函數。

最常會用到的大概就是 explode() 吧，它的原型是：

```
array explode ( string separator, string input);
```

這個函數類似於 Perl 語言的 split 函數。它可以將字串以 separator 爲分隔點，將字串 input 分成好幾個部份存在 array 中傳回。例如要從一個 email address 中取得 domain name 的話，可以利用下列程式：

範例程式 1-29 explode_exam.php

```
<?
$email_address = “007.audi@aaa.bbb.ccc.ddd.eee”;
$email_array = explode (“@”, $email_address);
printf(“Account: %s\n”, $email_array[0]);
printf(“Domain Name: %s\n”, $email_array[1]);
?>
```

有了 explode 來將字串分離成陣列，那麼相反地也有將陣列組合成一個字串的 implode() 或是叫做 join() 。

```
$new_email_address = implode(“@”, $email_array);
```

```
string strtok ( string input, string separator);
```

strtok() 不像 explode() 一次就將整個字串打散，而是一次只針對字串的一部份（稱爲 tokens）。如果每次只處理字串裡的單一文字，strtok() 也是個可以選擇的函數。

```
string substr ( string string, int start [, int length]);
```

substr() 是一個允許我們由指定的起點與終點取出字串的函數。例如我們若想取出某一字串的某幾個字元的話，就可以使用：

```
$test_string = “This is a test String”;
printf ( “%s”, substr($test_string, 0, 4);       // 顯示 This
printf ( “%s”, substr($test_string, 8);          // 顯示 a test String
```

```
printf ( “%s”, substr($test_string, -6);          // 顯示 String
printf ( “%s”, substr($test_string, 10, 5);       // 顯示什麼呢?
printf ( “%s’, substr($test_string, 8, -7);       // 顯示什麼呢?
```

1.3.3 Operations of Strings – 字串的操作

除了字串的格式化與合併、分離化，還有許多字串的操作，以下就略提一些比較會常用到的函數，其他的就有待各位從 PHP 手冊中發掘 PHP 對於字串處理到底是有多麼強了了。☺

取得字串長度：

```
int strlen ( string str);
```

字串比較：

```
int strcmp (string str1, string str2);
int strncmp (string str1, string str2, int len);
int strcasecmp (string str1, string str2);
```

在字串中尋找字元，子字串：

```
int strpos (string haystack, string needle [, int offset]);
string strstr (string haystack, string needle);
string stristr (string haystack, string needle);
string strchr (string haystack, string needle);
string strrchr (string haystack, string needle);
```

字串反轉：

```
string strrev ( string str);
```

字串代換：

```
string str_replace ( mixed search, mixed replace, mixed subject [, int &count]);
string str_ireplace ( mixed search, mixed replace, mixed subject [, int &count]);
string substr_replace ( mixed search, mixed replace, int start [, int length]);
```

字串編碼：

```
string chr (int ascii);
int ord ( string str);
string crypt ( string str [, string salt]);
string md5 ( string str [, bool raw_output]);
string sha1 ( string str [, bool raw_output]);
```

1.3.4 Regular Expression – 正規表示式

PHP 支援兩種正規表示式語法：POSIX 與 Perl（PCRE, Perl-compatible regular expression）。以下介紹的是比較簡單的 POSIX 語法，如果想了解 PCRE 的話，請自行參考 PHP 手冊。☺

正規表示式是用以在文章中描述樣式的一種方式。在 PHP 裡比對正規表示式比較像使用 strstr() 函數，因爲是在字串裡比對字串，而不僅只是字串相等的比較。

除了精確的樣式比對外，我們也可以利用特殊字元指示出其潛在意義。

首先，我們可以用「.」字元來取代任何除了換行字元外的單一字元。例如正規表示式的

```
.at
```

可以與 2at, cat, sat, mat, …等相符。

除了與任意字元相符外，我們還可以明確的指定要與所指定的字元相符。例如

```
[a-z] at
```

可以與 cat, sat, mat 相符，但不與 2at 相符。

同樣地，我們也可以指定不得屬於某一範圍的字元集，只要加個「^」即可：

```
[^a-zA-Z] at
```

用於 POSIX 格式正規表示式的字元類別

類別	比對
[[:alnum:]]	英數字字元
[[:alpha:]]	英文字元
[[:lower:]]	小寫字元
[[:upper:]]	大寫字元
[[:digit:]]	數字字元
[[:xdigit:]]	十六進位數字字元
[[:punct:]]	標點符號字元
[[:blank:]]	跳格與空白字元
[[:space:]]	空白字元
[[:cntrl:]]	控制字元
[[:print]]	所有可見字元
[[:graph:]]	除空白字元外的所有可見字元

有時候我們會希望指明某些特定字串與字元類別會出現一次以上，在正規表示式中有兩個特殊符號用以說明這種情形：「+」（表示該樣式出現 1 次或 1 次以上）與「*」表示該樣式出現 0 次或 0 次以上。例如 [[:alnum:]]+ 表示至少有一個英數字元。

將表示式分爲幾個子表示式常會很有用，如此便可以表示「至少這些字串之一其後一定會連接著那些字串之一」。這裡可以用括號（）來達成此目的。例如：

```
(very )*large
```

與 large, very large, very very large 等等都是相符的。

除了可以用 +,* 來表示一個子表示式可以重複多次外，也可以使用大括號{}來表示出現的次數，例如：

(very){1,3}

與 very, very very, very very very 等也都是相符的。

在字串中子字串相符之外，我們可能也需要指定某些特定子字串應出現於字串的開頭或是結尾，這時候就會應用到「^」與「$」兩個符號了。

「^」符號表示用於正規表示式起點，表示樣式須位於搜尋字串起點；而「$」則表示樣式須位於搜尋字串結尾。

例如下式表示位於字串起點的 www 樣式：

```
^www
```

下式則表位於字串終點的.edu.tw 樣式：

```
\.edu\.tw$
```

這樣表示字串本身是 a-z 的任何一個字元所組成的：

```
^[a-z]$
```

最後，下式爲一個簡單的 email address validate check。

```
if (!eregi('^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$', $email_address) {
    // email address is invalid
} else {
    // email address is valid
}
```

而 Regular Expression 除了檢查是否相符外，也可以同時取得一些相符的字串，以下爲上面 email address 檢查的小應用，有興趣的人就自行再研究研究嚕！

```
if (!eregi('^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9\-]+)\.([a-zA-Z0-9\-\.]+)$',
$email_address, $match) {
    // email address is invalid
} else {
    // email address is valid
    $account = $match[1];
    $host = $match[2];
    $domain = $match[3];
}
```

1.4 I/O Related – I/O 相關使用

在能利用 PHP 對網頁及使用者輸入的資料作處理後，我們來看看該怎麼把資料儲存下來。

基本上有兩種方式讓我們儲存資料，一是存成檔案，另一種就是存進資料庫了。資料庫部份晚些會敘述到，讓我們先看看該怎麼把資料存進檔案。

1.4.1 File Read/Write – 檔案的讀寫

寫入檔案的步驟有三：

1. 開啓檔案，若該檔案不存在則先產生該檔。
2. 寫入資料至該檔
3. 關閉檔案

同樣地，關閉檔案的步驟也有三步驟：

1. 開啓檔案，若無法開啓時，則要能辨別出並跳出此部份程式。
2. 自檔案中讀取內容。
3. 關閉檔案。

在 PHP 裡開啓檔案用到的是 fopen() 函數。在開啓檔案時須指出其開啓模式，是唯讀模式呢、寫入模式呢、還是讀寫模式。而寫入檔案時，也要考慮到是覆寫原有檔案的內容，或者是附加於檔案最後。還有一個是比較不常用到的，那就是以文字模式，還是二進位模式來開啓檔案。

```
resource fopen ( string filename, string mode [, int use_include_path]);
bool fclose ( resource handle);
int fputs ( resource handle, string string [, int length]);
string fgets ( resource handle [, int length]);
```

上式便爲 fopen(), fclose() 及兩個常用到的檔寫讀寫函數 — fputs(), fgets() 之原型。如果我們要將之前的學生各科分數寫進一個檔案的話，可以再加上利用 fopen() 加上 fputs() 函數寫成類似以下的程式碼：

範例程式 1-30 fopen_exam1.php

```
<?
$fp = fopen ( $filename, “w”);
if ( $fp != NULL ) {
    for ( $i = 0; $i < $students; $i++ ) {
        fputs ( $fp, “Student %02d:\t”, $i);
        fputs ( $fp, “Chinese: %02d\t”, $score[$i][‘chinese’]);
```

```
            fputs ( $fp, “English: %02d\t”, $score[$i][‘english’]);
            fputs ( $fp, “Computer: %02d\n”, $score[$i][‘computer’]);
      }
      fclose ( $fp);
}
?>
```

如果是反過來要從一個檔案中讀取呢，那也是類似的情形，只不過應該要改用 fgets() 函數來讓我們一行一行的讀取。

範例程式 1-31 fopen_exam2.php

```
<?
$fp = fopen ( $filename, “r”);
if ( $fp != NULL ) {
      $i = 0;
      while ( $line = fgets ( $fp) ) {
            $line = chop($line);
            list ($student_no, $score1, $score2, $score3 ) = explode(“\t”, $line);
            $score2[$i][‘chinese’] = $score1;
            $score2[$i][‘english’] = $score2;
            $score2[$i][‘computer’] = $score3;
            $i++;
      }
      fclose($fp);
}
?>
```

以上便是簡單的檔案讀寫方法，但是 PHP 還有一些非常方便的檔案讀寫函數可以使用。下面就列出一些可能會用到的函數以供參考。

```
bool feof ( resource handle);
      // Tests for end-of-file on a file pointer
string fgetc ( resource handle);
      // Gets character from file pointer
array fgetcsv ( resource handle, int length [, string delimiter [, string enclosure]]);
      // Gets line from file pointer and parse for CSV fields
string fgetss ( resource handle, int length [, string allowable_tags]);
      // Gets line from file pointer and strip HTML tags
string file_get_contents ( string filename [, int use_include_path [, resource context]]);
      // Reads entire file into a string
```

```
int file_put_contents ( string filename, string data [, int flags [, resource context]]);
    // Write a string to a file
array file ( string filename [, int use_include_path [, resource context]]);
    // Reads entire file into an array
int readfile ( string filename [, bool use_include_path [, resource context]]);
    // Outputs a file
```

1.4.2 File Uploads – 檔案上傳

老師們常常會需要收學生們繳交上來的作業，在沒有電腦幫忙的年代，不是親力親爲外，就是找個學生當小老師幫忙。在電腦網路發達的現在，則常會請學生用 email 或是 FTP 檔案傳輸的方式上傳作業。不過 email 信箱容量不大容易爆，用 FTP 的話，還得先教會這群小朋友使用。如果是用他們這世代最熟悉的瀏覽器上傳的話，大概不用五分鐘就解決了。我們這就來看看該怎麼用 PHP 寫個檔案上傳的程式吧！

範例程式 1-32 file_upload.html

```
<form enctype="multipart/form-data" action="file_upload.php" method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
Send this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

上面是一個可以讓使用者點選檔案上傳的表單，它比一較表單多了 enctype 這個部份，這可是一定要的喔！雖然說新的瀏覽器軟體看到<input type=”file”>就知道可以上傳檔案，可是難保有小朋友還用著舊舊的 IE 5.0 或是舊版不相容的軟體。

使用者選好檔案並按下 [Send File] 鈕後，我們該如何在程式中得到這個檔案呢? PHP 都幫大家想好了，一個檔案會有五個變數來讓我們存取，一起來看看：

```
$_FILE [‘userfile’][‘name’]; // 使用者端的檔案名稱
$_FILE [‘userfile’][‘type’];  // 檔案的 MIME 類別，例如 “application/msword”
$_FILE [‘userfile’][‘size’];  // 檔案的大小
$_FILE [‘userfile’][‘tmp_name’];      // 檔案在伺服器上的暫時檔名
$_FILE [‘userfile’][‘error’]; // 上傳的錯誤碼
```

取得這五個變數就算檔案上傳成功了嗎？不不不，因爲檔案上傳是會先放在伺服器上一個暫時的目錄底下，要是沒有把它做進一步處理的話，等到這頁的 PHP 程式結束，該檔案也就會在伺服器上消失了。我們來看看該怎麼處理：

範例程式 1-33　　file_uploaded.php

```
<?
$uploaddir = “/var/www/uploads”;
$uploadfile = $uploaddir . $_FILE[‘userfile’][‘name’];

print “<pre>\n”;
if ( move_uploaded_file ( $_FILE[‘userfile’][‘tmp_name’], $uploadfile) ) {
    printf(“File %s was successfully uploaded\n”, $_FILE[‘userfile’][‘name’]);
    print “Here is some debugging info:\n”;
} else {
    print “Possible file upload attack! Here is some debugging info:\n”;
}
print_r ( $_FILES);
print “</pre>\n”;
?>
```

將檔案利用 move_uploaded_file() 函數移到我們想放置檔案的地方後，才算上傳成功喔！

1.4.3 Operations of Filesystems – 檔案系統的操作

除了開、讀、寫及關檔案，檔案系統也有一些操作，以下就節選一些，其他許多的操作，也是有待各位伯樂去發掘了。

```
bool chgrp ( string filename, mixed group);
    // Changes file group
bool chmod ( string filename, int mode);
    // Changes file mode
bool chown ( string filename, mixed user);
    // Changes file owner
bool copy ( string source, string dest);
    // Copies file
bool delete ( string filename [, resource context]);
bool unlink ( string filename [, resource context]);
    // Deletes a file
string basename ( string path [, string suffix]);
    // Returns filename component of path
string dirname ( string path);
```

```
    // Returns directory name component of path
bool file_exists ( string filename);
    // Checks whether a file or directory exists
int filesize ( string filename);
    // Gets file size
bool is_dir ( string filename);
    // Tells whether the filename is a directory
bool is_file ( string filename);
    // Tells whether the filename is a regular file
bool mkdir ( string pathname [, int mode [, resource context]]);
    // Makes directory
bool rmdir ( string dirname [, resource context]);
    // Removes directory
bool rename ( string oldname, string newname [, resource context]);
    // Renames a file
bool rewind ( resource handle);
    // Rewind the position of a file pointer
array stat ( string filename);
    // Gives information about a file
```

1.4.4 SESSIONS

HTTP 本身是一種 Stateless（無狀態）的協定，也就是說 HTTP 沒有辦法分辨連續兩次的網頁要求是不是來自同一個使用者端。試著想像有個學生要上傳檔案繳交作業，在第一個畫面我們先用帳戶及密碼來認證這個學生，可是因爲 HTTP 本身的協定所影響，我們並不知道下一次的網頁要求是不是同一位學生，忘了有沒認證過倒還好，若是有另一個學生同時在線上上傳檔案，那…不就糗大了。而這小節要介紹的 Session 控制就是讓我們能夠追蹤使用者。

PHP 的 Session 是由 Session ID（一個隨機的密碼）來控制。Session ID 是由 PHP 產生，並儲存在用戶端直到 session 終了。它就像一把用來開啓 session 變數的鑰匙。Session 變數是儲存在伺服器端，而 Session ID 則是用戶端唯一可看到的資訊。藉由 Session ID 我們便可以存取在同一個 session 期間用的 session 變數。

Session 的使用通常有四個步驟：

1. 啓用 session
2. 註冊 session 變數
3. 使用 session 變數
4. 刪除已註冊變數與結束 session

啓用 session 的方法很簡單，只要呼叫如下的函數即可：

```
bool session_start ( void );
```

註冊 session 變數也不難，只要將變數名稱傳給如下的函數就可以了：

```
bool session_register ( mixed name [, mixed ...])
```

使用 session 變數多了一點小步驟，我們可能需要先檢查這個 session 變數是否已註冊，然後就可以用該變數來存取它或是從系統預設的 $_SESSION 關聯式陣列中存取它。

範例程式 1-34 session_exam.php

```
<?
if ( session_is_registered (“my_session_var”) ) {
    print “my_session_var = “ . $my_session_var . “\n”;
}

if ( isset ($_SESSION[‘my_session_var’]) ) {
    print “my_session_var = “ . $my_session_var . “\n”;
}
?>
```

刪除 session 變數以及清除 session 也只要各呼叫一個函數即可。

```
bool session_unregister ( string name);
    // Unregister a global variable from the current session
bool session_destroy ( void );
    // Destroys all data registered to a session
```

Session 也可以設定超過時間自動過期（expire），以免有的小朋友喜歡開著視窗到處跑…

```
int session_cache_expire ( [int new_cache_expire]);
```

以上就是 session 控制的大概用法，真正的使用經驗就靠各位自己去體驗了。☺

Chapter 2. MySQL

2.1 RDBMS Concepts – 關聯式資料庫概念

關聯式資料庫是目前最普遍的資料庫型式，而它是以可靠的關聯式理論爲基礎，雖然我們在使用 MySQL 上並不需要瞭解關聯式理論，但仍必瞭一些基本的資料庫觀念。

如果有用過微軟 Office 系列軟體中的 Access，那麼它本身其實也是個簡單的資料庫。

2.1.1 Table – 資料表

關聯式資料庫是由關聯式所組成的，說得簡單些，就是一個個資料表。資料表顧名思義就是有資料所組成的表格。如果曾用過如 Lotus 1-2-3, Excel 的電子試算表，那麼就已經使用過關聯式的資料表了。

讓我們來看看一個範例：

表格 2-1　　學生個人資料表

STUDENTS

StudentID	Name	Address	Tel
485081148	林黛王	台北市大安區…	02-2707-5215
485082326	賴笨任	台北市中正區…	02-2344-4752
485083371	洪小卡	台北縣永和市…	02-2921-4209

這個資料表有一個名稱（Students），一些對應到不同資料的欄位以及一些對應到每個學生的資料列。

2.1.2 Column – 欄位

在資料表中的每個欄位有著獨一無二的名稱並包含著不同的資料，而每個欄位則有個相關聯的資料型態。舉例來說，在表格 2-1 的學生個人資料表中，我們可以看到 StudentID 存的是整數型態的學號，而其他三個欄位則是字串。欄位有時可以稱做是屬性（attribute）

2.1.3 Row – 資料列

資料表中的每一個資料列表示一個不同的學生，而因爲表格格式的關係，它們有著相同的屬性。資料列有時也稱作資料錄（record）或是群組（tuple）。

2.1.4 Value – 值

每一個資料列是由一組對應著每個欄位的不同值所組成的，而每個值必須符合該欄位所規定的資料型態。

2.1.5 Key – 鍵

我們必須有一個方法來辨識每位學生，而姓名通常不是個好方法。以一個非常普通的姓名 — 王小明爲例，假若班上有二個王小明的話，雖然我們可以由其他如住址、電話、生日等其他資料來辨別是大王小明還是小王小明。但這樣還是相當麻煩且在資料庫中需要一個以上的欄位才能識別。

是故我們有了學號這個唯一的值來識別不同的人。而在資料庫中用來做辨識的欄位稱作是鍵或者是主鍵，而鍵值也可以由多個欄位來組成，不過通常主鍵都是只有一個欄位。

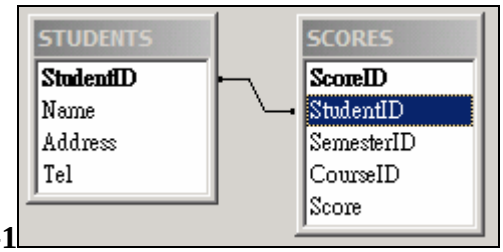
資料庫通常是由許多資料表所組成並且會使用到鍵值當做是某個資料表到另一個資料表的參數。例如在表格 2-2 中，我們加入了第二個資料表到資料庫中，而新的資料表則儲存了每位學生各學期各科的期末分數。

表格 2-2　　學生成績表

SCORES

ScoreID	StudentID	SemesterID	CourseID	Score
1	485081148	9301	0102	90.5
2	485082326	9301	0102	89.0
3	485083371	9301	0102	91.0
4	485082326	9301	0105	93.0

在學生成績表中的每一列表示某位學號的學生在某一學期的某科目分數，而不再是表示各學生的個人資料。我們可以清楚的知道每一列的分數是屬於哪一位學生的，因爲我們儲存了他們的 StudentID 值。所以我們可以知道林黛玉這位學生在 9301 學期的 0102 科目拿了 90.5 的高分。



圖表 2-1

這樣的關係在關聯式資料庫中被稱之爲外部鍵（foreign key），StudentID 是 STUDENTS 資料表的主鍵，可是當其出現在別的資料表像是本例的 SCORES 時，它便成爲此表的外部鍵了。

也許有人會覺得納悶，爲什麼要選擇兩個分離的資料表，而不單純的把所有學生的個人資料及成績都儲存在同一個資料表中呢？我們可以看看學生個人資料表中，ScoreID 爲 2 及 4 的這兩筆，它們是同一位學生同一學期，但不同科目的成績，如果我們把個人資料也儲存在同一個資料表中時，那麼個人資料就重複儲存了，不是嗎？除了空間的浪費外，如果學生的個人資料改變了，那麼所有有該位學生成績的資料列也得跟著修改，不然就會發生資料不一致了！

2.1.6 Schema – 規格

對於資料庫中所有資料表的完整設計稱作是資料庫規格（database schema），它就像是資料庫的設計藍圖一般。而規格必須顯示出和資料表相關的各個欄位以及其資料型態，並且指出每個資料表的主鍵和外部鍵。規格內並不含有任何資料，但是我們可以在規格中使用一些範例來做爲說明。我們可以利用實體關係圖(entity relationship diagram）或是使用文字形式來顯式規格，例如：

STUDENTS (StudentID, Name, Address, Tel)

SCORES (ScoreID, StudentID, SemesterID, CourseID, Score)

在規格中使用實底線是表示說此欄位爲資料表的主鍵，而使用虛底線則表示爲資料表的外部鍵。

2.1.7 Relation – 關係

外部鍵是用來表示兩個資料表間的關係（relationship）的。例如，圖表 2-1 中從 SCORES 到 STUDENTS 的連結就是表示 SCORES 資料表中的某一欄和 STUDENTS 資料表中某一欄的關係。

在關聯式資料庫中存在有三種基本的關聯，而這三種是依據有關聯兩方的數量多寡來做分類，分別爲一對一、一對多以及多對多。

一對一的關係就是指彼此都只有一樣是有關係的。舉例來說，若是把 Address

自 STUDENTS 資料表中取出來存放至另一個 ADDRESSES 資料表，則它們之間就會有一對一的關係。

而 SCORES 與 STUDENTS 間的關係就是一對多的關係，一個 StudentID 在 SCORES 資料表裡有多筆相對應的資料。

而多對多的關係是說，一個資料表的許多資料列會連結到另一個資料表的許多資料。假設我們另有兩個資料表名爲 TEACHERS 與 COURSES，一個科目可能有多個老師任教，而這些老師可能又教了其他科目。這種多對多的關係可能需要再多建立一個資料表才能很清楚的知道哪些老師與哪些科目相關。

2.2 Advanced MySQL – MySQL 進階

由於許多資料庫的管理都是在文字指示符號下下達指令，所以有關於 MySQL 資料庫的管理部份，待我們在下一章使用 phpMyAdmin 這個以 PHP 寫成的網頁式 MySQL 管理介面後，再來討論之。

我們先來看看 MySQL 能支援我們哪些部份來學習資料庫的使用。

2.2.1 Database Privilege System – 資料庫權限

一般資料庫系統都會有自己的權限系統，與作業系統的權限系統是不一樣的。

權限是指對於某一個物件所能執行的特定動作，而且是和使用者有關。它的觀念和檔案使用權限相似。

當我們要存取 MySQL 時，我們也需要一位資料庫使用者的權限，讓我們能取得、新增、修改、刪除資料表內的資料，甚至是新增、刪除表格或者是管理一個資料庫。

2.2.2 Data Types – 資料型態

MySQL 也支援 ANSI SQL92 所規範的資料型態，甚至新增了一些好用的資料型態，一起來看看一些常用的資料型態：

Numeric Types

Type	Bytes	From	To
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807
FLOAT4	4	-	-
FLOAT8	8	-	-
DOUBLE	8	-	-
REAL	8	-	-

Date and Time Types

Type	“Zero” value
DATETIME	‘0000-00-00 00:00:00’
DATE	‘0000-00-00’
TIMESTAMP	00000000000000 (length depends on display size)
TIME	’00:00:00’
YEAR	0000

String Types

Type	Max. size	Bytes
TINYTEXT or TINYBLOB	2^8 – 1	255
TEXT or BLOB	2^16 – 1 (64k -1)	65535
MEDIUMTEXT or MEDIUMBLOB	2^24 – 1 (16M -1)	16777215
LONGBLOB	2^32 – 1 (4G -1)	4294967295

The CHAR and VARCHAR Types

Value	CHAR(4)	Storage required	VARCHAR(4)	Storage required
‘’	‘’	4 bytes	‘’	1 byte
‘ab’	‘ab ‘	4 bytes	‘ab’	3 bytes
‘abcd’	‘abcd’	4 bytes	‘abcd’	5 bytes
‘abcdefgh’	‘abcd’	4 bytes	‘abcd’	5 bytes

2.2.3 Select Syntax – 如何在資料表中取得想要的資料

```
SELECT [STRAIGHT_JOIN]
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
        [HIGH_PRIORITY] [DISTINCT | DISTINCTROW | ALL]
    select_expression,...
    [INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ...
      [HAVING where_definition]
      [ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC] ,...]
      [LIMIT [offset,] rows]
      [PROCEDURE procedure_name]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

以上是 SQL SELECT syntax，看起來好像非常複雜，但我們只學所需要的就足夠了。

例如我們想取出某一位學生的地址，就可以下

```
SELECT Address FROM STUDENTS WHERE StudentID =  學號;
```

意思是自 STUDENTS 資料表中取得 StudentID 爲「學號」這位學生的 Address 資料。

如果我們也同時想取得電話呢，就可以下

```
SELECT Address, Tel FROM STUDENTS WHERE StudentID =  學號;
```

若是想取得整列資料或是覺得這樣一個個欄位指定太麻煩了，就可以以「*」來代表所有的資料欄位，一次取得該學生的所有個人資料。

當然也可以不加上 where 及其之後的 where clause，但是這樣就會把該資料表中的所有學生資料都傳回來，就看各位如何去使用了。

```
SELECT t.NAME, s.sum(score) as SUM
    FROM STUDENTS t, SCORES s
    WHERE t.StudentID BETWEEN 485080000 AND 485089999 AND t.StudentID
= s.StudentID
    GROUP BY t.StudentID
    ORDER BY SUM DESC
    LIMIT 10
```

以上是一個取出學號介於 485080000 與 485089999 間總分最高的前十名之學生姓名及總分的 SQL 指令碼，其實 SQL 指令要寫個一兩頁並不是那麼困難。☺

2.2.4 Insert Syntax – 如何在資料表中新增資料

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    VALUES ((expression | DEFAULT),...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expression, ... ]
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name
    SET col_name=(expression | DEFAULT), ...
    [ ON DUPLICATE KEY UPDATE col_name=expression, ... ]
```

以上是 SQL INSERT syntax，看起來也非常複雜，但我們也是學所需要的就足夠了。

例如最近新來了一位轉學生，我們想要在 STUDENTS 資料表裡新增一筆這位張大頭同學的個人資料，我們可以有下列兩種 SQL 指令碼寫法：

```
INSERT INTO STUDENTS VALUES (485084100, '張大頭', '台北市大安區…', '02-2394-0112');
or
INSERT INTO STUDENTS (StudentID, Name, Address, Tel) VALUES (485084100, '張大頭', '台北市大安區…', '02-2394-0112');
```

第一種指令法寫法必須在程式寫作者的腦中自動的將欄位及其值的順序對應自動配對好，一有差錯就糗大了。

第二種指令法寫法多了(StudentID, Name, Address, Tel)等字元，這是在指定後方 VALUES 欄位的值的對應欄位。這種用法常會在一些欄位有預設值或是自動增加值（Auto Increment，流水號即爲一種應用）時用到。

在 INSERT syntax 中提到的第二種方法常會在資料搬移、重組時用到，這裡就暫不說明。而第三種方法跟上面的指定欄位類似，就不多贅述了。

2.2.5 Update Syntax – 如何在資料表中修改已有的資料

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2, ...]
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT #]
```

以上即是 SQL UPDATE syntax，看起來也是有點複雜，但我們還是只學所需要的就足夠了。

如果今天一位學生某科成績被錯植了，要更動的話，我們就可以如此下 SQL 指令：

```
UPDATE SCORES SET Score = 85 WHERE StudentID = 485081326 AND
SemesterID = 9301 AND CourseID = 0102;
```

若是連科目代碼都打錯了呢，那麼可以如此的寫 SQL 指令：

```
UPDATE SCORES SET Score = 85, CourseID = 0103 WHERE StudentID =
485081326 AND SemesterID = 9301 AND CourseID = 0102;
```

2.2.6 Delete Syntax – 如何在資料表中刪除資料

```
DELETE [LOW_PRIORITY] [QUICK] FROM table_name
        [WHERE where_definition]
        [ORDER BY ...]
        [LIMIT rows]

or

DELETE [LOW_PRIORITY] [QUICK] table_name[.*] [,table_name[.*] ...]
        FROM table-references
        [WHERE where_definition]

or

DELETE [LOW_PRIORITY] [QUICK]
        FROM table_name[.*], [table_name[.*] ...]
        USING table-references
        [WHERE where_definition]
```

以上爲 SQL DELETE syntax，看起來不像太複雜，我們就來看看該如些刪除資料吧。

如果某科成績在鍵入時整個都看錯了，那麼一筆筆更新可能不是很方便，有時候最快的作法是完全的破壞再重新建設☺

```
DELETE FROM SCORES WHERE SemesterID = 9301 and CourseID = 0102;
```

這樣就可以把所以 SemesterID 爲 9301 且 CourseID 爲 0102 的資料列全部刪除了。

以上就是資料庫一些操作的簡單做法，不過千萬得注意，在更新與刪除資料時千萬別「誤改」或「誤刪」了不該更動的資料喔！

Chapter 3. PHP with MySQL

除了之前所介紹的 PHP 一些函數外，它也能很方便快速的存取資料庫。接著我們就開始看看該怎麼把功能強大的 PHP 語言跟世人離不開的資料庫結合起來吧！

3.1 PHP MySQL functions

PHP 支援著許多資料庫，像 MySQL、Oracle、Sybase、MS-SQL、LDAP、ODBC、…等它都可以很方面的連結上，不過一般大學生及教育上最常用到的就是 MySQL 這個功能、效能都不輸給商業化資料庫軟體的 OpenSource 資料庫軟體。

接著介紹怎麼利用 PHP 的 MySQL 函數來在 PHP 中直接存取 MySQL！

以下是常會用到的 PHP MySQL 函數：

```
resource mysql_connect ( [string server [, string username [, string password [, bool new_link [, int client_flags]]]]]);
    // Open a connection to a MySQL Server
resource mysql_pconnect ( [string server [, string username [, string password [, int client_flags]]]]);
    // Open a persistent connection to a MySQL server
bool mysql_select_db ( string database_name [, resource link_identifier]);
    // Select a MySQL database
bool mysql_close ( [resource link_identifier]);
    // Close MySQL connection
resource mysql_query ( string query [, resource link_identifier]);
    // Send a MySQL query
array mysql_fetch_row ( resource result);
    // Get a result row as an enumerated array
object mysql_fetch_object ( resource result);
    // Fetch a result row as an object
int mysql_affected_rows ( [resource link_identifier]);
    // Get number of affected rows in previous MySQL operation
bool mysql_free_result ( resource result);
    // Free result memory
int mysql_insert_id ( [resource link_identifier]);
    // Get the ID generated from the previous INSERT operation
string mysql_error ( [resource link_identifier]);
    // Returns the text of the error message from previous MySQL operation
```

比如我們剛才的 STUDENTS 與 SCORES 存在一台名叫 moon 機器上的 my_class 資料庫上。要列出所有學生的個人資料，程式可以很簡單的寫成下式：

範例程式 3-1　　mysql_exam1.php

```
<?
    mysql_connect(“moon”, “MY_DB_USERNAME”, “MY_DB_PASSWORD”);
    mysql_select_db(“my_class”);
    $sql_str = “select * from STUDENTS”;
    $result = mysql_query($sql_str);
    if ( $result ) {
        while ( list($studentid, $name, $address, $tel) = mysql_fetch_row($result) ) {
            printf(“學號: $studentid\t 姓名: $name\t 住址: $address\t 電話:$tel\n”);
        }
        mysql_free_result($result);
    }
    mysql_close();
?>
```

以上程式其實有點小問題，就是它沒有檢查資料庫開始是否成功，與資料欄的對應是否正確。我們可以改成如下式這般：

範例程式 3-2　　mysql_exam2.php

```
<?
    if ( ($db = mysql_connect(“moon”, “MY_DB_USERNAME”,
“MY_DB_PASSWORD”)) == NULL || mysql_select_db(“my_class”) == FALSE ) {
        print “資料庫連結失敗!\n”;
        exit;
    }
    $sql_str = “select * from STUDENTS”;
    $result = mysql_query($sql_str);
    if ( $result ) {
        while ( $object = mysql_fetch_object($result) ) {
            printf(“學號: %s\t 姓名: %s\t 住址: %s\t 電話:%s\n”,
$object->StudentID, $object->Name, $object->Address, $object->Tel);
        }
        mysql_free_result($result);
    }
    mysql_close();
?>
```

這麼簡單就把 PHP 跟資料庫連結起來了，夠快吧！不過還有更棒的喔！☺

3.2 Useful Tools/Library – 有用的工具、函數

3.2.1 phpMyAdmin

phpMyAdmin 簡單的說就是一套以 PHP 語言寫成的 MySQL 的管理工具。透過此一程式，可以直接利用瀏覽器透過網頁介面去管理 MySQL，不需要到系統上去用一些較不友善的文字介面。而有關於 phpMyAdmin 的使用，就請大家參考 http://dob.tnc.edu.tw/themes/old/showPage.php?s=1343&t=136 網頁的說明。

3.2.2 ADOdb

PHP 雖然在支援資料庫連結方面非常的廣泛。但，不幸地，在 PHP 中所有資料庫的存取大概都會有些微的不同。之前看到 PHP MySQL 的函數都以 mysql_ 開頭，而 MS-SQL 的函數則是大多以 mssql_ 起始，可遇到了 Oracle 又變成 oci，看了頭都有點大。而且並不是這個資料庫有的函數，在另一個資料庫就會有相對應的函數。所幸，有了一個名爲 ADOdb 的資料庫封裝函式庫的出現，讓我們先看看 ADOdb 的官方中文版手冊：http://www.ifin.net.tw/adodb/adodb_tutorial.htm 。

就以上一小節我們寫的 mysql_exam2.php 來修改，使用 ADOdb 的函式庫時，大概可以寫成下式這樣：

範例程式 3-3 adodb_exam.php

```
<?
    $database = "mysql";
    include_once("adodb.inc.php");
    $adoconn = ADONewConnection($database);
    $adoconn->Connect("my_class"  , "MY_DB_USERNAME",
"MY_DB_PASSWORD", "moon");

    $sql_str = "select * from STUDENTS";
    $result = $adoconn->Execute($sql_str);
    if ( $result ) {
        while ( $object = $result->FetchNextObject() ) {
            printf("學號: %s\t 姓名: %s\t 住址: %s\t 電話:%s\n",
$object->StudentID, $object->Name, $object->Address, $object->Tel);
        }
        $result->close();
    }
    $adoconn->close();
?>
```

看起來很像又不大像的樣子，好像都改成物件導向式的操作，但這也沒什麼

了不起啊！其實啊，除了物件導向外，只要把第一行的 mysql 改成 mssql 或是 oci8，下面的物件導向 SQL 指令碼可以幾乎不用更動到，就把後端資料庫換成 MS-SQL 或是 Oracle 了喔，很神奇吧！

3.2.3 jpGraph

看到許多網站上的動態圖片，或是自動產生的統計圖片好羨幕喔！不需要看著別人網站流口水，有了 PHP，我們也可以自己來畫個幾張！

jpGraph 是以 PHP 加上 GD Library，折線圖、長條圖、圓餅圖甚至 3D 雷達圖都可以輕輕鬆鬆的畫出來呢！

趕快來看看它的範例圖片吧！http://203.72.185.101/jpgraph/sample/

Chapter 4. Some Projects

4.1 phpBB

phpBB 是目前網路論壇中使用相當廣泛的軟體，若是大家有常在網路世界中遨遊的話，相信一定遇過它！比如市網新版討論區、pcdvd 的數位影音討論群組、或是 IBM 小黑的 TP 非官方情報站，都是使用 phpBB 架設的，它的功能強大，也非常的 scalable，大家有興趣也可以試著架一個給學生來討論事務用，不用大家都跑去 yahoo 給它們賺流量及廣告☺

4.2 Xoops

Xoops 號稱是一整套模組化的小型入口網站軟體套件，舉凡討論區、精華文章、投票功能、友站連結、FAQ 問題與解答、電子相簿、新聞區、…等，都是一個個模組，可以視需要的加入入口網站。也有一些學校已使用這個套件來架設學校網頁。這樣的優點是快速、方便，但缺點就是大家長相不會差太多，而模組化又讓它無法太彈性。Anyway, 大家還是可以試著玩看看☺

Lab

A. 請試著撰寫一個有六個文字輸入（input type=text）的表單（form），並在 PHP 程式中計算此六個值的和、乘積以及平均值。

B. 請以陣列方式重新撰寫上個程式。
Hint: for loop or array_sum();

C. 請在上個程式中利用陣列的排序函數對此六個值做升冪排序。
Hint: asort();

D. 請找一篇英文文章，依其英文單字出現的次數，列出最常見的十個單字。
Hint: explode(), asort();

E. 請試著將上個 Lab 的英文文章存入伺服器裡。
Hint: fopen(), fputs(), fclose() or file_put_contents();

F. 請將之前 Lab 所用的英文文章先存入本機的純文字檔案，再撰寫一上傳介面與程式上傳該檔案至伺服器，檔名請維持不變。
Hint: move_uploaded_file();

G. 請試著實作一認證頁面，並將上個 Lab 的上傳介面整合入通過認證後的畫面，然後讓通過認證的使用者才可上傳，未通過認證者導回至認證頁面。（帳號與密碼請先寫死在程式裡）
Hint: Session functions, header();

H. 請試著在 MySQL 資料庫中新增一個 passwd 資料表，準備記錄著帳號與密碼的對應，密碼部份請以 md5 編碼過後儲存。
Hint: md5(), strlen(), Using phpMyAdmin

I. 請整合上兩個 Lab，將認證部份的帳號及密碼改由 MySQL 中取得。
Hint: strcmp() or select clause…

J. 請試著在 MySQL 資料庫中新增一個 uploads 資料表，準備記錄著哪個帳號在何時、何地上傳了哪些檔案。
Hint: date(), $_SERVER[“REMOTE_ADDR”];

K. 整合以上認證、上傳、記錄的功能，完成一個簡單不實用的檔案上傳系統。

Homework

Homework Upload System

從古早的自己或小老師人工收作業，到請學生 email 或 FTP 上傳作業，何不再進一步到請學生利用網頁介面上傳作業呢！這樣連檔案整理都不大需要，程式都作好一切了，而且一目瞭然…

那個 13 班 38 號，每次都不交作業，或是晚個一兩天交，還說是他有寄，我信箱爆了沒收到。這下賴不掉了吧！

Note：別忘了加上認證功能喔！不然真的得開啓檔案或是問遍所教授的十來個班級才知道這檔案是誰上傳喔！

Online Scoring System

有了家庭作業上傳系統後，就再接再厲，來個線上評分系統吧！學生的作業線上開啓，線上評完分存入資料庫。學生可以隨時來檢查自己的作業做的如何，缺了哪些東西。學期末時我們再也不用再手忙腳亂的一個個學生去催作業、算平均了，連所教授十幾個班級的各班平均、標準差都可以畫成美美的圖出來呢！

那個 13 班 38 號終於知死來抱我大腿了，真是大快人心啊！可我才不會讓他這麼輕易的 pass 哩，頂多全班加 5 分讓他 49 分死當！

Note：可千萬要做好備份喔，免得電腦硬碟一出問題，那就成一場空了！

Reference

- http://www.php.net/
- http://www.mysql.com/
- http://sourceforge.net/projects/phpmyadmin/
- http://php.weblogs.com/ADODB/
- http://www.aditus.nu/jpgraph/
- http://www.phpbb.com/
- http://www.xoops.com/
- http://www.hotscripts.com/PHP/